# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro</td>
<td>1</td>
</tr>
<tr>
<td>Introduction to functions reference manual</td>
<td></td>
</tr>
<tr>
<td>Functions by category</td>
<td>2</td>
</tr>
<tr>
<td>Functions by name</td>
<td>21</td>
</tr>
<tr>
<td>Date and time functions</td>
<td>39</td>
</tr>
<tr>
<td>Mathematical functions</td>
<td>66</td>
</tr>
<tr>
<td>Matrix functions</td>
<td>73</td>
</tr>
<tr>
<td>Programming functions</td>
<td>80</td>
</tr>
<tr>
<td>Random-number functions</td>
<td>94</td>
</tr>
<tr>
<td>Selecting time-span functions</td>
<td>105</td>
</tr>
<tr>
<td>Statistical functions</td>
<td>107</td>
</tr>
<tr>
<td>String functions</td>
<td>149</td>
</tr>
<tr>
<td>Trigonometric functions</td>
<td>181</td>
</tr>
<tr>
<td>Subject and author index</td>
<td>184</td>
</tr>
</tbody>
</table>
Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, [U] 27 Overview of Stata estimation commands; [R] regress; and [D] reshape. The first example is a reference to chapter 27, Overview of Stata estimation commands, in the User’s Guide; the second is a reference to the regress entry in the Base Reference Manual; and the third is a reference to the reshape entry in the Data Management Reference Manual.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM] Getting Started with Stata for Mac
[GSU] Getting Started with Stata for Unix
[GSW] Getting Started with Stata for Windows
[U] Stata User’s Guide
[R] Stata Base Reference Manual
[BAYES] Stata Bayesian Analysis Reference Manual
[BMA] Stata Bayesian Model Averaging Reference Manual
[FN] Stata Functions Reference Manual
[XT] Stata Longitudinal-Data/Panel-Data Reference Manual
[MI] Stata Multiple-Imputation Reference Manual
[SVY] Stata Survey Data Reference Manual
[TABLES] Stata Customizable Tables and Collected Results Reference Manual
[I] Stata Index

This manual describes the functions allowed by Stata. For information on Mata functions, see [M-4] Intro.

A quick note about missing values: Stata denotes a numeric missing value by ., .a, .b, ..., or .z. A string missing value is denoted by "" (the empty string). Here any one of these may be referred to by missing. If a numeric value $x$ is missing, then $x \geq . \,$ is true. If a numeric value $x$ is not missing, then $x < . \,$ is true.

See [U] 12.2.1 Missing values for details.

Reference


Also see

[U] 1.3 What’s new
**Date and time functions**

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

Cofc(e_{tc}) the \( e_{tc} \) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of \( e_{tc} \) (ms. without leap seconds since 01jan1960 00:00:00.000)

cofC(e_{tc}) the \( e_{tc} \) datetime (ms. without leap seconds since 01jan1960 00:00:00.000) of \( e_{tc} \) (ms. with leap seconds since 01jan1960 00:00:00.000)

Cofd(e_d) the \( e_{tc} \) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date \( e_d \) at time 00:00:00.000

cofd(e_d) the \( e_{tc} \) datetime (ms. since 01jan1960 00:00:00.000) of date \( e_d \) at time 00:00:00.000

daily(s_1,s_2[,Y]) a synonym for date(s_1,s_2[,Y])

date(s_1,s_2[,Y]) the \( e_d \) date (days since 01jan1960) corresponding to \( s_1 \) based on \( s_2 \) and \( Y \)

datediff(e_{d1},e_{d2},s_u[,s_{nl}]) the difference, rounded down to an integer, from \( e_{d1} \) to \( e_{d2} \) in \( s_u \) units of days, months, or years with \( s_{nl} \) the nonleap-year anniversary for \( e_{d1} \) on 29feb

datediff_frac(e_{d1},e_{d2},s_u[,s_{nl}]) the difference, including the fractional part, from \( e_{d1} \) to \( e_{d2} \) in \( s_u \) units of days, months, or years with \( s_{nl} \) the nonleap-year anniversary for \( e_{d1} \) on 29feb

datepart(e_d,s_u) the integer year, month, or day of \( e_d \) with \( s_u \) specifying year, month, or day

day(e_d) the numeric day of the month corresponding to \( e_d \)

daysinmonth(e_d) the number of days in the month of \( e_d \)

dayssince(d_e,d) a synonym for dayssinceweekday(e_d,d)

dayssinceweekday(e_d,d) the number of days until \( e_d \) since previous day-of-week \( d \)

daysuntil(d_e,d) a synonym for daysuntilweekday(e_d,d)

daysuntilweekday(e_d,d) the number of days from \( e_d \) until next day-of-week \( d \)

dhms(e_d,h,m,s) the \( e_{tc} \) datetime (ms. since 01jan1960 00:00:00.000) corresponding to \( e_d, h, m, \) and \( s \)

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

dofb(e_b,"cal") the \( e_d \) date (days since 01jan1960) corresponding to \( e_b \)

dofC(e_{tc}) the \( e_d \) date (days since 01jan1960) of datetime \( e_{tc} \) (ms. with leap seconds since 01jan1960 00:00:00.000)

dofc(e_{tc}) the \( e_d \) date (days since 01jan1960) of datetime \( e_{tc} \) (ms. since 01jan1960 00:00:00.000)

dofh(e_h) the \( e_d \) date (days since 01jan1960) of the start of half-year \( e_h \)

dofm(e_m) the \( e_d \) date (days since 01jan1960) of the start of month \( e_m \)

dofq(e_q) the \( e_d \) date (days since 01jan1960) of the start of quarter \( e_q \)

dofw(e_w) the \( e_d \) date (days since 01jan1960) of the start of week \( e_w \)

dofy(e_y) the \( e_d \) date (days since 01jan1960) of 01jan in year \( e_y \)

dow(e_d) the numeric day of the week corresponding to date \( e_d \); 0 = Sunday, 1 = Monday, . . . , 6 = Saturday
doy($e_d$)  
the numeric day of the year corresponding to date $e_d$

firstdayofmonth($e_d$)  
the $e_d$ date of the first day of the month of $e_d$

firstdowofmonth($M,Y,d$)  
a synonym for firstweekdayofmonth($M,Y,d$)

firstweekdayofmonth($M,Y,d$)  
the $e_d$ date of the first day-of-week $d$ in month $M$ of year $Y$

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($ms$)  
$ms/3,600,000$

isleapsecond($e_{tC}$)  
1 if $e_{tC}$ is a leap second; otherwise, 0

isleapyear($Y$)  
1 if $Y$ is a leap year; otherwise, 0

lastdowofmonth($M,Y,d$)  
a synonym for lastweekdayofmonth($M,Y,d$)

lastweekdayofmonth($M,Y,d$)  
the $e_d$ date of the last day-of-week $d$ in month $M$ of year $Y$

lastdowofmonth($M,Y,d$)  
the $e_d$ date of the last day of the month of $e_d$

lastweekdayofmonth($M,Y,d$)  
the $e_d$ date of the last day-of-week $d$ in month $M$ of year $Y$

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

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

minutes($ms$)  
$ms/60,000$

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

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

mofd($e_d$)  
the $e_m$ monthly date (months since 1960m1) containing date $e_d$

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

monthly($s_1,s_2,Y$)  
the $e_m$ monthly date (months since 1960m1) corresponding to $s_1$ based on $s_2$ and $Y$; $Y$ specifies topyear; see date()

msofhours($h$)  
$h \times 3,600,000$

msofminutes($m$)  
$m \times 60,000$

msofseconds($s$)  
$s \times 1,000$

nextbirthday($e_d_{DOB},e_d[snl]$)  
the $e_d$ date of the first birthday after $e_d$ for date of birth $e_d_{DOB}$ with $snl$ the nonleap-year birthday for 29feb birthdates

nextdow($e_d,d$)  
a synonym for nextweekday($e_d,d$)

nextleapyear($Y$)  
the first leap year after year $Y$

nextweekday($e_d,d$)  
the $e_d$ date of the first day-of-week $d$ after $e_d$

now()  
the current $e_{tc}$ datetime
Functions by category

previousbirthday\( (e_d \text{DOB}, e_d[snl]) \)
the \(e_d\) date of the birthday immediately before \(e_d\) for date of birth \(e_d \text{DOB}\) with \(snl\) the nonleap-year birthday for 29feb birthdates

previousdow\( (e_d, d) \)
a synonym for previousweekday\( (e_d, d) \)

previousleapyear\( (Y) \)
the leap year immediately before \(Y\)

previousweekday\( (e_d, d) \)
the \(e_d\) date of the last day-of-week \(d\) before \(e_d\)

qofd\( (e_d) \)
the \(e_q\) quarterly date (quarters since 1960q1) containing date \(e_d\)

quarter\( (e_d) \)
the numeric quarter of the year corresponding to \(e_d\)

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()

seconds\( (ms) \)
\(ms/1,000\)

ss\( (etc) \)
the second corresponding to datetime \(etc\) (ms. since 01jan1960 00:00:00.000)

ssC\( (etcC) \)
the second corresponding to datetime \(etcC\) (ms. with leap seconds since 01jan1960 00:00:00.000)

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

today\( () \)
today’s \(e_d\) date

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 \( n - 1 < x \leq n \); \( x \) (not “.”) if \( x \) is missing, meaning that \( \text{ceil}(x) = x \)
cloglog(x)  the complementary log-log of \( x \)
comb(n,k)  the combinatorial function \( \frac{n!}{k!(n-k)!} \)
digamma(x)  the digamma() function, \( d \ln \Gamma(x)/dx \)
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 \( n \leq x < n + 1 \); \( x \) (not “.”) if \( x \) is missing, meaning that \( \text{floor}(x) = x \)
int(x)  the integer obtained by truncating \( x \) toward 0 (thus, \( \text{int}(5.2) = 5 \) and \( \text{int}(-5.8) = -5 \)); \( x \) (not “.”) if \( x \) is missing, meaning that \( \text{int}(x) = x \)
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)  \( \ln \{ \Gamma(x) \} \)
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 \( x_1, x_2, \ldots, x_n \)
min(x_1,x_2,...,x_n)  the minimum value of \( x_1, x_2, \ldots, x_n \)
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; \( \text{missing} \) if only one argument is missing or if the two arguments are two different types of \( \text{missing} \)
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 \( \text{round}(x) = x \) and that \( \text{round}(x,y) = y \) 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 \( \text{missing} \) 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 \(\text{lngamma}(x) = d^2 \ln \Gamma(x)/dx^2\)

trunc(x)  
a synonym for \(\text{int}(x)\)

**Matrix functions**

cholesky(M)  
the Cholesky decomposition of the matrix: if \(R = \text{cholesky}(S)\), then \(RR^T = S\)

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

colnfreeparms(M)  
the number of free parameters in columns of \(M\)

colnnumb(M,s)  
the column number of \(M\) associated with column name \(s\); \(\text{missing}\) 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 \(M\)

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[floor(i),floor(j)], the \(i,j\) element of the matrix named \(s\); \(\text{missing}\) 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

invvech(M)  
a symmetric matrix formed by filling in the columns of the lower triangle from a row or column vector

invvecp(M)  
a symmetric matrix formed by filling in the columns of the upper triangle from a row or column vector

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 \(\max_{i,j}\left\{|x_{ij} - y_{ij}|/(|y_{ij}| + 1)\right\}\)

nullmat(matname)  
use with the row-join (,) and column-join (\) operators

roweqnumb(M,s)  
the equation number of \(M\) associated with row equation \(s\); \(\text{missing}\) if the row equation cannot be found

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

rownnumb(M,s)  
the row number of \(M\) associated with row name \(s\); \(\text{missing}\) if the row cannot be found

rowsof(M)  
the number of rows of \(M\)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sweep($M, i$)</td>
<td>matrix $M$ with $i$th row/column swept</td>
</tr>
<tr>
<td>trace($M$)</td>
<td>the trace of matrix $M$</td>
</tr>
<tr>
<td>vec($M$)</td>
<td>a column vector formed by listing the elements of $M$, starting with the</td>
</tr>
<tr>
<td></td>
<td>first column and proceeding column by column</td>
</tr>
<tr>
<td>vecdiag($M$)</td>
<td>the row vector containing the diagonal of matrix $M$</td>
</tr>
<tr>
<td>vech($M$)</td>
<td>a column vector formed by listing the lower triangle elements of $M$</td>
</tr>
<tr>
<td>vecp($M$)</td>
<td>a column vector formed by listing the upper triangle elements of $M$</td>
</tr>
</tbody>
</table>

**Programming functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autocode($x, n, x_0, x_1$)</td>
<td>partitions the interval from $x_0$ to $x_1$ into $n$ equal-length intervals and returns the upper bound of the interval that contains $x$ or the upper bound of the first or last interval if $x &lt; x_0$ or $x &gt; x_1$, respectively</td>
</tr>
<tr>
<td>byteorder()</td>
<td>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</td>
</tr>
<tr>
<td>c(name)</td>
<td>the value of the system or constant result $c(name)$ (see [P] creturn)</td>
</tr>
<tr>
<td>_caller()</td>
<td>version of the program or session that invoked the currently running program; see [P] version</td>
</tr>
<tr>
<td>chop($x, \epsilon$)</td>
<td>round($x$) if $\text{abs}(x - \text{round}(x)) &lt; \epsilon$; otherwise, $x$; or $x$ if $x$ is missing</td>
</tr>
<tr>
<td>clip($x, a, b$)</td>
<td>$x$ if $a &lt; x &lt; b$, $b$ if $x \geq b$, $a$ if $x \leq a$, or missing if $x$ is missing or if $a &gt; b$; $x$ if $x$ is missing</td>
</tr>
<tr>
<td>cond($x, a, b[,c]$)</td>
<td>$a$ 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</td>
</tr>
<tr>
<td>e(name)</td>
<td>the value of stored result $e(name)$; see [U] 18.8 Accessing results calculated by other programs</td>
</tr>
<tr>
<td>e(sample)</td>
<td>1 if the observation is in the estimation sample and 0 otherwise</td>
</tr>
<tr>
<td>epsdouble()</td>
<td>the machine precision of a double-precision number</td>
</tr>
<tr>
<td>epsfloat()</td>
<td>the machine precision of a floating-point number</td>
</tr>
<tr>
<td>fileexists($f$)</td>
<td>1 if the file specified by $f$ exists; otherwise, 0</td>
</tr>
<tr>
<td>fileread($f$)</td>
<td>the contents of the file specified by $f$</td>
</tr>
<tr>
<td>filereaderror($s$)</td>
<td>0 or positive integer, said value having the interpretation of a return code</td>
</tr>
<tr>
<td>fwrite($f, s[,r]$)</td>
<td>writes the string specified by $s$ to the file specified by $f$ and returns the number of bytes in the resulting file</td>
</tr>
<tr>
<td>float($x$)</td>
<td>the value of $x$ rounded to float precision</td>
</tr>
<tr>
<td>fmtwidth($fmtstr$)</td>
<td>the output length of the %fmt contained in $fmtstr$; missing if $fmtstr$ does not contain a valid %fmt</td>
</tr>
<tr>
<td>frval()</td>
<td>returns values of variables stored in other frames</td>
</tr>
<tr>
<td>_frval()</td>
<td>programmer’s version of frval()</td>
</tr>
<tr>
<td>has_eprop(name)</td>
<td>1 if name appears as a word in e(properties); otherwise, 0</td>
</tr>
</tbody>
</table>
inlist(z,a,b,...) 1 if z is a member of the remaining arguments; otherwise, 0
inrange(z,a,b) 1 if it is known that a ≤ z ≤ b; otherwise, 0
irecode(x,x₁,...,xₙ) missing if x is missing or x₁,...,xₙ is not weakly increasing; 0 if x ≤ x₁; 1 if x₁ < x ≤ x₂; 2 if x₂ < x ≤ x₃; ...; n if x > xₙ

displays

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(x₁,x₂,...,xₙ) a synonym for missing(x₁,x₂,...,xₙ)
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(x₁,x₂,...,xₙ) 1 if any xᵢ evaluates to missing; otherwise, 0

r(name) the value of the stored result r(name); see [U] 18.8 Accessing results calculated by other programs
recode(x,x₁,...,xₙ) missing if x₁,...,xₙ is not weakly increasing; x if x is missing; x₁ if x ≤ x₁; x₂ if x ≤ x₂, ...; otherwise, xₙ if x > x₁, x₂, ... xₙ₋₁. xᵢ ≥ . is interpreted as xᵢ = +∞
replay() 1 if the first nonblank character of local macro ‘0’ is a comma, or if ‘0’ is empty
return(name) the value of the to-be-stored result r(name); see [P] return
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
rchisquare(df) χ², 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
<table>
<thead>
<tr>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rhypergeometric(N,K,n)</code></td>
</tr>
<tr>
<td><code>rigaussian(m,a)</code></td>
</tr>
<tr>
<td><code>rlaplace(m,b)</code></td>
</tr>
<tr>
<td><code>rlogistic()</code></td>
</tr>
<tr>
<td><code>rlogistic(s)</code></td>
</tr>
<tr>
<td><code>rlogistic(m,s)</code></td>
</tr>
<tr>
<td><code>rnbinomial(n,p)</code></td>
</tr>
<tr>
<td><code>rnormal()</code></td>
</tr>
<tr>
<td><code>rnormal(m)</code></td>
</tr>
<tr>
<td><code>rnormal(m,s)</code></td>
</tr>
<tr>
<td><code>rpoisson(m)</code></td>
</tr>
<tr>
<td><code>rt(df)</code></td>
</tr>
<tr>
<td><code>runiform()</code></td>
</tr>
<tr>
<td><code>runiform(a,b)</code></td>
</tr>
<tr>
<td><code>runiformint(a,b)</code></td>
</tr>
<tr>
<td><code>rweibull(a,b)</code></td>
</tr>
<tr>
<td><code>rweibull(a,b,g)</code></td>
</tr>
<tr>
<td><code>rweibullph(a,b)</code></td>
</tr>
<tr>
<td><code>rweibullph(a,b,g)</code></td>
</tr>
</tbody>
</table>

**Selecting time-span functions**

- `tin(d_1,d_2)` true if \(d_1 \leq t \leq d_2\), where \(t\) is the time variable previously `tsset`
- `twithin(d_1,d_2)` true if \(d_1 < t < d_2\), where \(t\) is the time variable previously `tsset`

**Statistical functions**

- `betaden(a,b,x)` the probability density of the beta distribution, where \(a\) and \(b\) are the shape parameters; 0 if \(x < 0\) or \(x > 1\)
- `binomial(n,k,\theta)` the probability of observing \(\text{floor}(k)\) or fewer successes in \(\text{floor}(n)\) trials when the probability of a success on one trial is \(\theta\); 0 if \(k < 0\); or 1 if \(k > n\)
- `binomialp(n,k,p)` the probability of observing \(\text{floor}(k)\) successes in \(\text{floor}(n)\) trials when the probability of a success on one trial is \(p\)
- `binomialtail(n,k,\theta)` the probability of observing \(\text{floor}(k)\) or more successes in \(\text{floor}(n)\) trials when the probability of a success on one trial is \(\theta\); 1 if \(k < 0\); or 0 if \(k > n\)
**Functions by category**

binormal($h,k,\rho$) the joint cumulative distribution $\Phi(h,k,\rho)$ of bivariate normal with correlation $\rho$

cauychy($a,b,x$) the cumulative Cauchy distribution with location parameter $a$ and scale parameter $b$

cauychyden($a,b,x$) the probability density of the Cauchy distribution with location parameter $a$ and scale parameter $b$

cauychytail($a,b,x$) the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter $a$ and scale parameter $b$

chi2($df,x$) the cumulative $\chi^2$ distribution with $df$ degrees of freedom; 0 if $x < 0$

chi2den($df,x$) the probability density of the $\chi^2$ distribution with $df$ degrees of freedom; 0 if $x < 0$

chi2tail($df,x$) the reverse cumulative (upper tail or survivor) $\chi^2$ distribution with $df$ degrees of freedom; 1 if $x < 0$

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

dgammapdana($a,x$) $\frac{\partial^2 P(a,x)}{\partial a^2}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$

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

dgammapd($a,x$) $\frac{\partial P(a,x)}{\partial x}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$

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

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

exponential($b,x$) the cumulative exponential distribution with scale $b$

exponentialden($b,x$) the probability density function of the exponential distribution with scale $b$

exponentialtail($b,x$) the reverse cumulative exponential distribution with scale $b$

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

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 $x > 1$
12 Functions by category

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

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

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

\[ \text{igaussiantail}(m,a) \] 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, \( \theta \) (\( \theta \) = probability of success on one trial) such that the probability of observing \( \text{floor}(k) \) or fewer successes in \( \text{floor}(n) \) trials is \( p \)

\[ \text{invbinomialtail}(n,k,p) \] the inverse of the right cumulative binomial; that is, \( \theta \) (\( \theta \) = 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 the cumulative Cauchy distribution: if \( \text{cauchy}(a,b,x) = p \), then \( \text{invcauchy}(a,b,p) = x \)

\[ \text{invcauchytail}(a,b,p) \] the inverse of the reverse cumulative Cauchy distribution: if \( \text{cauchytail}(a,b,x) = p \), then \( \text{invcauchytail}(a,b,p) = x \)

\[ \text{invchi2}(df,p) \] the inverse of \( \text{chi2}() \): if \( \text{chi2}(df,x) = p \), then \( \text{invchi2}(df,p) = x \)

\[ \text{invchi2tail}(df,p) \] the inverse of \( \text{chi2tail}() \): if \( \text{chi2tail}(df,x) = p \), 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

\[ \text{inveponential}(b,p) \] the inverse cumulative exponential distribution with scale \( b \): if \( \text{exponential}(b,x) = p \), then \( \text{inveponential}(b,p) = x \)

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

\[ \text{invF}(df1,df2,p) \] the inverse cumulative \( F \) distribution: if \( F(df1,df2,x) = p \), then \( \text{invF}(df1,df2,p) = x \)

\[ \text{invFtail}(df1,df2,p) \] the inverse reverse cumulative (upper tail or survivor) \( F \) distribution: if \( \text{Ftail}(df1,df2,x) = p \), then \( \text{invFtail}(df1,df2,p) = x \)

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

\[ \text{invgammatail}(a,p) \] the inverse reverse cumulative (upper tail or survivor) gamma distribution: if \( \text{gammatail}(a,x) = p \), then \( \text{invgammatail}(a,p) = x \)

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

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

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

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

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

invlogistic(p)  the inverse cumulative logistic distribution: if logistic(x) = p,
then invlogistic(p) = x

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

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

invlogisticntail(p)  the inverse reverse cumulative logistic distribution: if
logisticntail(x) = p, then invlogisticntail(p) = x

invlogisticntail(s,p)  the inverse reverse cumulative logistic distribution: if
logisticntail(s,x) = p, then invlogisticntail(s,p) = x

invlogisticntail(m,s,p)  the inverse reverse cumulative logistic distribution: if
logisticntail(m,s,x) = p, then
invlogisticntail(m,s,p) = x

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

invnbinomialntail(n,k,q)  the value of the negative binomial parameter, p, such that
q = nbinomialntail(n,k,p)

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

invnchi2tail(df,np,p)  the inverse reverse cumulative (upper tail or survivor) non-
central \( \chi^2 \) distribution: if nchi2tail(df,np,x) = p, then
invnchi2tail(df,np,p) = x

invnF(df1,df2,np,p)  the inverse cumulative noncentral \( F \) distribution: if
nF(df1,df2,np,f) = p, then invnF(df1,df2,np,p) = f

invnFtail(df1,df2,np,p)  the inverse reverse cumulative (upper tail or survivor) non-
central \( F \) distribution: if nFtail(df1,df2,np,f) = p, then
invnFtail(df1,df2,np,p) = f

invnibeta(a,b,np,p)  the inverse cumulative noncentral beta distribution: if
nibeta(a,b,np,x) = p, then invnibeta(a,b,np,p) = x

invnnormal(p)  the inverse cumulative standard normal distribution: if normal(z) = p,
then invnnormal(p) = z

invnt(df,np,p)  the inverse cumulative noncentral Student’s \( t \) distribution: if
nt(df,np,t) = p, then invnt(df,np,p) = t

invnttail(df,np,p)  the inverse reverse cumulative (upper tail or survivor) non-
central Student’s \( t \) distribution: if nttail(df,np,t) = p, then
invnttail(df,np,p) = t

invpoisson(k,p)  the Poisson mean such that the cumulative Poisson distribution eval-
uated at k is p: if poisson(m,k) = p, then invpoisson(k,p) = m

invpoissonntail(k,q)  the Poisson mean such that the reverse cumulative Poisson dis-
bution evaluated at k is q: if poissonntail(m,k) = q, then
invpoissonntail(k,q) = m

invt(df,p)  the inverse cumulative Student’s \( t \) distribution: if t(df,t) = p,
then invt(df,p) = t
invttail(df,p)

the inverse reverse cumulative (upper tail or survivor) Student’s
$t$ distribution: if $ttail(df,t) = p$, then $invttail(df,p) = t$

invtukeyprob(k,df,p)

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

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) distribu-
tion with shape $a$, scale $b$, and location $g$: if $weibullphtail(a,b,g,x) = p$,
then $invweibullphtail(a,b,g,p) = x$

invweibullphtail(a,b,g,p)

the inverse reverse cumulative Weibull (proportional hazards) distribu-
tion 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

lnigaussianden(m,a,x)

the natural logarithm of the inverse Gaussian density with mean $m$
and shape parameter $a$

lniwishartden(df,V,X)

the natural logarithm of the density of the inverse Wishart distribution;
missing if $df \leq n - 1$

lnlaplaceden(m,b,x)

the natural logarithm of the density of the Laplace distribution with
mean $m$ and scale parameter $b$

lnmvnormalden(M,V,X)

the natural logarithm of the multivariate normal density

lnnormal(z)

the natural logarithm of the cumulative standard normal distribution

lnnormalden(z)

the natural logarithm of the standard normal density, $N(0,1)$
lnnormalden($x, \sigma$) 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 $s\pi/\sqrt{3}$

logisticictail($x$) the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$

logisticictail($s, x$) the reverse cumulative logistic distribution with mean 0, scale $s$, and standard deviation $s\pi/\sqrt{3}$

logisticictail($m, s, x$) 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 $\chi^2$ distribution; 0 if $x < 0$

nchi2den($df, np, x$) the probability density of the noncentral $\chi^2$ distribution; 0 if $x < 0$

nchi2tail($df, np, x$) the reverse cumulative (upper tail or survivor) noncentral $\chi^2$ distribution; 1 if $x < 0$

nF($df_1, df_2, np, f$) the cumulative noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 0 if $f < 0$

nFden($df_1, df_2, np, f$) the probability density function of the noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 0 if $f < 0$

nFtail($df_1, df_2, np, f$) the reverse cumulative (upper tail or survivor) noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 1 if $f < 0$

nibeta($a, b, np, x$) the cumulative noncentral beta distribution; 0 if $x < 0$; or 1 if $x > 1$

normal($z$) the cumulative standard normal distribution
normalden(z)
the standard normal density, \( N(0, 1) \)

normalden(x, \sigma)
the normal density with mean 0 and standard deviation \( \sigma \)

normalden(x, \mu, \sigma)
the normal density with mean \( \mu \) and standard deviation \( \sigma \), \( N(\mu, \sigma^2) \)

npnchi2(df, x, p)
the noncentrality parameter, \( np \), for noncentral \( \chi^2 \): if \( \chi^2(df, np, x) = p \), then \( npnchi2(df, x, p) = np \)

npnF(df1, df2, f, p)
the noncentrality parameter, \( np \), for the noncentral \( F \): if \( F(df1, df2, np, f) = p \), then \( npnF(df1, df2, f, p) = np \)

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

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

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

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

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

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

poissontail(m, k)
the probability of observing \( \text{floor}(k) \) 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

