Date and time functions

- `bofd("cal", ed)` returns the `eb` business date corresponding to `ed`.
- `Cdhms(ed, h, m, s)` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to `ed`, `h`, `m`, `s`.
- `Chms(h, m, s)` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to `h`, `m`, `s` on 01jan1960.
- `Clock(s1, s2[, Y])` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to `s1` based on `s2` and `Y`.
- `clock(s1, s2[, Y])` returns the `etC` datetime (ms. since 01jan1960 00:00:00.000) corresponding to `s1` based on `s2` and `Y`.
- `Cmdyhms(M, D, Y, h, m, s)` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to `M`, `D`, `Y`, `h`, `m`, `s`.
- `Cofc(etC)` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of `etC` (ms. without leap seconds since 01jan1960 00:00:00.000).
- `cofC(etC)` returns the `etC` datetime (ms. without leap seconds since 01jan1960 00:00:00.000) of `etC` (ms. with leap seconds since 01jan1960 00:00:00.000).
- `Cofd(ed)` returns the `etC` datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date `ed` at time 00:00:00.000.
- `cofd(ed)` returns the `etC` datetime (ms. since 01jan1960 00:00:00.000) of date `ed` at time 00:00:00.000.
- `daily(s1, s2[, Y])` is a synonym for `date(s1, s2[, Y])`.
- `date(s1, s2[, Y])` returns the `ed` date (days since 01jan1960) corresponding to `s1` based on `s2` and `Y`.
- `day(ed)` returns the numeric day of the month corresponding to `ed`.
- `dhms(ed, h, m, s)` returns the `etC` datetime (ms. since 01jan1960 00:00:00.000) corresponding to `ed`, `h`, `m`, and `s`.
- `dofb(eb, "cal")` returns the `ed` date (days since 01jan1960) corresponding to `eb`.
- `dofC(etC)` returns the `ed` date (days since 01jan1960) of datetime `etC` (ms. with leap seconds since 01jan1960 00:00:00.000).
dofc(\texttt{e_{tc}}) \quad \text{the e_d date (days since 01jan1960) of datetime e_{tc} (ms. since 01jan1960 00:00:00.000)}

dofh(\texttt{e_h}) \quad \text{the e_d date (days since 01jan1960) of the start of half-year e_h}

dofm(\texttt{e_m}) \quad \text{the e_d date (days since 01jan1960) of the start of month e_m}

dofq(\texttt{e_q}) \quad \text{the e_d date (days since 01jan1960) of the start of quarter e_q}

dofw(\texttt{e_w}) \quad \text{the e_d date (days since 01jan1960) of 01jan in year e_y}

dofy(\texttt{e_y}) \quad \text{the numeric day of the year corresponding to date e_d; 0 = Sunday, 1 = Monday, ..., 6 = Saturday}

doy(\texttt{e_d}) \quad \text{the numeric day of the week corresponding to date e_d}

halfyear(\texttt{e_d}) \quad \text{the numeric half of the year corresponding to date e_d}

halfyearly(\texttt{s_1,s_2[ ,Y]}) \quad \text{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()}

hh(\texttt{e_{tc}}) \quad \text{the hour corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)}

hhC(\texttt{e_{tC}}) \quad \text{the hour corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)}

hms(\texttt{h,m,s}) \quad \text{the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960}

hodf(\texttt{e_d}) \quad \text{the e_h half-yearly date (half years since 1960h1) containing date e_d}

hours(\texttt{ms}) \quad \text{the e_d date (days since 01jan1960) corresponding to M, D, Y}

mdy(M,D,Y) \quad \text{the e_d date (days since 01jan1960) corresponding to M, D, Y}

mdyhms(M,D,Y,h,m,s) \quad \text{the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s}

minutes(\texttt{ms}) \quad \text{the minute corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)}

mm(\texttt{e_{tc}}) \quad \text{the minute corresponding to datetime e_{tc} (ms. with leap seconds since 01jan1960 00:00:00.000)}

mmC(\texttt{e_{tC}}) \quad \text{the minute corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)}

mofd(\texttt{e_d}) \quad \text{the e_m monthly date (months since 1960m1) containing date e_d}

month(\texttt{e_d}) \quad \text{the numeric month corresponding to date e_d}

monthly(\texttt{s_1,s_2[ ,Y]}) \quad \text{the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and Y; Y specifies topyear; see date()}

msofhours(\texttt{h}) \quad h \times 3,600,000

msofminutes(\texttt{m}) \quad m \times 60,000

msofseconds(\texttt{s}) \quad s \times 1,000

qofd(\texttt{e_d}) \quad \text{the e_q quarterly date (quarters since 1960q1) containing date e_d}

quarter(\texttt{e_d}) \quad \text{the numeric quarter of the year corresponding to date e_d}

quarterly(\texttt{s_1,s_2[ ,Y]}) \quad \text{the e_q quarterly date (quarters since 1960q1) corresponding to s_1 based on s_2 and Y; Y specifies topyear; see date()}

seconds(\texttt{ms}) \quad \text{the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)}

ss(\texttt{e_{tc}}) \quad \text{the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)}

ssC(\texttt{e_{tC}}) \quad \text{the second corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)}
Functions by category

*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

*th(l)*  
convenience function to make typing half-yearly dates in expressions easier

*tm(l)*  
convenience function to make typing monthly dates in expressions easier

*tq(l)*  
convenience function to make typing quarterly dates in expressions easier

