Date and time functions

- bofd("cal", ed) the ed business date corresponding to ed
- Cdhms(ed, h, m, s) the etC datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to ed, h, m, s
- Chms(h, m, s) the etC datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
- Clock(s1, s2[ , Y]) the etC datetime (ms. since 01jan1960 00:00:00.000) corresponding to s1 based on s2 and Y
- clock(s1, s2[ , Y]) the etc datetime (ms. since 01jan1960 00:00:00.000) corresponding to ed (ms. without leap seconds since 01jan1960 00:00:00.000)
- Cmdyhms(M, D, Y, h, m, s) the etC datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
- Cofc(etC) 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) 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) the etC datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date ed at time 00:00:00.000
- cofd(ed) the etC datetime (ms. since 01jan1960 00:00:00.000) of date ed at time 00:00:00.000
- daily(s1, s2[ , Y]) a synonym for date(s1, s2[ , Y])
- date(s1, s2[ , Y]) the ed date (days since 01jan1960) corresponding to s1 based on s2 and Y
- day(ed) the numeric day of the month corresponding to ed
- dhms(ed, h, m, s) the etc datetime (ms. since 01jan1960 00:00:00.000) corresponding to ed, h, m, and s
- dofb(eb, "cal") the ed date (days since 01jan1960) of date etC (ms. with leap seconds since 01jan1960 00:00:00.000)
- dofC(etC) the ed date (days since 01jan1960) of date etC (ms. with leap seconds since 01jan1960 00:00:00.000)
2 Functions by category

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 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 $e_d$

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()

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)  
the $e_d$ date (days since 01jan1960) corresponding to $M$, $D$, $Y$

mdy(M,D,Y)  
the $e_d$ date (days since 01jan1960) corresponding to $M$, $D$, $Y$, $h$, $m$, $s$

mofd(e_d)  
the numeric month corresponding to date $e_d$

month(e_d)  
the numeric month of the year 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()

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

mm(e_{tc})  
the minute corresponding to datetime $e_{tC}$ (ms. with leap seconds 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)

mofd(e_d)  
the numeric month corresponding to date $e_d$

months(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()

ms/3,600,000

ms/60,000

ms/1,000

secondsofhours(h)

secondsofminutes(m)

secondsofseconds(s)

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()

ms/1,000

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)
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 toyear; 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 toyear; 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 lnΓ(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
4 Functions by category

- `invClogLog(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(x_1,x_2,...,x_n)` the maximum value of x_1, x_2, ..., x_n
- `min(x_1,x_2,...,x_n)` the minimum value of x_1, x_2, ..., x_n
- `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) = d^2 lnΓ(x)/dx^2
- `trunc(x)` a synonym for int(x)

Matrix functions

- `cholesky(M)` the Cholesky decomposition of the matrix: if R = cholesky(S), then RR^T = S
- `coleqnumb(M,s)` the equation number of M associated with column equation s; missing if the column equation cannot be found
- `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
- `corr(M)` the correlation matrix of the variance matrix
- `det(M)` the determinant of matrix M
**Functions by category**

### Mathematical functions

- **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 \( \max_{i,j} \left\{ \frac{|x_{ij} - y_{ij}|}{(|y_{ij}| + 1)} \right\} \)
- **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
- **rownfreparms(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,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 \( x \)
- **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

epsdouble() the machine precision of a double-precision number

epsfloat() the machine precision of a floating-point number

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

fileread(\(f\)) the contents of the file specified by \(f\)

filereadererror(s) \(0\) or positive integer, said value having the interpretation of a return code

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

float(\(x\)) the value of \(x\) rounded to float precision

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

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

inlist(\(z, a, b, \ldots\)) \(1\) 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, \(0\)

irecode(\(x, x_1, \ldots, x_n\)) \(missing\) 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 \(x > x_n\)

matrix(exp) restricts name interpretation to scalars and matrices; see scalar()

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

mi(\(x_1, x_2, \ldots, x_n\)) a synonym for missing(\(x_1, x_2, \ldots, x_n\))

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

missing(\(x_1, x_2, \ldots, x_n\)) the value of the stored result \(r(name)\); see [U] 18.8 Accessing results calculated by other programs