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

tukeyprob(k, df, x)
the cumulative Tukey’s Studentized range distribution with \( k \) ranges and \( df \) degrees of freedom; 0 if \( x < 0 \)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbrev((s, n))</td>
<td>name (s), abbreviated to a length of (n)</td>
</tr>
<tr>
<td>char((n))</td>
<td>the character corresponding to ASCII or extended ASCII code (n); &quot;&quot; if (n) is not in the domain</td>
</tr>
<tr>
<td>collatorlocale((loc, type))</td>
<td>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</td>
</tr>
<tr>
<td>collatorversion((loc))</td>
<td>the version string of a collator based on locale (loc)</td>
</tr>
<tr>
<td>indexnot((s_1, s_2))</td>
<td>the position in ASCII string (s_1) of the first character of (s_1) not found in ASCII string (s_2), or 0 if all characters of (s_1) are found in (s_2)</td>
</tr>
<tr>
<td>plural((n, s))</td>
<td>the plural of (s) if (n \neq \pm 1)</td>
</tr>
<tr>
<td>plural((n, s_1, s_2))</td>
<td>the plural of (s_1), as modified by or replaced with (s_2), if (n \neq \pm 1)</td>
</tr>
<tr>
<td>real((s))</td>
<td>(s) converted to numeric or missing</td>
</tr>
<tr>
<td>regexcapture((n))</td>
<td>subexpression (n) from a previous regexm() or regexmatch() match</td>
</tr>
<tr>
<td>regexcapturenamed((grp))</td>
<td>subexpression corresponding to matching group named (grp) in regular expression from a previous regexm() or regexmatch() match</td>
</tr>
<tr>
<td>regexm((s, re))</td>
<td>a match of a regular expression, which evaluates to 1 if regular expression (re) is satisfied by the ASCII string (s); otherwise, 0</td>
</tr>
<tr>
<td>regexmatch((s, re[, noc[, std[, nlalt]]]]))</td>
<td>a match of a regular expression, which evaluates to 1 if regular expression (re) is satisfied by the ASCII string (s); otherwise, 0</td>
</tr>
<tr>
<td>regexr((s_1, re, s_2))</td>
<td>replaces the first substring within ASCII string (s_1) that matches (re) with ASCII string (s_2) and returns the resulting string</td>
</tr>
<tr>
<td>regexreplace((s_1, re, s_2[, noc[, fmt[, std[, nlalt]]]])))</td>
<td>replaces the first substring within ASCII string (s_1) that matches (re) with ASCII string (s_2) and returns the resulting string</td>
</tr>
<tr>
<td>regexreplaceall((s_1, re, s_2[, noc[, fmt[, std[, nlalt]]]])))</td>
<td>replaces all substrings within ASCII string (s_1) that match (re) with ASCII string (s_2) and returns the resulting string</td>
</tr>
<tr>
<td>regexs((n))</td>
<td>subexpression (n) from a previous regexm() or regexmatch() match, where (0 \leq n &lt; 10)</td>
</tr>
<tr>
<td>soundex((s))</td>
<td>the soundex code for a string, (s)</td>
</tr>
<tr>
<td>soundex_nara((s))</td>
<td>the U.S. Census soundex code for a string, (s)</td>
</tr>
<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>
</tbody>
</table>
string(n,s) a synonym for strofreal(n,s)
strtrim(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(s1,s2) 1 if s1 matches the pattern s2; otherwise, 0
strofreal(n) n converted to a string
strofreal(n,s) n converted to a string using the specified display format
strpos(s1,s2) the position in s1 at which s2 is first found, 0 if s2 does not occur, and 1 if s2 is empty
strproper(s) a string with the first ASCII letter and any other letters immediately following characters that are not letters capitalized; all other ASCII letters converted to lowercase
strreverse(s) the reverse of ASCII string s
strrpos(s1,s2) the position in s1 at which s2 is last found, 0 if s2 does not occur, and 1 if s2 is empty
strrtrim(s) s without trailing blanks (ASCII space character char(32))
strtoname(s[,p]) s translated into a Stata 13 compatible name
strtrim(s) s without leading and trailing blanks (ASCII space character char(32)); equivalent to strltrim(strrtrim(s))
strupper(s) uppercase ASCII characters in string s
subinstruct(s1,s2,s3,n) s1, where the first n occurrences in s1 of s2 have been replaced with s3
subinword(s1,s2,s3,n) s1, where the first n occurrences in s1 of s2 as a word have been replaced with s3
substr(s,n1,n2) the substring of s, starting at n1, for a length of n2
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,n1,n2) the Unicode substring of s, starting at character n1, for n2 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(s1,s2[,loc]) compares two Unicode strings
ustrcompareex(s1,s2,loc,case,cslv,norm,num,alt,fr) compares two Unicode strings
ustrfix(s[,rep]) replaces each invalid UTF-8 sequence with a Unicode character
ustrfrom(s,enc,mode) converts the string s in encoding enc to a UTF-8 encoded Unicode string
ustrinvalidcnt(s) the number of invalid UTF-8 sequences in s
ustrleft(s,n) the first n Unicode characters of the Unicode string s
ustrlen(s)  the number of characters in the Unicode string s
ustrtolower(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(s1,s2[,n])  the position in s1 at which s2 is first found; otherwise, 0
ustrregexm(s,re[,noc])  performs a match of a regular expression and evaluates to 1 if regular expression re is satisfied by the Unicode string s; otherwise, 0
ustrregexra(s1,re,s2[,noc])  replaces all substrings within the Unicode string s1 that match re with s2 and returns the resulting string
ustrregexrf(s1,re,s2[,noc])  replaces the first substring within the Unicode string s1 that matches re with s2 and returns the resulting string
ustrreverse(s)  the reverse of Unicode string s
ustrright(s,n)  the last n Unicode characters of the Unicode string s
ustrrpos(s1,s2[,n])  the position in s1 at which s2 is last found; otherwise, 0
ustrrtrim(s)  remove trailing Unicode whitespace characters and blanks from the Unicode string s
ustrsortkey(s[,loc])  generates a null-terminated byte array that can be used by the sort command to produce the same order as ustrcompare()
ustrsortkeyex(s,loc,st,case,cslv,norm,num,alt,fr)  generates a null-terminated byte array that can be used by the sort command to produce the same order as ustrcompare()
ustrtitle(s[,loc])  a string with the first characters of Unicode words titlecased and other characters lowercased
ustrto(s,enc,mode)  converts the Unicode string s in UTF-8 encoding to a string in encoding enc
ustrtohex(s[,n])  escaped hex digit string of s up to 200 Unicode characters
ustrtoname(s[,p])  string s translated into a Stata name
ustrtrim(s)  removes leading and trailing Unicode whitespace characters and blanks from the Unicode string s
ustrunescape(s)  the Unicode string corresponding to the escaped sequences of s
ustrupper(s[,loc])  uppercase all characters in string s under the given locale loc
ustrword(s,n[,loc])  the nth Unicode word in the Unicode string s
ustrwordcount(s[,loc])  the number of nonempty Unicode words in the Unicode string s
usubinstr(s1,s2,s3,n)  replaces the first n occurrences of the Unicode string s2 with the Unicode string s3 in s1
usubstr(s,n1,n2)  the Unicode substring of s, starting at n1, for a length of n2
word(s,n)  the nth word in s; missing (""") if n is missing
wordbreaklocale(loc, type)  the most closely related locale supported by ICU from loc if type is 1, the actual locale where the word-boundary analysis data come from if type is 2; or an empty string is returned for any other type

wordcount(s)  the number of words in s

Trigonometric functions

\[ \text{acos}(x) \]  the radian value of the arccosine of \( x \)
\[ \text{acosh}(x) \]  the inverse hyperbolic cosine of \( x \)
\[ \text{asin}(x) \]  the radian value of the arcsine of \( x \)
\[ \text{asinh}(x) \]  the inverse hyperbolic sine of \( x \)
\[ \text{atan}(x) \]  the radian value of the arctangent of \( x \)
\[ \text{atan2}(y, x) \]  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
\[ \text{atanh}(x) \]  the inverse hyperbolic tangent of \( x \)
\[ \text{cos}(x) \]  the cosine of \( x \), where \( x \) is in radians
\[ \text{cosh}(x) \]  the hyperbolic cosine of \( x \)
\[ \text{sin}(x) \]  the sine of \( x \), where \( x \) is in radians
\[ \text{sinh}(x) \]  the hyperbolic sine of \( x \)
\[ \text{tan}(x) \]  the tangent of \( x \), where \( x \) is in radians
\[ \text{tanh}(x) \]  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
Functions by name

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abbrev(s,n)</code></td>
<td>name s, abbreviated to a length of n</td>
</tr>
<tr>
<td><code>abs(x)</code></td>
<td>the absolute value of x</td>
</tr>
<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>age(e_dDOB,e_d[ ,snl])</code></td>
<td>the age in integer years on e_d for date of birth e_dDOB with snl the nonleap-year birthday for 29feb birthdates</td>
</tr>
<tr>
<td><code>age_frac(e_dDOB,e_d[ ,snl])</code></td>
<td>the age in years, including the fractional part, on e_d for date of birth e_dDOB with snl the nonleap-year birthday for 29feb birthdates</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>autocode(x,n,x_0,x_1)</code></td>
<td>partitions the interval from x_0 to x_1 into n equal-length intervals and returns the upper bound of the interval that contains x or the upper bound of the first or last interval if x &lt; x_0 or x &gt; x_1, respectively</td>
</tr>
<tr>
<td><code>betaden(a,b,x)</code></td>
<td>the probability density of the beta distribution, where a and b are the shape parameters; 0 if x &lt; 0 or x &gt; 1</td>
</tr>
<tr>
<td><code>binomial(n,k,θ)</code></td>
<td>the probability of observing floor(k) or fewer successes in floor(n) trials when the probability of a success on one trial is θ; 0 if k &lt; 0; or 1 if k &gt; n</td>
</tr>
<tr>
<td><code>binomialp(n,k,p)</code></td>
<td>the probability of observing floor(k) successes in floor(n) trials when the probability of a success on one trial is p</td>
</tr>
<tr>
<td><code>binomialtail(n,k,θ)</code></td>
<td>the probability of observing floor(k) or more successes in floor(n) trials when the probability of a success on one trial is θ; 1 if k &lt; 0; or 0 if k &gt; n</td>
</tr>
<tr>
<td><code>cauchy(a,b,x)</code></td>
<td>the cumulative Cauchy distribution with location parameter a and scale parameter b</td>
</tr>
<tr>
<td><code>birthday(e_dDOB,Y[ ,snl])</code></td>
<td>the e_d date of the birthday in year Y for date of birth e_dDOB with snl the nonleap-year birthday for 29feb birthdates</td>
</tr>
<tr>
<td><code>bofd(&quot;cal&quot;,e_d)</code></td>
<td>the e_b business date corresponding to e_d</td>
</tr>
<tr>
<td><code>byteorder()</code></td>
<td>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</td>
</tr>
<tr>
<td><code>c(name)</code></td>
<td>the value of the system or constant result c(name) (see [P] creturn)</td>
</tr>
<tr>
<td><code>_caller()</code></td>
<td>version of the program or session that invoked the currently running program; see [P] version</td>
</tr>
<tr>
<td><code>cauchy(a,b,x)</code></td>
<td>the cumulative Cauchy distribution with location parameter a and scale parameter b</td>
</tr>
<tr>
<td>function</td>
<td>description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cauchyden ((a, b, x))</td>
<td>the probability density of the Cauchy distribution with location parameter (a) and scale parameter (b)</td>
</tr>
<tr>
<td>cauchytail ((a, b, x))</td>
<td>the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter (a) and scale parameter (b)</td>
</tr>
<tr>
<td>Cdhms ((e_d, h, m, s))</td>
<td>the (e_{tc}) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to (e_d, h, m, s)</td>
</tr>
<tr>
<td>ceil ((x))</td>
<td>the unique integer (n) such that (n \leq x &lt; n + 1); (x) (not “.”) if (x) is missing, meaning that ceil(.a) = .a</td>
</tr>
<tr>
<td>char ((n))</td>
<td>the character corresponding to ASCII or extended ASCII code (n); &quot;&quot; if (n) is not in the domain</td>
</tr>
<tr>
<td>chi2 ((df, x))</td>
<td>the cumulative (\chi^2) distribution with (df) degrees of freedom; 0 if (x &lt; 0)</td>
</tr>
<tr>
<td>chi2den ((df, x))</td>
<td>the probability density of the (\chi^2) distribution with (df) degrees of freedom; 0 if (x &lt; 0)</td>
</tr>
<tr>
<td>chi2tail ((df, x))</td>
<td>the reverse cumulative (upper tail or survivor) (\chi^2) distribution with (df) degrees of freedom; 1 if (x &lt; 0)</td>
</tr>
<tr>
<td>Chms ((h, m, s))</td>
<td>the (e_{tc}) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to (h, m, s) on 01jan1960</td>
</tr>
<tr>
<td>chop ((x, \epsilon))</td>
<td>round ((x)) if (\text{abs}(x - \text{round}(x)) &lt; \epsilon); otherwise, (x); or (x) if (x) is missing</td>
</tr>
<tr>
<td>cholesky ((M))</td>
<td>the Cholesky decomposition of the matrix: if (R = \text{cholesky}(S)), then (RR^T = S)</td>
</tr>
<tr>
<td>clip ((x, a, b))</td>
<td>(x) if (a &lt; x &lt; b), (b) if (x \geq b), (a) if (x \leq a), or missing if (x) is missing</td>
</tr>
<tr>
<td>Clock ((s_1, s_2[Y]))</td>
<td>the (e_{tc}) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to (s_1) based on (s_2) and (Y)</td>
</tr>
<tr>
<td>clock ((s_1, s_2[Y]))</td>
<td>the (e_{tc}) datetime (ms. since 01jan1960 00:00:00.000) corresponding to (s_1) based on (s_2) and (Y)</td>
</tr>
<tr>
<td>Clockdiff ((e_{tc1}, e_{tc2}, s_u))</td>
<td>the (e_{tc}) datetime difference, rounded down to an integer, from (e_{tc1}) to (e_{tc2}) in (s_u) units of days, hours, minutes, seconds, or milliseconds</td>
</tr>
<tr>
<td>clockdiff ((e_{tc1}, e_{tc2}, s_u))</td>
<td>the (e_{tc}) datetime difference, rounded down to an integer, from (e_{tc1}) to (e_{tc2}) in (s_u) units of days, hours, minutes, seconds, or milliseconds</td>
</tr>
<tr>
<td>Clockdiff_frac ((e_{tc1}, e_{tc2}, s_u))</td>
<td>the (e_{tc}) datetime difference, including the fractional part, from (e_{tc1}) to (e_{tc2}) in (s_u) units of days, hours, minutes, seconds, or milliseconds</td>
</tr>
<tr>
<td>clockdiff_frac ((e_{tc1}, e_{tc2}, s_u))</td>
<td>the (e_{tc}) datetime difference, including the fractional part, from (e_{tc1}) to (e_{tc2}) in (s_u) units of days, hours, minutes, seconds, or milliseconds</td>
</tr>
<tr>
<td>Clockpart ((e_{tc}, s_u))</td>
<td>the integer year, month, day, hour, minute, second, or millisecond of (e_{tc}) with (s_u) specifying which time part</td>
</tr>
<tr>
<td>clockpart ((e_{tc}, s_u))</td>
<td>the integer year, month, day, hour, minute, second, or millisecond of (e_{tc}) with (s_u) specifying which time part</td>
</tr>
<tr>
<td>cloglog ((x))</td>
<td>the complementary log-log of (x)</td>
</tr>
<tr>
<td>Cmdyhms ((M, D, Y, h, m, s))</td>
<td>the (e_{tc}) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to (M, D, Y, h, m, s)</td>
</tr>
<tr>
<td>Cofc ((e_{tc}))</td>
<td>the (e_{tc}) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of (e_{tc}) (ms. without leap seconds since 01jan1960 00:00:00.000)</td>
</tr>
</tbody>
</table>
Functions by name

- \( \text{datediff}(a, b) \) : the difference in days between date \( a \) and date \( b \).
- \( \text{day}(c) \) : the day of the week of \( c \) (1 for Sunday, 7 for Saturday).
- \( \text{cofC}(e) \) : the numeric day of the month corresponding to \( e \).
- \( \text{daysinmonth}(e) \) : the number of days in the month of \( e \).
- \( \text{daysuntilweekday}(e) \) : the number of days from \( e \) until next day-of-week \( d \).
- \( \text{daysinweek}(e) \) : the number of days from \( e \) until next day-of-week \( d \).
- \( \text{colsof}(e) \) : the number of columns in \( e \).
- \( \text{colnfreeparms}(e) \) : the number of free parameters in columns of \( e \).
- \( \text{collatorversion}(e) \) : the version string of a collator based on locale \( e \).
- \( \text{cofCofd}(e) \) : the equation number of \( e \).
- \( \text{colnumb}(M) \) : the column number of \( M \) associated with column name \( s \); missing if the column cannot be found.
- \( \text{collocale}(loc, type) \) : the most closely related locale supported by ICU from \( loc \) if \( type \) is 2; the actual locale where the collation data comes from if \( type \) is not 2.
24 Functions by name

dgammapdadx(a,x) \frac{\partial^2 P(a,x)}{\partial a \partial x}, where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)
dgammapdx(a,x) \frac{\partial P(a,x)}{\partial x}, where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)
dgammapdxadx(a,x) \frac{\partial^2 P(a,x)}{\partial x^2}, where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)
dhms(e_d,h,m,s) the \( e_{tc} \) datetime (ms. since Jan 1960 00:00:00.000) corresponding to \( e_d, h, m, \) and \( s \)
diag(M) the square, diagonal matrix created from the row or column vector
dig0cnt(M) the number of zeros on the diagonal of \( M \)
digamma(x) the \text{digamma}(x) \text{function, } d \ln \Gamma(x)/dx
dmy(D,M,Y) the \( e_d \) date (days since Jan 1960) corresponding to \( D, M, Y \)
dofb(e_b,"calw") the \( e_d \) datetime corresponding to \( e_b \)
dofC(e_{tc}) the \( e_d \) date (days since Jan 1960) of datetime \( e_{tc} \) (ms. with leap seconds since Jan 1960 00:00:00.000)
dofc(e_{tc}) the \( e_d \) date (days since Jan 1960) of datetime \( e_{tc} \) (ms. since Jan 1960 00:00:00.000)
dofh(e_h) the \( e_d \) date (days since Jan 1960) of the start of half-year \( e_h \)
dofm(e_m) the \( e_d \) date (days since Jan 1960) of the start of month \( e_m \)
dofq(e_q) the \( e_d \) date (days since Jan 1960) of the start of quarter \( e_q \)
dofw(e_w) the \( e_d \) date (days since Jan 1960) of the start of week \( e_w \)
dofy(e_y) the \( e_d \) date (days since Jan 1960) of Jan in year \( e_y \)
dow(e_d) the numeric day of the week corresponding to date \( e_d \); 0 = Sunday, 1 = Monday, \ldots, 6 = Saturday
doy(e_d) the numeric day of the year corresponding to date \( e_d \)
dunnettprob(k,df,x) the cumulative multiple range distribution that is used in Dunnett’s multiple-comparison method with \( k \) ranges and \( df \) degrees of freedom; 0 if \( x < 0 \)
e(name) the value of stored result e(name); see [U] 18.8 Accessing results calculated by other programs
el(s,i,j) \([\text{floor}(i),\text{floor}(j)]\), 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
epsdouble() the machine precision of a double-precision number
epsfloat() the machine precision of a floating-point number
exp(x) the exponential function \( e^x \)
expml1(x) \( e^x - 1 \) with higher precision than \( \exp(x) - 1 \) for small values of \( x \)
exponential(b,x) the cumulative exponential distribution with scale \( b \)
exponentialalden(b,x) the probability density function of the exponential distribution with scale \( b \)
exponentialaltail(b,x) 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: \( F(df_1,df_2,f) = \int_0^f \text{Fden}(df_1,df_2,t) \, dt; 0 \) 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 \( f < 0 \)
fileexists($f$) 1 if the file specified by $f$ exists; otherwise, 0
filered($f$) the contents of the file specified by $f$
filereaderror($s$) 0 or positive integer, said value having the interpretation of a return code
filewrite($f, s[,r]$) writes the string specified by $s$ to the file specified by $f$ and returns the number of bytes in the resulting file
firstdayofmonth($e_d$) the $e_d$ date of the first day of the month of $e_d$
firstdowofmonth($M,Y,d$) a synonym for firstweekdayofmonth($M,Y,d$)
firstweekdayofmonth($M,Y,d$) the $e_d$ date of the first day-of-week $d$ in month $M$ of year $Y$
float($x$) the value of $x$ rounded to float precision
floor($x$) the unique integer $n$ such that $n \leq x < n + 1$; $x$ (not \( \).\) if $x$ is missing, meaning that floor(.a) = .a
fmtwidth($fmtstr$) the output length of the \%fmt contained in $fmtstr$; missing if $fmtstr$ does not contain a valid \%fmt
frval() returns values of variables stored in other frames
_Frval() programmer's version of frval() 
Ftail($df_1, df_2, f$) the reverse cumulative (upper tail or survivor) $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom; 1 if $f < 0$
gammaden($a,b,g,x$) the probability density function of the gamma distribution; 0 if $x < g$
gamma($a,x$) the cumulative gamma distribution with shape parameter $a$; 0 if $x < 0$
gammaptail($a,x$) the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter $a$; 1 if $x < 0$
get($systemname$) a copy of Stata internal system matrix $systemname$
hadamard($M,N$) a matrix whose $i, j$ element is $M[i,j] \cdot N[i,j]$ (if $M$ and $N$ are not the same size, this function reports a conformability error)
halfyear($e_d$) the numeric half of the year corresponding to date $e_d$
halfyearly($s_1,s_2,[Y]$) the $e_h$ half-yearly date (half-years since 1960h1) corresponding to $s_1$ based on $s_2$ and $Y$; $Y$ specifies toyear; see date()
has่าง_eprop($name$) 1 if $name$ appears as a word in e(properties); otherwise, 0
hh($e_{tc}$) the hour corresponding to datetime $e_{tc}$ (ms. since 01jan1960 00:00:00.000)
hhC($e_{tC}$) the hour corresponding to datetime $e_{tC}$ (ms. with leap seconds since 01jan1960 00:00:00.000)
hms($h,m,s$) the $e_{tc}$ datetime (ms. since 01jan1960 00:00:00.000) corresponding to $h$, $m$, $s$ on 01jan1960
hofd($e_d$) the $e_h$ half-yearly date (half years since 1960h1) containing date $e_d$
hours($ms$) $ms$/3,600,000
hypergeometric($N,K,n,k$) the cumulative probability of the hypergeometric distribution
hypergeometricp($N,K,n,k$) the hypergeometric probability of $k$ successes out of a sample of size $n$, from a population of size $N$ containing $K$ elements that have the attribute of interest
I($n$) an $n \times n$ identity matrix if $n$ is an integer; otherwise, a round($n$) $\times$ round($n$) identity matrix
ibeta(\(a,b,x\)) the cumulative beta distribution with shape parameters \(a\) and \(b\); 0 if \(x < 0\); or 1 if \(x > 1\)

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

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

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

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

indexnot(\(s_1,s_2\)) the position in ASCII string \(s_1\) of the first character of \(s_1\) not found in ASCII string \(s_2\), or 0 if all characters of \(s_1\) are found in \(s_2\)

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

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

int(\(x\)) the integer obtained by truncating \(x\) toward 0 (thus, \(\text{int}(5.2) = 5\) and \(\text{int}(-5.8) = -5\); \(x\) (not “.”) if \(x\) is missing, meaning that \(\text{int}(.a) = .a\)

inv(\(M\)) the inverse of the matrix \(M\)

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 \(\text{chi2}(df,x) = p\), then \(\text{invchi2}(df,p) = x\)

invchi2tail(\(df,p\)) the inverse of chi2tail(): if \(\text{chi2tail}(df,x) = p\), then \(\text{invchi2tail}(df,p) = x\)

invcloglog(\(x\)) the inverse of the complementary log-log function of \(x\)

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

invexponential(\(b,p\)) the inverse cumulative exponential distribution with scale \(b\); if \(\text{exponential}(b,x) = p\), 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 \(\text{Ftail}(df_1,df_2,f) = p\), then \(\text{invFtail}(df_1,df_2,p) = f\)

invgammap(\(a,p\)) the inverse cumulative gamma distribution: if \(\text{gammap}(a,x) = p\), then \(\text{invgammap}(a,p) = x\)
\begin{itemize}
  \item \texttt{invgammaptail}(a,p) \quad \text{the inverse reverse cumulative (upper tail or survivor) gamma distribution}: \text{if } \texttt{gammaptail}(a,x) = p, \text{ then } \texttt{invgammaptail}(a,p) = x
  \item \texttt{invbeta}(a,b,p) \quad \text{the inverse cumulative beta distribution}: \text{if } \texttt{ibeta}(a,b,x) = p, \text{ then } \texttt{invbeta}(a,b,p) = x
  \item \texttt{invbetatail}(a,b,p) \quad \text{the inverse reverse cumulative (upper tail or survivor) beta distribution}: \text{if } \texttt{betatail}(a,b,x) = p, \text{ then } \texttt{invbetatail}(a,b,p) = x
  \item \texttt{invgaussian}(m,a,p) \quad \text{the inverse of \texttt{igaussian}}(): \text{if } \texttt{igaussian}(m,a,x) = p, \text{ then } \texttt{invgaussian}(m,a,p) = x
  \item \texttt{invgaussianantail}(m,a,p) \quad \text{the inverse of \texttt{igaussianantail}}(): \text{if } \texttt{igaussianantail}(m,a,x) = p, \text{ then } \texttt{invgaussianantail}(m,a,p) = x
  \item \texttt{inlaplace}(m,b,p) \quad \text{the inverse of \texttt{laplace}}(): \text{if } \texttt{laplace}(m,b,x) = p, \text{ then } \texttt{inlaplace}(m,b,p) = x
  \item \texttt{inlaplacetail}(m,b,p) \quad \text{the inverse of \texttt{laplacetail}}(): \text{if } \texttt{laplacetail}(m,b,x) = p, \text{ then } \texttt{inlaplacetail}(m,b,p) = x
  \item \texttt{invlogistic}(p) \quad \text{the inverse cumulative logistic distribution}: \text{if } \texttt{logistic}(x) = p, \text{ then } \texttt{invlogistic}(p) = x
  \item \texttt{invlogistic}(s,p) \quad \text{the inverse cumulative logistic distribution}: \text{if } \texttt{logistic}(s,x) = p, \text{ then } \texttt{invlogistic}(s,p) = x
  \item \texttt{invlogistic}(m,s,p) \quad \text{the inverse cumulative logistic distribution}: \text{if } \texttt{logistic}(m,s,x) = p, \text{ then } \texttt{invlogistic}(m,s,p) = x
  \item \texttt{invlogistictail}(p) \quad \text{the inverse reverse cumulative logistic distribution}: \text{if } \texttt{logistictail}(x) = p, \text{ then } \texttt{invlogistictail}(p) = x
  \item \texttt{invlogistictail}(s,p) \quad \text{the inverse cumulative logistic distribution}: \text{if } \texttt{logistictail}(s,x) = p, \text{ then } \texttt{invlogistictail}(s,p) = x
  \item \texttt{invlogistictail}(m,s,p) \quad \text{the inverse cumulative logistic distribution}: \text{if } \texttt{logistictail}(m,s,x) = p, \text{ then } \texttt{invlogistictail}(m,s,p) = x
  \item \texttt{invlogit}(x) \quad \text{the inverse of the logit function of } x
  \item \texttt{invnbinomial}(n,k,q) \quad \text{the value of the negative binomial parameter, } p, \text{ such that } q = \texttt{nbinomial}(n,k,p)
  \item \texttt{invnbinomialtail}(n,k,q) \quad \text{the value of the negative binomial parameter, } p, \text{ such that } q = \texttt{nbinomialtail}(n,k,p)
  \item \texttt{invnchi2}(df,np,p) \quad \text{the inverse cumulative noncentral } \chi^2 \text{ distribution}: \text{if } \texttt{nchi2}(df,np,x) = p, \text{ then } \texttt{invnchi2}(df,np,p) = x
  \item \texttt{invnchi2tail}(df,np,p) \quad \text{the inverse reverse cumulative (upper tail or survivor) noncentral } \chi^2 \text{ distribution}: \text{if } \texttt{nchi2tail}(df,np,x) = p, \text{ then } \texttt{invnchi2tail}(df,np,p) = x
  \item \texttt{invnF}(df_1,df_2,np,p) \quad \text{the inverse cumulative noncentral } F \text{ distribution}: \text{if } \texttt{nF}(df_1,df_2,np,f) = p, \text{ then } \texttt{invnF}(df_1,df_2,np,p) = f
  \item \texttt{invnFtail}(df_1,df_2,np,p) \quad \text{the inverse reverse cumulative (upper tail or survivor) noncentral } F \text{ distribution}: \text{if } \texttt{nFtail}(df_1,df_2,np,f) = p, \text{ then } \texttt{invnFtail}(df_1,df_2,np,p) = f
  \item \texttt{invnibeta}(a,b,np,p) \quad \text{the inverse cumulative noncentral beta distribution}: \text{if } \texttt{nibeta}(a,b,np,x) = p, \text{ then } \texttt{invnibeta}(a,b,np,p) = x
  \item \texttt{invnormal}(p) \quad \text{the inverse cumulative standard normal distribution}: \text{if } \texttt{normal}(z) = p, \text{ then } \texttt{invnormal}(p) = z
\end{itemize}
The document contains a list of 28 functions by name, including:

- `invpoisson(df,np,p)`
- `invntail(df,np,p)`
- `invpoisson(k,p)`
- `invpoissontail(k,q)`
- `invsym(M)`
- `invvech(M)`
- `invvecp(M)`
- `invweibull(a,b,p)`
- `invweibull(a,b,g,p)`
- `invweibullph(a,b,p)`
- `invweibullph(a,b,g,p)`
- `invweibullphtail(a,b,p)`
- `invweibullphtail(a,b,g,p)`
- `invtukeyprob(k,df,p)`
- `invweibulltail(a,b,p)`
- `invweibulltail(a,b,g,p)`

Each function is described with a brief explanation and conditions, such as:

- `invpoisson(df,np,p)` returns the Poisson mean such that the cumulative Poisson distribution evaluated at `k = p` if `poisson(m,k) = p`, then `invpoisson(k,p) = m`.

The functions are designed to work with various distributions and parameters, providing cumulative, reverse cumulative, and inverse cumulative calculations.
**isleapsecond**($e_{tC}$) \[1 \text{ if } e_{tC} \text{ is a leap second; otherwise, } 0\]

**isleapyear**($Y$) \[1 \text{ if } Y \text{ is a leap year; otherwise, } 0\]

**issymmetric**($M$) \[1 \text{ if the matrix is symmetric; otherwise, } 0\]

**J**($r,c,z$) \[the \ r \times c \text{ matrix containing elements } z\]

**laplace**($m,b,x$) \[the cumulative Laplace distribution with mean } m \text{ and scale parameter } b\]

**laplaceden**($m,b,x$) \[the probability density of the Laplace distribution with mean } m \text{ and scale parameter } b\]

**laplacetail**($m,b,x$) \[the reverse cumulative (upper tail or survivor) Laplace distribution with mean } m \text{ and scale parameter } b\]

**lastdayofmonth**($e_d$) \[the } e_d \text{ date of the last day of the month of } e_d\]

**lastdowofmonth**($M,Y,d$) \[a synonym for } lastweekdayofmonth(M,Y,d)\]

**lastweekdayofmonth**($M,Y,d$) \[the } e_d \text{ date of the last day-of-week } d \text{ in month } M \text{ of year } Y\]

**ln**($x$) \[the natural logarithm, } \ln(x)\]

**ln1m**($x$) \[the natural logarithm of } 1 - x \text{ with higher precision than } \ln(1 - x) \text{ for small values of } |x|\]

**ln1p**($x$) \[the natural logarithm of } 1 + x \text{ with higher precision than } \ln(1 + x) \text{ for small values of } |x|\]

**lncauchyden**($a,b,x$) \[the natural logarithm of the density of the Cauchy distribution with location parameter } a \text{ and scale parameter } b\]

**lnfactorial**($n$) \[the natural log of } n \text{ factorial } = \ln(n!)\]

**lngamma**($x$) \[\ln\{\Gamma(x)\}\]

**lnigammaden**($a,b,x$) \[the natural logarithm of the inverse gamma density, where } a \text{ is the shape parameter and } b \text{ is the scale parameter}\]

**lnnigaussianden**($m,a,x$) \[the natural logarithm of the inverse Gaussian density with mean } m \text{ and shape parameter } a\]

**lniwishartden**($df,V,X$) \[the natural logarithm of the density of the inverse Wishart distribution; missing if } df \leq n - 1\]

**lnlaplaceden**($m,b,x$) \[the natural logarithm of the density of the Laplace distribution with mean } m \text{ and scale parameter } b\]

**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**($x,\mu,\sigma$) \[the natural logarithm of the normal density with mean } \mu \text{ and standard deviation } \sigma, \ N(\mu,\sigma^2)\]

**lnwishartden**($df,V,X$) \[the natural logarithm of the density of the Wishart distribution; missing if } df \leq n - 1\]

**log**($x$) \[a synonym for } \ln(x)\]

**log10**($x$) \[the base-10 logarithm of } x\]

**log1m**($x$) \[a synonym for } \ln1m(x)\]

**log1p**($x$) \[a synonym for } \ln1p(x)\]
logistic\( (x) \) the cumulative logistic distribution with mean 0 and standard deviation \( \pi / \sqrt{3} \)

logistic\( (s, x) \) the cumulative logistic distribution with mean 0, scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logistic\( (m, s, x) \) the cumulative logistic distribution with mean \( m \), scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logistic\( \text{den} (x) \) the density of the logistic distribution with mean 0 and standard deviation \( \pi / \sqrt{3} \)

logistic\( \text{den} (s, x) \) the density of the logistic distribution with mean 0, scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logistic\( \text{den} (m, s, x) \) the density of the logistic distribution with mean \( m \), scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logistictail\( (x) \) the reverse cumulative logistic distribution with mean 0 and standard deviation \( \pi / \sqrt{3} \)

logistictail\( (s, x) \) the reverse cumulative logistic distribution with mean 0, scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logistictail\( (m, s, x) \) the reverse cumulative logistic distribution with mean \( m \), scale \( s \), and standard deviation \( s\pi / \sqrt{3} \)

logit\( (x) \) the log of the odds ratio of \( x \), \( \logit(x) = \ln \left\{ x / (1 - x) \right\} \)

matmissing\( (M) \) 1 if any elements of the matrix are missing; otherwise, 0

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

matuniform\( (r, c) \) the \( r \times c \) matrices containing uniformly distributed pseudorandom numbers on the interval \( (0, 1) \)

max\( (x_1, x_2, \ldots, x_n) \) the maximum value of \( x_1, x_2, \ldots, x_n \)

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

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

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

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

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

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

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

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

min\( (x_1, x_2, \ldots, x_n) \) the minimum value of \( x_1, x_2, \ldots, x_n \)

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

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

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

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

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

minutes\( (ms) \) \( ms \) / 60,000

missing\( (x_1, x_2, \ldots, x_n) \) 1 if any \( x_i \) evaluates to missing; otherwise, 0

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

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

mofd($e_d$) the $e_m$ monthly date (months since 1960m1) containing date $e_d$.