*tw(l)*  
convenience function to make typing weekly dates in expressions easier

*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()*

*wofd(e_d)*  
the *e_w* weekly date (weeks since 1960w1) containing date *e_d*

*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*

Mathematical functions

*abs(x)*  
the absolute value of *x*

*ceil(x)*  
the unique integer *n* such that *n* − 1 < *x* ≤ *n*; *x* (not “.”) if *x* is missing, meaning that *ceil(.a) = .a*

*cloglog(x)*  
the complementary log-log of *x*

*comb(n,k)*  
the combinatorial function *n!/\{k!(n−k)!\}*

*digamma(x)*  
the digamma() function, *d ln Γ(x)/dx*

*exp(x)*  
the exponential function *e^x*

*expm1(x)*  
e^*x* − 1 with higher precision than *exp(x) − 1* for small values of |*x*|

*floor(x)*  
the unique integer *n* such that *n* ≤ *x* < *n* + 1; *x* (not “.”) if *x* is missing, meaning that *floor(.a) = .a*

*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*
invclloglog(x)  
the inverse of the complementary log-log function of x

invlogit(x)  
the inverse of the logit function of x

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 |x|

ln1p(x)  
the natural logarithm of 1 + x with higher precision than ln(1 + x) for small values of |x|

lnfactorial(n)  
the natural log of n factorial = ln(n!)

lngamma(x)  
ln\{Γ(x)\}

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)

logit(x)  
the log of the odds ratio of x, logit(x) = ln{x/(1 − x)}

max(x₁, x₂, ..., xₙ)  
the maximum value of x₁, x₂, ..., xₙ

min(x₁, x₂, ..., xₙ)  
the minimum value of x₁, x₂, ..., xₙ

mod(x, y)  
the modulus of x with respect to y

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

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

sign(x)  
the sign of x: −1 if x < 0, 0 if x = 0, 1 if x > 0, or missing if x is missing

sqrt(x)  
the square root of x

sum(x)  
the running sum of x, treating missing values as zero

trigamma(x)  
the second derivative of lngamma(x) = d² lnΓ(x)/dx²

trunc(x)  
a synonym for int(x)

Matrix functions

cholesky(M)  
the Cholesky decomposition of the matrix: if R = cholesky(S), then RRᵀ = S

coleqnumb(M, s)  
the equation number of M associated with column equation s; missing if the column equation cannot be found

colinfreeparms(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

corr(M)  
the correlation matrix of the variance matrix

det(M)  
the determinant of matrix M
Functions by category

- **diag(M)**: the square, diagonal matrix created from the row or column vector
- **diag0cnt(M)**: the number of zeros on the diagonal of \( M \)
- **el(s,i,j)**: \( s[\text{floor}(i),\text{floor}(j)] \), 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
- **get(systemname)**: a copy of Stata internal system matrix \( \text{systemname} \)
- **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)
- **I(n)**: an \( n \times n \) identity matrix if \( n \) is an integer; otherwise, a \( \text{round}(n) \times \text{round}(n) \) identity matrix
- **inv(M)**: the inverse of the matrix \( M \)
- **invsym(M)**: the inverse of \( M \) if \( M \) is positive definite
- **issymmetric(M)**: 1 if the matrix is symmetric; otherwise, 0
- **J(r,c,z)**: the \( r \times c \) matrix containing elements \( z \)
- **matmissing(M)**: 1 if any elements of the matrix are missing; otherwise, 0
- **matuniform(r,c)**: the \( r \times c \) matrices containing uniformly distributed pseudorandom numbers on the interval \((0,1)\)
- **mreldif(X,Y)**: the relative difference of \( X \) and \( Y \), where the relative difference is defined as \( \max_{i,j} \{|x_{ij} - y_{ij}|/(|y_{ij}| + 1)\} \)
- **nullmat(matname)**: use with the row-join (,) and column-join (\) operators
- **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 \( M \)
- **sweep(M,i)**: matrix \( M \) with \( i \)th row/column swept
- **trace(M)**: the trace of matrix \( M \)
- **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 \( M \)

Programming functions

- **autocode(x,n,x0,x1)**: partitions the interval from \( x_0 \) to \( x_1 \) into \( n \) equal-length intervals and returns the upper bound of the interval that contains \( x \)
- **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
- **chop(x, \epsilon)**: \( \text{round}(x) \) if \( \text{abs}(x - \text{round}(x)) < \epsilon \); otherwise, \( x \); or \( x \) if \( x \) is missing
- **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
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

\( e(name) \)
the value of stored result \( e(name) \); see [U] 18.8 Accessing results calculated by other programs

\( e(sample) \)
1 if the observation is in the estimation sample and 0 otherwise

\( \text{epsdouble()} \)
the machine precision of a double-precision number

\( \text{epsfloat()} \)
the machine precision of a floating-point number

\( \text{fileexists(f)} \)
1 if the file specified by \( f \) exists; otherwise, 0

\( \text{fileread(f)} \)
the contents of the file specified by \( f \)

\( \text{filereaderror(s)} \)
0 or positive integer, said value having the interpretation of a return code

\( \text{fwrite(f,s[,r])} \)
writes the string specified by \( s \) to the file specified by \( f \) and returns the number of bytes in the resulting file

\( \text{float(x)} \)
the value of \( x \) rounded to float precision

\( \text{fmtwidth(fmtstr)} \)
the output length of the \%fmt contained in \( fmtstr \); missing if \( fmtstr \) does not contain a valid \%fmt