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

recode(\(x, x_1, \ldots, x_n\)) 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 \(x < x_2\); \ldots; otherwise, \(x_n\) if \(x > x_1, x_2, \ldots, x_{n-1}\). \(x_i\) is interpreted as \(x_i = +\infty\)

replay() \(1\) if the first nonblank character of local macro ‘0’ is a comma, or if ‘0’ is empty

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

s(name)
Functions by category

**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
- **rchi2(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 $\pi/\sqrt{3}$
- **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
- **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
Selecting time-span functions

\[
\begin{align*}
tin(d_1, d_2) & \quad \text{true if } d_1 \leq t \leq d_2, \text{ where } t \text{ is the time variable previously } \texttt{tsset} \\
twithin(d_1, d_2) & \quad \text{true if } d_1 < t < d_2, \text{ where } t \text{ is the time variable previously } \texttt{tsset}
\end{align*}
\]

Statistical functions

\[
\begin{align*}
\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 } \lfloor k \rfloor \text{ or fewer successes in } \lfloor n \rfloor \text{ trials when the probability of a success on one trial is } \theta; \ 0 \text{ if } k < 0; \text{ or } 1 \text{ if } k > n \\
\text{binomialp}(n, k, p) & \quad \text{the probability of observing } \lfloor k \rfloor \text{ successes in } \lfloor n \rfloor \text{ trials when the probability of a success on one trial is } p \\
\text{binomialtail}(n, k, \theta) & \quad \text{the probability of observing } \lfloor k \rfloor \text{ or more successes in } \lfloor n \rfloor \text{ trials when the probability of a success on one trial is } \theta; \ 1 \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); \ 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); \ 0 \text{ if } x < 0 \\
\text{dgammapdax}(a, x) & \quad \frac{\partial^2 P(a, x)}{\partial a \partial x}, \text{ where } P(a, x) = \text{gammap}(a, x); \ 0 \text{ if } x < 0 \\
\text{dgammapdx}(a, x) & \quad \frac{\partial P(a, x)}{\partial x}, \text{ where } P(a, x) = \text{gammap}(a, x); \ 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); \ 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
F(df1,df2,f)

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

Fden(df1,df2,f)

the probability density function of the $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom; $0$ if $f < 0$

Ftail(df1,df2,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 $x < 0$

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

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 $x \leq 0$

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 $p$

invbinomialtail(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 $p$

invcauchy(a,b,p)

the inverse of cauchy(): if cauchy($a, b, x$) = $p$, then

$\text{invcauchy}(a, b, p) = x$

invcauchytail(a,b,p)

the inverse of cauchytail(): if cauchytail($a, b, x$) = $p$, then

$\text{invcauchytail}(a, b, p) = x$

invchi2(df,p)

the inverse of chi2(): if chi2($df, x$) = $p$, then

$\text{invchi2}(df, p) = x$

invchi2tail(df,p)

the inverse of chi2tail(): if chi2tail($df, x$) = $p$, then

$\text{invchi2tail}(df, p) = x$

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 $\text{exponential}(b, x) = p$, 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\((df_1,df_2,p)\) the inverse cumulative \(F\) distribution: if \(F(df_1,df_2,f) = p\), then \(\text{invF}(df_1,df_2,p) = f\)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