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

monthly($s_1,s_2,Y$) the $e_m$ monthly date (months since 1960m1) corresponding to $s_1$ based on $s_2$ and $Y$; $Y$ specifies topyear; see date() the relative difference of $X$ and $Y$, where the relative difference is defined as $\max_{i,j}\{|x_{ij} - y_{ij}|/(|y_{ij}| + 1)\}$.

msofhours($h$) $h \times 3,600,000$.

msofminutes($m$) $m \times 60,000$.

msoseconds($s$) $s \times 1,000$.

nbetaden($a,b,np,x$) the probability density function of the noncentral beta distribution; $0$ if $x < 0$ or $x > 1$.

nbinomial($n,k,p$) the cumulative probability of the negative binomial distribution.

nbinomialp($n,k,p$) the negative binomial probability.

nbinomialtail($n,k,p$) the reverse cumulative probability of the negative binomial distribution.

nchi2($df,np,x$) the cumulative noncentral $\chi^2$ distribution; $0$ if $x < 0$.

nchi2den($df,np,x$) the probability density of the noncentral $\chi^2$ distribution; $0$ if $x < 0$.

nchi2tail($df,np,x$) the reverse cumulative (upper tail or survivor) noncentral $\chi^2$ distribution; $1$ if $x < 0$.

nextbirthday($e_{dDOB},e_d,s_{nl}$) the $e_d$ date of the first birthday after $e_d$ for date of birth $e_{dDOB}$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates.

nextdow($e_{d_1},d$) a synonym for nextweekday($e_{d_1},d$).

nextleapyear($Y$) the first leap year after year $Y$.

nextweekd($e_{d_1},d$) the $e_d$ date of the first day-of-week $d$ after $e_d$.

nF($df_1,df_2,np,f$) the cumulative noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; $0$ if $f < 0$.

nFden($df_1,df_2,np,f$) the probability density function of the noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; $0$ if $f < 0$.

nFtail($df_1,df_2,np,f$) the reverse cumulative (upper tail or survivor) noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; $1$ if $f < 0$.

nibeta($a,b,np,x$) the cumulative noncentral beta distribution; $0$ if $x < 0$; or $1$ if $x > 1$.

normal($z$) the cumulative standard normal distribution.

normalden($z$) the standard normal density, $N(0,1)$.

normalden($x,\sigma$) the normal density with mean $0$ and standard deviation $\sigma$.

normalden($x,\mu,\sigma$) the normal density with mean $\mu$ and standard deviation $\sigma$, $N(\mu,\sigma^2)$.

now() the current $e_{tc}$ datetime.
npnchi2(df, x, p)

the noncentrality parameter, \( np \), for noncentral \( \chi^2 \); if

\[ \text{nchi2}(df, np, x) = p \], then \( \text{npnchi2}(df, x, p) = np \)

npnF(df1, df2, f, p)

the noncentrality parameter, \( np \), for the noncentral \( F \); if

\[ \text{nF}(df1, df2, np, f) = p \], then \( \text{npnF}(df1, df2, f, p) = np \)

npnt(df, t, p)

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

nt(df, np, t)

the cumulative noncentral Student’s \( t \) distribution with \( df \) degrees of freedom and noncentrality parameter \( np \)

ntden(df, np, t)

the probability density function of the noncentral Student’s \( t \) distribution with \( df \) degrees of freedom and noncentrality parameter \( np \)

nttail(df, np, t)

the reverse cumulative (upper tail or survivor) noncentral Student’s \( t \) distribution with \( df \) degrees of freedom and noncentrality parameter \( np \)

nullmat(matname)

use with the row-join (,) and column-join (\) operators

plural(n, s)

the plural of \( s \) if \( n \neq \pm 1 \)

plural(n, s1, s2)

the plural of \( s1 \), as modified by or replaced with \( s2 \), if \( n \neq \pm 1 \)

poisson(m, k)

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

poissonp(m, k)

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

poissontail(m, k)

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

previousbirthday(e_dDOB, e_d[ , snl])

the \( e_d \) date of the birthday immediately before \( e_d \) for date of birth \( e_dDOB \) with \( snl \) the nonleap-year birthday for 29feb birthdates

previousdow(e_d, d)

a synonym for \( \text{previousweekday}(e_d, d) \)

previousleapyear(Y)

the leap year immediately before year \( Y \)

previousweekday(e_d, d)

the \( e_d \) date of the last day-of-week \( d \) before \( e_d \)

qofd(e_d)

the \( e_q \) quarterly date (quarters since 1960q1) containing date \( e_d \)

quarter(e_d)

the numeric quarter of the year corresponding to date \( e_d \)

quarterly(s1, s2[ , Y])

the \( e_q \) quarterly date (quarters since 1960q1) corresponding to \( s1 \) based on \( s2 \) and \( Y \); \( Y \) specifies \( \text{topyear} \); see \text{date()}

r(name)

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

rbeta(a, b)

beta\((a,b)\) random variates, where \( a \) and \( b \) are the beta distribution shape parameters

rbinomial(n, p)

binomial\((n,p)\) random variates, where \( n \) is the number of trials and \( p \) is the success probability

rcauchy(a, b)

Cauchy\((a,b)\) random variates, where \( a \) is the location parameter and \( b \) is the scale parameter

rchisq(df)

\( \chi^2 \), with \( df \) degrees of freedom, random variates

recode(x, x1, \ldots, xn)

\textit{missing} if \( x_1, x_2, \ldots, x_n \) is not weakly increasing; \( x \) if \( x \) is missing; \( x_1 \) if \( x \leq x_1 \); \( x_2 \) if \( x \leq x_2 \); \ldots; otherwise, \( x_n \) if \( x > x_1, x_2, \ldots, x_{n-1} \). \( x_i \geq \) is interpreted as \( x_i = +\infty \)

real(s)

\( s \) converted to numeric or \textit{missing}
Functions by name

regexcapture(n)  
subexpression $n$ from a previous regexm() or regexmatch() match

regexcapturenamed(grp)  
subexpression corresponding to matching group named $grp$ in regular expression from a previous regexm() or regexmatch() match

regex(s,re)  
a match of a regular expression, which evaluates to 1 if regular expression $re$ is satisfied by the ASCII string $s$; otherwise, 0

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

regexr(s1,re,s2)  
replaces the first substring within ASCII string $s1$ that matches $re$ with ASCII string $s2$ and returns the resulting string

regexreplace(s1,re,s2[ ,noc[ ,fmt[ ,std[ ,nlalt ] ] ] ] )  
replaces the first substring within ASCII string $s1$ that matches $re$ with ASCII string $s2$ and returns the resulting string

regexreplaceall(s1,re,s2[ ,noc[ ,fmt[ ,std[ ,nlalt ] ] ] ] )  
replaces all substrings within ASCII string $s1$ that match $re$ with ASCII string $s2$ and returns the resulting string

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

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

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

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

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

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

rhypergeometric(N,K,n)  
hypergeometric random variates

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

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

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

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

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

rnbinomial(n,p)  
negative binomial random variates

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

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

rnormal(m,s)  
normal(m,s) (Gaussian) random variates, where $m$ is the mean and $s$ is the standard deviation
round($x,y$) or round($x$)

$x$ rounded in units of $y$ or $x$ rounded to the nearest integer if the argument $y$ is omitted; $x$ (not “.”) if $x$ is missing (meaning that round(.a) = .a and that round(.a,y) = .a if $y$ is not missing) and if $y$ is missing, then “.” is returned.

roweqnumb($M,s$)

the equation number of $M$ associated with row equation $s$; missing if the row equation cannot be found.

rownfreeparms($M$)

the number of free parameters in rows of $M$.

rownumb($M,s$)

the row number of $M$ associated with row name $s$; missing if the row cannot be found.

rowsof($M$)

the number of rows of $M$.

rpoisson($m$)

Poisson($m$) random variates, where $m$ is the distribution mean.

rt($df$)

Student’s $t$ random variates, where $df$ is the degrees of freedom.

runiform()

uniformly distributed random variates over the interval (0,1).

runiform($a,b$)

uniformly distributed random variates over the interval ($a,b$).

runiformint($a,b$)

uniformly distributed random integer variates on the interval [a,b].

rweibull($a,b$)

Weibull variates with shape $a$ and scale $b$.

rweibull($a,b,g$)

Weibull variates with shape $a$, scale $b$, and location $g$.

rweibullph($a,b$)

Weibull (proportional hazards) variates with shape $a$ and scale $b$.

rweibullph($a,b,g$)

Weibull (proportional hazards) variates with shape $a$, scale $b$, and location $g$.

s(name)

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

scalar($exp$)

restricts name interpretation to scalars and matrices.

seconds($ms$)

$ms/1,000$.

sign($x$)

the sign of $x$: -1 if $x<0$, 0 if $x=0$, 1 if $x>0$, or missing if $x$ is missing.

sin($x$)

the sine of $x$, where $x$ is in radians.

sinh($x$)

the hyperbolic sine of $x$.

smallestdouble()

the smallest double-precision number greater than zero.

soundex($s$)

the soundex code for a string, $s$.

soundex_nara($s$)

the U.S. Census soundex code for a string, $s$.

sqrt($x$)

the square root of $x$.

ss($et_c$)

the second corresponding to datetime $et_c$ (ms. since 01jan1960 00:00:00.000).

ssC($et_{C}$)

the second corresponding to datetime $et_{C}$ (ms. with leap seconds since 01jan1960 00:00:00.000).

strcat($s_1,s_2$)

there is no strcat() function; instead the addition operator is used to concatenate strings.

strdup($s_1,n$)

there is no strdup() function; instead the multiplication operator is used to create multiple copies of strings.

string($n$)

a synonym for strofreal($n$).

string($n,s$)

a synonym for strofreal($n,s$).

strittrim($s$)

$s$ 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(s₁,s₂) 1 if s₁ matches the pattern s₂; otherwise, 0
strofreal(n) n converted to a string
strofreal(n,s) n converted to a string using the specified display format
strpos(s₁,s₂) the position in s₁ at which s₂ is first found, 0 if s₂ does not occur,
and 1 if s₂ is empty
strproper(s) a string with the first ASCII letter and any other letters immediately
following characters that are not letters capitalized; all other
ASCII letters converted to lowercase
strreverse(s) the reverse of ASCII string s
strrpos(s₁,s₂) the position in s₁ at which s₂ is last found, 0 if s₂ does not occur,
and 1 if s₂ is empty
strrtrim(s) s without trailing blanks (ASCII space character char(32))
strtoname(s[],p) s translated into a Stata 13 compatible name
strtrim(s) s without leading and trailing blanks (ASCII space character char(32)); equivalent to strltrim(strrtrim(s))
strupper(s) uppercase ASCII characters in string s
subinestr(s₁,s₂,s₃,n) s₁, where the first n occurrences in s₁ of s₂ have been replaced
with s₃
substr(s,n₁,n₂) the substring of s, starting at n₁, for a length of n₂
sum(x) the running sum of x, treating missing values as zero
sweep(M,i) matrix M with i-th row/column swept
t(df,t) the cumulative Student’s t distribution with df degrees of freedom
the tangent of x, where x is in radians
the hyperbolic tangent of x
convenience function to make typing dates and times in expressions
easier
tc(l) convenience function to make typing dates and times in expressions
easier
td(l) convenience function to make typing dates in expressions easier
tden(df,t) the probability density function of Student’s t distribution
th(l) convenience function to make typing half-yearly dates in expressions
easier
tin(d₁,d₂) true if d₁ ≤ t ≤ d₂, where t is the time variable previously tsset
convenience function to make typing monthly dates in expressions
easier
tm(l) escaped decimal or hex digit strings of up to 200 bytes of s
tobytess(s[],n) today’s e_d date
today() convenience function to make typing quarterly dates in expressions
easier
trace(M) the trace of matrix M
trigamma(x) the second derivative of lngamma(x) = d² lnΓ(x)/dx²
36 Functions by name

trunc($x$)  a synonym for \texttt{int($x$)}

ttail($df,t$)  the reverse cumulative (upper tail or survivor) Student’s $t$ distribution; the probability $T > t$

tukeyprob($k,df,x$)  the cumulative Tukey’s Studentized range distribution with $k$ ranges and $df$ degrees of freedom; 0 if $x < 0$

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

twithin($d_1,d_2$)  \texttt{true} if $d_1 < t < d_2$, where $t$ is the time variable previously \texttt{tsset}

uchar($n$)  the Unicode character corresponding to Unicode code point $n$ or an empty string if $n$ is beyond the Unicode code-point range

udstrlen($s$)  the number of display columns needed to display the Unicode string $s$ in the Stata Results window

udsubstr($s,n_1,n_2$)  the Unicode substring of $s$, starting at character $n_1$, for $n_2$ display columns

uisdigit($s$)  1 if the first Unicode character in $s$ is a Unicode decimal digit; otherwise, 0

uisletter($s$)  1 if the first Unicode character in $s$ is a Unicode letter; otherwise, 0

ustrcompare($s_1,s_2[,loc]$)  compares two Unicode strings

ustrcompareex($s_1,s_2,st,case,cslv,norm,num,alt,fr$)  compares two Unicode strings

ustrfix($s[,rep]$)  replaces each invalid UTF-8 sequence with a Unicode character

ustrfrom($s,enc,mode$)  converts the string $s$ in encoding \texttt{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 \texttt{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 \texttt{norm}

ustrpos($s_1,s_2[,n]$)  the position in $s_1$ at which $s_2$ is first found; otherwise, 0

ustrregexm($s,re[,noc]$)  performs a match of a regular expression and evaluates to 1 if regular expression $re$ is satisfied by the Unicode string $s$; otherwise, 0

ustrregexra($s_1,re,s_2[,noc]$)  replaces all substrings within the Unicode string $s_1$ that match $re$ with $s_2$ and returns the resulting string

ustrregexrf($s_1,re,s_2[,noc]$)  replaces the first substring within the Unicode string $s_1$ that matches $re$ with $s_2$ and returns the resulting string

ustrregexs($n$)  subexpression $n$ from a previous \texttt{ustrregexm()} match

ustrreverse($s$)  the reverse of Unicode string $s$

ustrright($s,n$)  the last $n$ Unicode characters of the Unicode string $s$

ustrrpos($s_1,s_2[,n]$)  the position in $s_1$ at which $s_2$ is last found; otherwise, 0

ustrrtrim($s$)  remove trailing Unicode whitespace characters and blanks from the Unicode string $s$
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>ustrupper(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>usubstr(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>
<tr>
<td><code>vec(M)</code></td>
<td>a column vector formed by listing the elements of <code>M</code>, starting with the first column and proceeding column by column</td>
</tr>
<tr>
<td><code>vecdiag(M)</code></td>
<td>the row vector containing the diagonal of matrix <code>M</code></td>
</tr>
<tr>
<td><code>vech(M)</code></td>
<td>a column vector formed by listing the lower triangle elements of <code>M</code></td>
</tr>
<tr>
<td><code>vecp(M)</code></td>
<td>a column vector formed by listing the upper triangle elements of <code>M</code></td>
</tr>
<tr>
<td><code>week(\text{\textit{ed}})</code></td>
<td>the numeric week of the year corresponding to date <code>\textit{ed}</code>, the \texttt{%td} encoded date (days since 01jan1960)</td>
</tr>
<tr>
<td><code>weekly(s1,s2[,Y])</code></td>
<td>the <code>\textit{ew}</code> weekly date (weeks since 1960w1) corresponding to <code>s1</code> based on <code>s2</code> and <code>Y</code>; <code>Y</code> specifies \texttt{topyear}; see \texttt{date()}</td>
</tr>
<tr>
<td><code>weibull(a,b,x)</code></td>
<td>the cumulative Weibull distribution with shape <code>a</code> and scale <code>b</code></td>
</tr>
<tr>
<td><code>weibull(a,b,g,x)</code></td>
<td>the cumulative Weibull distribution with shape <code>a</code>, scale <code>b</code>, and location <code>g</code></td>
</tr>
<tr>
<td><code>weibullden(a,b,x)</code></td>
<td>the probability density function of the Weibull distribution with shape <code>a</code> and scale <code>b</code></td>
</tr>
<tr>
<td><code>weibullden(a,b,g,x)</code></td>
<td>the probability density function of the Weibull distribution with shape <code>a</code>, scale <code>b</code>, and location <code>g</code></td>
</tr>
<tr>
<td><code>weibullph(a,b,x)</code></td>
<td>the cumulative Weibull (proportional hazards) distribution with shape <code>a</code> and scale <code>b</code></td>
</tr>
<tr>
<td><code>weibullph(a,b,g,x)</code></td>
<td>the cumulative Weibull (proportional hazards) distribution with shape <code>a</code>, scale <code>b</code>, and location <code>g</code></td>
</tr>
<tr>
<td><code>weibullphden(a,b,x)</code></td>
<td>the probability density function of the Weibull (proportional hazards) distribution with shape <code>a</code> and scale <code>b</code></td>
</tr>
<tr>
<td><code>weibullphden(a,b,g,x)</code></td>
<td>the probability density function of the Weibull (proportional hazards) distribution with shape <code>a</code>, scale <code>b</code>, and location <code>g</code></td>
</tr>
</tbody>
</table>
### Functions by name

- **weibullphtail**(a, b, x)  
  the reverse cumulative Weibull (proportional hazards) distribution with shape \(a\) and scale \(b\)

- **weibullphtail**(a, b, g, x)  
  the reverse cumulative Weibull (proportional hazards) distribution with shape \(a\), scale \(b\), and location \(g\)

- **weibulltail**(a, b, x)  
  the reverse cumulative Weibull distribution with shape \(a\) and scale \(b\)

- **weibulltail**(a, b, g, x)  
  the reverse cumulative Weibull distribution with shape \(a\), scale \(b\), and location \(g\)

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

- **word**(s, n)  
  the \(n\)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 \(s\)

- **year**(e_d)  
  the numeric year corresponding to date \(e_d\)

- **yearly**(s_1, s_2[, Y])  
  the \(e_y\) yearly date (year) corresponding to \(s_1\) based on \(s_2\) and \(Y\); \(Y\) specifies topyear; see **date()**

- **yh**(Y, H)  
  the \(e_h\) half-yearly date (half-years since 1960h1) corresponding to year \(Y\), half-year \(H\)

- **ym**(Y, M)  
  the \(e_m\) monthly date (months since 1960m1) corresponding to year \(Y\), month \(M\)

- **yofd**(e_d)  
  the \(e_y\) yearly date (year) containing date \(e_d\)

- **yq**(Y, Q)  
  the \(e_q\) quarterly date (quarters since 1960q1) corresponding to year \(Y\), quarter \(Q\)

- **yw**(Y, W)  
  the \(e_w\) weekly date (weeks since 1960w1) corresponding to year \(Y\), week \(W\)

### Also see

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

Date and time functions

<table>
<thead>
<tr>
<th>Contents</th>
<th>Functions</th>
<th>Remarks and examples</th>
<th>Methods and formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>Also see</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Contents

- **age**(*e_dDOB*, *e_d*, *s_nl*)
  - the age in integer years on *e_d* for date of birth *e_dDOB* with *s_nl* the nonleap-year birthday for 29feb birthdates

- **age_frac**(*e_dDOB*, *e_d*, *s_nl*)
  - the age in years, including the fractional part, on *e_d* for date of birth *e_dDOB* with *s_nl* the nonleap-year birthday for 29feb birthdates

- **birthday**(*e_dDOB*, *Y*, *s_nl*)
  - the *e_d* date of the birthday in year *Y* for date of birth *e_dDOB* with *s_nl* the nonleap-year birthday for 29feb birthdates

- **bofd**("cal", *e_d*)
  - the *e_b* business date corresponding to *e_d*

- **Cdhms**(*e_d*, *h*, *m*, *s*)
  - the *e_tC* datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to *e_d*, *h*, *m*, *s*

- **Chms**(*h*, *m*, *s*)
  - the *e_tC* datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to *h*, *m*, *s* on 01jan1960

- **Clock**(*s_1*, *s_2*, *Y*)
  - the *e_tC* datetime (ms. since 01jan1960 00:00:00.000) corresponding to *s_1* based on *s_2* and *Y*

- **clockdiff**(*e_tc1*, *e_tc2*, *s_u*)
  - the *e_tC* datetime difference, rounded down to an integer, from *e_tc1* to *e_tc2* in *s_u* units of days, hours, minutes, seconds, or milliseconds

- **Clockdiff_frac**(*e_tc1*, *e_tc2*, *s_u*)
  - the *e_tC* datatetime difference, including the fractional part, from *e_tc1* to *e_tc2* in *s_u* units of days, hours, minutes, seconds, or milliseconds

- **Cmdyhms**(*M*, *D*, *Y*, *h*, *m*, *s*)
  - the *e_tC* datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to *M*, *D*, *Y*, *h*, *m*, *s*

- **Cofc**(*e_tc*)
  - the *e_tC* datetime (ms. without leap seconds since 01jan1960 00:00:00.000) of *e_tc* (ms. with leap seconds since 01jan1960 00:00:00.000)

- **cofC**(*e_tc*)
  - the *e_tc* datatetime (ms. without leap seconds since 01jan1960 00:00:00.000) of *e_tc* (ms. with leap seconds since 01jan1960 00:00:00.000)
Cofd($e_d$)
the \textit{e}_{tc} datetime (ms. with leap seconds since 01jan1960 00:00:00:000) of date $e_d$ at time 00:00:00.000

cofd($e_d$)
the \textit{e}_{tc} datetime (ms. since 01jan1960 00:00:00:000) of date $e_d$ at time 00:00:00.000

daily($s_1,s_2[ , Y ]$)
a synonym for \textit{date}(s_1,s_2[ , Y ])
date($s_1,s_2[ , Y ]$)
the $e_d$ date (days since 01jan1960) corresponding to $s_1$ based on $s_2$ and $Y$
datediff($e_{d1},e_{d2},s_u[ , s_n l ]$)
the difference, rounded down to an integer, from $e_{d1}$ to $e_{d2}$ in $s_u$ units of days, months, or years with $s_n l$ the nonleap-year anniversary for $e_{d1}$ on 29feb

datediff\_frac($e_{d1},e_{d2},s_u[ , s_n l ]$)
the difference, including the fractional part, from $e_{d1}$ to $e_{d2}$ in $s_u$ units of days, months, or years with $s_n l$ the nonleap-year anniversary for $e_{d1}$ on 29feb

datepart($e_d,s_u$)
the integer year, month, or day of $e_d$ with $s_u$ specifying year, month, or day
day($e_d$)
the numeric day of the month corresponding to $e_d$
daysinmonth($e_d$)
the number of days in the month of $e_d$
dayssincecwday($e_d,d$) a synonym for \textit{dayssinceweekday}($e_d,d$)
dayssinceweekday($e_d,d$)
the number of days until $e_d$ since previous day-of-week $d$
daysuntildow($e_d,d$) a synonym for \textit{daysuntilweekday}($e_d,d$)
daysuntilow($e_d,d$) the number of days from $e_d$ until next day-of-week $d$
dhms($e_d,h,m,s$)
the \textit{e}_{tc} datetime (ms. since 01jan1960 00:00:00:000) corresponding to $e_d$, $h$, $m$, and $s$
dmy($D,M,Y$)
the $e_d$ date (days since 01jan1960) corresponding to $D$, $M$, $Y$
dofb($e_b,"cal"$)
the $e_d$ date corresponding to $e_b$
dofC($e_{tC}$)
the $e_d$ date (days since 01jan1960) of datatime $e_{tC}$ (ms. with leap seconds since 01jan1960 00:00:00.000)
dofc($e_{tc}$)
the $e_d$ date (days since 01jan1960) of datatime $e_{tc}$ (ms. since 01jan1960 00:00:00.000)
dofh($e_h$)
the $e_d$ date (days since 01jan1960) of the start of half-year $e_h$
dofm($e_m$)
the $e_d$ date (days since 01jan1960) of the start of month $e_m$
dofq($e_q$)
the $e_d$ date (days since 01jan1960) of the start of quarter $e_q$
dofw($e_w$)
the $e_d$ date (days since 01jan1960) of the start of week $e_w$
dofy($e_y$)
the $e_d$ date (days since 01jan1960) of 01jan in year $e_y$
dow($e_d$) the numeric day of the week corresponding to date $e_d$; 0 = Sunday, 1 = Monday, \ldots, 6 = Saturday
doy($e_d$)
the numeric day of the year corresponding to date $e_d$
firstdayofmonth($e_d$)
the $e_d$ date of the first day of the month of $e_d$
firstdowofmonth($M,Y,d$) a synonym for \textit{firstweekdayofmonth}($M,Y,d$)
firstweekdayofmonth($M,Y,d$)
the $e_d$ date of the first day-of-week $d$ in month $M$ of year $Y$
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 \textit{topyear}; see \textit{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 $e_{d}$ half-yearly date (half years since 1960h1) containing date $e_{d}$

hours($ms$) $ms/3,600,000$

isleapsecond($e_{tC}$) 1 if $e_{tC}$ is a leap second; otherwise, 0

isleapyear($Y$) 1 if $Y$ is a leap year; otherwise, 0

lastdowofmonth($M,Y,d$) a synonym for lastweekdayofmonth($M,Y,d$)

lastweekdayofmonth($M,Y,d$) the $e_{d}$ date of the last day-of-week $d$ in month $M$ of year $Y$

mdy($M,D,Y$) the $e_{d}$ date (days since 01jan1960) corresponding to $M$, $D$, $Y$

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

minutes($ms$) $ms/60,000$

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

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

mofd($e_{d}$) the $e_{m}$ monthly date (months since 1960m1) containing date $e_{d}$

month($e_{d}$) the numeric month corresponding to date $e_{d}$

monthly($s_{1},s_{2}[,Y]$) the $e_{m}$ monthly date (months since 1960m1) corresponding to $s_{1}$ based on $s_{2}$ and $Y$; $Y$ specifies topyear; see date()

msofhours($h$) $h \times 3,600,000$

msofminutes($m$) $m \times 60,000$

msofseconds($s$) $s \times 1,000$

nextbirthday($e_{dDOB},e_{d}[[,s_{nl}]])$ the $e_{d}$ date of the first birthday after $e_{d}$ for date of birth $e_{dDOB}$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates

nextdow($e_{d},d$) a synonym for nextweekday($e_{d},d$)

nextleapyear($Y$) the first leap year after year $Y$

nextweekday($e_{d},d$) the $e_{d}$ date of the first day-of-week $d$ after $e_{d}$

now() the current $e_{tc}$ datetime

previousbirthday($e_{dDOB},e_{d}[[,s_{nl}]])$ the $e_{d}$ date of the birthday immediately before $e_{d}$ for date of birth $e_{dDOB}$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates

previousdow($e_{d},d$) a synonym for previousweekday($e_{d},d$)

previousleapyear($Y$) the leap year immediately before year $Y$

previousweekday($e_{d},d$) the $e_{d}$ date of the last day-of-week $d$ before $e_{d}$

qofd($e_{d}$) the $e_{q}$ quarterly date (quarters since 1960q1) containing date $e_{d}$

quarter($e_{d}$) the numeric quarter of the year corresponding to date $e_{d}$
quarterly($s_1, s_2[, Y]$) the $e_q$ quarterly date (quarters since 1960q1) corresponding to $s_1$
  based on $s_2$ and $Y$; $Y$ specifies topyear; see date()

seconds($ms$) $ms/1,000$

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)

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

today() today’s $e_d$ date

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 $W$
Date and time functions

Stata’s date and time functions are described with examples in [U] 25 Working with dates and times, [D] Datetime, [D] Datetime durations, and [D] Datetime relative dates. What follows is a technical description. We use the following notation:

- $e_b$ %tb business calendar date (days)
- $e_{tc}$ %tc encoded datetime (ms. since 01jan1960 00:00:00.000)
- $e_{TC}$ %tC encoded datetime (ms. with leap seconds since 01jan1960 00:00:00.000)
- $e_d$ %td encoded date (days since 01jan1960)
- $e_w$ %tw encoded weekly date (weeks since 1960w1)
- $e_m$ %tm encoded monthly date (months since 1960m1)
- $e_q$ %tq encoded quarterly date (quarters since 1960q1)
- $e_h$ %th encoded half-yearly date (half-years since 1960h1)
- $e_y$ %ty encoded yearly date (years)
- $M$ month, 1–12
- $D$ day of month, 1–31
- $Y$ year, 0100–9999
- $h$ hour, 0–23
- $m$ minute, 0–59
- $s$ second, 0–59 or 60 if leap seconds
- $ms$ milliseconds
- $W$ week number, 1–52
- $Q$ quarter number, 1–4
- $H$ half-year number, 1 or 2
- $d$ numeric day of the week, 0 = Sunday, 1 = Monday, . . . , 6 = Saturday

The date and time functions, where integer arguments are required, allow noninteger values and use the floor() of the value.

A Stata date-and-time variable is recorded as the number of milliseconds, days, weeks, etc., depending upon the units, from 01jan1960. Negative values indicate dates and times before 01jan1960. Allowable dates and times are those between 01jan0100 and 31dec9999, inclusive, but all functions are based on the Gregorian calendar, and values do not correspond to historical dates before Friday, 15oct1582.

**age($e_d$DOB, $e_d$[, $s_{nl}$])**

Description: the age in integer years on $e_d$ for date of birth $e_d$DOB with $s_{nl}$ the nonleap-year birthday for 29feb birthdates

$s_{nl}$ specifies when someone born on 29feb becomes another year older in nonleap years. $s_{nl}$ = "01mar" (the default) means the birthday is taken to be 01mar. $s_{nl}$ = "28feb" means the birthday is taken to be 28feb. See Methods and formulas.

When $e_d < e_d$DOB, the result is missing.

Domain $e_d$DOB: $e_d$ dates 01jan0101 to 31dec9998 (integers −678,985 to 2,936,184)

Domain $e_d$: $e_d$ dates 01jan0101 to 31dec9998 (integers −678,985 to 2,936,184)

Domain $s_{nl}$: strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)

Range: integers 0 to 9897 or missing
**age_frac(e_dDOB, e_d[, s_nl])**

Description: the age in years, including the fractional part, on e_d for date of birth e_dDOB with s_nl the nonleap-year birthday for 29feb birthdates

s_nl specifies when someone born on 29feb becomes another year older in nonleap years. s_nl = "01mar" (the default) means the birthday is taken to be 01mar. s_nl = "28feb" means the birthday is taken to be 28feb. See Methods and formulas.

When e_d < e_dDOB, the result is missing.

**Domain e_dDOB:** e_d dates 01jan0100 to 31dec9999 (integers -679,350 to 2,936,549)

**Domain e_d:** e_d dates 01jan0100 to 31dec9999 (integers -679,350 to 2,936,549)

**Domain s_nl:** strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)

**Range:** reals 0 to 9897.997... or missing

**birthday(e_dDOB,Y[, s_nl])**

Description: the e_d date of the birthday in year Y for date of birth e_dDOB with s_nl the nonleap-year birthday for 29feb birthdates

s_nl specifies when someone born on 29feb becomes another year older in nonleap years. s_nl = "01mar" (the default) means the birthday is taken to be 01mar. s_nl = "28feb" means the birthday is taken to be 28feb. See Methods and formulas.