\( \text{frval()} \)
returns values of variables stored in other frames

\( \text{_frval()} \)
programmer’s version of \( \text{frval()} \)

\( \text{has_e-prop(name)} \)
1 if \( name \) appears as a word in \( e(properties) \); otherwise, 0

\( \text{inlist(z,a,b,...)} \)
1 if \( z \) is a member of the remaining arguments; otherwise, 0

\( \text{inrange(z,a,b)} \)
1 if it is known that \( a \leq z \leq b \); otherwise, 0

\( \text{irecode(x,x_1,...,x_n)} \)

\( \text{matrix(exp)} \)
restricts name interpretation to scalars and matrices; see \( \text{scalar()} \)

\( \text{maxbyte()} \)
the largest value that can be stored in storage type byte

\( \text{maxdouble()} \)
the largest value that can be stored in storage type double

\( \text{maxfloat()} \)
the largest value that can be stored in storage type float

\( \text{maxint()} \)
the largest value that can be stored in storage type int

\( \text{maxlong()} \)
the largest value that can be stored in storage type long

\( \text{mi(x_1,x_2,...,x_n)} \)
a synonym for \( \text{missing(x_1,x_2,...,x_n)} \)

\( \text{minbyte()} \)
the smallest value that can be stored in storage type byte

\( \text{mindouble()} \)
the smallest value that can be stored in storage type double

\( \text{minfloat()} \)
the smallest value that can be stored in storage type float

\( \text{minint()} \)
the smallest value that can be stored in storage type int

\( \text{minlong()} \)
the smallest value that can be stored in storage type long

\( \text{missing(x_1,x_2,...,x_n)} \)
the value of \( x_i \) evaluates to missing; otherwise, 0

\( \text{r(name)} \)
the value of the stored result \( r(name) \); see [U] 18.8 Accessing results calculated by other programs

\( \text{recode(x,x_1,...,x_n)} \)
missing if \( x_1,x_2,...,x_n \) 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 \( x_i = +\infty \)

\( \text{replay()} \)
1 if the first nonblank character of local macro ‘0’ is a comma, or if ‘0’ is empty
Functions by category

```
return(name)  the value of the to-be-stored result r(name); see [P]
        return
        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
        smallestdouble() the smallest double-precision number greater than zero
```

Random-number functions

```
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
rchisq(df)  chi-squared, with df degrees of freedom, random variates
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 π/√3
rlogistic(s)  logistic variates with mean 0, scale s, and standard deviation sπ/√3
rlogistic(m,s)  logistic variates with mean m, scale s, and standard deviation
            sπ/√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
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]
weibull(a,b)  Weibull variates with shape a and scale b
weibull(a,b,g)  Weibull variates with shape a, scale b, and location g
weibullph(a,b)  Weibull (proportional hazards) variates with shape a and scale b
weibullph(a,b,g)  Weibull (proportional hazards) variates with shape a, scale b, and
            location g
```
Selecting time-span functions

\[ \text{tin}(d_1, d_2) \quad \text{true if } d_1 \leq t \leq d_2, \text{ where } t \text{ is the time variable previously } tsset \]

\[ \text{twithin}(d_1, d_2) \quad \text{true if } d_1 < t < d_2, \text{ where } t \text{ is the time variable previously } tsset \]

Statistical functions

\[ \text{betaden}(a, b, x) \quad \text{the probability density of the beta distribution, where } a \text{ and } b \text{ are the shape parameters; } 0 \text{ if } x < 0 \text{ or } x > 1 \]

\[ \text{binomial}(n, k, \theta) \quad \text{the probability of observing floor}(k) \text{ or fewer successes in } floor(n) \text{ trials when the probability of a success on one trial is } \theta; \text{ } 0 \text{ if } k < 0; \text{ or } 1 \text{ if } k > n \]

\[ \text{binomialp}(n, k, p) \quad \text{the probability of observing floor}(k) \text{ successes in } floor(n) \text{ trials when the probability of a success on one trial is } p \]

\[ \text{binomialtail}(n, k, \theta) \quad \text{the probability of observing floor}(k) \text{ or more successes in } floor(n) \text{ trials when the probability of a success on one trial is } \theta; \text{ } 0 \text{ if } k < 0; \text{ or } 0 \text{ if } k > n \]

\[ \text{binormal}(h, k, \rho) \quad \text{the joint cumulative distribution } \Phi(h, k, \rho) \text{ of bivariate normal with correlation } \rho \]

\[ \text{cauchy}(a, b, x) \quad \text{the cumulative Cauchy distribution with location parameter } a \text{ and scale parameter } b \]

\[ \text{cauchyden}(a, b, x) \quad \text{the probability density of the Cauchy distribution with location parameter } a \text{ and scale parameter } b \]

\[ \text{cauchytail}(a, b, x) \quad \text{the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter } a \text{ and scale parameter } b \]

\[ \text{chi2}(df, x) \quad \text{the cumulative } \chi^2 \text{ distribution with } df \text{ degrees of freedom; } 0 \text{ if } x < 0 \]

\[ \text{chi2den}(df, x) \quad \text{the probability density of the chi-squared distribution with } df \text{ degrees of freedom; } 0 \text{ if } x < 0 \]

\[ \text{chi2tail}(df, x) \quad \text{the reverse cumulative (upper tail or survivor) } \chi^2 \text{ distribution with } df \text{ degrees of freedom; } 1 \text{ if } x < 0 \]