invnchi2\text{tail}\((df,np,p)\) the inverse reverse cumulative (upper tail or survivor) noncentral \(\chi^2\) distribution: if \(\operatorname{nchi2\text{tail}}(df,np,x) = p\), then \(\text{invnchi2\text{tail}}(df,np,p) = x\)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `invnF(df_1, df_2, np, p)` | the inverse cumulative noncentral F distribution: if $\text{nF}(df_1, df_2, np, f) = p$, 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 $\text{nFtail}(df_1, df_2, np, f) = p$, then `invnFtail(df_1, df_2, np, p) = f`
| `invnibeta(a, b, np, p)` | the inverse cumulative noncentral beta distribution: if $\text{ibeta}(a, b, np, x) = p$, 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 $\text{nt}(df, np, t) = p$, then `invnt(df, np, p) = t`
| `invnttail(df, np, p)` | the inverse reverse cumulative (upper tail or survivor) noncentral Student's t distribution: if $\text{nttail}(df, np, t) = p$, 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 cumulative Poisson distribution evaluated at $k$ is $q$: if poissontail($m, k$) = $q$, then `invpoissontail(k, q) = m`
| `invt(df, p)` | the inverse cumulative Student's t distribution: if $\text{t}(df, t) = p$, then `invt(df, p) = t`
| `invttail(df, p)` | the inverse reverse cumulative (upper tail or survivor) Student's t distribution: if $\text{ttail}(df, t) = p$, 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
| `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`
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 \(b\)
lncauchyden(\(a, b, x\)) the natural logarithm of the density of the Cauchy distribution with location parameter \(a\) and scale parameter \(b\)
lnigammaden(\(a, b, x\)) the natural logarithm of the inverse gamma density, where \(a\) is the shape parameter and \(b\) is the scale parameter
lnigaussian(\(m, a, x\)) 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(\(x, \sigma\)) the natural logarithm of the normal density with mean 0 and standard deviation \(\sigma\)
lnnormalden(\(x, \mu, \sigma\)) the natural logarithm of the normal density with mean \(\mu\) and standard deviation \(\sigma\), \(N(\mu, \sigma^2)\)
lnwishartden(\(df, V, X\)) the natural logarithm of the density of the Wishart distribution; missing if \(df \leq n - 1\)
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 \(s\pi/\sqrt{3}\)
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 χ² distribution; 0 if x < 0
nchi2den(df,np,x)  the probability density of the noncentral χ² distribution; 0 if x < 0
nchi2tail(df,np,x)  the reverse cumulative (upper tail or survivor) noncentral χ² distribution; 1 if x < 0
nF(df1,df2,np,f)  the cumulative noncentral F distribution with df₁ numerator and df₂ denominator degrees of freedom and noncentrality parameter np; 0 if f < 0
nFden(df1,df2,np,f)  the probability density function of the noncentral F distribution with df₁ numerator and df₂ denominator degrees of freedom and noncentrality parameter np; 0 if f < 0
nFtail(df1,df2,np,f)  the reverse cumulative (upper tail or survivor) noncentral F distribution with df₁ numerator and df₂ 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(μ, σ²)
npnchi2(dx,p)  the noncentrality parameter, np, for noncentral χ²:
      nchi2(dx,p) = p, then npnchi2(dx,p) = np
npnF(df1,df2,f,p)  the noncentrality parameter, np, for the noncentral F:
      nF(df1,df2,p) = p, then npnF(df1,df2,f,p) = np
npnt(df,t,p)  the noncentrality parameter, np, for the noncentral Student’s t distribution: if npnt(df,t,p) = 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
ntail(df,np,t)  the reverse cumulative (upper tail or survivor) noncentral Student’s t distribution with df degrees of freedom and noncentrality parameter np
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
t(df,t)  the cumulative Student’s t distribution with df degrees of freedom
tden(df,t)  the probability density function of Student’s t distribution
tail(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
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 ≠ ±1
plural(n, s1, s2) the plural of s1, as modified by or replaced with s2, if n ≠ ±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 ≤ n < 10
soundex(s) the soundex code for a string, s
soundex_nara(s) the U.S. Census soundex code for a string, s
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcat($s_1, s_2$)</td>
<td>there is no strcat() function; instead the addition operator is used to concatenate strings</td>
</tr>
<tr>
<td>strdup($s_1, n$)</td>
<td>there is no strdup() function; instead the multiplication operator is used to create multiple copies of strings</td>