**Domain e_dDOB:** e_d dates 01jan0100 to 31dec9999 (integers -679,350 to 2,936,549)

**Domain Y:** integers 0100 to 9999 (but probably 1800 to 2100)

**Domain s_nl:** strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)

**Range:** e_d dates 01jan0100 to 31dec9999 (integers -679,350 to 2,936,549) or missing

**bofd("cal", e_d)**

Description: the e_b business date corresponding to e_d

**Domain cal:** business calendar names and formats

**Domain e_d:** e_d as defined by business calendar named cal

**Range:** as defined by business calendar named cal

**Cdhms(e_d, h, m, s)**

Description: the e_tC datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to e_d, h, m, s

**Domain e_d:** e_d dates 01jan0100 to 31dec9999 (integers -679,350 to 2,936,549)

**Domain h:** integers 0 to 23

**Domain m:** integers 0 to 59

**Domain s:** reals 0.000 to 60.999

**Range:** e_tC datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers -58,695,840,000,000 to 253,717,919,999,999 + number of leap seconds) or missing
Chms\((h,m,s)\)
Description: the \(e_{tC}\) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to \(h, m, s\) on 01jan1960
Domain \(h\): integers 0 to 23
Domain \(m\): integers 0 to 59
Domain \(s\): reals 0.000 to 60.999
Range: \(e_{tC}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds) or missing

Clock\((s_1,s_2[,Y])\)
Description: the \(e_{tC}\) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to \(s_1\) based on \(s_2\) and \(Y\)

Function Clock() works the same as function clock() except that Clock() returns a leap second–adjusted \(t_{C}\) value rather than an unadjusted \(t_c\) value. Use Clock() only if original time values have been adjusted for leap seconds.

Domain \(s_1\): strings
Domain \(s_2\): strings
Domain \(Y\): integers 1000 to 9998 (but probably 2001 to 2099)
Range: \(e_{tC}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds) or missing

clock\((s_1,s_2[,Y])\)
Description: the \(e_{tc}\) datetime (ms. since 01jan1960 00:00:00.000) corresponding to \(s_1\) based on \(s_2\) and \(Y\)

\(s_1\) contains the date, time, or both, recorded as a string, in virtually any format. Months can be spelled out, abbreviated (to three characters), or indicated as numbers; years can include or exclude the century; blanks and punctuation are allowed.

\(s_2\) is any permutation of M, D, [##]Y, h, m, and s, with their order defining the order that month, day, year, hour, minute, and second occur (and whether they occur) in \(s_1\). ##, if specified, indicates the default century for two-digit years in \(s_1\). For instance, \(s_2 = "MD19Y hm"\) would translate \(s_1 = "11/15/91 21:14"\) as 15nov1991 21:14. The space in "MD19Y hm" was not significant and the string would have translated just as well with "MD19Yhm".

\(Y\) provides an alternate way of handling two-digit years. \(Y\) specifies the largest year that is to be returned when a two-digit year is encountered; see function date() below. If neither ## nor \(Y\) is specified, clock() returns missing when it encounters a two-digit year.

Domain \(s_1\): strings
Domain \(s_2\): strings
Domain \(Y\): integers 1000 to 9998 (but probably 2001 to 2099)
Range: \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\)) or missing
Clockdiff($e_{tC1}, e_{tC2}, s_u$)
Description: the $e_{tC}$ datetime difference, rounded down to an integer, from $e_{tC1}$ to $e_{tC2}$ in $s_u$ units of days, hours, minutes, seconds, or milliseconds
Note that $\text{Clockdiff}(e_{tC1}, e_{tC2}, s_u) = -\text{Clockdiff}(e_{tC2}, e_{tC1}, s_u)$.
Domain $e_{tC1}$: $e_{tC}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds)
Domain $e_{tC2}$: $e_{tC}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds)
Domain $s_u$: strings "day" or "d" for day; "hour" or "h" for hour; "minute", "min", or "m" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)
Range: integers $-312,413,759,999,999$ to $312,413,759,999,999$ or $missing$

clockdiff($e_{tc1}, e_{tc2}, s_u$)
Description: the $e_{tc}$ datetime difference, rounded down to an integer, from $e_{tc1}$ to $e_{tc2}$ in $s_u$ units of days, hours, minutes, seconds, or milliseconds
Note that $\text{clockdiff}(e_{tc1}, e_{tc2}, s_u) = -\text{clockdiff}(e_{tc2}, e_{tc1}, s_u)$.
Domain $e_{tc1}$: $e_{tc}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$)
Domain $e_{tc2}$: $e_{tc}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$)
Domain $s_u$: strings "day" or "d" for day; "hour" or "h" for hour; "minute", "min", or "m" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)
Range: integers $-312,413,759,999,999$ to $312,413,759,999,999$ or $missing$

Clockdiff\_frac($e_{tC1}, e_{tC2}, s_u$)
Description: the $e_{tC}$ datetime difference, including the fractional part, from $e_{tC1}$ to $e_{tC2}$ in $s_u$ units of days, hours, minutes, seconds, or milliseconds
Note that $\text{Clockdiff\_frac}(e_{tC1}, e_{tC2}, s_u) = -\text{Clockdiff\_frac}(e_{tC2}, e_{tC1}, s_u)$.
Domain $e_{tC1}$: $e_{tC}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds)
Domain $e_{tC2}$: $e_{tC}$ datetimes 01jan0000 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds)
Domain $s_u$: strings "day" or "d" for day; "hour" or "h" for hour; "minute", "min", or "m" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)
Range: reals $-312,413,759,999,999$ to $312,413,759,999,999$ or $missing$
clockdiff_frac(e_{tc1}, e_{tc2}, s_u)

Description: the $e_{tc}$ datetime difference, including the fractional part, from $e_{tc1}$ to $e_{tc2}$ in $s_u$ units of days, hours, minutes, seconds, or milliseconds

Note that

\[
\text{clockdiff}_{\text{frac}}(e_{tc1}, e_{tc2}, s_u) = -\text{clockdiff}_{\text{frac}}(e_{tc2}, e_{tc1}, s_u).
\]

Domain $e_{tc1}$: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$)

Domain $e_{tc2}$: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$)

Domain $s_u$: strings "day" or "d" for day; "hour" or "h" for hour; "minute", "min", or "m" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)

Range: reals $-312,413,759,999,999$ to $312,413,759,999,999$ or missing

Clockpart($e_{tc}$, $s_u$)

Description: the integer year, month, day, hour, minute, second, or millisecond of $e_{tc}$ with $s_u$ specifying which time part

Domain $e_{tc}$: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds)

Domain $s_u$: strings "year" or "y" for year; "month" or "mon" for month; "day" or "d" for day; "hour" or "h" for hour; "minute" or "min" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)

Range: integers 0 to 9999 or missing

clockpart($e_{tc}$, $s_u$)

Description: the integer year, month, day, hour, minute, second, or millisecond of $e_{tc}$ with $s_u$ specifying which time part

Domain $e_{tc}$: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$)

Domain $s_u$: strings "year" or "y" for year; "month" or "mon" for month; "day" or "d" for day; "hour" or "h" for hour; "minute" or "min" for minute; "second", "sec", or "s" for second; and "millisecond" or "ms" for millisecond (case insensitive)

Range: integers 0 to 9999 or missing
Cmdyhms (M, D, Y, h, m, s)

Description: the $et_C$ datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to $M$, D, Y, h, m, s

Domain $M$: integers 1 to 12
Domain $D$: integers 1 to 31
Domain $Y$: integers 0100 to 9999 (but probably 1800 to 2100)
Domain $h$: integers 0 to 23
Domain $m$: integers 0 to 59
Domain $s$: reals 0.000 to 60.999

Range: $et_C$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$ + number of leap seconds) or missing
Cofc\((e_{tc})\)

Description: the \(e_{tc}\) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of \(e_{tc}\) (ms. without leap seconds since 01jan1960 00:00:00.000)

Domain \(e_{tc}\): \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\))

Range: \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds)

cofC\((e_{tc})\)

Description: the \(e_{tc}\) datetime (ms. without leap seconds since 01jan1960 00:00:00.000) of \(e_{tc}\) (ms. with leap seconds since 01jan1960 00:00:00.000)

Domain \(e_{tc}\): \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds)

Range: \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds)

Cofd\((e_d)\)

Description: the \(e_{tc}\) datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date \(e_d\) at time 00:00:00.000

Domain \(e_d\): \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

Range: \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds)

cofd\((e_d)\)

Description: the \(e_{tc}\) datetime (ms. since 01jan1960 00:00:00.000) of date \(e_d\) at time 00:00:00.000

Domain \(e_d\): \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

Range: \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\))

daily\((s_1,s_2]\ [,Y\])\)

Description: a synonym for date\((s_1,s_2]\ [,Y\])\)

\texttt{date(s1[,s2,Y])}

Description: the ed date (days since 01jan1960) corresponding to \( s_1 \) based on \( s_2 \) and \( Y \)

\( s_1 \) contains the date, recorded as a string, in virtually any format. Months can be spelled out, abbreviated (to three characters), or indicated as numbers; years can include or exclude the century; blanks and punctuation are allowed.

\( s_2 \) is any permutation of M, D, and [##]Y, with their order defining the order that month, day, and year occur in \( s_1 \). #, if specified, indicates the default century for two-digit years in \( s_1 \). For instance, \( s_2 = "MD19Y" \) would translate \( s_1 = "11/15/91" \) as 15nov1991.

\( Y \) provides an alternate way of handling two-digit years. When a two-digit year is encountered, the largest year, \texttt{topyear}, that does not exceed \( Y \) is returned.

\begin{align*}
\text{date}("1/15/08","MDY",1999) &= 15jan1908 \\
\text{date}("1/15/08","MDY",2019) &= 15jan2008 \\
\text{date}("1/15/51","MDY",2000) &= 15jan1951 \\
\text{date}("1/15/50","MDY",2000) &= 15jan1950 \\
\text{date}("1/15/49","MDY",2000) &= 15jan1949 \\
\text{date}("1/15/01","MDY",2050) &= 15jan2001 \\
\text{date}("1/15/00","MDY",2050) &= 15jan2000
\end{align*}

If neither # nor \( Y \) is specified, \texttt{date()} returns \texttt{missing} when it encounters a two-digit year. See Working with two-digit years in [D] Datetime conversion for more information.

Domain \( s_1 \): strings

Domain \( s_2 \): strings

Domain \( Y \): integers 1000 to 9998 (but probably 2001 to 2099)

Range: \( e_d \) dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549) or \texttt{missing}

\texttt{datediff(e_d1,e_d2,s_u[,s_nl])}

Description: the difference, rounded down to an integer, from \( e_d1 \) to \( e_d2 \) in \( s_u \) units of days, months, or years with \( s_nl \) the nonleap-year anniversary for \( e_d1 \) on 29feb

\( s_nl \) specifies the anniversary when \( e_d1 \) is on 29feb. \( s_nl = "01mar" \) (the default) means the anniversary is taken to be 01mar. \( s_nl = "28feb" \) means the anniversary is taken to be 28feb. See Methods and formulas.

Note that \texttt{datediff}(\( e_d1,e_d2,s_u,s_nl \)) = −\texttt{datediff}(\( e_d2,e_d1,s_u,s_nl \)).

Domain \( e_d1 \): \( e_d \) dates 01jan0101 to 31dec9999 (integers −678,985 to 2,936,184)

Domain \( e_d2 \): \( e_d \) dates 01jan0101 to 31dec9999 (integers −678,985 to 2,936,184)

Domain \( s_u \): strings "day" or "d" for day; "month", "mon", or "m" for month; and "year" or "y" for year (case insensitive)

Domain \( s_nl \): strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)

Range: integers −3,615,169 to 3,615,169 or \texttt{missing}
datediff_frac($e_{d1}, e_{d2}, s_u[, s_{nl}]$)
Description: the difference, including the fractional part, from $e_{d1}$ to $e_{d2}$ in $s_u$ units of days, months, or years with $s_{nl}$ the nonleap-year anniversary for $e_{d1}$ on 29feb
$s_{nl}$ specifies the anniversary when $e_{d1}$ is on 29feb. $s_{nl} =$ "01mar" (the default) means the anniversary is taken to be 01mar. $s_{nl} =$ "28feb" means the anniversary is taken to be 28feb. See Methods and formulas.
Note that $\text{datediff\_frac}(e_{d1}, e_{d2}, s_u, s_{nl}) = -\text{datediff\_frac}(e_{d2}, e_{d1}, s_u, s_{nl})$.
Domain $e_{d1}$: $e_d$ dates 01jan0101 to 31dec9998 (integers $-678,985$ to $2,936,184$)
Domain $e_{d2}$: $e_d$ dates 01jan0101 to 31dec9998 (integers $-678,985$ to $2,936,184$)
Domain $s_u$: strings "day" or "d" for day; "month", "mon", or "m" for month; and "year" or "y" for year (case insensitive)
Domain $s_{nl}$: strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)
Range: reals $-3,615,169$ to $3,615,169$ or missing

datepart($e_d, s_u$)
Description: the integer year, month, or day of $e_d$ with $s_u$ specifying year, month, or day
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $s_u$: strings "day" or "d" for day; "month", "mon", or "m" for month; and "year" or "y" for year (case insensitive)
Range: integers 1 to 9999 or missing

day($e_d$)
Description: the numeric day of the month corresponding to $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Range: integers 1 to 31 or missing

daysinmonth($e_d$)
Description: the number of days in the month of $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Range: integers 28 to 31 or missing

dayssincegedow($e_d, d$)
Description: a synonym for dayssinceweekday($e_d, d$)

dayssinceweekday($e_d, d$)
Description: the number of days until $e_d$ since previous day-of-week $d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $d$: integers 0 to 6 (0=Sunday, 1=Monday, ..., 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)
Range: integers 1 to 7 or missing

daysuntildow($e_d, d$)
Description: a synonym for daysuntilweekday($e_d, d$)
daysuntilweekday\((e_d,d)\)
Description: the number of days from \(e_d\) until next day-of-week \(d\)
Domain \(e_d\): \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to 2,936,549)
Domain \(d\): integers 0 to 6 (0=Sunday, 1=Monday, \ldots, 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)
Range: integers 1 to 7 or missing

dhms\((e_d,h,m,s)\)
Description: the \(e_d\) datetime (ms. since 01jan1960 00:00:00.000) corresponding to \(e_d, h, m,\) and \(s\)
Domain \(e_d\): \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to 2,936,549)
Domain \(h\): integers 0 to 23
Domain \(m\): integers 0 to 59
Domain \(s\): reals 0.000 to 59.999
Range: \(e_d\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to 253,717,919,999,999) or missing

dmy\((D,M,Y)\)
Description: the \(e_d\) date (days since 01jan1960) corresponding to \(D, M, Y\)
Domain \(D\): integers 1 to 31
Domain \(M\): integers 1 to 12
Domain \(Y\): integers 0100 to 9999 (but probably 1800 to 2100)
Range: \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to 2,936,549) or missing

dофb\((e_b,"cal")\)
Description: the \(e_d\) datetime corresponding to \(e_b\)
Domain \(e_b\): \(e_b\) as defined by business calendar named \(cal\)
Domain \(cal\): business calendar names and formats
Range: as defined by business calendar named \(cal\)

dофC\((e_{tc})\)
Description: the \(e_d\) date (days since 01jan1960) of datetime \(e_{tc}\) (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain \(e_{tc}\): \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to 253,717,919,999,999+number of leap seconds)
Range: \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to 2,936,549)

dофc\((e_{tc})\)
Description: the \(e_d\) date (days since 01jan1960) of datetime \(e_{tc}\) (ms. since 01jan1960 00:00:00.000)
Domain \(e_{tc}\): \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to 253,717,919,999,999)
Range: \(e_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to 2,936,549)
dofh\( (e_h) \)
Description: the \( e_d \) date (days since 01jan1960) of the start of half-year \( e_h \)
Domain \( e_h \): \( e_h \) dates 0100h1 to 9999h2 (integers −3,720 to 16,079)
Range: \( e_d \) dates 01jan0100 to 01jul9999 (integers −679,350 to 2,936,366)

dofm\( (e_m) \)
Description: the \( e_d \) date (days since 01jan1960) of the start of month \( e_m \)
Domain \( e_m \): \( e_m \) dates 0100m1 to 9999m12 (integers −22,320 to 96,479)
Range: \( e_d \) dates 01jan0100 to 01dec9999 (integers −679,350 to 2,936,519)

dofq\( (e_q) \)
Description: the \( e_d \) date (days since 01jan1960) of the start of quarter \( e_q \)
Domain \( e_q \): \( e_q \) dates 0100q1 to 9999q4 (integers −7,440 to 32,159)
Range: \( e_d \) dates 01jan0100 to 01oct9999 (integers −679,350 to 2,936,458)

dofw\( (e_w) \)
Description: the \( e_d \) date (days since 01jan1960) of the start of week \( e_w \)
Domain \( e_w \): \( e_w \) dates 0100w1 to 9999w52 (integers −96,720 to 418,079)
Range: \( e_d \) dates 01jan0100 to 24dec9999 (integers −679,350 to 2,936,542)

dofy\( (e_y) \)
Description: the \( e_d \) date (days since 01jan1960) of 01jan in year \( e_y \)
Domain \( e_y \): \( e_y \) dates 0100 to 9999 (integers 0100 to 9999)
Range: \( e_d \) dates 01jan0100 to 01jan9999 (integers −679,350 to 2,936,185)

dow\( (e_d) \)
Description: the numeric day of the week corresponding to date \( e_d \); 0 = Sunday, 1 = Monday, \ldots, 6 = Saturday
Domain \( e_d \): \( e_d \) dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 0 to 6 or missing

doy\( (e_d) \)
Description: the numeric day of the year corresponding to date \( e_d \)
Domain \( e_d \): \( e_d \) dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 366 or missing

firstdayofmonth\( (e_d) \)
Description: the \( e_d \) date of the first day of the month of \( e_d \)
Domain \( e_d \): \( e_d \) dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: \( e_d \) dates 01jan0100 to 01dec9999 (integers −679,350 to 2,936,519) or missing

firstdowofmonth\( (M,Y,d) \)
Description: a synonym for firstweekdayofmonth\( (M,Y,d) \)
54 Date and time functions

firstweekdayofmonth\((M, Y, d)\)
Description: the \(e_d\) date of the first day-of-week \(d\) in month \(M\) of year \(Y\)
Domain \(M\): integers 1 to 12
Domain \(Y\): integers 0100 to 9999 (but probably 1800 to 2100)
Domain \(d\): integers 0 to 6 (0=Sunday, 1=Monday, ..., 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)
Range: \(e_d\) dates 01jan0100 to 07dec9999 (integers −679,350 to 2,936,525) or missing

halfyear\((e_d)\)
Description: the numeric half of the year corresponding to date \(e_d\)
Domain \(e_d\): \(e_d\) dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1, 2, or missing

halfyearly\((s_1, s_2[, Y])\)
Description: the \(e_h\) half-yearly date (half-years since 1960h1) corresponding to \(s_1\) based on \(s_2\) and \(Y\); \(Y\) specifies toyear; see date()
Domain \(s_1\): strings
Domain \(s_2\): strings "HY" and "YH"; \(Y\) may be prefixed with ##
Domain \(Y\): integers 1000 to 9998 (but probably 2001 to 2099)
Range: \(e_h\) dates 0100h1 to 9999h2 (integers −3,720 to 16,079) or missing

hh\((e_{tc})\)
Description: the hour corresponding to datetime \(e_{tc}\) (ms. since 01jan1960 00:00:00.000)
Domain \(e_{tc}\): \(e_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999)
Range: integers 0 through 23 or missing

hhC\((e_{tC})\)
Description: the hour corresponding to datetime \(e_{tC}\) (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain \(e_{tC}\): \(e_{tC}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999+number of leap seconds)
Range: integers 0 through 23 or missing

hms\((h, m, s)\)
Description: the \(e_{tc}\) datetime (ms. since 01jan1960 00:00:00.000) corresponding to \(h, m, s\) on 01jan1960
Domain \(h\): integers 0 to 23
Domain \(m\): integers 0 to 59
Domain \(s\): reals 0.000 to 59.999
Range: datetimes 01jan1960 00:00:00.000 to 01jan1960 23:59:59.999 (integers 0 to 86,399,999 or missing)
**holf**($e_d$)  
**Description:** the $e_h$ half-yearly date (half years since 1960h1) containing date $e_d$  
**Domain** $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)  
**Range:** $e_h$ dates 0100h1 to 9999h2 (integers $-3,720$ to $16,079$)

**hours**($ms$)  
**Description:** $ms/3,600,000$  
**Domain** $ms$: real; milliseconds  
**Range:** real or **missing**

**isleapsecond**($e_{tC}$)  
**Description:** 1 if $e_{tC}$ is a leap second; otherwise, 0  
**Domain** $e_{tC}$: $e_{tC}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$+number of leap seconds)  
**Range:** 0, 1, or **missing**

**isleapyear**($Y$)  
**Description:** 1 if $Y$ is a leap year; otherwise, 0  
**Domain** $Y$: integers 0100 to 9999 (but probably 1800 to 2100)  
**Range:** 0, 1, or **missing**

**lastdayofmonth**($e_d$)  
**Description:** the $e_d$ date of the last day of the month of $e_d$  
**Domain** $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)  
**Range:** $e_d$ dates 31jan0100 to 31dec9999 (integers $-679,320$ to $2,936,549$) or **missing**

**lastdowofmonth**($M, Y, d$)  
**Description:** a synonym for **lastweekdayofmonth**($M, Y, d$)

**lastweekdayofmonth**($M, Y, d$)  
**Description:** the $e_d$ date of the last day-of-week $d$ in month $M$ of year $Y$  
**Domain** $M$: integers 1 to 12  
**Domain** $Y$: integers 0100 to 9999 (but probably 1800 to 2100)  
**Domain** $d$: integers 0 to 6 (0=Sunday, 1=Monday, ..., 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)  
**Range:** $e_d$ dates 25jan0100 to 31dec9999 (integers $-679,326$ to $2,936,549$) or **missing**

**mdy**($M, D, Y$)  
**Description:** the $e_d$ date (days since 01jan1960) corresponding to $M$, $D$, $Y$  
**Domain** $M$: integers 1 to 12  
**Domain** $D$: integers 1 to 31  
**Domain** $Y$: integers 0100 to 9999 (but probably 1800 to 2100)  
**Range:** $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$) or **missing**
**mdyhms**\((M, D, Y, h, m, s)\)

**Description:**
the e\(_{tc}\) datetime (ms. since 01jan1960 00:00:00.000) corresponding to \(M, D, Y, h, m, s\)

**Domain**
- \(M\): integers 1 to 12
- \(D\): integers 1 to 31
- \(Y\): integers 0100 to 9999 (but probably 1800 to 2100)
- \(h\): integers 0 to 23
- \(m\): integers 0 to 59
- \(s\): reals 0.000 to 59.999

**Range:**
e\(_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers \(-58,695,840,000,000\) to \(253,717,919,999,999\)) or missing

**minutes**\((ms)\)

**Description:**
\(ms/60,000\)

**Domain**
ms: real; milliseconds

**Range:**
real or missing

**mm**\((e\(_{tc}\))\)

**Description:**
the minute corresponding to datetime e\(_{tc}\) (ms. since 01jan1960 00:00:00.000)

**Domain**
e\(_{tc}\):
e\(_{tc}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers \(-58,695,840,000,000\) to \(253,717,919,999,999\))

**Range:**
integers 0 through 59 or missing

**mmC**\((e\(_{tC}\))\)

**Description:**
the minute corresponding to datetime e\(_{tC}\) (ms. with leap seconds since 01jan1960 00:00:00.000)

**Domain**
e\(_{tC}\):
e\(_{tC}\) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers \(-58,695,840,000,000\) to \(253,717,919,999,999\) + number of leap seconds)

**Range:**
integers 0 through 59 or missing

**mofd**\((e_d)\)

**Description:**
the e\(_m\) monthly date (months since 1960m1) containing date e\(_d\)

**Domain**
e\(_d\):
e\(_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

**Range:**
e\(_m\) dates 0100m1 to 9999m12 (integers \(-22,320\) to \(96,479\))

**month**\((e_d)\)

**Description:**
the numeric month corresponding to date e\(_d\)

**Domain**
e\(_d\):
e\(_d\) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

**Range:**
integers 1 to 12 or missing
monthly($s_1, s_2, Y$)

Description: the $e_m$ monthly date (months since 1960m1) corresponding to $s_1$ based on $s_2$ and $Y$; $Y$ specifies toyear; see date()

Domain $s_1$: strings
Domain $s_2$: strings "MY" and "YM"; $Y$ may be prefixed with ##
Domain $Y$: integers 1000 to 9998 (but probably 2001 to 2099)
Range: $e_m$ dates 0100m1 to 9999m12 (integers $-22,320$ to $96,479$) or missing

msofhours($h$)

Description: $h \times 3,600,000$

Domain $h$: real; hours
Range: real or missing; milliseconds

msofminutes($m$)

Description: $m \times 60,000$

Domain $m$: real; minutes
Range: real or missing; milliseconds

msofseconds($s$)

Description: $s \times 1,000$

Domain $s$: real; seconds
Range: real or missing; milliseconds

nextbirthday($e_d\text{DOB}, e_d, s_{nl}$)

Description: the $e_d$ date of the first birthday after $e_d$ for date of birth $e_d\text{DOB}$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates

$s_{nl}$ specifies when someone born on 29feb becomes another year older in nonleap years. $s_{nl} =$ "01mar" (the default) means the birthday is taken to be 01mar. $s_{nl} =$ "28feb" means the birthday is taken to be 28feb. See Methods and formulas.

Domain $e_d\text{DOB}$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $s_{nl}$: strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)
Range: $e_d$ dates 01jan0101 to 31dec9999 (integers $-678,985$ to $2,936,549$) or missing

nextdow($e_d, d$)

Description: a synonym for nextweekday($e_d, d$)

nextleapyear($Y$)

Description: the first leap year after year $Y$

Domain $Y$: integers 0100 to 9999 (but probably 1800 to 2100)
Range: integers 1584 to 9996 or missing
nextweekday\((e_d, d)\)
Description: the $e_d$ date of the first day-of-week $d$ after $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $d$: integers 0 to 6 (0=Sunday, 1=Monday, ..., 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)
Range: $e_d$ dates 02jan0100 to 31dec9999 (integers $-679,349$ to $2,936,549$) or missing

now()
Description: the current $e_{tc}$ datetime
Range: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$)

previousbirthday\((e_d\text{DOB}, e_d[s_{nl}]\)\)
Description: the $e_d$ date of the birthday immediately before $e_d$ for date of birth $e_d\text{DOB}$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates
$s_{nl}$ specifies when someone born on 29feb becomes another year older in nonleap years. $s_{nl} = "01mar"$ (the default) means the birthday is taken to be 01mar. $s_{nl} = "28feb"$ means the birthday is taken to be 28feb. See Methods and formulas.
Domain $e_d\text{DOB}$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $s_{nl}$: strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)
Range: $e_d$ dates 01jan0100 to 31dec9998 (integers $-679,350$ to $2,936,184$) or missing

previousdow\((e_d, d)\)
Description: a synonym for previousweekday\((e_d, d)\)

previousleapyear\((Y)\)
Description: the leap year immediately before year $Y$
Domain $Y$: integers 0100 to 9999 (but probably 1800 to 2100)
Range: integers 1584 to 9996 or missing

previousweekday\((e_d, d)\)
Description: the $e_d$ date of the last day-of-week $d$ before $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Domain $d$: integers 0 to 6 (0=Sunday, 1=Monday, ..., 6=Saturday); alternatively, strings with the first two or more letters of the day of week (case insensitive)
Range: $e_d$ dates 01jan0100 to 30dec9999 (integers $-679,350$ to $2,936,548$) or missing

qofd\((e_d)\)
Description: the $e_q$ quarterly date (quarters since 1960q1) containing date $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Range: $e_q$ dates 0100q1 to 9999q4 (integers $-7,440$ to $32,159$)
quarter($e_d$)
Description: the numeric quarter of the year corresponding to date $e_d$
Domain $e_d$: $e_d$ dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)
Range: integers 1 to 4 or missing

quarterly($s_1, s_2, Y$)
Description: the $e_q$ quarterly date (quarters since 1960q1) corresponding to $s_1$ based on $s_2$ and $Y$; $Y$ specifies topyear; see date()
Domain $s_1$: strings
Domain $s_2$: strings "QY" and "YQ"; $Y$ may be prefixed with ##
Domain $Y$: integers 1000 to 9998 (but probably 2001 to 2099)
Range: $e_q$ dates 0100q1 to 9999q4 (integers $-7,440$ to 32,159) or missing

seconds($ms$)
Description: $ms/1,000$
Domain $ms$: real; milliseconds
Range: real or missing

ss($e_{tc}$)
Description: the second corresponding to datetime $e_{tc}$ (ms. since 01jan1960 00:00:00.000)
Domain $e_{tc}$: $e_{tc}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$)
Range: real 0.000 through 59.999 or missing

ssC($e_{tC}$)
Description: the second corresponding to datetime $e_{tC}$ (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain $e_{tC}$: $e_{tC}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$+number of leap seconds)
Range: real 0.000 through 60.999 or missing

tC($l$)
Description: convenience function to make typing dates and times in expressions easier
Same as tc(), except returns leap second–adjusted values; for example, typing tC(29nov2007 9:15) is equivalent to typing 1511946900000, whereas tC(29nov2007 9:15) is 1511946923000.
Domain $l$: datetime literal strings 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
Range: $e_{tC}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers $-58,695,840,000,000$ to $253,717,919,999,999$+number of leap seconds)
to\texttt{c(l)}
\begin{description}
\item[Description:] convenience function to make typing dates and times in expressions easier
\item[Domain \texttt{l}:] datetime literal strings 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
\item[Range:] e_{tc} datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
\end{description}

\noindent For example, typing \texttt{tc(2jan1960 13:42)} is equivalent to typing 13572000; the date but not the time may be omitted, and then 01jan1960 is assumed; the seconds portion of the time may be omitted and is assumed to be 0.000; \texttt{tc(11:02)} is equivalent to typing 39720000.

\texttt{td(l)}
\begin{description}
\item[Description:] convenience function to make typing dates in expressions easier
\item[Domain \texttt{l}:] date literal strings 01jan0100 to 31dec9999
\item[Range:] e_{d} dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))
\end{description}

\texttt{th(l)}
\begin{description}
\item[Description:] convenience function to make typing half-yearly dates in expressions easier
\item[Domain \texttt{l}:] half-year literal strings 0100h1 to 9999h2
\item[Range:] e_{h} dates 0100h1 to 9999h2 (integers \(-3,720\) to \(16,079\))
\end{description}

\texttt{tm(l)}
\begin{description}
\item[Description:] convenience function to make typing monthly dates in expressions easier
\item[Domain \texttt{l}:] month literal strings 0100m1 to 9999m12
\item[Range:] e_{m} dates 0100m1 to 9999m12 (integers \(-22,320\) to \(96,479\))
\end{description}

\texttt{today()}
\begin{description}
\item[Description:] today’s e_{d} date
\item[Range:] e_{d} dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))
\end{description}

\texttt{tq(l)}
\begin{description}
\item[Description:] convenience function to make typing quarterly dates in expressions easier
\item[Domain \texttt{l}:] quarter literal strings 0100q1 to 9999q4
\item[Range:] e_{q} dates 0100q1 to 9999q4 (integers \(-7,440\) to \(32,159\))
\end{description}


**tw(l)**
Description: convenience function to make typing weekly dates in expressions easier
For example, typing `tw(1960w2)` is equivalent to typing `1`.