\[ \text{dgammapda}(a, x) \quad \frac{\partial P(a, x)}{\partial a}, \text{ where } P(a, x) = \text{gammap}(a, x); \text{ } 0 \text{ if } x < 0 \]

\[ \text{dgammapdada}(a, x) \quad \frac{\partial^2 P(a, x)}{\partial a^2}, \text{ where } P(a, x) = \text{gammap}(a, x); \text{ } 0 \text{ if } x < 0 \]

\[ \text{dgammapdadx}(a, x) \quad \frac{\partial^2 P(a, x)}{\partial a \partial x}, \text{ where } P(a, x) = \text{gammap}(a, x); \text{ } 0 \text{ if } x < 0 \]

\[ \text{dgammapdxdx}(a, x) \quad \frac{\partial^2 P(a, x)}{\partial x^2}, \text{ where } P(a, x) = \text{gammap}(a, x); \text{ } 0 \text{ if } x < 0 \]

\[ \text{dunnettprob}(k, df, x) \quad \text{the cumulative multiple range distribution that is used in Dunnett’s multiple-comparison method with } k \text{ ranges and } df \text{ degrees of freedom; } 0 \text{ if } x < 0 \]

\[ \text{exponential}(b, x) \quad \text{the cumulative exponential distribution with scale } b \]

\[ \text{exponentialden}(b, x) \quad \text{the probability density function of the exponential distribution with scale } b \]

\[ \text{exponentialtail}(b, x) \quad \text{the reverse cumulative exponential distribution with scale } b \]
Functions by category

\[ F(df_1, df_2, f) \]
the cumulative \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom: \( F(df_1, df_2, f) = \int_0^f F_{\text{den}}(df_1, df_2, t) \, dt; 0 \text{ if } f < 0 \)

\[ F_{\text{den}}(df_1, df_2, f) \]
the probability density function of the \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom; 0 if \( f < 0 \)

\[ F_{\text{tail}}(df_1, df_2, f) \]
the reverse cumulative (upper tail or survivor) \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom; 1 if \( f < 0 \)

\[ \text{gammaden}(a, b, g, x) \]
the probability density function of the gamma distribution; 0 if \( x < g \)

\[ \text{gammmap}(a, x) \]
the cumulative gamma distribution with shape parameter \( a \); 0 if \( x < 0 \)

\[ \text{gammaptail}(a, x) \]
the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter \( a \); 1 if \( x < 0 \)

\[ \text{hypergeometric}(N, K, n, k) \]
the cumulative probability of the hypergeometric distribution

\[ \text{hypergeometricp}(N, K, n, k) \]
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

\[ \text{ibeta}(a, b, x) \]
the cumulative beta distribution with shape parameters \( a \) and \( b \); 0 if \( x < 0 \); or 1 if \( x > 1 \)

\[ \text{ibetatail}(a, b, x) \]
the reverse cumulative (upper tail or survivor) beta distribution with shape parameters \( a \) and \( b \); 1 if \( x < 0 \); or 0 if \( x > 1 \)

\[ \text{igaussian}(m, a, x) \]
the cumulative inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 0 if \( x \leq 0 \)

\[ \text{igaussianden}(m, a, x) \]
the probability density of the inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 0 if \( x \leq 0 \)

\[ \text{igaussiantail}(m, a, x) \]
the reverse cumulative (upper tail or survivor) inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 1 if \( x \leq 0 \)

\[ \text{invbinomial}(n, k, p) \]
the inverse of the cumulative binomial; that is, \( \theta (\theta = \text{probability of success on one trial}) \) such that the probability of observing \( \text{floor}(k) \) or fewer successes in \( \text{floor}(n) \) trials is \( p \)

\[ \text{invbinomialaltail}(n, k, p) \]
the inverse of the right cumulative binomial; that is, \( \theta (\theta = \text{probability of success on one trial}) \) such that the probability of observing \( \text{floor}(k) \) or more successes in \( \text{floor}(n) \) trials is \( p \)

\[ \text{invcauchy}(a, b, p) \]
the inverse of cauchy(); if \( \text{cauchy}(a, b, x) = p \), then \( \text{invcauchy}(a, b, p) = x \)

\[ \text{invcauchytail}(a, b, p) \]
the inverse of cauchytail(); if \( \text{cauchytail}(a, b, x) = p \), then \( \text{invcauchytail}(a, b, p) = x \)

\[ \text{invchi2}(df, p) \]
the inverse of \( \chi^2() \); if \( \chi^2(df, x) = p \), then \( \text{invchi2}(df, p) = x \)

\[ \text{invchi2tail}(df, p) \]
the inverse of \( \chi^2\text{tail}() \); if \( \chi^2\text{tail}(df, x) = p \), then \( \text{invchi2tail}(df, p) = x \)

\[ \text{invdunnettprob}(k, df, p) \]
the inverse cumulative multiple range distribution that is used in Dunnett’s multiple-comparison method with \( k \) ranges and \( df \) degrees of freedom

\[ \text{invexponential}(b, p) \]
the inverse cumulative exponential distribution with scale \( b \); if \( \text{exponential}(b, x) = p \), then \( \text{invexponential}(b, p) = x \)
invexponentialtail(b,p)  the inverse reverse cumulative exponential distribution with scale b: if \( \text{exponentialtail}(b,x) = p \), then \( \text{invexponentialtail}(b,p) = x \)

invF(df1,df2,p)  the inverse cumulative \( F \) distribution: if \( F(df_1,df_2,f) = p \), then \( \text{invF}(df_1,df_2,p) = f \)