</tr>
<tr>
<td>string($n$)</td>
<td>a synonym for strofreal($n$)</td>
</tr>
<tr>
<td>string($n, s$)</td>
<td>a synonym for strofreal($n, s$)</td>
</tr>
<tr>
<td>stritrim($s$)</td>
<td>$s$ with multiple, consecutive internal blanks (ASCII space character char(32)) collapsed to one blank</td>
</tr>
<tr>
<td>strlen($s$)</td>
<td>the number of characters in ASCII $s$ or length in bytes</td>
</tr>
<tr>
<td>strlower($s$)</td>
<td>lowercase ASCII characters in string $s$</td>
</tr>
<tr>
<td>strltrim($s$)</td>
<td>$s$ without leading blanks (ASCII space character char(32))</td>
</tr>
<tr>
<td>strmatch($s_1, s_2$)</td>
<td>1 if $s_1$ matches the pattern $s_2$; otherwise, 0</td>
</tr>
<tr>
<td>strofreal($n$)</td>
<td>$n$ converted to a string</td>
</tr>
<tr>
<td>strofreal($n, s$)</td>
<td>$n$ converted to a string using the specified display format</td>
</tr>
<tr>
<td>strpos($s_1, s_2$)</td>
<td>the position in $s_1$ at which $s_2$ is first found; otherwise, 0</td>
</tr>
<tr>
<td>strproper($s$)</td>
<td>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</td>
</tr>
<tr>
<td>strrreverse($s$)</td>
<td>reverses the ASCII string $s$</td>
</tr>
<tr>
<td>strrpos($s_1, s_2$)</td>
<td>the position in $s_1$ at which $s_2$ is last found; otherwise, 0</td>
</tr>
<tr>
<td>strrtrim($s$)</td>
<td>$s$ without trailing blanks (ASCII space character char(32))</td>
</tr>
<tr>
<td>strtoname($s[p]$)</td>
<td>$s$ translated into a Stata 13 compatible name</td>
</tr>
<tr>
<td>strtrim($s$)</td>
<td>$s$ without leading and trailing blanks (ASCII space character char(32)); equivalent to strltrim(strrtrim($s$))</td>
</tr>
<tr>
<td>strupper($s$)</td>
<td>uppercase ASCII characters in string $s$</td>
</tr>
<tr>
<td>subinstr($s_1, s_2, s_3, n$)</td>
<td>$s_1$, where the first $n$ occurrences in $s_1$ of $s_2$ have been replaced with $s_3</td>
</tr>
<tr>
<td>subinword($s_1, s_2, s_3, n$)</td>
<td>$s_1$, where the first $n$ occurrences in $s_1$ of $s_2$ as a word have been replaced with $s_3</td>
</tr>
<tr>
<td>substr($s, n_1, n_2$)</td>
<td>the substring of $s$, starting at $n_1$, for a length of $n_2</td>
</tr>
<tr>
<td>tobytes($s[n]$)</td>
<td>escaped decimal or hex digit strings of up to 200 bytes of $s$</td>
</tr>
<tr>
<td>uchar($n$)</td>
<td>the Unicode character corresponding to Unicode code point $n$ or an empty string if $n$ is beyond the Unicode code-point range</td>
</tr>
<tr>
<td>ustrlen($s$)</td>
<td>the number of display columns needed to display the Unicode string $s$ in the Stata Results window</td>
</tr>
<tr>
<td>usubstr($s, n_1, n_2$)</td>
<td>the Unicode substring of $s$, starting at character $n_1$, for $n_2$ display columns</td>
</tr>
<tr>
<td>uisdigit($s$)</td>
<td>1 if the first Unicode character in $s$ is a Unicode decimal digit; otherwise, 0</td>
</tr>
<tr>
<td>uisletter($s$)</td>
<td>1 if the first Unicode character in $s$ is a Unicode letter; otherwise, 0</td>
</tr>
<tr>
<td>ustrcompare($s_1, s_2[loc]$)</td>
<td>compares two Unicode strings</td>
</tr>
<tr>
<td>ustrcompareex($s_1, s_2, loc, st, case, cslv, norm, num, alt, fr$)</td>
<td>compares two Unicode strings</td>
</tr>
<tr>
<td>ustrfix($s[rep]$)</td>
<td>replaces each invalid UTF-8 sequence with a Unicode character</td>
</tr>
</tbody>
</table>
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.

ustrregrexra($s_1$, $re$, $s_2$, $noc$) 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($s$) reverses the 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 $n$th 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 $s_1$.

ustrstr($s$, $n_1$, $n_2$) the Unicode substring of $s$, starting at $n_1$, for a length of $n_2$. 
word(s,n) the n\text{th} word in s; \textit{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

\texttt{acos(x)} the radian value of the arccosine of \( x \)
\texttt{acosh(x)} the inverse hyperbolic cosine of \( x \)
\texttt{asin(x)} the radian value of the arcsine of \( x \)
\texttt{asinh(x)} the inverse hyperbolic sine of \( x \)
\texttt{atan(x)} the radian value of the arctangent of \( x \)
\texttt{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
\texttt{atanh(x)} the inverse hyperbolic tangent of \( x \)
\texttt{cos(x)} the cosine of \( x \), where \( x \) is in radians
\texttt{cosh(x)} the hyperbolic cosine of \( x \)
\texttt{sin(x)} the sine of \( x \), where \( x \) is in radians
\texttt{sinh(x)} the hyperbolic sine of \( x \)
\texttt{tan(x)} the tangent of \( x \), where \( x \) is in radians
\texttt{tanh(x)} the hyperbolic tangent of \( x \)

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