Domain `l`: week literal strings `0100w1` to `9999w52`
Range: `e_w` dates `0100w1` to `9999w52` (integers `-96,720` to `418,079`)

**week(e_d)**
Description: the numeric week of the year corresponding to date `e_d`, the `%td` encoded date (days since 01jan1960)
Note: The first week of a year is the first 7-day period of the year.

Domain `e_d`: `e_d` dates `01jan0100` to `31dec9999` (integers `-679,350` to `2,936,549`)
Range: integers 1 to 52 or `missing`

**weekly(s_1,s_2,Y)**
Description: the `e_w` weekly date (weeks since 1960w1) corresponding to `s_1` based on `s_2` and `Y`; `Y` specifies `topyear`; see `date()`

Domain `s_1`: strings
Domain `s_2`: strings "WY" and "YW"; `Y` may be prefixed with `##`
Domain `Y`: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `e_w` dates `0100w1` to `9999w52` (integers `-96,720` to `418,079`) or `missing`

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

Domain `e_d`: `e_d` dates `01jan0100` to `31dec9999` (integers `-679,350` to `2,936,549`)
Range: `e_w` dates `0100w1` to `9999w52` (integers `-96,720` to `418,079`)

**year(e_d)**
Description: the numeric year corresponding to date `e_d`

Domain `e_d`: `e_d` dates `01jan0100` to `31dec9999` (integers `-679,350` to `2,936,549`)
Range: integers 100 to 9999 (but probably 1800 to 2100)

**yearly(s_1,s_2,Y)**
Description: the `e_y` yearly date (year) corresponding to `s_1` based on `s_2` and `Y`; `Y` specifies `topyear`; see `date()`

Domain `s_1`: strings
Domain `s_2`: string "Y"; `Y` may be prefixed with `##`
Domain `Y`: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `e_y` dates `0100` to `9999` (integers `0100` to `9999`) or `missing`
Date and time functions

\( yh(Y,H) \)

**Description:** the \( e_h \) half-yearly date (half-years since 1960h1) corresponding to year \( Y \), half-year \( H \)

**Domain** \( Y \): integers 1000 to 9999 (but probably 1800 to 2100)

**Domain** \( H \): integers 1, 2

**Range:** \( e_h \) dates 1000h1 to 9999h2 (integers \(-1,920\) to \(16,079\))

\( ym(Y,M) \)

**Description:** the \( e_m \) monthly date (months since 1960m1) corresponding to year \( Y \), month \( M \)

**Domain** \( Y \): integers 1000 to 9999 (but probably 1800 to 2100)

**Domain** \( M \): integers 1 to 12

**Range:** \( e_m \) dates 1000m1 to 9999m12 (integers \(-11,520\) to \(96,479\))

\( yofd(e_d) \)

**Description:** the \( e_y \) yearly date (year) containing date \( e_d \)

**Domain** \( e_d \): \( e_d \) dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

**Range:** \( e_y \) dates 0100 to 9999 (integers 0100 to 9999)

\( yq(Y,Q) \)

**Description:** the \( e_q \) quarterly date (quarters since 1960q1) corresponding to year \( Y \), quarter \( Q \)

**Domain** \( Y \): integers 1000 to 9999 (but probably 1800 to 2100)

**Domain** \( Q \): integers 1 to 4

**Range:** \( e_q \) dates 1000q1 to 9999q4 (integers \(-3,840\) to \(32,159\))

\( yw(Y,W) \)

**Description:** the \( e_w \) weekly date (weeks since 1960w1) corresponding to year \( Y \), week \( W \)

**Domain** \( Y \): integers 1000 to 9999 (but probably 1800 to 2100)

**Domain** \( W \): integers 1 to 52

**Range:** \( e_w \) dates 1000w1 to 9999w52 (integers \(-49,920\) to \(418,079\))

Remarks and examples

Stata’s date and time functions are described with examples in [U] 25 Working with dates and times, [D] Datetime, [D] Datetime durations, and [D] Datetime relative dates.

Video example

How to create a date variable from a date stored as a string

Methods and formulas

The functions \( \text{age}() \) and \( \text{age\_frac}() \) are based on \( \text{datediff}() \) and \( \text{datediff\_frac}() \), respectively,

\[
\text{age}(e_d\text{DOB},e_d,s_{nl}) = \text{datediff}(e_d\text{DOB},e_d,"year",s_{nl})
\]
and

$$\text{age\_frac}(e_{\text{DOB}}, e_d, s_{\text{nl}}) = \text{datediff\_frac}(e_{\text{DOB}}, e_d, "year", s_{\text{nl}})$$

when $e_d \geq e_{\text{DOB}}$. When $e_d < e_{\text{DOB}}$, age() and age\_frac() return missing (.)

datediff$(e_{d1}, e_{d2}, "year", s_{nl})$ returns an integer that is the number of years between $e_{d1}$ and $e_{d2}$. Assume $e_{d2} \geq e_{d1}$. If the month and day of $e_{d2}$ are the same or after the month and day of $e_{d1}$, it returns $\text{year}(e_{d2}) - \text{year}(e_{d1})$. If the month and day of $e_{d2}$ are before the month and day of $e_{d1}$, it returns $\text{year}(e_{d2}) - \text{year}(e_{d1}) - 1$.

If $e_{d2} < e_{d1}$, the result is calculated using

$$\text{datediff}(e_{d1}, e_{d2}, "year", s_{nl}) = -\text{datediff}(e_{d2}, e_{d1}, "year", s_{nl})$$

This formula also holds for units of "month" and "day" and for datediff\_frac() .

datediff$(e_{d1}, e_{d2}, "year", s_{nl})$ has an optional fourth argument, $s_{nl}$, that applies only to a starting date $e_{d1}$ on 29feb when the ending date $e_{d2}$ is not in a leap year. There are two possible values for $s_{nl}$: either "01mar" (with equivalents "1mar", "mar01", "mar1") or "28feb" ("feb28"). When "01mar" is specified and $e_{d1}$ is on 29feb, datediff() increases by one in nonleap years when $e_{d2}$ goes to 01mar. When "28feb" is specified and $e_{d1}$ is on 29feb, it increases by one in nonleap years when $e_{d2}$ goes to 28feb.

In other words, $s_{nl}$ sets the anniversary date (or birthday) in nonleap years for starting dates (or dates of birth) on 29feb. When the fourth argument is omitted, it is as if "01mar" was specified.

Regardless of the value of $s_{nl}$, when $e_{d1}$ is on 29feb, datediff(...,"year",...) increases by one in leap years when $e_{d2}$ goes to 29feb.

datediff\_frac$(e_{d1}, e_{d2}, "year", s_{nl})$ is defined similarly. datediff\_frac(...,"year",...) is exactly an integer and equal to datediff(...,"year",...) for days $e_{d2}$ on which datediff() increases by one from the day previous to $e_{d2}$.

The fractional part of datediff\_frac$(e_{d1}, e_{d2}, "year", s_{nl})$ is calculated by first counting the number of days, $d_1$, from the closest date prior to $e_{d2}$ that has an exact integer value of datediff\_frac(...,"year",...) to $e_{d2}$. Then number of the days, $d_2$, from $e_{d2}$ to the closest following date that has an exact integer value of datediff\_frac() is determined. The fractional part is $d_1/(d_1 + d_2)$, and $d_1 + d_2$ is either 365 or 366.

For examples, see example 1 and example 3 in [D] Datetime durations.

datediff$(e_{d1}, e_{d2}, "month", s_{nl})$ and datediff\_frac$(e_{d1}, e_{d2}, "month", s_{nl})$ follow the corresponding definitions with "year". datediff(...,"month",...) increases to an integer multiple of 12 when datediff(...,"year",...) increases by one from the day previous to $e_{d2}$.

datediff\_frac(...,"month",...) is exactly 12 times datediff\_frac(...,"year",...) when datediff\_frac(...,"year",...) is an integer.

datediff$(e_{d1}, e_{d2}, "month", s_{nl})$ increases by one from the day previous to $e_{d2}$ when day($e_{d2}$) = day($e_{d1}$). If there is no day($e_{d1}$) in the month, then it increases by one on the first day of the next month. For example, if $e_{d1}$ is on 30aug, then datediff(...,"month",...) increases by one when $e_{d2}$ goes to 30sep. If $e_{d1}$ is on 31aug, then datediff(...,"month",...) increases by one when $e_{d2}$ goes to 01oct.

The optional fourth argument, $s_{nl}$, again sets the date, either "01mar" or "28feb", when datediff(...,"month",...) increases by one when $e_{d1}$ is on 29feb.

datediff\_frac(...,"month",...) is defined like datediff\_frac(...,"year",...). Days on which datediff\_frac(...,"month",...) is an exact integer are determined, and the fractional part for other days is determined by interpolating between these days. The denominator of the fractional part is 28, 29, 30, or 31.
See example 2 of datediff() and datediff_frac() for months in [D] Datetime durations.

datediff($e_{d1}, e_{d2}, "day", s_{nl}$) and datediff_frac($e_{d1}, e_{d2}, "day", s_{nl}$) have no such complications. Both are equal to $e_{d2} - e_{d1}$ and are always integers. The optional fourth argument has no bearing on the calculation and is ignored.

clockdiff($e_{tc1}, e_{tc2}, s_{u}$) and clockdiff_frac($e_{tc1}, e_{tc2}, s_{u}$) take the difference $e_{tc2} - e_{tc1}$, which is in milliseconds, and converts the difference to the units specified by $s_{u}$, days ($24 \times 60 \times 60 \times 1000$ milliseconds), hours ($60 \times 60 \times 1000$ milliseconds), minutes ($60 \times 1000$ milliseconds), or seconds ($1000$ milliseconds). clockdiff() rounds the result down to an integer, whereas clockdiff_frac() retains the fractional part of the difference.

Clockdiff($e_{tC1}, e_{tC2}, s_{u}$) and Clockdiff_frac($e_{tC1}, e_{tC2}, s_{u}$) are similar to clockdiff() and clockdiff_frac() except they are used with datetime/C values (times with leap seconds) rather than datetime/c values (times without leap seconds). In almost all cases, Clockdiff() and Clockdiff_frac() give the same results as clockdiff() and clockdiff_frac() with the datetime/C values converted to datetime/c values. They only differ when either or both of times $e_{tC1}$ and $e_{tC2}$ are close to a leap second and the units are days, hours, or minutes. By “close”, we mean within a day, hour, or minute of the leap second, respectively, for the chosen unit, and less than or equal to the leap second.

Stata system file leapseconds.maint lists the dates on which leap seconds occurred. To view the file, type

```
  . viewsource leapseconds.maint
```

For times close to leap seconds or times that are leap seconds, Clockdiff() and Clockdiff_frac() base their calculations on there being a minute consisting of 61 seconds, an hour of $60 \times 60 + 1 = 3,601$ seconds, and a day of $24 \times 60 \times 60 + 1 = 86,401$ seconds before the leap second (and including the leap second).

For example, 31dec2016 23:59:60 is a leap second, so the time difference between 31dec2016 23:59:00 and 01jan2017 00:00:00 is a minute that consists of 61 seconds. The time difference between $e_{tC1} = 31dec2016 23:59:00$ and $e_{tC2} = 31dec2016 23:59:59$ is 59 seconds. So

\[
\text{Clockdiff_frac}(e_{tC1}, e_{tC2}, "minute") = 59/61 = 0.9672 \text{ minute.}
\]

For times further away from the leap second, say, $e_{tC1} = 31dec2016 23:58:00$ and $e_{tC2} = 01jan2017 00:02:01$, having a leap second between these times has no effect on the result. In this case, Clockdiff_frac($e_{tC1}, e_{tC2}, "minute") = 4 + 1/60 = 4.0167 \text{ minutes.\ 01jan2017 00:02:00 is considered the “anniversary” minute of 31dec2016 23:58:00, so the difference between these times is exactly 4 minutes. Increasing the ending time by a second gives the result 4 + 1/60 minutes. This is, of course, the same result produced by clockdiff_frac(...,"minute") with the datetime/C values converted to datetime/c.

For units of days or hours, the logic of the calculation is similar. For units of seconds or milliseconds, the results are straightforward. The arguments $e_{tC1}$ and $e_{tC2}$ are numbers of milliseconds, so

\[
\text{Clockdiff_frac}(e_{tC1}, e_{tC2}, "millisecond") = e_{tC2} - e_{tC1}
\]

and

\[
\text{Clockdiff_frac}(e_{tC1}, e_{tC2}, "second") = (e_{tC2} - e_{tC1})/1000
\]
References


Also see

[FN] **Functions by category**

[D] *Datetime* — Date and time values and variables

[D] *Datetime durations* — Obtaining and working with durations

[D] *Datetime relative dates* — Obtaining dates and date information from other dates

[D] *egen* — Extensions to generate

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

[M-5] *date()* — Date and time manipulation

[U] **13.3 Functions**

[U] **25 Working with dates and times**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(x)</td>
<td>the absolute value of x</td>
</tr>
<tr>
<td>ceil(x)</td>
<td>the unique integer ( n ) such that ( n - 1 &lt; x \leq n ); if ( x ) is missing, meaning that ( \text{ceil}(x) = x )</td>
</tr>
<tr>
<td>cloglog(x)</td>
<td>the complementary log-log of ( x )</td>
</tr>
<tr>
<td>comb(n,k)</td>
<td>the combinatorial function ( n!/{k!(n-k)!} )</td>
</tr>
<tr>
<td>digamma(x)</td>
<td>the digamma() function, ( d\ln\Gamma(x)/dx )</td>
</tr>
<tr>
<td>exp(x)</td>
<td>the exponential function ( e^x )</td>
</tr>
<tr>
<td>expm1(x)</td>
<td>( e^x - 1 ) with higher precision than ( \exp(x) - 1 ) for small values of (</td>
</tr>
<tr>
<td>floor(x)</td>
<td>the unique integer ( n ) such that ( n \leq x &lt; n + 1 ); if ( x ) is missing, meaning that ( \text{floor}(x) = x )</td>
</tr>
<tr>
<td>int(x)</td>
<td>the integer obtained by truncating ( x ) toward 0 (thus, ( \text{int}(5.2) = 5 ) and ( \text{int}(-5.8) = -5 )); if ( x ) is missing, meaning that ( \text{int}(x) = x )</td>
</tr>
<tr>
<td>invcloglog(x)</td>
<td>the inverse of the complementary log-log function of ( x )</td>
</tr>
<tr>
<td>invlogit(x)</td>
<td>the inverse of the logit function of ( x )</td>
</tr>
<tr>
<td>ln(x)</td>
<td>the natural logarithm, ( \ln(x) )</td>
</tr>
<tr>
<td>ln1m(x)</td>
<td>the natural logarithm of ( 1 - x ) with higher precision than ( \ln(1-x) ) for small values of (</td>
</tr>
<tr>
<td>ln1p(x)</td>
<td>the natural logarithm of ( 1 + x ) with higher precision than ( \ln(1+x) ) for small values of (</td>
</tr>
<tr>
<td>lnfactorial(n)</td>
<td>the natural log of ( n ) factorial = ( \ln(n!) )</td>
</tr>
<tr>
<td>lngamma(x)</td>
<td>( \ln{\Gamma(x)} )</td>
</tr>
<tr>
<td>log(x)</td>
<td>a synonym for ( \ln(x) )</td>
</tr>
<tr>
<td>log10(x)</td>
<td>the base-10 logarithm of ( x )</td>
</tr>
<tr>
<td>log1m(x)</td>
<td>a synonym for ( \ln1m(x) )</td>
</tr>
<tr>
<td>log1p(x)</td>
<td>a synonym for ( \ln1p(x) )</td>
</tr>
<tr>
<td>logit(x)</td>
<td>the log of the odds ratio of ( x ), ( \logit(x) = \ln {x/(1-x)} )</td>
</tr>
<tr>
<td>max(x1,x2,...,xn)</td>
<td>the maximum value of ( x_1,x_2,...,x_n )</td>
</tr>
<tr>
<td>min(x1,x2,...,xn)</td>
<td>the minimum value of ( x_1,x_2,...,x_n )</td>
</tr>
<tr>
<td>mod(x,y)</td>
<td>the modulus of ( x ) with respect to ( y )</td>
</tr>
<tr>
<td>reldif(x,y)</td>
<td>the “relative” difference (</td>
</tr>
</tbody>
</table>
round($x, y$) or round($x$) \[ x \text{ rounded in units of } y \text{ or } x \text{ rounded to the nearest integer if the argument } y \text{ is omitted; } x \text{ (not ".") if } x \text{ is missing (meaning that round(.a) = .a and that round(.a, y) = .a if } y \text{ is not missing) and if } y \text{ is missing, then "." is returned}

sign($x$) \[ \text{the sign of } x: -1 \text{ if } x < 0, 0 \text{ if } x = 0, 1 \text{ if } x > 0, \text{ or } \text{missing if } x \text{ is missing}

sqrt($x$) \[ \text{the square root of } x

sum($x$) \[ \text{the running sum of } x, \text{ treating missing values as zero}

trigamma($x$) \[ \text{the second derivative of } \ln\Gamma(x) = d^2 \ln\Gamma(x)/dx^2

trunc($x$) \[ \text{a synonym for } \text{int}(x)

**Functions**

abs($x$) \[ \text{Description: the absolute value of } x
\text{Domain: } -8\times10^3 \text{ to } 8\times10^7
\text{Range: } 0 \text{ to } 8\times10^7

ceil($x$) \[ \text{Description: the unique integer } n \text{ such that } n - 1 < x \leq n; x \text{ (not ".") if } x \text{ is missing, meaning that ceil(.a) = .a}
\text{Also see } \text{floor}(x), \text{int}(x), \text{and } \text{round}(x).
\text{Domain: } -8\times10^7 \text{ to } 8\times10^7
\text{Range: } \text{integers in } -8\times10^7 \text{ to } 8\times10^7

cloglog($x$) \[ \text{Description: the complementary log-log of } x
\text{cloglog}(x) = \ln\{-\ln(1 - x)\}
\text{Domain: } 0 \text{ to } 1
\text{Range: } -8\times10^7 \text{ to } 8\times10^7

comb($n,k$) \[ \text{Description: the combinatorial function } n!/(k!(n - k)!)\text{Domain } n: \text{ integers 1 to } 1\times10^5
\text{Domain } k: \text{ integers 0 to } n
\text{Range: } 0 \text{ to } 8\times10^7 \text{ or } \text{missing}

digamma($x$) \[ \text{Description: the digamma()} \text{ function, } d\ln\Gamma(x)/dx
\text{This is the derivative of } \ln\Gamma(x). \text{ The } \text{digamma}(x) \text{ function is sometimes called the psi function, } \psi(x).
\text{Domain: } -1\times10^15 \text{ to } 8\times10^7
\text{Range: } -8\times10^7 \text{ to } 8\times10^7 \text{ or } \text{missing}
exp($x$)
Description: the exponential function $e^x$
This function is the inverse of $\ln(x)$. To compute $e^x - 1$ with high precision for small values of $|x|$, use $\expm1(x)$.
Domain: $-8e+307$ to $709$
Range: $0$ to $8e+307$

expm1($x$)
Description: $e^x - 1$ with higher precision than $\exp(x) - 1$ for small values of $|x|$
Domain: $-8e+307$ to $709$
Range: $-1$ to $8e+307$

floor($x$)
Description: the unique integer $n$ such that $n \leq x < n + 1$; $x$ (not “.”) if $x$ is missing, meaning that $\text{floor}(.a) = .a$
Also see $\text{ceil}(x)$, $\text{int}(x)$, and $\text{round}(x)$.
Domain: $-8e+307$ to $8e+307$
Range: integers in $-8e+307$ to $8e+307$

int($x$)
Description: the integer obtained by truncating $x$ toward 0 (thus, $\text{int}(5.2) = 5$ and $\text{int}(-5.8) = -5$); $x$ (not “.”) if $x$ is missing, meaning that $\text{int}(.a) = .a$
One way to obtain the closest integer to $x$ is $\text{int}(x+\text{sign}(x)/2)$, which simplifies to $\text{int}(x+0.5)$ for $x \geq 0$. However, use of the $\text{round()}$ function is preferred. Also see $\text{round}(x)$, $\text{ceil}(x)$, and $\text{floor}(x)$.
Domain: $-8e+307$ to $8e+307$
Range: integers in $-8e+307$ to $8e+307$

invcloglog($x$)
Description: the inverse of the complementary log-log function of $x$
\[
\text{invcloglog}(x) = 1 - \exp\{-\exp(x)\}
\]
Domain: $-8e+307$ to $8e+307$
Range: $0$ to $1$ or $\text{missing}$

invlogit($x$)
Description: the inverse of the logit function of $x$
\[
\text{invlogit}(x) = \frac{\exp(x)}{1 + \exp(x)}
\]
Domain: $-8e+307$ to $8e+307$
Range: $0$ to $1$ or $\text{missing}$
ln(x)
Description: the natural logarithm, ln(x)

This function is the inverse of exp(x). The logarithm of x in base b can be calculated via \( \log_b(x) = \log_a(x)/\log_a(b) \). Hence,
\[
\log_5(x) = \frac{\ln(x)}{\ln(5)} = \frac{\log(x)}{\log(5)} = \frac{\log_{10}(x)}{\log_{10}(5)} \\
\log_2(x) = \frac{\ln(x)}{\ln(2)} = \frac{\log(x)}{\log(2)} = \frac{\log_{10}(x)}{\log_{10}(2)}
\]

You can calculate \( \log_b(x) \) by using the formula that best suits your needs. To compute \( \ln(1-x) \) and \( \ln(1+x) \) with high precision for small values of \(|x|\), use \( \text{ln1m}(x) \) and \( \text{ln1p}(x) \), respectively.

**Domain:** 1e–323 to 8e+307  
**Range:** −744 to 709

\( \text{ln1m}(x) \)
Description: the natural logarithm of \( 1 - x \) with higher precision than \( \ln(1-x) \) for small values of \(|x|\)

**Domain:** −8e+307 to 1 – c(epsdouble)  
**Range:** −37 to 709

\( \text{ln1p}(x) \)
Description: the natural logarithm of \( 1 + x \) with higher precision than \( \ln(1+x) \) for small values of \(|x|\)

**Domain:** −1 + c(epsdouble) to 8e+307  
**Range:** −37 to 709

\( \text{lncfactorial}(n) \)
Description: the natural log of \( n \) factorial = \( \ln(n!) \)

To calculate \( n! \), use \( \text{round}(\exp(\text{lncfactorial}(n)),1) \) to ensure that the result is an integer. Logs of factorials are generally more useful than the factorials themselves because of overflow problems.

**Domain:** integers 0 to 1e+305  
**Range:** 0 to 8e+307

\( \text{lngamma}(x) \)
Description: \( \ln\{\Gamma(x)\} \)

Here the gamma function, \( \Gamma(x) \), is defined by \( \Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt \). For integer values of \( x > 0 \), this is \( \ln((x-1)!) \).

\( \text{lngamma}(x) \) for \( x < 0 \) returns a number such that \( \exp(\text{lngamma}(x)) \) is equal to the absolute value of the gamma function, \( \Gamma(x) \). That is, \( \text{lngamma}(x) \) always returns a real (not complex) result.

**Domain:** −2,147,483,648 to 1e+305 (excluding negative integers)  
**Range:** −8e+307 to 8e+307

log(x)
Description: a synonym for \( \ln(x) \)
log10(x)
Description: the base-10 logarithm of x
Domain: 1e–323 to 8e+307
Range: −323 to 308

log1m(x)
Description: a synonym for ln1m(x)

log1p(x)
Description: a synonym for ln1p(x)

logit(x)
Description: the log of the odds ratio of x, logit(x) = ln {x/(1 − x)}
Domain: 0 to 1 (exclusive)
Range: −8e+307 to 8e+307 or missing

max(x1,x2,...,xn)
Description: the maximum value of x1, x2, ..., xn
Unless all arguments are missing, missing values are ignored.
max(2,10,.,7) = 10
max(.,.,.) = .
Domain x1: −8e+307 to 8e+307 or missing
Domain x2: −8e+307 to 8e+307 or missing
...
Domain xn: −8e+307 to 8e+307 or missing
Range: −8e+307 to 8e+307 or missing

min(x1,x2,...,xn)
Description: the minimum value of x1, x2, ..., xn
Unless all arguments are missing, missing values are ignored.
min(2,10,.,7) = 2
min(.,.,.) = .
Domain x1: −8e+307 to 8e+307 or missing
Domain x2: −8e+307 to 8e+307 or missing
...
Domain xn: −8e+307 to 8e+307 or missing
Range: −8e+307 to 8e+307 or missing
mod($x$, $y$)
Description: the modulus of $x$ with respect to $y$

$$\text{mod}(x, y) = x - y \text{ floor}(x/y)$$

Domain $x$: $-8e+307$ to $8e+307$
Domain $y$: $0$ to $8e+307$
Range: $0$ to $8e+307$

reldif($x$, $y$)
Description: 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

Domain $x$: $-8e+307$ to $8e+307$ or missing
Domain $y$: $-8e+307$ to $8e+307$ or missing
Range: $0$ to $8e+307$ or missing

round($x$, $y$) or round($x$)
Description: $x$ 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

For $y = 1$, or with $y$ omitted, this amounts to the closest integer to $x$; round(5.2, 1) is 5, as is round(4.8, 1); round(-5.2, 1) is -5, as is round(-4.8, 1). The rounding definition is generalized for $y \neq 1$. With $y = 0.01$, for instance, $x$ is rounded to two decimal places; round(sqrt(2), .01) is 1.41. $y$ may also be larger than 1; round(28, 5) is 30, which is 28 rounded to the closest multiple of 5. For $y = 0$, the function is defined as returning $x$ unmodified.

For values of $x$ exactly at midpoints, where it may not be clear whether to round up or down, $x$ is always rounded up to the larger value. For example, round(4.5) is 5 and round(-4.5) is -4. Note that rounding a number is based on the floating-point number representation of the number instead of the number itself. So round() is sensitive to representation errors and precision limits. For example, 0.15 has no exact floating-point number representation. Therefore, round(0.15, 0.1) is 0.1 instead of 0.2. See [U] 13.12 Precision and problems therein for details.

Also see int($x$), ceil($x$), and floor($x$).

Domain $x$: $-8e+307$ to $8e+307$
Domain $y$: $-8e+307$ to $8e+307$
Range: $-8e+307$ to $8e+307$

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

Domain: $-8e+307$ to $8e+307$ or missing
Range: $-1$, 0, 1 or missing
sqrt(x)
Description: the square root of $x$
Domain: 0 to 8e+307
Range: 0 to 1e+154

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

For example, following the command `generate y=sum(x)`, the $j$th observation on $y$ contains the sum of the first through $j$th observations on $x$. See [D] `egen` for an alternative sum function, `total()`, that produces a constant equal to the overall sum.

Domain: all real numbers or `missing`
Range: $-8e+307$ to $8e+307$ (excluding `missing`)

trigamma(x)
Description: the second derivative of $\ln\Gamma(x) = d^2 \ln\Gamma(x)/dx^2$

The `trigamma()` function is the derivative of `digamma(x)`.

Domain: $-1e+15$ to $8e+307$
Range: 0 to $8e+307$ or `missing`

trunc(x)
Description: a synonym for `int(x)`

Video example
How to round a continuous variable

References


Also see

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

**Contents**

- **cholesky(M)**
  - the Cholesky decomposition of the matrix: if $R = \text{cholesky}(S)$, 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 $M$

- **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)**
  - $s[floor(i),floor(j)]$, the $i,j$ element of the matrix named $s$; *missing* if $i$ or $j$ are out of range or if matrix $s$ does not exist

- **get(systemname)**
  - a copy of Stata internal system matrix $systemname$

- **hadamard(M,N)**
  - a matrix whose $i,j$ element is $M[i,j] \cdot N[i,j]$ (if $M$ and $N$ are not the same size, this function reports a conformability error)

- **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

- **invvech(M)**
  - a symmetric matrix formed by filling in the columns of the lower triangle from a row or column vector

- **invvecp(M)**
  - a symmetric matrix formed by filling in the columns of the upper triangle from a row or column vector

- **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
**Matrix functions**

- `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 $M$
- `vech(M)` - a column vector formed by listing the lower triangle elements of $M$
- `vecp(M)` - a column vector formed by listing the upper triangle elements of $M$

**Functions**

We divide the basic matrix functions into two groups, according to whether they return a matrix or a scalar:

<table>
<thead>
<tr>
<th>Matrix functions returning a matrix</th>
<th>Matrix functions returning a scalar</th>
</tr>
</thead>
</table>

**Matrix functions returning a matrix**

In addition to the functions listed below, see [P] matrix svd for singular value decomposition, [P] matrix symeigen for eigenvalues and eigenvectors of symmetric matrices, and [P] matrix eigenvalues for eigenvalues of nonsymmetric matrices.

- `cholesky(M)`
  Description: the Cholesky decomposition of the matrix: if $R = \text{cholesky}(S)$, then $RR^T = S$
  $R^T$ indicates the transpose of $R$. Row and column names are obtained from $M$.
  Domain: $n \times n$, positive-definite, symmetric matrices
  Range: $n \times n$ lower-triangular matrices

- `corr(M)`
  Description: the correlation matrix of the variance matrix
  Row and column names are obtained from $M$.
  Domain: $n \times n$ symmetric variance matrices
  Range: $n \times n$ symmetric correlation matrices

- `diag(M)`
  Description: the square, diagonal matrix created from the row or column vector
  Row and column names are obtained from the column names of $M$ if $M$ is a row vector or from the row names of $M$ if $M$ is a column vector.
  Domain: $1 \times n$ and $n \times 1$ vectors
  Range: $n \times n$ diagonal matrices

- `get(systemname)`
  Description: a copy of Stata internal system matrix `systemname`
  This function is included for backward compatibility with previous versions of Stata.
  Domain: existing names of system matrices
  Range: matrices
hadamard(\(M, N\))
Description: 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)

Domain \(M\): \(m \times n\) matrices
Domain \(N\): \(m \times n\) matrices
Range: \(m \times n\) matrices

I(\(n\))
Description: an \(n \times n\) identity matrix if \(n\) is an integer; otherwise, a \(\text{round}(n) \times \text{round}(n)\) identity matrix

Domain: real scalars 1 to c(max_matdim)
Range: identity matrices

inv(\(M\))
Description: the inverse of the matrix \(M\)

If \(M\) is singular, this will result in an error.

The function invsym() should be used in preference to inv() because invsym() is more accurate. The row names of the result are obtained from the column names of \(M\), and the column names of the result are obtained from the row names of \(M\).

Domain: \(n \times n\) nonsingular matrices
Range: \(n \times n\) matrices

invsym(\(M\))
Description: the inverse of \(M\) if \(M\) is positive definite

If \(M\) is not positive definite, rows will be inverted until the diagonal terms are zero or negative; the rows and columns corresponding to these terms will be set to 0, producing a g2 inverse. The row names of the result are obtained from the column names of \(M\), and the column names of the result are obtained from the row names of \(M\).