invFtail(df1,df2,p)  the inverse reverse cumulative (upper tail or survivor) \( F \) distribution: if \( \text{Tail}(df_1,df_2,f) = p \), then \( \text{invFtail}(df_1,df_2,p) = f \)

invgammap(a,p)  the inverse cumulative gamma distribution: if \( \text{gammmap}(a,x) = p \), then \( \text{invgammap}(a,p) = x \)

invgammaptail(a,p)  the inverse reverse cumulative (upper tail or survivor) gamma distribution: if \( \text{gammmap}(a,x) = p \), then \( \text{invgammap}(a,p) = x \)

invibeta(a,b,p)  the inverse cumulative beta distribution: if \( \text{ibeta}(a,b,x) = p \), then \( \text{invibeta}(a,b,p) = x \)

invibetatail(a,b,p)  the inverse reverse cumulative (upper tail or survivor) beta distribution: if \( \text{ibetatail}(a,b,x) = p \), then \( \text{invibetatail}(a,b,p) = x \)

invgaussian(m,a,p)  the inverse of \( \text{igaussian}() \): if \( \text{igaussian}(m,a,x) = p \), then \( \text{invgaussian}(m,a,p) = x \)

invgaussiantail(m,a,p)  the inverse of \( \text{igaussiantail}() \): if \( \text{igaussiantail}(m,a,x) = p \), then \( \text{invgaussiantail}(m,a,p) = x \)

invlaplace(m,b,p)  the inverse of \( \text{laplace}() \): if \( \text{laplace}(m,b,x) = p \), then \( \text{invlaplace}(m,b,p) = x \)

invlaplacetail(m,b,p)  the inverse of \( \text{laplacetail}() \): if \( \text{laplacetail}(m,b,x) = p \), then \( \text{invlaplacetail}(m,b,p) = x \)

invlogistic(p)  the inverse cumulative logistic distribution: if \( \text{logistic}(x) = p \), then \( \text{invlogistic}(p) = x \)

invlogistic(s,p)  the inverse cumulative logistic distribution: if \( \text{logistic}(s,x) = p \), then \( \text{invlogistic}(s,p) = x \)

invlogistic(m,s,p)  the inverse cumulative logistic distribution: if \( \text{logistic}(m,s,x) = p \), then \( \text{invlogistic}(m,s,p) = x \)

invlogistictail(p)  the inverse reverse cumulative logistic distribution: if \( \text{logistictail}(x) = p \), then \( \text{invlogistictail}(p) = x \)

invlogistictail(s,p)  the inverse reverse cumulative logistic distribution: if \( \text{logistictail}(s,x) = p \), then \( \text{invlogistictail}(s,p) = x \)

invlogistictail(m,s,p)  the inverse reverse cumulative logistic distribution: if \( \text{logistictail}(m,s,x) = p \), then \( \text{invlogistictail}(m,s,p) = x \)

invnbinomial(n,k,q)  the value of the negative binomial parameter, \( p \), such that \( q = \text{nbinomial}(n,k,p) \)

invnbinomialtail(n,k,q)  the value of the negative binomial parameter, \( p \), such that \( q = \text{nbinomialtail}(n,k,p) \)

invnchi2(df,np,p)  the inverse cumulative noncentral \( \chi^2 \) distribution: if \( \text{nchi2}(df,np,x) = p \), then \( \text{invnchi2}(df,np,p) = x \)

invnchi2tail(df,np,p)  the inverse reverse cumulative (upper tail or survivor) noncentral \( \chi^2 \) distribution: if \( \text{nchi2tail}(df,np,x) = p \), then \( \text{invnchi2tail}(df,np,p) = x \)
The inverse cumulative noncentral $F$ distribution: if \( \text{invF}(df_1, df_2, np, p) = f \), then \( \text{invF}(df_1, df_2, np, p) = f \).

The inverse reverse cumulative (upper tail or survivor) noncentral $F$ distribution: if \( \text{nF}(df_1, df_2, np, f) = p \), then \( \text{invF}(df_1, df_2, np, p) = f \).

The inverse cumulative noncentral beta distribution: if \( \text{invbeta}(a, b, np, x) = p \), then \( \text{invbeta}(a, b, np, p) = x \).

The inverse cumulative standard normal distribution: if \( \text{normal}(z) = p \), then \( \text{invnormal}(p) = z \).

The inverse cumulative noncentral Student’s $t$ distribution: if \( \text{nt}(df, np, t) = p \), then \( \text{invnt}(df, np, p) = t \).

The inverse reverse cumulative (upper tail or survivor) noncentral Student’s $t$ distribution: if \( \text{nttail}(df, np, t) = p \), then \( \text{invnttail}(df, np, p) = t \).

The Poisson mean such that the cumulative Poisson distribution evaluated at $k$ is $p$: if \( \text{poisson}(m, k) = p \), then \( \text{invpoisson}(k, p) = m \).

The Poisson mean such that the cumulative Poisson distribution evaluated at $k$ is $q$: if \( \text{poisson}(m, k) = q \), then \( \text{invpoisson}(k, q) = m \).

The inverse cumulative Student’s $t$ distribution: if \( t(df, t) = p \), then \( \text{invt}(df, p) = t \).

The inverse reverse cumulative (upper tail or survivor) Student’s $t$ distribution: if \( ttail(df, t) = p \), then \( \text{invttail}(df, p) = t \).