Domain: \(n \times n\) symmetric matrices
Range: \(n \times n\) symmetric matrices

invvech(\(M\))
Description: a symmetric matrix formed by filling in the columns of the lower triangle from a row or column vector

Domain: \(n(n+1)/2 \times 1\) and \(1 \times n(n+1)/2\) vectors
Range: \(n \times n\) matrices

invvecp(\(M\))
Description: a symmetric matrix formed by filling in the columns of the upper triangle from a row or column vector

Domain: \(n(n+1)/2 \times 1\) and \(1 \times n(n+1)/2\) vectors
Range: \(n \times n\) matrices

J(\(r, c, z\))
Description: the \(r \times c\) matrix containing elements \(z\)

Domain \(r\): integer scalars 1 to c(max_matdim)
Domain \(c\): integer scalars 1 to c(max_matdim)
Domain \(z\): scalars \(-8e+307\) to \(8e+307\)
Range: \(r \times c\) matrices
matuniform(r,c)
Description: the \( r \times c \) matrices containing uniformly distributed pseudorandom numbers on the interval \((0, 1)\)
Domain \( r \): integer scalars 1 to \( c(max\_matdim) \)
Domain \( c \): integer scalars 1 to \( c(max\_matdim) \)
Range: \( r \times c \) matrices

nullmat(matname)
Description: use with the row-join (,) and column-join (\) operators

Consider the following code fragment, which is an attempt to create the vector \((1, 2, 3, 4)\):

```stata
forvalues i = 1/4 {
    mat v = (v, 'i')
}
```

The above program will not work because, the first time through the loop, \( v \) will not yet exist, and thus forming \((v, 'i')\) makes no sense. nullmat() relaxes that restriction:

```stata
forvalues i = 1/4 {
    mat v = (nullmat(v), 'i')
}
```

The nullmat() function informs Stata that if \( v \) does not exist, the function row-join is to be generalized. Joining nothing with ‘i’ results in ‘i’. Thus the first time through the loop, \( v = (1) \) is formed. The second time through, \( v \) does exist, so \( v = (1, 2) \) is formed, and so on.

nullmat() can be used only with the , and \ operators.

Domain: matrix names, existing and nonexistent
Range: matrices including null if matname does not exist

sweep(M,i)
Description: matrix \( M \) with \( i \)th row/column swept

The row and column names of the resultant matrix are obtained from \( M \), except that the \( n \)th row and column names are interchanged. If \( B = sweep(A, k) \), then

\[
B_{kk} = \frac{1}{A_{kk}} \\
B_{ik} = -\frac{A_{ik}}{A_{kk}}, \quad i \neq k \\
B_{kj} = \frac{A_{kj}}{A_{kk}}, \quad j \neq k \\
B_{ij} = A_{ij} - \frac{A_{ik}A_{kj}}{A_{kk}}, \quad i \neq k, j \neq k
\]

Domain \( M \): \( n \times n \) matrices
Domain \( i \): integer scalars 1 to \( n \)
Range: \( n \times n \) matrices
Matrix functions

\text{vec}(M)
Description: a column vector formed by listing the elements of \( M \), starting with the first column and proceeding column by column  
Domain: matrices  
Range: column vectors \((n \times 1 \text{ matrices})\)

\text{vecdiag}(M)
Description: the row vector containing the diagonal of matrix \( M \)  
\text{vecdiag()} is the opposite of \text{diag()}. The row name is set to \( r1 \); the column names are obtained from the column names of \( M \).  
Domain: \( n \times n \) matrices  
Range: \( 1 \times n \) vectors

\text{vech}(M)
Description: a column vector formed by listing the lower triangle elements of \( M \)  
Domain: \( n \times n \) matrices  
Range: \( n(n+1)/2 \times 1 \) vectors

\text{vecp}(M)
Description: a column vector formed by listing the upper triangle elements of \( M \)  
Domain: \( n \times n \) matrices  
Range: \( n(n+1)/2 \times 1 \) vectors

Matrix functions returning a scalar

\text{coleqnumb}(M,s)
Description: the equation number of \( M \) associated with column equation \( s \); \text{missing} if the column equation cannot be found  
Domain \( M \): matrices  
Domain \( s \): strings  
Range: integer scalars 1 to \( \text{c(max_matdim)} \) or \text{missing}

\text{colnfreeparms}(M)
Description: the number of free parameters in columns of \( M \)  
Domain: matrices  
Range: integer scalars 0 to \( \text{c(max_matdim)} \)

\text{colnumb}(M,s)
Description: the column number of \( M \) associated with column name \( s \); \text{missing} if the column cannot be found  
Domain \( M \): matrices  
Domain \( s \): strings  
Range: integer scalars 1 to \( \text{c(max_matdim)} \) or \text{missing}

\text{colsof}(M)
Description: the number of columns of \( M \)  
Domain: matrices  
Range: integer scalars 1 to \( \text{c(max_matdim)} \)
\textbf{det}(M)

Description: the determinant of matrix \( M \)
Domain: \( n \times n \) (square) matrices
Range: scalars \(-8e+307\) to \( 8e+307 \)

\textbf{diag0cnt}(M)

Description: the number of zeros on the diagonal of \( M \)
Domain: \( n \times n \) (square) matrices
Range: integer scalars 0 to \( n \)

\textbf{el}(s,i,j)

Description: \( s[floor(i),floor(j)], \) the \( i,j \) element of the matrix named \( s; \) missing if \( i \) or \( j \)
    are out of range or if matrix \( s \) does not exist
Domain \( s: \) strings containing matrix name
Domain \( i: \) scalars 1 to \( c(max\_matdim) \)
Domain \( j: \) scalars 1 to \( c(max\_matdim) \)
Range: scalars \(-8e+307\) to \( 8e+307 \) or \textit{missing}

\textbf{issymmetric}(M)

Description: 1 if the matrix is symmetric; otherwise, 0
Domain \( M: \) matrices
Range: integers 0 and 1

\textbf{matmissing}(M)

Description: 1 if any elements of the matrix are missing; otherwise, 0
Domain \( M: \) matrices
Range: integers 0 and 1

\textbf{mreldif}(X,Y)

Description: the relative difference of \( X \) and \( Y, \) where the relative difference is defined as
    \[
    \max_{i,j}\{\frac{|x_{ij} - y_{ij}|}{(|y_{ij}| + 1)}\}
    \]
Domain \( X: \) matrices
Domain \( Y: \) matrices with same number of rows and columns as \( X \)
Range: scalars \(-8e+307\) to \( 8e+307 \)

\textbf{roweqnumb}(M,s)

Description: the equation number of \( M \) associated with row equation \( s; \) \textit{missing} if the row equation
    cannot be found
Domain \( M: \) matrices
Domain \( s: \) strings
Range: integer scalars 1 to \( c(max\_matdim) \) or \textit{missing}

\textbf{rownfreeparms}(M)

Description: the number of free parameters in rows of \( M \)
Domain: \( M: \) matrices
Range: integer scalars 0 to \( c(max\_matdim) \)
Matrix functions

\begin{align*}
\text{rownumb}(M,s) \\
\text{Description: } & \text{the row number of } M \text{ associated with row name } s; \text{ missing if the row cannot be found} \\
\text{Domain } M: & \text{ matrices} \\
\text{Domain } s: & \text{ strings} \\
\text{Range: } & \text{integer scalars 1 to } c(\text{max\_matdim}) \text{ or missing}
\end{align*}

\begin{align*}
\text{rowsof}(M) \\
\text{Description: } & \text{the number of rows of } M \\
\text{Domain: } & \text{ matrices} \\
\text{Range: } & \text{integer scalars 1 to } c(\text{max\_matdim})
\end{align*}

\begin{align*}
\text{trace}(M) \\
\text{Description: } & \text{the trace of matrix } M \\
\text{Domain: } & n \times n \text{ (square) matrices} \\
\text{Range: } & \text{scalars } -8e+307 \text{ to } 8e+307
\end{align*}

Jacques Salomon Hadamard (1865–1963) was born in Versailles, France. He had a tumultuous childhood, eating elephant meat to survive and enduring the premature deaths of two younger sisters. Hadamard taught while working on his doctorate, which he obtained in 1892 from École Normale Supérieure. His dissertation is recognized as the first examination of singularities. Hadamard published a paper on the Riemann zeta function, for which he was awarded the Grand Prix des Sciences Mathématiques in 1892. Shortly after, he became a professor at the University of Bordeaux and made many significant contributions over the course of four years. For example, in 1893 he published a paper on determinant inequalities, giving rise to Hadamard matrices. Then in 1896, he used complex analysis to prove the prime number theorem, and he was awarded the Bordin Prize by the Academy of Sciences for his work on dynamic trajectories. In the following years, he published books on two-dimensional and three-dimensional geometry, as well as an influential paper on functional analysis. He was elected to presidency of the French Mathematical Society in 1906 and as chair of mechanics at the Collège de France in 1909. Faced with the tragic deaths of two of his sons during World War I, Hadamard buried himself in his work. He continued to publish outstanding work in new areas, including probability theory, education, and psychology. In 1956, he was awarded the CNRS Gold Medal for his many contributions.

Reference


Also see

[FN] Functions by category
[D] egen — Extensions to generate
[D] generate — Create or change contents of variable
[M-4] Intro — Categorical guide to Mata functions
[U] 13.3 Functions
[U] 14.8 Matrix functions
Programming functions

Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>autocode</strong>($x, n, x_0, x_1$)</td>
<td>partitions the interval from $x_0$ to $x_1$ into $n$ equal-length intervals and returns the upper bound of the interval that contains $x$ or the upper bound of the first or last interval if $x &lt; x_0$ or $x &gt; x_1$, respectively</td>
</tr>
<tr>
<td><strong>byteorder()</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>c(name)</strong></td>
<td>the value of the system or constant result $c(name)$ (see [P] creturn)</td>
</tr>
<tr>
<td><strong>_caller()</strong></td>
<td>version of the program or session that invoked the currently running program; see [P] version</td>
</tr>
<tr>
<td><strong>chop($x, \epsilon$)</strong></td>
<td>$\text{round}(x)$ if $\text{abs}(x - \text{round}(x)) &lt; \epsilon$; otherwise, $x$; or $x$ if $x$ is missing</td>
</tr>
<tr>
<td><strong>clip($x, a, b$)</strong></td>
<td>$x$ if $a &lt; x &lt; b$, $b$ if $x \geq b$, $a$ if $x \leq a$, or missing if $x$ is missing or if $a &gt; b$; $x$ if $x$ is missing</td>
</tr>
<tr>
<td><strong>cond($x, a, b[ , c ]$)</strong></td>
<td>$a$ 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</td>
</tr>
<tr>
<td><strong>e(name)</strong></td>
<td>the value of stored result $e(name)$; see [U] 18.8 Accessing results calculated by other programs</td>
</tr>
<tr>
<td><strong>e(sample)</strong></td>
<td>1 if the observation is in the estimation sample and 0 otherwise</td>
</tr>
<tr>
<td><strong>epsdouble()</strong></td>
<td>the machine precision of a double-precision number</td>
</tr>
<tr>
<td><strong>epsfloat()</strong></td>
<td>the machine precision of a floating-point number</td>
</tr>
<tr>
<td><strong>fileexists($f$)</strong></td>
<td>1 if the file specified by $f$ exists; otherwise, 0</td>
</tr>
<tr>
<td><strong>fileread($f$)</strong></td>
<td>the contents of the file specified by $f$</td>
</tr>
<tr>
<td><strong>filereaderror($s$)</strong></td>
<td>0 or positive integer, said value having the interpretation of a return code</td>
</tr>
<tr>
<td><strong>fwrite($f, s[,r]$)</strong></td>
<td>writes the string specified by $s$ to the file specified by $f$ and returns the number of bytes in the resulting file</td>
</tr>
<tr>
<td><strong>float($x$)</strong></td>
<td>the value of $x$ rounded to float precision</td>
</tr>
<tr>
<td><strong>fmtwidth($fmtstr$)</strong></td>
<td>the output length of the $%fmt$ contained in $fmtstr$; missing if $fmtstr$ does not contain a valid $%fmt$</td>
</tr>
<tr>
<td><strong>frval()</strong></td>
<td>returns values of variables stored in other frames</td>
</tr>
<tr>
<td><strong>_frval()</strong></td>
<td>programmer’s version of <strong>frval()</strong></td>
</tr>
<tr>
<td><strong>has_evarp(name)</strong></td>
<td>1 if name appears as a word in e(properties); otherwise, 0</td>
</tr>
<tr>
<td><strong>inlist($z, a, b, \ldots$)</strong></td>
<td>1 if $z$ is a member of the remaining arguments; otherwise, 0</td>
</tr>
<tr>
<td><strong>inrange($z, a, b$)</strong></td>
<td>1 if it is known that $a \leq z \leq b$; otherwise, 0</td>
</tr>
<tr>
<td><strong>irecode($x, x_1, \ldots, x_n$)</strong></td>
<td>missing if $x$ is missing or $x_1, \ldots, x_n$ is not weakly increasing; 0 if $x \leq x_1$; 1 if $x_1 &lt; x \leq x_2$; 2 if $x_2 &lt; x \leq x_3$; \ldots; $n$ if $x &gt; x_n$</td>
</tr>
</tbody>
</table>
programming functions 81

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, ..., x_n) a synonym for missing(x_1, x_2, ..., 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, ..., x_n) 1 if any x_i evaluates to missing; otherwise, 0

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

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

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

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
Functions

autocode\((x,n,x_0,x_1)\)
Description: partitions the interval from \(x_0\) to \(x_1\) into \(n\) equal-length intervals and returns the upper bound of the interval that contains \(x\) or the upper bound of the first or last interval if \(x < x_0\) or \(x > x_1\), respectively
This function is an automated version of `recode()`. See [U] 26 Working with categorical data and factor variables for an example.

The algorithm for `autocode()` is
\[
\text{if } (n \geq 0 \land x_0 \geq x \land x_1 \geq x \land n \leq 0 \land x_0 \geq x_1) \\
\text{then return missing} \\
\text{if } x \geq . \text{, then return } x \\
\text{otherwise} \\
\text{for } i = 1 \text{ to } n - 1 \\
xmap = x_0 + i \times (x_1 - x_0)/n \\
\text{if } x \leq xmap \text{ then return } xmap \\
\text{end} \\
\text{otherwise} \\
\text{return } x_1
\]

Domain \(x\): \(-8e+307\) to \(8e+307\)
Domain \(n\): integers 1 to 10,000
Domain \(x_0\): \(-8e+307\) to \(8e+307\)
Domain \(x_1\): \(x_0\) to \(8e+307\)
Range: \(x_0\) to \(x_1\)

byteorder()
Description: 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
Consider the number 1 written as a 2-byte integer. On some computers (called hilo), it is written as “00 01”, and on other computers (called lohi), it is written as “01 00” (with the least significant byte written first). There are similar issues for 4-byte integers, 4-byte floats, and 8-byte floats. Stata automatically handles byte-order differences for Stata-created files. Users need not be concerned about this issue. Programmers producing custom binary files can use `byteorder()` to determine the native byte ordering; see [P] file.

Range: 1 and 2

c\((name)\)
Description: the value of the system or constant result `c\((name)\)` (see [P] `creturn`)
Referencing `c\((name)\)` will return an error if the result does not exist.

Domain: names
Range: real values, strings, or missing
__caller__

Description: version of the program or session that invoked the currently running program; see [P] version

The current version at the time of this writing is 18.1, so 18.1 is the upper end of this range. If Stata 18.2 were the current version, 18.2 would be the upper end of this range, and likewise, if Stata 19 were the current version, 19 would be the upper end of this range. This is a function for use by programmers.

Range: 1 to 18.0

chop\((x, \epsilon)\)

Description: \(\text{round}(x)\) if \(\text{abs}(x - \text{round}(x)) < \epsilon\); otherwise, \(x\); or \(x\) if \(x\) is missing

Domain \(x\): \(-8e+307\) to \(8e+307\)
Domain \(\epsilon\): \(-8e+307\) to \(8e+307\)
Range: \(-8e+307\) to \(8e+307\)

clip\((x,a,b)\)

Description: \(x\) 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

If \(a\) or \(b\) is missing, this is interpreted as \(a = -\infty\) or \(b = +\infty\), respectively.

Domain \(x\): \(-8e+307\) to \(8e+307\)
Domain \(a\): \(-8e+307\) to \(8e+307\)
Domain \(b\): \(-8e+307\) to \(8e+307\)
Range: \(-8e+307\) to \(8e+307\)

cond\((x,a,b,c)\)

Description: \(a\) 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

Note that expressions such as \(x > 2\) will never evaluate to missing.

\[
\begin{align*}
\text{cond}(x>2,50,70) & \text{ returns 50 if } x > 2 \text{ (includes } x \geq .) \\
\text{cond}(x>2,50,70) & \text{ returns 70 if } x \leq 2 \\
\text{cond}(\text{missing}(x), ., \text{cond}(x>2,50,70)) & \text{ returns . if } x \text{ is missing, returns 50 if } x > 2, \text{ and returns 70 if } x \leq 2 \\
\end{align*}
\]

If you need a case for missing values in the above examples, try

\[
\begin{align*}
\text{cond}(\text{missing}(x), ., \text{cond}(x>2,50,70)) & \text{ returns . if } x \text{ is missing, returns 50 if } x > 2, \text{ and returns 70 if } x \leq 2 \\
\end{align*}
\]

If the first argument is a scalar that may contain a missing value or a variable containing missing values, the fourth argument has an effect.

\[
\begin{align*}
\text{cond}(\text{wage},1,0,.)) & \text{ returns 1 if } \text{wage} \text{ is not zero and not missing} \\
\text{cond}(\text{wage},1,0,.)) & \text{ returns 0 if } \text{wage} \text{ is zero} \\
\text{cond}(\text{wage},1,0,.)) & \text{ returns . if } \text{wage} \text{ is missing} \\
\end{align*}
\]

Caution: If the first argument to cond() is a logical expression, that is, \(\text{cond}(x>2,50,70,.))\), the fourth argument is never reached.

Domain \(x\): \(-8e+307\) to \(8e+307\) or missing; \(0 \Rightarrow \text{false}\), otherwise interpreted as true
Domain \(a\): numbers and strings
Domain \(b\): numbers if \(a\) is a number; strings if \(a\) is a string
Domain \(c\): numbers if \(a\) is a number; strings if \(a\) is a string
Range: \(a, b, \text{ and } c\)
**e(name)**
Description: the value of stored result e(name); see [U] 18.8 Accessing results calculated by other programs

- e(name) = scalar missing if the stored result does not exist
- e(name) = specified matrix if the stored result is a matrix
- e(name) = scalar numeric value if the stored result is a scalar

Domain: names
Range: strings, scalars, matrices, or missing

**e(sample)**
Description: 1 if the observation is in the estimation sample and 0 otherwise
Range: 0 and 1

**epsdouble()**
Description: the machine precision of a double-precision number

- If \( d < \text{epsdouble()} \) and (double) \( x = 1 \), then \( x + d = \) (double) 1. This function takes no arguments, but the parentheses must be included.

Range: a double-precision number close to 0

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

- If \( d < \text{epsfloat()} \) and (float) \( x = 1 \), then \( x + d = \) (float) 1. This function takes no arguments, but the parentheses must be included.

Range: a floating-point number close to 0

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

- If the file exists but is not readable, \text{fileexists()} will still return 1, because it does exist. If the “file” is a directory, \text{fileexists()} will return 0.

Domain: filenames
Range: 0 and 1

**fileread(f)**
Description: the contents of the file specified by \( f \)

- If the file does not exist or an I/O error occurs while reading the file, then “fileread() error #” is returned, where # is a standard Stata error return code.

Domain: filenames
Range: strings
filereaderror(s)
Description: 0 or positive integer, said value having the interpretation of a return code
It is used like this

. generate strL s = fileread(filename) if fileexists(filename)
. assert filereaderror(s)==0

or this

. generate strL s = fileread(filename) if fileexists(filename)
. generate rc = filereaderror(s)

That is, filereaderror(s) is used on the result returned by fileread(filename) to determine whether an I/O error occurred.
In the example, we only fileread() files that fileexists(). That is not required. If the file does not exist, that will be detected by filereaderror() as an error. The way we showed the example, we did not want to read missing files as errors. If we wanted to treat missing files as errors, we would have coded

. generate strL s = fileread(filename)
. assert filereaderror(s)==0

or

. generate strL s = fileread(filename)
. generate rc = filereaderror(s)

Domain: strings
Range: integers

filewrite(f,s[,r])
Description: writes the string specified by s to the file specified by f and returns the number of bytes in the resulting file
If the optional argument r is specified as 1, the file specified by f will be replaced if it exists. If r is specified as 2, the file specified by f will be appended to if it exists. Any other values of r are treated as if r were not specified; that is, f will only be written to if it does not already exist.
When the file f is freshly created or is replaced, the value returned by filewrite() is the number of bytes written to the file, strlen(s). If r is specified as 2, and thus filewrite() is appending to an existing file, the value returned is the total number of bytes in the resulting file; that is, the value is the sum of the number of the bytes in the file as it existed before filewrite() was called and the number of bytes newly written to it, strlen(s).
If the file exists and r is not specified as 1 or 2, or an error occurs while writing to the file, then a negative number (#) is returned, where abs(#) is a standard Stata error return code.

Domain f: filenames
Domain s: strings
Domain r: integers 1 or 2
Range: integers
**float(\(x\))**  
Description: the value of \(x\) rounded to float precision

Although you may store your numeric variables as byte, int, long, float, or double, Stata converts all numbers to double before performing any calculations. Consequently, difficulties can arise in comparing numbers that have no finite binary representation.

For example, if the variable \(x\) is stored as a float and contains the value 1.1 (a repeating “decimal” in binary), the expression \(x==1.1\) will evaluate to false because the literal 1.1 is the double representation of 1.1, which is different from the float representation stored in \(x\). (They differ by \(2.384 \times 10^{-8}\).) The expression \(x==\text{float}(1.1)\) will evaluate to true because the float() function converts the literal 1.1 to its float representation before it is compared with \(x\). (See [U] 13.12 Precision and problems therein for more information.)

| Domain: | \(-1\times 10^{38}\) to \(1\times 10^{38}\) |
| Range:  | \(-1\times 10^{38}\) to \(1\times 10^{38}\) |

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

For example, fmtwidth("%9.2f") returns 9 and fmtwidth("%tc") returns 18.

| Range: | strings |

**frval(lvar,var)**  
Description: returns values of variables stored in other frames

The frame functions frval() and _frval() access values of variables in frames outside the current frame. If you do not know what a frame is, see [D] frames intro.

The two functions do the same thing, but frval() is easier to use, and it is safer. _frval() is a programmer’s function.

\(lvar\) is the name of a variable created by frlink that links the current frame to another frame.

\(var\) is the name of a variable in the other frame.

Returned is the value of \(var\) from the observation in the other frame that matches the observation in the current frame.

**Example 1:** The current frame contains data on persons. Among the variables in the current frame is countyid containing the county in which each person lives.

Frame frcounty contains data on counties. In these data, variable countyid also records the county’s ID, and the other variables record county characteristics.

In the current frame, you have previously created variable linkcnty that links the current frame to frcounty. You did this by typing

```
.frlink m:1 countyid, frame(frcounty) generate(linkcnty)
```

Thus, you can now type

```
.generate rel_income = income / frval(linkcnty, median_income)
```
income is an existing variable in the current frame. median_income is an existing variable in frcounty. rel_income will be a new variable in the current frame, containing the income of each person divided by the median income of the county in which they live.

Example 2: It is usual to name frames after dataset names and to name link variables after frame names. Here is an example of this, following the names used above:

```
use persons, clear
frame create county
frame county: use county
frlink m:1 countyid, frame(county)
generate rel_income = income / frval(county, median_income)
```

Domain lvar: the name of a variable created by frlink that links the current frame to another frame

Domain var: any variable (string or numeric) in the frame to which lvar links; varname abbreviation is allowed

Range: range of var, plus missing value (missing value is defined as . when var contains numeric data and "" when var contains string data; missing value is returned for observations in the current frame that are unmatched in the other frame)

```
frval(lvar, var, unm)
```

Description: the frval() function described above but with a third argument unm

frval() returns the value of var from the observation in the frame linked using lvar that matches the observation in the current frame and the value in unm if there is no matching observation.

For example, type

```
generate median_inc = frval(county, median_income, .a)
```

to create new variable median_inc in the current frame, containing median_income from the other frame, or .a when there is no matched observation in the other frame.

Domain lvar: the name of a variable created by frlink that links the current frame to another frame

Domain var: any variable (string or numeric) in the frame to which lvar links; varname abbreviation is allowed

Domain unm: any numeric value if var is numeric; any string value when var is string

Range: range of var, plus unm

```
__frval(frm, var, i)
```

Description: programmer’s version of frval()

It is useful for those wishing to write their own frlink and create special (or at least different) effects.

__frval() returns values of variables stored in other frames. It returns var’s ith observation (var[i]) from the frame frm; see [D] frames intro.

If i is outside the valid range of observations for the frame, __frval() returns missing.
For example, you have two datasets in memory. The current frame is named `default` and contains 57 observations. The other dataset, we will assume, is stored in frame `xdata`. It contains different variables but on the same 57 observations. The two datasets are in the same order so that observation 1 in `default` corresponds to observation 1 in `xdata`, observation 2 to observation 2, and so on. You can type

```
. generate hrlywage = income / _frval(xdata, hrswrked, n)
```

This will divide values of `income` stored in `default` by values of `hrswrked` stored in `xdata`.

The first thing to notice is that `_frval()`’s first two arguments are not expressions. You just type the name of the frame and the name of the variable without embedding them in quotes. We specified `xdata` for the frame name and and `hrswrked` for the variable name.

The second thing to notice is that the third argument is an expression. To emphasize that, let’s change the example. Assume that `xdata` contains 58 instead of 57 observations. Assume that observation 1 in `default` corresponds to observation 2 in `xdata`, observation 2 corresponds to observation 3, and so on. There is no observation in `default` that corresponds to observation 1 in `xdata`. In this case, you type

```
. generate hrlywage = income / _frval(xdata, hrswrked, _n+1)
```

These examples are artificial. You will normally use `_frval()` by creating a variable in `default` that contains the corresponding observation numbers in `xdata`. If the variable were called `xobsno`, then in the first example, `xobsno` would contain 1, 2, ..., 57.

In the second example, `xobsno` would contain 2, 3, ..., 58.

In another example, `xobsno` might contain 9, 6, ..., 32, which is to say, the numbers 2, 3, ..., 58, but permuted to reflect the datasets’ jumbled order.

In yet another example, `xobsno` might contain 9, 6, 9, ..., 32, which is to say, observation 1 and 3 in `default` both correspond to observation 9 in `xdata`. `xdata` in this example might record geographic location and in `default`, persons in observations 1 and 3 live in the same locale.

And in a final example, `xobsno` might contain all the above and missing values (._). The missing values would indicate observations in `default` that have no corresponding observation in `xdata`. If observations 7 and 11 contained missing, that means there would be no observations in `xdata` corresponding to observations 7 and 11 in `default`. `_frval()` has a second syntax that allows you to specify the value returned when there are no corresponding observations; see below.)

Regardless of the complexity of the example, the value of `xobsno` in observation \( j \) is the corresponding observation number \( i \) in `xdata`. Regardless of complexity, to create new variable `hrlywage` in `default`, you would type

```
. generate hrlywage = income / _frval(xdata, hrswrked, xobsno)
```

That leaves only the question of how to generate `xobsno` in all the above situations, and it is easy to do. See `[D] frlink`. 
There are two more things to know.

First, variables across frames are distinct. If the variable we have been calling `income` in `default` were named `x`, and the variable `hrswrked` in `xdata` were also named `x`, you would type

```
. generate hrlywage = x / _frval(xdata, x, xobsno)
```

Second, although we have demonstrated the use of `_frval()` with numeric variables, it works with string variables too. If `var` is a string variable name, `_frval()` returns a string result.

Domain `frm`: any existing framename
Domain `var`: any existing variable name in `frm`; varname abbreviation is allowed
Domain `i`: any numeric values including missing values even though the nonmissing values should be integers in the range 1 to `frm`’s `_N`; nonintegers will be interpreted as the corresponding integer obtained by truncation, and values outside the range will be treated as if they were missing value
Range: range of `var` in `frm` plus missing value; numeric missing value (`.`) when `var` is numeric, and string missing value (`""`) when `var` is string

```
    _frval(frm, var, i, v)
```

Description: the `_frval()` function described above but with a fourth argument `v`

`_frval()` returns values of variables stored in other frames. It returns `var`’s `i`th observation (`var[i]`) from the frame `frm`.

When `v` is specified, `_frval()` returns `v` if `var[i]` is missing or if `i` is outside the valid range of observations.

```
. generate hwage = income / _frval(xdata, hrswrked, xobsno, .z)
. generate hwage = income / _frval(xdata, hrswrked, xobsno, avg)
```

In the first case, `.z` is returned for observations in which `xobsno` contains values that are out of range. In the second case, the value recorded in variable `avg` is returned.

Domain `frm`: any existing framename
Domain `var`: any existing variable name in `frm`; varname abbreviation is allowed
Domain `i`: any numeric values including missing values even though the nonmissing values should be integers in the range 1 to `frm`’s `_N`; nonintegers will be interpreted as the corresponding integer obtained by truncation, and values outside the range will be treated as if they were missing value
Domain `v`: any numeric value when `var` is numeric; any string value when `var` is string (can be a constant or vary observation by observation)
Range: range of `var` in `frm` plus `v`

```
    has_eprop(name)
```

Description: 1 if `name` appears as a word in `e(properties)`; otherwise, 0
Domain: names
Range: 0 or 1
inlist(z,a,b,...)
Description: 1 if z is a member of the remaining arguments; otherwise, 0
All arguments must be reals or all must be strings. The number of arguments is between 2 and 250 for reals and between 2 and 10 for strings.
Domain: all reals or all strings
Range: 0 or 1

inrange(z,a,b)
Description: 1 if it is known that a ≤ z ≤ b; otherwise, 0
The following ordered rules apply:
z ≥ . returns 0.
a ≥ . and b = . returns 1.
a ≥ . returns 1 if z ≤ b; otherwise, it returns 0.
b ≥ . returns 1 if a ≤ z; otherwise, it returns 0.
Otherwise, 1 is returned if a ≤ z ≤ b.
If the arguments are strings, “.” is interpreted as "".
Domain: all reals or all strings
Range: 0 or 1

irecode(x,x_1,x_2,x_3,...,x_n)
Description: missing if x is missing or x_1,...,x_n is not weakly increasing; 0 if x ≤ x_1; 1 if x_1 < x ≤ x_2; 2 if x_2 < x ≤ x_3; ...; n if x > x_n
Also see autocode() and recode() for other styles of recode functions.
Domain x: −8e+307 to 8e+307
Domain x_i: −8e+307 to 8e+307
Range: nonnegative integers

matrix(exp)
Description: restricts name interpretation to scalars and matrices; see scalar()
Domain: any valid expression
Range: evaluation of exp

maxbyte()
Description: the largest value that can be stored in storage type byte
This function takes no arguments, but the parentheses must be included.
Range: one integer number

maxdouble()
Description: the largest value that can be stored in storage type double
This function takes no arguments, but the parentheses must be included.
Range: one double-precision number

maxfloat()
Description: the largest value that can be stored in storage type float
This function takes no arguments, but the parentheses must be included.
Range: one floating-point number
maxint()  
Description:  the largest value that can be stored in storage type int  
This function takes no arguments, but the parentheses must be included.  
Range:  one integer number

maxlong()  
Description:  the largest value that can be stored in storage type long  
This function takes no arguments, but the parentheses must be included.  
Range:  one integer number

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

minbyte()  
Description:  the smallest value that can be stored in storage type byte  
This function takes no arguments, but the parentheses must be included.  
Range:  one integer number

mindouble()  
Description:  the smallest value that can be stored in storage type double  
This function takes no arguments, but the parentheses must be included.  
Range:  one double-precision number

minfloat()  
Description:  the smallest value that can be stored in storage type float  
This function takes no arguments, but the parentheses must be included.  
Range:  one floating-point number

minint()  
Description:  the smallest value that can be stored in storage type int  
This function takes no arguments, but the parentheses must be included.  
Range:  one integer number

minlong()  
Description:  the smallest value that can be stored in storage type long  
This function takes no arguments, but the parentheses must be included.  
Range:  one integer number
missing($x_1, x_2, \ldots, x_n$)
Description: 1 if any $x_i$ evaluates to missing; otherwise, 0
Stata has two concepts of missing values: a numeric missing value (., .a, .b, \ldots, .z) and a string missing value (" "). missing() returns 1 (meaning true) if any expression $x_i$ evaluates to missing. If $x$ is numeric, missing($x$) is equivalent to $x \geq \ldots$. If $x$ is string, missing($x$) is equivalent to $x=="\"$.
Domain $x_i$: any string or numeric expression
Range: 0 and 1

$r(name)$
Description: the value of the stored result $r(name)$; see [U] 18.8 Accessing results calculated by other programs
\[ r(name) = \begin{cases} \text{scalar missing} & \text{if the stored result does not exist} \\ \text{specified matrix} & \text{if the stored result is a matrix} \\ \text{scalar numeric value} & \text{if the stored result is a scalar that can be interpreted as a number} \end{cases} \]
Domain: names
Range: strings, scalars, matrices, or missing

recode($x, x_1, x_2, \ldots, x_n$)
Description: missing if $x_1, x_2, \ldots, x_n$ is not weakly increasing; $x$ if $x$ is missing; $x_1$ if $x \leq x_1$; $x_2$ if $x \leq x_2, \ldots$; otherwise, $x_n$ if $x > x_1, x_2, \ldots, x_{n-1}$. $x_i \geq \ldots$ is interpreted as $x_i = +\infty$
Also see autocode() and irecode() for other styles of recode functions.
Domain $x$: $-8e+307$ to $8e+307$ or missing
Domain $x_1$: $-8e+307$ to $8e+307$
Domain $x_2$: $x_1$ to $8e+307$
\ldots
Domain $x_n$: $x_{n-1}$ to $8e+307$
Range: $x_1, x_2, \ldots, x_n$ or missing

replay()
Description: 1 if the first nonblank character of local macro ‘0’ is a comma, or if ‘0’ is empty
This is a function for use by programmers writing estimation commands; see [P] ereturn.
Range: integers 0 and 1, meaning false and true, respectively

return($name$)
Description: the value of the to-be-stored result $r(name)$; see [P] return
\[ \begin{align*} r(name) &= \text{scalar missing} & \text{if the stored result does not exist} \\ r(name) &= \text{specified matrix} & \text{if the stored result is a matrix} \\ r(name) &= \text{scalar numeric value} & \text{if the stored result is a scalar} \end{align*} \]
Domain: names
Range: strings, scalars, matrices, or missing
\( s(name) \)

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

\( s(name) = . \) if the stored result does not exist

Domain: names
Range: strings or missing

\( \text{scalar}(exp) \)

Description: restricts name interpretation to scalars and matrices

Names in expressions can refer to names of variables in the dataset, names of matrices, or names of scalars. Matrices and scalars can have the same names as variables in the dataset. If names conflict, Stata assumes that you are referring to the name of the variable in the dataset.

\( \text{matrix}() \) and \( \text{scalar}() \) explicitly state that you are referring to matrices and scalars. \( \text{matrix}() \) and \( \text{scalar}() \) are the same function; scalars and matrices may not have the same names and so cannot be confused. Typing \( \text{scalar}(x) \) makes it clear that you are referring to the scalar or matrix named \( x \) and not the variable named \( x \), should there happen to be a variable of that name.

Domain: any valid expression
Range: evaluation of \( exp \)

\( \text{smallestdouble}() \)

Description: the smallest double-precision number greater than zero

If \( 0 < d < \text{smallestdouble}() \), then \( d \) does not have full double precision; these are called the denormalized numbers. This function takes no arguments, but the parentheses must be included.

Range: a double-precision number close to 0

References


Also see

[FN] Functions by category
[D] egen — Extensions to generate
[D] generate — Create or change contents of variable
[M-4] Programming — Programming functions
[U] 13.3 Functions
<table>
<thead>
<tr>
<th>Contents</th>
<th>Functions</th>
<th>Remarks and examples</th>
<th>Methods and formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>References</td>
<td>Also see</td>
<td></td>
</tr>
</tbody>
</table>

**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)**: $\chi^2$, with df degrees of freedom, random variates.
- **rexpontial(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()** and **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()** and **rnormal(m)**: 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,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()** and **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].
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rweibull(a, b)</td>
<td>Weibull variates with shape a and scale b</td>
</tr>
<tr>
<td>rweibull(a, b, g)</td>
<td>Weibull variates with shape a, scale b, and location g</td>
</tr>
<tr>
<td>rweibullph(a, b)</td>
<td>Weibull (proportional hazards) variates with shape a and scale b</td>
</tr>
<tr>
<td>rweibullph(a, b, g)</td>
<td>Weibull (proportional hazards) variates with shape a, scale b, and location g</td>
</tr>
</tbody>
</table>

### Functions

The term “pseudorandom number” is used to emphasize that the numbers are generated by formulas and are thus not truly random. From now on, we will drop the “pseudo” and just say random numbers.

For information on setting the random-number seed, see [R] set seed.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>runiform()</td>
<td>Uniformly distributed random variates over the interval (0, 1)</td>
<td>c(epsdouble) to 1 − c(epsdouble)</td>
<td></td>
</tr>
<tr>
<td>runiform(a, b)</td>
<td>Uniformly distributed random variates over the interval (a, b)</td>
<td>c(mindouble) to c(maxdouble)</td>
<td>a + c(epsdouble) to b − c(epsdouble)</td>
</tr>
<tr>
<td>runiformint(a, b)</td>
<td>Uniformly distributed random integer variates on the interval [a, b]</td>
<td>−2^53 to 2^53 (may be nonintegral)</td>
<td>−2^53 to 2^53 (may be nonintegral)</td>
</tr>
<tr>
<td>rbeta(a, b)</td>
<td>Beta(a,b) random variates, where a and b are the beta distribution shape parameters</td>
<td>0.05 to 1e+5</td>
<td>0 to 1 (exclusive)</td>
</tr>
</tbody>
</table>
rbinomial(n,p)
Description: binomial(n,p) random variates, where n is the number of trials and p is the success probability
Besides using the standard methodology for generating random variates from a given distribution, rbinomial() uses the specialized algorithms of Kachitvichyanukul (1982), Kachitvichyanukul and Schmeiser (1988), and Kemp (1986).
Domain n: 1 to 1e+11
Domain p: 1e–8 to 1–1e–8
Range: 0 to n

rcauchy(a,b)
Description: Cauchy(a,b) random variates, where a is the location parameter and b is the scale parameter
Domain a: −1e+300 to 1e+300
Domain b: 1e–100 to 1e+300
Range: c(mindouble) to c(maxdouble)

rchi2(df)
Description: χ², with df degrees of freedom, random variates
Domain df: 2e–4 to 2e+8
Range: 0 to c(maxdouble)

rexponential(b)
Description: exponential random variates with scale b
Domain b: 1e–323 to 8e+307
Range: 1e–323 to 8e+307

rgamma(a,b)
Description: gamma(a,b) random variates, where a is the gamma shape parameter and b is the scale parameter
Methods for generating gamma variates are taken from Ahrens and Dieter (1974), Best (1983), and Schmeiser and Lal (1980).
Domain a: 1e–4 to 1e+8
Domain b: c(smallestdouble) to c(maxdouble)
Range: 0 to c(maxdouble)

rhypergeometric(N,K,n)
Description: hypergeometric random variates
The distribution parameters are integer valued, where N is the population size, K is the number of elements in the population that have the attribute of interest, and n is the sample size.
Besides using the standard methodology for generating random variates from a given distribution, rhypergeometric() uses the specialized algorithms of Kachitvichyanukul (1982) and Kachitvichyanukul and Schmeiser (1985).
Domain N: 2 to 1e+6
Domain K: 1 to N–1
Domain n: 1 to N–1
Range: max(0,n – N + K) to min(K,n)
random-number functions

\( \texttt{rigaussian}(m,a) \)
\[\text{Description: inverse Gaussian random variates with mean } m \text{ and shape parameter } a\]
\textit{rigaussian()} is based on a method proposed by Michael, Schucany, and Haas (1976).
\[\text{Domain } m: \ 1\text{e}–10 \text{ to } 1000\]
\[\text{Domain } a: \ 0.001 \text{ to } 1\text{e}+10\]
\[\text{Range: } 0 \text{ to } c(\text{maxdouble})\]

\( \texttt{rlaplace}(m,b) \)
\[\text{Description: Laplace}(m,b) \text{ random variates with mean } m \text{ and scale parameter } b\]
\[\text{Domain } m: \ -1\text{e}+300 \text{ to } 1\text{e}+300\]
\[\text{Domain } b: \ 1\text{e}–300 \text{ to } 1\text{e}+300\]
\[\text{Range: } c(\text{mindouble}) \text{ to } c(\text{maxdouble})\]

\( \texttt{rlogistic()} \)
\[\text{Description: logistic variates with mean 0 and standard deviation } \pi/\sqrt{3}\]
\[\text{The variates } x \text{ are generated by } x = \text{invlogistic}(0,1,u), \text{ where } u \text{ is a random uniform(0,1) variate.}\]
\[\text{Range: } c(\text{mindouble}) \text{ to } c(\text{maxdouble})\]

\( \texttt{rlogistic}(s) \)
\[\text{Description: logistic variates with mean 0, scale } s, \text{ and standard deviation } s\pi/\sqrt{3}\]
\[\text{The variates } x \text{ are generated by } x = \text{invlogistic}(0,s,u), \text{ where } u \text{ is a random uniform(0,1) variate.}\]
\[\text{Domain } s: \ 0 \text{ to } c(\text{maxdouble})\]
\[\text{Range: } c(\text{mindouble}) \text{ to } c(\text{maxdouble})\]

\( \texttt{rlogistic}(m,s) \)
\[\text{Description: logistic variates with mean } m, \text{ scale } s, \text{ and standard deviation } s\pi/\sqrt{3}\]
\[\text{The variates } x \text{ are generated by } x = \text{invlogistic}(m,s,u), \text{ where } u \text{ is a random uniform(0,1) variate.}\]
\[\text{Domain } m: \ c(\text{mindouble}) \text{ to } c(\text{maxdouble})\]
\[\text{Domain } s: \ 0 \text{ to } c(\text{maxdouble})\]
\[\text{Range: } c(\text{mindouble}) \text{ to } c(\text{maxdouble})\]

\( \texttt{rnbinomial}(n,p) \)
\[\text{Description: negative binomial random variates}\]
\[\text{If } n \text{ is integer valued, } \texttt{rnbinomial}() \text{ returns the number of failures before the } n\text{th success, where the probability of success on a single trial is } p. \ n \text{ can also be nonintegral.}\]
\[\text{Domain } n: \ 1\text{e}–4 \text{ to } 1\text{e}+5\]
\[\text{Domain } p: \ 1\text{e}–4 \text{ to } 1–1\text{e}–4\]
\[\text{Range: } 0 \text{ to } 2^{53} – 1\]
### rnormal()

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

Range: c(mindouble) to c(maxdouble)

### rnormal(m)

Description: normal(m,1) (Gaussian) random variates, where \( m \) is the mean and the standard deviation is 1

Domain \( m \): c(mindouble) to c(maxdouble)

Range: c(mindouble) to c(maxdouble)

### rnormal(m,s)

Description: normal(m,s) (Gaussian) random variates, where \( m \) is the mean and \( s \) is the standard deviation

The methods for generating normal (Gaussian) random variates are taken from Knuth (1998, 122–128); Marsaglia, MacLaren, and Bray (1964); and Walker (1977).

Domain \( m \): c(mindouble) to c(maxdouble)

Domain \( s \): 0 to c(maxdouble)

Range: c(mindouble) to c(maxdouble)

### rpoisson(m)

Description: Poisson(m) random variates, where \( m \) is the distribution mean

Poisson variates are generated using the probability integral transform methods of Kemp and Kemp (1990, 1991) and the method of Kachitvichyanukul (1982).

Domain \( m \): 1e–6 to 1e+11

Range: 0 to 2^{53} – 1

### rt(df)

Description: Student’s \( t \) random variates, where \( df \) is the degrees of freedom

Student’s \( t \) variates are generated using the method of Kinderman and Monahan (1977, 1980).

Domain \( df \): 1 to 2^{53} – 1

Range: c(mindouble) to c(maxdouble)

### rweibull(a,b)

Description: Weibull variates with shape \( a \) and scale \( b \)

The variates \( x \) are generated by \( x = \text{invweibulltail}(a, b, 0, u) \), where \( u \) is a random uniform(0,1) variate.

Domain \( a \): 0.01 to 1e+6

Domain \( b \): 1e–323 to 8e+307

Range: 1e–323 to 8e+307
\textbf{rweibull(}a, b, g\textbf{)}

Description: Weibull variates with shape \(a\), scale \(b\), and location \(g\)

The variates \(x\) are generated by \(x = \text{invweibulltail}(a, b, g, u)\), where \(u\) is a random uniform(0,1) variate.

Domain \(a\): 0.01 to 1e+6
Domain \(b\): 1e–323 to 8e+307
Domain \(g\): \(-8e+307\) to \(8e+307\)
Range: \(g + c(\text{epsdouble})\) to 8e+307

\textbf{rweibullph(}a, b\textbf{)}

Description: Weibull (proportional hazards) variates with shape \(a\) and scale \(b\)

The variates \(x\) are generated by \(x = \text{invweibullphtail}(a, b, 0, u)\), where \(u\) is a random uniform(0,1) variate.

Domain \(a\): 0.01 to 1e+6
Domain \(b\): 1e–323 to 8e+307
Range: 1e–323 to 8e+307

\textbf{rweibullph(}a, b, g\textbf{)}

Description: Weibull (proportional hazards) variates with shape \(a\), scale \(b\), and location \(g\)

The variates \(x\) are generated by \(x = \text{invweibullphtail}(a, b, g, u)\), where \(u\) is a random uniform(0,1) variate.

Domain \(a\): 0.01 to 1e+6
Domain \(b\): 1e–323 to 8e+307
Domain \(g\): \(-8e+307\) to \(8e+307\)
Range: \(g + c(\text{epsdouble})\) to 8e+307

\textbf{Remarks and examples}

It is ironic that the first thing to note about random numbers is how to make them reproducible. Before using a random-number function, type

\texttt{set seed \#}

where \# is any integer between 0 and \(2^{31} - 1\), inclusive, to draw the same sequence of random numbers. It does not matter which integer you choose as your seed; they are all equally good. See \texttt{R} \textit{set seed}.

\texttt{runiform()} is the basis for all the other random-number functions because all the other random-number functions transform uniform \((0, 1)\) random numbers to the specified distribution.

\texttt{runiform()} implements the 64-bit Mersenne Twister (\texttt{mt64}), the stream 64-bit Mersenne Twister (\texttt{mt64s}), and the 32-bit “keep it simple stupid” (\texttt{kiss32}) random-number generators (RNGs) for generating uniform \((0, 1)\) random numbers. \texttt{runiform()} uses the \texttt{mt64} RNG by default.

\texttt{runiform()} uses the \texttt{kiss32} RNG only when the user version is less than 14 or when the RNG has been set to \texttt{kiss32}; see \texttt{P} \textit{version} for details about setting the user version. We recommend that you do not change the default RNG, but see \texttt{R} \textit{set rng} for details.
Technical note

Although we recommend that you use `runiform()`, we made generator-specific versions of `runiform()` available for advanced users who want to hardcode their generator choice. The function `runiform_mt64()` always uses the mt64 RNG to generate uniform (0, 1) random numbers, the function `runiform_mt64s()` always uses the mt64s RNG to generate uniform (0, 1) random numbers, the function `runiform_kiss32()` always uses the kiss32 RNG to generate uniform (0, 1) random numbers. In fact, generator-specific versions are available for all the implemented distributions. For example, `rnormal_mt64()`, `rnormal_mt64s`, and `rnormal_kiss32()` use transforms of mt64, mt64s, and kiss32 uniform variates, respectively, to generate standard normal variates.

Technical note

Both the mt64 and the kiss32 RNGs produce uniform variates that pass many tests for randomness. Many researchers prefer the mt64 to the kiss32 RNG because the mt64 generator has a longer period and a finer resolution and requires a higher dimension before patterns appear; see Matsumoto and Nishimura (1998).

The mt64 RNG has a period of $2^{19937} - 1$ and a resolution of $2^{-53}$; see Matsumoto and Nishimura (1998). Each stream of the mt64s RNG contains $2^{128}$ random numbers, and mt64s has a resolution of $2^{-53}$; see Haramoto et al. (2008). The kiss32 RNG has a period of about $2^{126}$ and a resolution of $2^{-32}$; see Methods and formulas below.

Technical note

This technical note explains how to restart a RNG from its current spot.

The current spot in the sequence of a RNG is part of the state of a RNG. If you tell me the state of a RNG, I know where it is in its sequence, and I can compute the next random number. The state of a RNG is a complicated object that requires more space than the integers used to seed a generator. For instance, an mt64 state is a 5011-digit, base-16 number preceded by three letters.

If you want to restart a RNG from where it left off, you should store the current state in a macro and then set the state of the RNG when you want to restart it. For example, suppose we set a seed and draw some random numbers.

```
    . set obs 3
    Number of observations (N) was 0, now 3.
    . set seed 12345
    . generate x = runiform()
    . list x
         +------+
        | x     |
        |-------|
        | .3576297 |
        | .4004426 |
        | .6893833 |
        +-------+
```


We store the state of the RNG so that we can pick up right here in the sequence.

```
. local rngstate "c(rngstate)"
```

We draw some more random numbers.

```
. replace x = runiform()
  (3 real changes made)
. list x

   x
  1.  .5597356
  2.  .5744513
  3.  .2076905
```

Now, we set the state of the RNG to where it was and draw those same random numbers again.

```
. set rngstate 'rngstate'
. replace x = runiform()
  (0 real changes made)
. list x

   x
  1.  .5597356
  2.  .5744513
  3.  .2076905
```

## Methods and formulas

All the nonuniform generators are based on the uniform \texttt{mt64}, \texttt{mt64s}, and \texttt{kiss32} RNGs.

The \texttt{mt64} RNG is well documented in Matsumoto and Nishimura (1998) and on their website [http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html](http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html). The \texttt{mt64} RNG implements the 64-bit version discussed at [http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt64.html](http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt64.html). The \texttt{mt64s} RNG is based on a method proposed by Haramoto et al. (2008). The default seed of all three RNGs is 123456789.

### kiss32 generator

The \texttt{kiss32} uniform RNG implemented in \texttt{runiform()} is based on George Marsaglia’s (G. Marsaglia, 1994, pers. comm.) 32-bit pseudorandom-integer generator \texttt{kiss32}. The integer \texttt{kiss32} RNG is composed of two 32-bit pseudorandom-integer generators and two 16-bit integer generators (combined to make one 32-bit integer generator). The four generators are defined by the recursions

\begin{align}
  x_n &= 69069 \cdot x_{n-1} + 1234567 \mod 2^{32} \\
  y_n &= y_{n-1}(I + L^{13})(I + R^{17})(I + L^5) \\
  z_n &= 65184(z_{n-1} \mod 2^{16}) + \text{int}(z_{n-1}/2^{16}) \\
  w_n &= 63663(w_{n-1} \mod 2^{16}) + \text{int}(w_{n-1}/2^{16})
\end{align}
In (2), the 32-bit word \( y_n \) is viewed as a \( 1 \times 32 \) binary vector; \( L \) is the \( 32 \times 32 \) matrix that produces a left shift of one (\( L \) has 1s on the first left subdiagonal, 0s elsewhere); and \( R \) is \( L \) transpose, affecting a right shift by one. In (3) and (4), \( \text{int}(x) \) is the integer part of \( x \).

The integer \texttt{kiss32 RNG} produces the 32-bit random integer

\[
R_n = x_n + y_n + z_n + 2^{16} w_n \mod 2^{32}
\]

The \texttt{kiss32 uniform RNG} implemented in \texttt{runiform()} takes the output from the integer \texttt{kiss32 RNG} and divides it by \( 2^{32} \) to produce a real number on the interval \((0, 1)\). (Zeros are discarded, and the first nonzero result is returned.)

The recursion (5)–(8) have, respectively, the periods

\[
\begin{align*}
2^{32} & \quad \text{(5)} \\
2^{32} - 1 & \quad \text{(6)} \\
(65184 \cdot 2^{16} - 2)/2 & \approx 2^{31} \quad \text{(7)} \\
(63663 \cdot 2^{16} - 2)/2 & \approx 2^{31} \quad \text{(8)}
\end{align*}
\]

Thus the overall period for the integer \texttt{kiss32 RNG} is

\[
2^{32} \cdot (2^{32} - 1) \cdot (65184 \cdot 2^{15} - 1) \cdot (63663 \cdot 2^{15} - 1) \approx 2^{126}
\]

When Stata first comes up, it initializes the four recursions in \texttt{kiss32} by using the seeds

\[
\begin{align*}
x_0 &= 123456789 \\
y_0 &= 521288629 \\
z_0 &= 362436069 \\
w_0 &= 2262615
\end{align*}
\]

Successive calls to the \texttt{kiss32 uniform RNG} implemented in \texttt{runiform()} then produce the sequence

\[
\frac{R_1}{2^{32}}, \quad \frac{R_2}{2^{32}}, \quad \frac{R_3}{2^{32}}, \quad \ldots
\]

Hence, the \texttt{kiss32 uniform RNG} implemented in \texttt{runiform()} gives the same sequence of random numbers in every Stata session (measured from the start of the session) unless you reinitialize the seed. The full seed is the set of four numbers \((x, y, z, w)\), but you can reinitialize the seed by simply issuing the command

\[
. \text{ set seed #}
\]

where \# is any integer between 0 and \( 2^{31} - 1 \), inclusive. When this command is issued, the initial value \( x_0 \) is set equal to \#, and the other three recursions are restarted at the seeds \( y_0, z_0, \) and \( w_0 \) given above. The first 100 random numbers are discarded, and successive calls to the \texttt{kiss32 uniform RNG} implemented in \texttt{runiform()} give the sequence

\[
\frac{R'_{101}}{2^{32}}, \quad \frac{R'_{102}}{2^{32}}, \quad \frac{R'_{103}}{2^{32}}, \quad \ldots
\]
However, if the command

```
    . set seed 123456789
```

is given, the first 100 random numbers are not discarded, and you get the same sequence of random numbers that the `kiss32` RNG produces when Stata restarts; also see [R] `set seed`.

## Acknowledgments

We thank the late George Marsaglia, formerly of Florida State University, for providing his `kiss32` RNG.

We thank John R. Gleason (retired) of Syracuse University for directing our attention to Wichura (1988) for calculating the cumulative normal density accurately, for sharing his experiences about techniques with us, and for providing C code to make the calculations.

We thank Makoto Matsumoto and Takuji Nishimura for deriving the Mersenne Twister and distributing their code for their generator so that it could be rapidly and effectively tested.

## References


Also see

[FN] Functions by category

[D] egen — Extensions to generate

[D] generate — Create or change contents of variable

[R] set rng — Set which random-number generator (RNG) to use

[R] set rngstream — Specify the stream for the stream random-number generator

[R] set seed — Specify random-number seed and state

[M-5] runiform( ) — Uniform and nonuniform pseudorandom variates

[U] 13.3 Functions
Selecting time-span functions

Contents Functions Also see

Contents

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

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

Functions

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

You must have previously \text{tsset} the data to use \text{tin}(); see \text{[TS]} \text{tsset}. When you \text{tsset} the data, you specify a time variable, \( t \), and the format on \( t \) states how it is recorded. You type \( d_1 \) and \( d_2 \) according to that format.

If \( t \) has a \%tc format, you could type \text{tin}(5jan1992 11:15, 14apr2002 12:25).

If \( t \) has a \%td format, you could type \text{tin}(5jan1992, 14apr2002).

If \( t \) has a \%tw format, you could type \text{tin}(1985w1, 2002w15).

If \( t \) has a \%tm format, you could type \text{tin}(1985m1, 2002m4).

If \( t \) has a \%tq format, you could type \text{tin}(1985q1, 2002q2).

If \( t \) has a \%th format, you could type \text{tin}(1985h1, 2002h1).

If \( t \) has a \%ty format, you could type \text{tin}(1985, 2002).

If \( t \) has a \%tb format, you could type \text{tin}(5jan1992, 14apr2002). This will work as expected even if the arguments of \text{tin}() are not business days.

Otherwise, \( t \) is just a set of integers, and you could type \text{tin}(12, 38).

The details of the \%t format do not matter. If your \( t \) is formatted \%tdmm/dd/yy so that 5jan1992 displays as 1/5/92, you would still type the date in day–month–year order: \text{tin}(5jan1992, 14apr2002).

Domain \( d_1 \): date or time literals or strings recorded in units of \( t \) previously \text{tsset} or blank to indicate no minimum date

Domain \( d_2 \): date or time literals or strings recorded in units of \( t \) previously \text{tsset} or blank to indicate no maximum date

Range: 0 and 1, 1 \( \Rightarrow \) true
twithin($d_1, d_2$)
Description: $true$ if $d_1 < t < d_2$, where $t$ is the time variable previously $t$set

See $tin()$ above; $twithin()$ is similar, except the range is exclusive.

Domain $d_1$: date or time literals or strings recorded in units of $t$ previously $t$set or blank to indicate no minimum date
Domain $d_2$: date or time literals or strings recorded in units of $t$ previously $t$set or blank to indicate no maximum date
Range: 0 and 1, 1 $\Rightarrow$ $true$

Also see

[FN] Functions by category
[D] egen — Extensions to generate
[D] generate — Create or change contents of variable
[U] 13.3 Functions
## Statistical functions

<table>
<thead>
<tr>
<th>Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>betaden(a, b, x)</code></td>
<td>the probability density of the beta distribution, where <code>a</code> and <code>b</code> are</td>
</tr>
<tr>
<td></td>
<td>the shape parameters; 0 if <code>x &lt; 0</code> or <code>x &gt; 1</code></td>
</tr>
<tr>
<td><code>binomial(n, k, θ)</code></td>
<td>the probability of observing <code>floor(k)</code> or fewer successes in <code>floor(n)</code></td>
</tr>
<tr>
<td></td>
<td>trials when the probability of a success on one trial is <code>θ</code>; 0 if <code>k &lt; 0</code></td>
</tr>
<tr>
<td></td>
<td>or 1 if <code>k &gt; n</code></td>
</tr>
<tr>
<td><code>binomialp(n, k, p)</code></td>
<td>the probability of observing <code>floor(k)</code> successes in <code>floor(n)</code> trials</td>
</tr>
<tr>
<td></td>
<td>when the probability of a success on one trial is <code>p</code></td>
</tr>
<tr>
<td><code>binomialtail(n, k, θ)</code></td>
<td>the probability of observing <code>floor(k)</code> or more successes in <code>floor(n)</code></td>
</tr>
<tr>
<td></td>
<td>trials when the probability of a success on one trial is <code>θ</code>; 1 if <code>k &lt; 0</code></td>
</tr>
<tr>
<td></td>
<td>or 0 if <code>k &gt; n</code></td>
</tr>
<tr>
<td><code>binormal(h, k, ρ)</code></td>
<td>the joint cumulative distribution <code>Φ(h, k, ρ)</code> of bivariate normal with</td>
</tr>
<tr>
<td></td>
<td>correlation <code>ρ</code></td>
</tr>
<tr>
<td><code>cauchy(a, b, x)</code></td>
<td>the cumulative Cauchy distribution with location parameter <code>a</code> and scale</td>
</tr>
<tr>
<td></td>
<td>parameter <code>b</code></td>
</tr>
<tr>
<td><code>cauchyden(a, b, x)</code></td>
<td>the probability density of the Cauchy distribution with location parameter</td>
</tr>
<tr>
<td></td>
<td><code>a</code> and scale parameter <code>b</code></td>
</tr>
<tr>
<td><code>cauchytail(a, b, x)</code></td>
<td>the reverse cumulative (upper tail or survivor) Cauchy distribution with</td>
</tr>
<tr>
<td></td>
<td>location parameter <code>a</code> and scale parameter <code>b</code></td>
</tr>
<tr>
<td><code>chi2(df, x)</code></td>
<td>the cumulative $\chi^2$ distribution with <code>df</code> degrees of freedom; 0 if</td>
</tr>
<tr>
<td></td>
<td><code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>chi2den(df, x)</code></td>
<td>the probability density of the $\chi^2$ distribution with <code>df</code> degrees of</td>
</tr>
<tr>
<td></td>
<td>freedom; 0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>chi2tail(df, x)</code></td>
<td>the reverse cumulative (upper tail or survivor) $\chi^2$ distribution with</td>
</tr>
<tr>
<td></td>
<td><code>df</code> degrees of freedom; 1 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dgammapda(a, x)</code></td>
<td>$\frac{\partial P(a,x)}{\partial a}$, where $P(a,x) = \text{gammap}(a,x)$;</td>
</tr>
<tr>
<td></td>
<td>0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dgammapdada(a, x)</code></td>
<td>$\frac{\partial^2 P(a,x)}{\partial a^2}$, where $P(a,x) = \text{gammap}(a,x)$;</td>
</tr>
<tr>
<td></td>
<td>0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dgammapdadx(a, x)</code></td>
<td>$\frac{\partial^2 P(a,x)}{\partial a \partial x}$, where $P(a,x) = \text{gammap}(a,x)$;</td>
</tr>
<tr>
<td></td>
<td>0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dgammapdx(a, x)</code></td>
<td>$\frac{\partial P(a,x)}{\partial x}$, where $P(a,x) = \text{gammap}(a,x)$;</td>
</tr>
<tr>
<td></td>
<td>0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dgammapdxdx(a, x)</code></td>
<td>$\frac{\partial^2 P(a,x)}{\partial x^2}$, where $P(a,x) = \text{gammap}(a,x)$;</td>
</tr>
<tr>
<td></td>
<td>0 if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>dunnettprob(k, df, x)</code></td>
<td>the cumulative multiple range distribution that is used in Dunnett's</td>
</tr>
<tr>
<td></td>
<td>multiple-comparison method with <code>k</code> ranges and <code>df</code> degrees of freedom; 0</td>
</tr>
<tr>
<td></td>
<td>if <code>x &lt; 0</code></td>
</tr>
<tr>
<td><code>exponential(b, x)</code></td>
<td>the cumulative exponential distribution with scale <code>b</code></td>
</tr>
<tr>
<td><code>exponentialden(b, x)</code></td>
<td>the probability density function of the exponential distribution with</td>
</tr>
<tr>
<td></td>
<td>scale <code>b</code></td>
</tr>
<tr>
<td><code>exponentialtail(b, x)</code></td>
<td>the reverse cumulative exponential distribution with scale <code>b</code></td>
</tr>
</tbody>
</table>
F(df1, df2, f)  the cumulative F distribution with df1 numerator and df2 denominator degrees of freedom: \( F(df1, df2, f) = \int_0^f F_{df1, df2}(t) dt \); 0 if \( f < 0 \)