The inverse cumulative Tukey’s Studentized range distribution with $k$ ranges and $df$ degrees of freedom.

The inverse cumulative Weibull distribution with shape $a$ and scale $b$: if \( \text{weibull}(a, b, x) = p \), then \( \text{invweibull}(a, b, p) = x \).

The inverse cumulative Weibull distribution with shape $a$, scale $b$, and location $g$: if \( \text{weibull}(a, b, g, x) = p \), then \( \text{invweibull}(a, b, g, p) = x \).

The inverse cumulative Weibull (proportional hazards) distribution with shape $a$ and scale $b$: if \( \text{weibullph}(a, b, x) = p \), then \( \text{invweibullph}(a, b, p) = x \).

The inverse cumulative Weibull (proportional hazards) distribution with shape $a$, scale $b$, and location $g$: if \( \text{weibullph}(a, b, g, x) = p \), then \( \text{invweibullph}(a, b, g, p) = x \).

The inverse reverse cumulative Weibull (proportional hazards) distribution with shape $a$ and scale $b$: if \( \text{weibullphtail}(a, b, x) = p \), then \( \text{invweibullphtail}(a, b, p) = x \).

The inverse reverse cumulative Weibull (proportional hazards) distribution with shape $a$, scale $b$, and location $g$: if \( \text{weibullphtail}(a, b, g, x) = p \), then \( \text{invweibullphtail}(a, b, g, p) = x \).

The inverse reverse cumulative Weibull distribution with shape $a$ and scale $b$: if \( \text{weibulltail}(a, b, x) = p \), then \( \text{invweibulltail}(a, b, p) = x \).

The inverse reverse cumulative Weibull distribution with shape $a$, scale $b$, and location $g$: if \( \text{weibulltail}(a, b, g, x) = p \), then \( \text{invweibulltail}(a, b, g, p) = x \).
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>laplace(m,b,x)</code></td>
<td>the cumulative Laplace distribution with mean ( m ) and scale parameter ( b )</td>
</tr>
<tr>
<td><code>laplaceden(m,b,x)</code></td>
<td>the probability density of the Laplace distribution with mean ( m ) and scale parameter ( b )</td>
</tr>
<tr>
<td><code>laplacetail(m,b,x)</code></td>
<td>the reverse cumulative (upper tail or survivor) Laplace distribution with mean ( m ) and scale parameter ( b )</td>
</tr>
<tr>
<td><code>lncauchyden(a,b,x)</code></td>
<td>the natural logarithm of the density of the Cauchy distribution with location parameter ( a ) and scale parameter ( b )</td>
</tr>
<tr>
<td><code>lnigammaden(a,b,x)</code></td>
<td>the natural logarithm of the inverse gamma density, where ( a ) is the shape parameter and ( b ) is the scale parameter</td>
</tr>
<tr>
<td><code>lnigaussianden(m,a)</code></td>
<td>the natural logarithm of the inverse Gaussian density with mean ( m ) and shape parameter ( a )</td>
</tr>
<tr>
<td><code>lniwishartden(df,V,X)</code></td>
<td>the natural logarithm of the density of the inverse Wishart distribution; missing if ( df \leq n - 1 )</td>
</tr>
<tr>
<td><code>lnmvnormalden(M,V,X)</code></td>
<td>the natural logarithm of the multivariate normal density</td>
</tr>
<tr>
<td><code>lnnormal(z)</code></td>
<td>the natural logarithm of the cumulative standard normal distribution</td>
</tr>
<tr>
<td><code>lnnormalden(z)</code></td>
<td>the natural logarithm of the standard normal density, ( N(0, 1) )</td>
</tr>
<tr>
<td><code>lnnormalden(x,σ)</code></td>
<td>the natural logarithm of the normal density with mean 0 and standard deviation ( σ )</td>
</tr>
<tr>
<td><code>lnnormalden(x,μ,σ)</code></td>
<td>the natural logarithm of the normal density with mean ( μ ) and standard deviation ( σ ), ( N(μ, σ^2) )</td>
</tr>
<tr>
<td><code>lnwishartden(df,V,X)</code></td>
<td>the natural logarithm of the density of the Wishart distribution; missing if ( df \leq n - 1 )</td>
</tr>
<tr>
<td><code>logistic(x)</code></td>
<td>the cumulative logistic distribution with mean 0 and standard deviation ( π/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logistic(s,x)</code></td>
<td>the cumulative logistic distribution with mean 0, scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logistic(m,s,x)</code></td>
<td>the cumulative logistic distribution with mean ( m ), scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logisticden(x)</code></td>
<td>the density of the logistic distribution with mean 0 and standard deviation ( π/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logisticden(s,x)</code></td>
<td>the density of the logistic distribution with mean 0, scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logisticden(m,s,x)</code></td>
<td>the density of the logistic distribution with mean ( m ), scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logistictail(x)</code></td>
<td>the reverse cumulative logistic distribution with mean 0 and standard deviation ( π/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logistictail(s,x)</code></td>
<td>the reverse cumulative logistic distribution with mean 0, scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>logistictail(m,s,x)</code></td>
<td>the reverse cumulative logistic distribution with mean ( m ), scale ( s ), and standard deviation ( sπ/\sqrt{3} )</td>
</tr>
<tr>
<td><code>nbetaden(a,b,np,x)</code></td>
<td>the probability density function of the noncentral beta distribution; 0 if ( x &lt; 0 ) or ( x &gt; 1 )</td>
</tr>
<tr>
<td><code>nbinomial(n,k,p)</code></td>
<td>the cumulative probability of the negative binomial distribution</td>
</tr>
</tbody>
</table>
\texttt{normal}(z) \quad \text{the cumulative standard normal distribution}

\texttt{normalden}(z) \quad \text{the standard normal density, \(N(0,1)\)}

\texttt{normalden}(x, \sigma) \quad \text{the normal density with mean 0 and standard deviation \(\sigma\)}

\texttt{normalden}(x, \mu, \sigma) \quad \text{the normal density with mean \(\mu\) and standard deviation \(\sigma\)}

\texttt{npnchi2}(df, x, p) \quad \text{the noncentrality parameter, \(np\), for noncentral \(\chi^2\): if \(\texttt{nchi2}(df, x, p) = p\), then \(\texttt{npnchi2}(df, x, p) = np\)}

\texttt{npnF}(df_1, df_2, f, p) \quad \text{the noncentrality parameter, \(np\), for the noncentral \(F\): if \(\texttt{nF}(df_1, df_2, np, f) = p\), then \(\texttt{npnF}(df_1, df_2, f, p) = np\)}

\texttt{npnt}(df, t, p) \quad \text{the noncentrality parameter, \(np\), for the noncentral Student’s \(t\) distribution: if \(\texttt{nt}(df, np, t) = p\), then \(\texttt{npnt}(df, t, p) = np\)}

\texttt{nt}(df, np, t) \quad \text{the cumulative noncentral Student’s \(t\) distribution with \(df\) degrees of freedom and noncentrality parameter \(np\)}

\texttt{ntden}(df, np, t) \quad \text{the probability density function of the noncentral Student’s \(t\) distribution with \(df\) degrees of freedom and noncentrality parameter \(np\)}

\texttt{nttail}(df, np, t) \quad \text{the reverse cumulative (upper tail or survivor) noncentral Student’s \(t\) distribution with \(df\) degrees of freedom and noncentrality parameter \(np\)}

\texttt{poisson}(m, k) \quad \text{the probability of observing \texttt{floor}(k) or fewer outcomes that are distributed as Poisson with mean \(m\)}

\texttt{poissonp}(m, k) \quad \text{the probability of observing \texttt{floor}(k) outcomes that are distributed as Poisson with mean \(m\)}

\texttt{poissontail}(m, k) \quad \text{the probability of observing \texttt{floor}(k) or more outcomes that are distributed as Poisson with mean \(m\)}

\texttt{t}(df, t) \quad \text{the cumulative Student’s \(t\) distribution with \(df\) degrees of freedom}

\texttt{tden}(df, t) \quad \text{the probability density function of Student’s \(t\) distribution}

\texttt{ttail}(df, t) \quad \text{the reverse cumulative (upper tail or survivor) Student’s \(t\) distribution; the probability \(T > t\)}

\texttt{tukeyprob}(k, df, x) \quad \text{the cumulative Tukey’s Studentized range distribution with \(k\) ranges and \(df\) degrees of freedom; 0 if \(x < 0\)}
14 Functions by category

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

String functions

abbrev(s, n) name s, abbreviated to a length of n
char(n) the character corresponding to ASCII or extended ASCII code n; "" if n is not in the domain
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)indexnot(s1, s2) the position in ASCII string s1 of the first character of s1 not found in ASCII string s2, or 0 if all characters of s1 are found in s2 plural(n, s) the plural of s if n \neq \pm 1 plural(n, s1, s2) the plural of s1, as modified by or replaced with s2, if n \neq \pm 1 real(s) s converted to numeric or missing regexm(s, re) performs a match of a regular expression and evaluates to 1 if regular expression re is satisfied by the ASCII string s; otherwise, 0 regexr(s1, re, s2) replaces the first substring within ASCII string s1 that matches re with ASCII string s2 and returns the resulting string regexs(n) subexpression n from a previous regexm() match, where 0 \leq n < 10 soundex(s) the soundex code for a string, s soundex_nara(s) the U.S. Census soundex code for a string, s
**Functions by category**

- **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; otherwise, `0`.
- **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**($s$): reverses the ASCII string `s`.
- **strrpos**($s_1, s_2$): the position in `$s_1$` at which `$s_2$` is last found; otherwise, `0`.
- **strrtrim**($s$): `s` without 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 `$n_2$`.
- **tobytes**($s[, n]$): escaped decimal or hex digit strings of up to 200 bytes of `s`.
- **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.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ustrfrom(s, enc, mode)</code></td>
<td>Converts the string <code>s</code> in encoding <code>enc</code> to a UTF-8 encoded Unicode string</td>
</tr>
<tr>
<td><code>ustrinvalidcnt(s)</code></td>
<td>The number of invalid UTF-8 sequences in <code>s</code></td>
</tr>
<tr>
<td><code>ustrleft(s, n)</code></td>
<td>The first <code>n</code> Unicode characters of the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrlen(s)</code></td>
<td>The number of characters in the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrlower(s[, loc])</code></td>
<td>Lowercase all characters of Unicode string <code>s</code> under the given locale <code>loc</code></td>
</tr>
<tr>
<td><code>ustrltrim(s)</code></td>
<td>Removes the leading Unicode whitespace characters and blanks from the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrnormalize(s, norm)</code></td>
<td>Normalizes Unicode string <code>s</code> to one of the five normalization forms specified by <code>norm</code></td>
</tr>
<tr>
<td><code>ustrpos(s1, s2[, n])</code></td>
<td>The position in <code>s1</code> at which <code>s2</code> is first found; otherwise, 0</td>
</tr>
<tr>
<td><code>ustrregexm(s, re[, noc])</code></td>
<td>Performs a match of a regular expression and evaluates to 1 if regular expression <code>re</code> is satisfied by the Unicode string <code>s</code>; otherwise, 0</td>
</tr>
<tr>
<td><code>ustrregexra(s1, re, s2[, noc])</code></td>
<td>Replaces all substrings within the Unicode string <code>s1</code> that match <code>re</code> with <code>s2</code> and returns the resulting string</td>
</tr>
<tr>
<td><code>ustrregexrf(s1, re, s2[, noc])</code></td>
<td>Replaces the first substring within the Unicode string <code>s1</code> that matches <code>re</code> with <code>s2</code> and returns the resulting string</td>
</tr>
<tr>
<td><code>ustrregexs(n)</code></td>
<td>Subexpression <code>n</code> from a previous <code>ustrregexm()</code> match</td>
</tr>
<tr>
<td><code>ustrreverse(s)</code></td>
<td>Reverses the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrright(s, n)</code></td>
<td>The last <code>n</code> Unicode characters of the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrrpos(s1, s2[, n])</code></td>
<td>The position in <code>s1</code> at which <code>s2</code> is last found; otherwise, 0</td>
</tr>
<tr>
<td><code>ustrrtrim(s)</code></td>
<td>Remove trailing Unicode whitespace characters and blanks from the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrsortkey(s[, loc])</code></td>
<td>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></td>
</tr>
<tr>
<td><code>ustrsortkeyex(s, loc, st, case, cslv, norm, num, alt, fr)</code></td>
<td>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></td>
</tr>
<tr>
<td><code>ustrtitle(s[, loc])</code></td>
<td>A string with the first characters of Unicode words titlecased and other characters lowercased</td>
</tr>
<tr>
<td><code>ustrto(s, enc, mode)</code></td>
<td>Converts the Unicode string <code>s</code> in UTF-8 encoding to a string in encoding <code>enc</code></td>
</tr>
<tr>
<td><code>ustrtohex(s[, n])</code></td>
<td>Escaped hex digit string of <code>s</code> up to 200 Unicode characters</td>
</tr>
<tr>
<td><code>ustrtoname(s[, p])</code></td>
<td>String <code>s</code> translated into a Stata name</td>
</tr>
<tr>
<td><code>ustrtrim(s)</code></td>
<td>Removes leading and trailing Unicode whitespace characters and blanks from the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrunescape(s)</code></td>
<td>The Unicode string corresponding to the escaped sequences of <code>s</code></td>
</tr>
<tr>
<td><code>ustruppper(s[, loc])</code></td>
<td>Uppercase all characters in string <code>s</code> under the given locale <code>loc</code></td>
</tr>
<tr>
<td><code>ustrword(s, n[, loc])</code></td>
<td>The <code>n</code>th Unicode word in the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>ustrwordcount(s[, loc])</code></td>
<td>The number of nonempty Unicode words in the Unicode string <code>s</code></td>
</tr>
<tr>
<td><code>usubinstr(s1, s2, s3, n)</code></td>
<td>Replaces the first <code>n</code> occurrences of the Unicode string <code>s2</code> with the Unicode string <code>s3</code> in <code>s1</code></td>
</tr>
<tr>
<td><code>ustrsubstr(s, n1, n2)</code></td>
<td>The Unicode substring of <code>s</code>, starting at <code>n1</code>, for a length of <code>n2</code></td>
</tr>
</tbody>
</table>
**word(s,n)**  
the $n$th 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 $s$

---

### Trigonometric functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>acos(x)</code></td>
<td>the radian value of the arccosine of $x$</td>
</tr>
<tr>
<td><code>acosh(x)</code></td>
<td>the inverse hyperbolic cosine of $x$</td>
</tr>
<tr>
<td><code>asin(x)</code></td>
<td>the radian value of the arcsine of $x$</td>
</tr>
<tr>
<td><code>asinh(x)</code></td>
<td>the inverse hyperbolic sine of $x$</td>
</tr>
<tr>
<td><code>atan(x)</code></td>
<td>the radian value of the arctangent of $x$</td>
</tr>
<tr>
<td><code>atan2(y, x)</code></td>
<td>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</td>
</tr>
<tr>
<td><code>atanh(x)</code></td>
<td>the inverse hyperbolic tangent of $x$</td>
</tr>
<tr>
<td><code>cos(x)</code></td>
<td>the cosine of $x$, where $x$ is in radians</td>
</tr>
<tr>
<td><code>cosh(x)</code></td>
<td>the hyperbolic cosine of $x$</td>
</tr>
<tr>
<td><code>sin(x)</code></td>
<td>the sine of $x$, where $x$ is in radians</td>
</tr>
<tr>
<td><code>sinh(x)</code></td>
<td>the hyperbolic sine of $x$</td>
</tr>
<tr>
<td><code>tan(x)</code></td>
<td>the tangent of $x$, where $x$ is in radians</td>
</tr>
<tr>
<td><code>tanh(x)</code></td>
<td>the hyperbolic tangent of $x$</td>
</tr>
</tbody>
</table>

---

Also see

[FN] Functions by name  
[D] egen — Extensions to generate  
[D] generate — Create or change contents of variable  
[M-4] Intro — Categorical guide to Mata functions  
[U] 13.3 Functions