Fden(df1, df2, f)  the probability density function of the F distribution with df1 numerator and df2 denominator degrees of freedom; 0 if \( f < 0 \)

Ftail(df1, df2, f)  the reverse cumulative (upper tail or survivor) F distribution with df1 numerator and df2 denominator degrees of freedom; 1 if \( f < 0 \)

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

gammmap(a, x)  the cumulative gamma distribution with shape parameter a; 0 if \( x < 0 \)

gammapntail(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 \( x < 0 \)

igaussianden(m, a, x)  the probability density of the inverse Gaussian distribution with mean m and shape parameter a; 0 if \( x < 0 \)

igaussiantail(m, a, x)  the reverse cumulative (upper tail or survivor) inverse Gaussian distribution with mean m and shape parameter a; 1 if \( x < 0 \)

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 floor\((k)\) or fewer successes in 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 floor\((k)\) or more successes in floor\((n)\) trials is \( p \)

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

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

invchi2(df, p)  the inverse of chi2(): if \( \text{chi2}(df, x) = p \), then \( \text{invchi2}(df, p) = x \)

invchi2tail(df, p)  the inverse of chi2tail(): if \( \text{chi2tail}(df, x) = p \), 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

invasymp(df, p)  the inverse cumulative asymptotic normal distribution with scale \( b \): if \( \text{asymnormal}(b, x) = p \), then \( \text{invasymp}(b, p) = x \)
\textbf{invexponentialtail}(b,p) \quad \text{the inverse reverse cumulative exponential distribution with scale } b:\ 
\text{if exponentialtail}(b,x) = p, \text{ then } \text{invexponentialtail}(b,p) = x

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

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

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

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

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

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

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

\textbf{invgaussianantail}(m,a,p) \quad \text{the inverse of } \text{igaussian}(): \text{ if } \text{igaussianantail}(m,a,x) = p, \text{ then } \text{invgaussianantail}(m,a,p) = x

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

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

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

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

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

\textbf{invlogisticx}(p) \quad \text{the inverse cumulative logistic distribution: if } \text{logisticx}(x) = p, \text{ then } \text{invlogisticx}(p) = x

\textbf{invlogisticx}(s,p) \quad \text{the inverse cumulative logistic distribution: if } \text{logisticx}(s,x) = p, \text{ then } \text{invlogisticx}(s,p) = x

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

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

\textbf{invnbinomialx}(n,k,q) \quad \text{the value of the negative binomial parameter, } p, \text{ such that } q = \text{nbinomialx}(n,k,p)

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

\textbf{invnchix2}(df,np,p) \quad \text{the inverse cumulative noncentral } \chi^2 \text{ distribution: if } \text{nchix2}(df,np,x) = p, \text{ then } \text{invnchix2}(df,np,p) = x
The inverse cumulative noncentral $F$ distribution: if $\text{nF}(df_1, df_2, np, f) = p$, then $\text{invnF}(df_1, df_2, np, p) = f$

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

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

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 $\text{t}(df, t) = p$, then $\text{invt}(df, p) = t$

The inverse reverse cumulative (upper tail or survivor) Student’s $t$ distribution: if $\text{ttail}(df, t) = p$, 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 $\text{invweibulltail}(a, b, g, p) = x$
laplace($m, b, x$)  the cumulative Laplace distribution with mean $m$ and scale parameter $b$
laplace_den($m, b, x$)  the probability density of the Laplace distribution with mean $m$ and scale parameter $b$
laplace_tail($m, b, x$)  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
lnigaussianden($m, a, x$)  the natural logarithm of the inverse Gaussian density with mean $m$ and shape parameter $a$
lniwishartden($df, V, X$)  the natural logarithm of the density of the inverse Wishart distribution; missing if $df \leq n - 1$
lnlaplace_den($m, b, x$)  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
lnnormal_den($z$)  the natural logarithm of the standard normal density, $N(0, 1)$
lnnormal_den($x, \sigma$)  the natural logarithm of the normal density with mean 0 and standard deviation $\sigma$
lnnormal_den($m, \mu, \sigma$)  the natural logarithm of the normal density with mean $\mu$ and standard deviation $\sigma$
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 $s\pi/\sqrt{3}$
logistic_den($x$)  the density of the logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
logistic_den($s, x$)  the density of the logistic distribution with mean 0, scale $s$, and standard deviation $s\pi/\sqrt{3}$
logistic_den($m, s, x$)  the density of the logistic distribution with mean $m$, scale $s$, and standard deviation $s\pi/\sqrt{3}$
logistic_tail($x$)  the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
logistic_tail($s, x$)  the reverse cumulative logistic distribution with mean 0, scale $s$, and standard deviation $s\pi/\sqrt{3}$
logistic_tail($m, s, x$)  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
nb$\text{inomialp}(n, k, p)$ the negative binomial probability
nb$n$ $\text{inomialtail}(n, k, p)$ the reverse cumulative probability of the negative binomial distribution
nc$\text{hi2}(df, np, x)$ the cumulative noncentral $\chi^2$ distribution; 0 if $x < 0$
nch$\text{i2den}(df, np, x)$ the probability density of the noncentral $\chi^2$ distribution; 0 if $x < 0$
nch$\text{i2tail}(df, np, x)$ the reverse cumulative (upper tail or survivor) noncentral $\chi^2$ distribution; 1 if $x < 0$
nF$\text{F}(df_1, df_2, np, f)$ the cumulative noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 0 if $f < 0$
nF$d$ $\text{den}(df_1, df_2, np, f)$ the probability density function of the noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 0 if $f < 0$
nF$t$ $\text{ail}(df_1, df_2, np, f)$ the reverse cumulative (upper tail or survivor) noncentral $F$ distribution with $df_1$ numerator and $df_2$ denominator degrees of freedom and noncentrality parameter $np$; 1 if $f < 0$
nb$\text{ibeta}(a, b, np, x)$ the cumulative noncentral beta distribution; 0 if $x < 0$; or 1 if $x > 1$
}\text{ormal}(z)$ the cumulative standard normal distribution
n$\text{ormalden}(z)$ the standard normal density, $N(0, 1)$
n$\text{ormalden}(x, \sigma)$ the normal density with mean 0 and standard deviation $\sigma$
n$\text{ormalden}(x, \mu, \sigma)$ the normal density with mean $\mu$ and standard deviation $\sigma$
n$\text{pnchi2}(df, x, p)$ the noncentrality parameter, $np$, for noncentral $\chi^2$: if $\text{nchi2}(df, np, x) = p$, then $\text{nppnchi2}(df, x, p) = np$
n$\text{pnF}(df_1, df_2, f, p)$ the noncentrality parameter, $np$, for the noncentral $F$: if $\text{nF}(df_1, df_2, np, f) = p$, then $\text{nppnF}(df_1, df_2, f, p) = np$
n$\text{pnt}(df, t, p)$ the noncentrality parameter, $np$, for the noncentral Student’s $t$ distribution: if $\text{nt}(df, np, t) = p$, then $\text{nppnt}(df, t, p) = np$
n$\text{nt}(df, np, t)$ the cumulative noncentral Student’s $t$ distribution with $df$ degrees of freedom and noncentrality parameter $np$
n$\text{ntden}(df, np, t)$ the probability density function of the noncentral Student’s $t$ distribution with $df$ degrees of freedom and noncentrality parameter $np$
n$\text{nttail}(df, np, t)$ the reverse cumulative (upper tail or survivor) noncentral Student’s $t$ distribution with $df$ degrees of freedom and noncentrality parameter $np$
po$\text{isson}(m, k)$ the probability of observing $\text{floor}(k)$ or fewer outcomes that are distributed as Poisson with mean $m$
po$\text{issonp}(m, k)$ the probability of observing $\text{floor}(k)$ outcomes that are distributed as Poisson with mean $m$
po$\text{ssontail}(m, k)$ the probability of observing $\text{floor}(k)$ or more outcomes that are distributed as Poisson with mean $m$
t$\text{t}(df, t)$ the cumulative Student’s $t$ distribution with $df$ degrees of freedom
t$\text{den}(df, t)$ the probability density function of Student’s $t$ distribution
t$\text{tail}(df, t)$ the reverse cumulative (upper tail or survivor) Student’s $t$ distribution; the probability $T > t$
tu$\text{keyprob}(k, df, x)$ 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 $g$
Functions

Statistical functions are listed alphabetically under the following headings:

- Beta and noncentral beta distributions
- Binomial distribution
- Cauchy distribution
- $\chi^2$ and noncentral $\chi^2$ distributions
- Dunnett’s multiple range distribution
- Exponential distribution
- $F$ and noncentral $F$ distributions
- Gamma distribution
- Hypergeometric distribution
- Inverse Gaussian distribution
- Laplace distribution
- Logistic distribution
- Negative binomial distribution
- Normal (Gaussian), binormal, and multivariate normal distributions
- Poisson distribution
- Student’s $t$ and noncentral Student’s $t$ distributions
- Tukey’s Studentized range distribution
- Weibull distribution
- Weibull (proportional hazards) distribution
- Wishart distribution

Beta and noncentral beta distributions

$\text{betaden}(a, b, x)$

Description: the probability density of the beta distribution, where $a$ and $b$ are the shape parameters; 0 if $x < 0$ or $x > 1$

The probability density of the beta distribution is

$$\text{betaden}(a, b, x) = \frac{x^{a-1}(1-x)^{b-1}}{\Gamma(a)\Gamma(b)\int_0^\infty t^{a-1}(1-t)^{b-1}dt} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}$$

Domain $a$: 1e–323 to 8e+307
Domain $b$: 1e–323 to 8e+307
Domain $x$: $-8e+307$ to $8e+307$; interesting domain is $0 \leq x \leq 1$
Range: 0 to 8e+307
ibeta\((a, b, x)\)

Description: the cumulative beta distribution with shape parameters \(a\) and \(b\); \(0\) if \(x < 0\); or \(1\) if \(x > 1\)

The cumulative beta distribution with shape parameters \(a\) and \(b\) is defined by

\[
I_x(a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1 - t)^{b-1} dt
\]

ibeta() returns the regularized incomplete beta function, also known as the incomplete beta function ratio. The incomplete beta function without regularization is given by \((\text{gamma}(a)\ast\text{gamma}(b)/\text{gamma}(a+b))\ast\text{ibeta}(a, b, x)\) or, better when \(a\) or \(b\) might be large, \(\exp(\text{lngamma}(a)\ast\text{lngamma}(b)\ast\text{lngamma}(a+b))\ast\text{ibeta}(a, b, x)\).

Here is an example of the use of the regularized incomplete beta function. Although Stata has a cumulative binomial function (see binomial()), the probability that an event occurs \(k\) or fewer times in \(n\) trials, when the probability of one event is \(p\), can be evaluated as \(\text{cond}(k==n, 1, 1-\text{ibeta}(k+1, n-k, p))\). The reverse cumulative binomial (the probability that an event occurs \(k\) or more times) can be evaluated as \(\text{cond}(k==0, 1, \text{ibeta}(k, n-k+1, p))\). See Press et al. (2007, 270–273) for a more complete description and for suggested uses for this function.

Domain \(a\): \(1e-10\) to \(1e+17\)
Domain \(b\): \(1e-10\) to \(1e+17\)
Domain \(x\): \(-8e+307\) to \(8e+307\); interesting domain is \(0 \leq x \leq 1\)
Range: \(0\) to \(1\)

ibetatail\((a, b, x)\)

Description: the reverse cumulative (upper tail or survivor) beta distribution with shape parameters \(a\) and \(b\); \(1\) if \(x < 0\); or \(0\) if \(x > 1\)

The reverse cumulative (upper tail or survivor) beta distribution with shape parameters \(a\) and \(b\) is defined by

\[
\text{ibetatail}(a, b, x) = 1 - \text{ibeta}(a, b, x) = \int_x^1 \text{betaden}(a, b, t) dt
\]

ibetatail() is also known as the complement to the incomplete beta function (ratio).

Domain \(a\): \(1e-10\) to \(1e+17\)
Domain \(b\): \(1e-10\) to \(1e+17\)
Domain \(x\): \(-8e+307\) to \(8e+307\); interesting domain is \(0 \leq x \leq 1\)
Range: \(0\) to \(1\)

invibeta\((a, b, p)\)

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

Domain \(a\): \(1e-10\) to \(1e+17\)
Domain \(b\): \(1e-10\) to \(1e+17\)
Domain \(p\): \(0\) to \(1\)
Range: \(0\) to \(1\)
invibetatail(\(a, b, p\))
Description: the inverse reverse cumulative (upper tail or survivor) beta distribution: if \(\text{ibetatail}(a, b, x) = p\), then \(\text{invibetatail}(a, b, p) = x\)
Domain \(a\): 1e–10 to 1e+17
Domain \(b\): 1e–10 to 1e+17
Domain \(p\): 0 to 1
Range: 0 to 1

nbetaden(\(a, b, np, x\))
Description: the probability density function of the noncentral beta distribution; 0 if \(x < 0\) or \(x > 1\)
The probability density function of the noncentral beta distribution is defined as
\[
\sum_{j=0}^{\infty} \frac{e^{-np/2}(np/2)^j}{\Gamma(j+1)} \left\{ \frac{\Gamma(a + b + j)}{\Gamma(a + j)\Gamma(b)} x^{a+j-1}(1 - x)^{b-1} \right\}
\]
where \(a\) and \(b\) are shape parameters, \(np\) is the noncentrality parameter, and \(x\) is the value of a beta random variable.
\(\text{nbetaden}(a, b, 0, x) = \text{betaden}(a, b, x)\), but \(\text{betaden()}\) is the preferred function to use for the central beta distribution. \(\text{nbetaden()}\) is computed using an algorithm described in Johnson, Kotz, and Balakrishnan (1995).
Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(np\): 0 to 1,000
Domain \(x\): -8e+307 to 8e+307; interesting domain is 0 \(\leq x \leq 1\)
Range: 0 to 8e+307

nibeta(\(a, b, np, x\))
Description: the cumulative noncentral beta distribution; 0 if \(x < 0\); or 1 if \(x > 1\)
The cumulative noncentral beta distribution is defined as
\[
I_x(a, b, np) = \sum_{j=0}^{\infty} \frac{e^{-np/2}(np/2)^j}{\Gamma(j+1)} I_x(a + j, b)
\]
where \(a\) and \(b\) are shape parameters, \(np\) is the noncentrality parameter, \(x\) is the value of a beta random variable, and \(I_x(a, b)\) is the cumulative beta distribution, \(\text{ibeta()}\).
\(\text{nibeta}(a, b, 0, x) = \text{ibeta}(a, b, x)\), but \(\text{ibeta()}\) is the preferred function to use for the central beta distribution. \(\text{nibeta()}\) is computed using an algorithm described in Johnson, Kotz, and Balakrishnan (1995).
Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(np\): 0 to 10,000
Domain \(x\): -8e+307 to 8e+307; interesting domain is 0 \(\leq x \leq 1\)
Range: 0 to 1
\text{invnibeta}(a, b, np, p)
\text{Description:} the inverse cumulative noncentral beta distribution: if 
\text{nibeta}(a, b, np, x) = p, then \text{invibeta}(a, b, np, p) = x 
\text{Domain } a: 1e^{-323} \text{ to } 8e+307 
\text{Domain } b: 1e^{-323} \text{ to } 8e+307 
\text{Domain } np: 0 \text{ to } 1,000 
\text{Domain } p: 0 \text{ to } 1 
\text{Range: } 0 \text{ to } 1 

\textbf{Binomial distribution} 

\text{binomialp}(n, k, p)
\text{Description:} the probability of observing \text{floor}(k) successes in \text{floor}(n) trials when the probability of a success on one trial is \text{p} 
\text{Domain } n: 1 \text{ to } 1e+6 
\text{Domain } k: 0 \text{ to } n 
\text{Domain } p: 0 \text{ to } 1 
\text{Range: } 0 \text{ to } 1 

\text{binomial}(n, k, \theta)
\text{Description:} the probability of observing \text{floor}(k) or fewer successes in \text{floor}(n) trials when the probability of a success on one trial is \text{\theta}; 0 \text{ if } k < 0; \text{ or } 1 \text{ if } k > n 
\text{Domain } n: 0 \text{ to } 1e+17 
\text{Domain } k: -8e+307 \text{ to } 8e+307; \text{ interesting domain is } 0 \leq k < n 
\text{Domain } \theta: 0 \text{ to } 1 
\text{Range: } 0 \text{ to } 1 

\text{binomialtail}(n, k, \theta)
\text{Description:} the probability of observing \text{floor}(k) or more successes in \text{floor}(n) trials when the probability of a success on one trial is \text{\theta}; 1 \text{ if } k < 0; \text{ or } 0 \text{ if } k > n 
\text{Domain } n: 0 \text{ to } 1e+17 
\text{Domain } k: -8e+307 \text{ to } 8e+307; \text{ interesting domain is } 0 \leq k < n 
\text{Domain } \theta: 0 \text{ to } 1 
\text{Range: } 0 \text{ to } 1 

\text{invbinomial}(n, k, p)
\text{Description:} the inverse of the cumulative binomial; that is, \text{\theta} (\text{\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 \text{p} 
\text{Domain } n: 1 \text{ to } 1e+17 
\text{Domain } k: 0 \text{ to } n-1 
\text{Domain } p: 0 \text{ to } 1 \text{ (exclusive)} 
\text{Range: } 0 \text{ to } 1
invbinomialtail(n, k, p)
Description: the inverse of the right cumulative binomial; that is, \( \theta \) (\( \theta \) = probability of success on one trial) such that the probability of observing \( \text{floor}(k) \) or more successes in \( \text{floor}(n) \) trials is \( p \)
Domain n: 1 to 1e+17
Domain k: 1 to n
Domain p: 0 to 1 (exclusive)
Range: 0 to 1

Cauchy distribution

cauchyden(a, b, x)
Description: the probability density of the Cauchy distribution with location parameter \( a \) and scale parameter \( b \)
Domain a: \(-1e+300\) to \(1e+300\)
Domain b: \(1e–100\) to \(1e+300\)
Domain x: \(-8e+307\) to \(8e+307\)
Range: 0 to \(8e+307\)

cauchy(a, b, x)
Description: the cumulative Cauchy distribution with location parameter \( a \) and scale parameter \( b \)
Domain a: \(-1e+300\) to \(1e+300\)
Domain b: \(1e–100\) to \(1e+300\)
Domain x: \(-8e+307\) to \(8e+307\)
Range: 0 to 1

cauchytail(a, b, x)
Description: the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter \( a \) and scale parameter \( b \)
\[ \text{cauchytail}(a, b, x) = 1 - \text{cauchy}(a, b, x) \]
Domain a: \(-1e+300\) to \(1e+300\)
Domain b: \(1e–100\) to \(1e+300\)
Domain x: \(-8e+307\) to \(8e+307\)
Range: 0 to 1

invcauchy(a, b, p)
Description: the inverse of \( \text{cauchy}() \): if \( \text{cauchy}(a, b, x) = p \), then
\[ \text{invcauchy}(a, b, p) = x \]
Domain a: \(-1e+300\) to \(1e+300\)
Domain b: \(1e–100\) to \(1e+300\)
Domain p: 0 to 1 (exclusive)
Range: \(-8e+307\) to \(8e+307\)
invcauchytail\((a, b, p)\)

**Description:** the inverse of \texttt{cauchytail()}: if \texttt{cauchytail}(\(a, b, x\)) = \(p\), then \n\texttt{invcauchytail}(\(a, b, p\)) = \(x\)

- **Domain** \(a\): \(-1e+300\) to \(1e+300\)
- **Domain** \(b\): \(1e–100\) to \(1e+300\)
- **Domain** \(p\): \(0\) to \(1\) (exclusive)
- **Range:** \(-8e+307\) to \(8e+307\)

\lncauchyden\((a, b, x)\)

**Description:** the natural logarithm of the density of the Cauchy distribution with location parameter \(a\) and scale parameter \(b\)

- **Domain** \(a\): \(-1e+300\) to \(1e+300\)
- **Domain** \(b\): \(1e–100\) to \(1e+300\)
- **Domain** \(x\): \(-8e+307\) to \(8e+307\)
- **Range:** \(-1650\) to \(230\)

---

**Augustin-Louis Cauchy** (1789–1857) was born in Paris, France. He obtained a degree in engineering with honors from École Polytechnique, where he would later teach mathematics. While working as a military engineer, he published two papers on polyhedra, one of which was a solution to a problem presented to him by Joseph-Louis Lagrange. In 1816, he won the Grand Prix for his work on wave propagation.

Cauchy’s contributions were numerous and far reaching, as evident by the many concepts and theorems named after him. Some examples include the Cauchy criterion for convergence, Cauchy’s theorem for finite groups, the Cauchy distribution, and the Cauchy stress tensor. His contributions were so vast that once all of his work was collected, it comprised 27 volumes. His name is engraved on the Eiffel Tower, along with 71 other scientists and mathematicians.

---

\(\chi^2\) and noncentral \(\chi^2\) distributions

\chi2den\((df, x)\)

**Description:** the probability density of the \(\chi^2\) distribution with \(df\) degrees of freedom; 0 if \(x < 0\)

\texttt{chi2den}(\(df, x\)) = \texttt{gammaden}(\(df/2, 2, 0, x\))

- **Domain** \(df\): \(2e–10\) to \(2e+17\) (may be nonintegral)
- **Domain** \(x\): \(-8e+307\) to \(8e+307\)
- **Range:** 0 to \(8e+307\)

\chi2\((df, x)\)

**Description:** the cumulative \(\chi^2\) distribution with \(df\) degrees of freedom; 0 if \(x < 0\)

\texttt{chi2}(\(df, x\)) = \texttt{gammap}(\(df/2, x/2\))

- **Domain** \(df\): \(2e–10\) to \(2e+17\) (may be nonintegral)
- **Domain** \(x\): \(-8e+307\) to \(8e+307\); interesting domain is \(x \geq 0\)
- **Range:** 0 to 1
chi2tail\( (df, x) \)
Description: the reverse cumulative (upper tail or survivor) \( \chi^2 \) distribution with \( df \) degrees of freedom; 1 if \( x < 0 \)
\( \chi2tail(df, x) = 1 - \chi2(df, x) \)
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( x \): \(-8e+307\) to \(8e+307\); interesting domain is \( x \geq 0 \)
Range: 0 to 1

invchi2\( (df, p) \)
Description: the inverse of \( \chi2() \): if \( \chi2(df, x) = p \), then \( \text{invchi2}(df, p) = x \)
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( p \): 0 to 1
Range: 0 to \(8e+307\)

invchi2tail\( (df, p) \)
Description: the inverse of \( \chi2tail() \): if \( \chi2tail(df, x) = p \), then \( \text{invchi2tail}(df, p) = x \)
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( p \): 0 to 1
Range: 0 to \(8e+307\)

nchi2den\( (df, np, x) \)
Description: the probability density of the noncentral \( \chi^2 \) distribution; 0 if \( x < 0 \)
\( df \) denotes the degrees of freedom, \( np \) is the noncentrality parameter, and \( x \) is the value of \( \chi^2 \).
\( nchi2den(df, 0, x) = chi2den(df, x) \), but \( \text{chi2den()} \) is the preferred function to use for the central \( \chi^2 \) distribution.
Domain \( df \): 2e–10 to 1e+6 (may be nonintegral)
Domain \( np \): 0 to 10,000
Domain \( x \): \(-8e+307\) to \(8e+307\)
Range: 0 to \(8e+307\)

nchi2\( (df, np, x) \)
Description: the cumulative noncentral \( \chi^2 \) distribution; 0 if \( x < 0 \)
The cumulative noncentral \( \chi^2 \) distribution is defined as
\[
\int_0^x e^{-t/2} e^{-np/2} \frac{t^{df/2}}{2^{df/2}} \sum_{j=0}^{\infty} \frac{t^{df/2+j-1} np^j}{\Gamma(df/2+j) 2^{2j} j!} dt
\]
where \( df \) denotes the degrees of freedom, \( np \) is the noncentrality parameter, and \( x \) is the value of \( \chi^2 \).
\( nchi2(df, 0, x) = chi2(df, x) \), but \( \text{chi2()} \) is the preferred function to use for the central \( \chi^2 \) distribution.
Domain \( df \): 2e–10 to 1e+6 (may be nonintegral)
Domain \( np \): 0 to 10,000
Domain \( x \): \(-8e+307\) to \(8e+307\); interesting domain is \( x \geq 0 \)
Range: 0 to 1
**nchi2tail\((df, np, x)\)**

Description: the reverse cumulative (upper tail or survivor) noncentral $\chi^2$ distribution; 1 if $x < 0$

- $df$ denotes the degrees of freedom, $np$ is the noncentrality parameter, and $x$ is the value of $\chi^2$.

- **Domain $df$:** 2e–10 to 1e+6 (may be nonintegral)
- **Domain $np$:** 0 to 10,000
- **Domain $x$:** $-8e+307$ to 8e+307
- **Range:** 0 to 1

**invnchi2\((df, np, p)\)**

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

- **Domain $df$:** 2e–10 to 1e+6 (may be nonintegral)
- **Domain $np$:** 0 to 10,000
- **Domain $p$:** 0 to 1
- **Range:** 0 to 8e+307

**invnchi2tail\((df, np, p)\)**

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

- **Domain $df$:** 2e–10 to 1e+6 (may be nonintegral)
- **Domain $np$:** 0 to 10,000
- **Domain $p$:** 0 to 1
- **Range:** 0 to 8e+307

**npnchi2\((df, x, p)\)**

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

- **Domain $df$:** 2e–10 to 1e+6 (may be nonintegral)
- **Domain $x$:** 0 to 8e+307
- **Domain $p$:** 0 to 1
- **Range:** 0 to 10,000

---

**Dunnett’s multiple range distribution**

**dunnettprob\((k, df, x)\)**

Description: the cumulative multiple range distribution that is used in Dunnett’s multiple-comparison method with $k$ ranges and $df$ degrees of freedom; 0 if $x < 0$

- **dunnettprob\((\)\)** is computed using an algorithm described in Miller (1981).

- **Domain $k$:** 2 to 1e+6
- **Domain $df$:** 2 to 1e+6
- **Domain $x$:** $-8e+307$ to 8e+307; interesting domain is $x \geq 0$
- **Range:** 0 to 1
invdunnettprob\( (k, df, p) \)

**Description:** the inverse cumulative multiple range distribution that is used in Dunnett’s multiple-comparison method with \( k \) ranges and \( df \) degrees of freedom

If \( \text{dunnettprob}(k, df, x) = p \), then \( \text{invdunnettprob}(k, df, p) = x \).

\( \text{invdunnettprob}() \) is computed using an algorithm described in Miller (1981).

**Domain**
- \( k \): 2 to 1e+6
- \( df \): 2 to 1e+6
- \( p \): 0 to 1 (right exclusive)

**Range:** 0 to 8e+307

Charles William Dunnett (1921–2007) was a Canadian statistician best known for his work on multiple-comparison procedures. He was born in Windsor, Ontario, and graduated in mathematics and physics from McMaster University. After naval service in World War II, Dunnett’s career included further graduate work, teaching, and research at Toronto, Columbia, the New York State Maritime College, the Department of National Health and Welfare in Ottawa, Cornell, Lederle Laboratories, and Aberdeen before he became Professor of Clinical Epidemiology and Biostatistics at McMaster University in 1974. He was President and Gold Medalist of the Statistical Society of Canada. Throughout his career, Dunnett took a keen interest in computing. According to Google Scholar, his 1955 paper on comparing treatments with a control has been cited over 4,000 times.

### Exponential distribution

**exponentialden\( (b, x) \)**

**Description:** the probability density function of the exponential distribution with scale \( b \)

The probability density function of the exponential distribution is

\[
\frac{1}{b} \exp(-x/b)
\]

where \( b \) is the scale and \( x \) is the value of an exponential variate.

**Domain**
- \( b \): 1e–323 to 8e+307
- \( x \): \(-8e+307\) to \( 8e+307 \); interesting domain is \( x \geq 0 \)

**Range:** 1e–323 to 8e+307

**exponential\( (b, x) \)**

**Description:** the cumulative exponential distribution with scale \( b \)

The cumulative distribution function of the exponential distribution is

\[
1 - \exp(-x/b)
\]

for \( x \geq 0 \) and 0 for \( x < 0 \), where \( b \) is the scale and \( x \) is the value of an exponential variate.

The mean of the exponential distribution is \( b \) and its variance is \( b^2 \).

**Domain**
- \( b \): 1e–323 to 8e+307
- \( x \): \(-8e+307\) to \( 8e+307 \); interesting domain is \( x \geq 0 \)

**Range:** 0 to 1
exponentialtail\( (b, x) \)
Description: the reverse cumulative exponential distribution with scale \( b \)

The reverse cumulative distribution function of the exponential distribution is

\[
\exp(-x/b)
\]

where \( b \) is the scale and \( x \) is the value of an exponential variate.

Domain \( b \): 1e–323 to 8e+307
Domain \( x \): –8e+307 to 8e+307; interesting domain is \( x \geq 0 \)
Range: 0 to 1

invexponential\( (b, p) \)
Description: the inverse cumulative exponential distribution with scale \( b \): if \n\( \text{exponential}(b, x) = p \), then \( \text{invexponential}(b, p) = x \)

Domain \( b \): 1e–323 to 8e+307
Domain \( p \): 0 to 1
Range: 1e–323 to 8e+307

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

Domain \( b \): 1e–323 to 8e+307
Domain \( p \): 0 to 1
Range: 1e–323 to 8e+307

F and noncentral F distributions

\( F\text{den}(df_1, df_2, f) \)
Description: the probability density function of the \( F \) distribution with \( df_1 \) numerator and \( df_2 \)

denominator degrees of freedom; 0 if \( f < 0 \)

The probability density function of the \( F \) distribution with \( df_1 \) numerator and \( df_2 \)
denominator degrees of freedom is defined as

\[
F\text{den}(df_1, df_2, f) = \frac{\Gamma\left(\frac{df_1+df_2}{2}\right)}{\Gamma\left(\frac{df_1}{2}\right)\Gamma\left(\frac{df_2}{2}\right)} \left(\frac{df_1}{df_2}\right)^{\frac{df_1}{2}} \cdot f^{\frac{df_1}{2}-1} \left(1 + \frac{df_1}{df_2} f\right)^{-\frac{1}{2}(df_1+df_2)}
\]

Domain \( df_1 \): 1e–323 to 8e+307 (may be nonintegral)
Domain \( df_2 \): 1e–323 to 8e+307 (may be nonintegral)
Domain \( f \): –8e+307 to 8e+307; interesting domain is \( f \geq 0 \)
Range: 0 to 8e+307
F(df1, df2, f)
Description: the cumulative F distribution with df1 numerator and df2 denominator degrees of freedom: F(df1, df2, f) = ∫₀⁻° Fden(df1, df2, t) dt; 0 if f < 0
Domain df1: 2e–10 to 2e+17 (may be nonintegral)
Domain df2: 2e–10 to 2e+17 (may be nonintegral)
Domain f: −8e+307 to 8e+307; interesting domain is f ≥ 0
Range: 0 to 1

Ftail(df1, df2, f)
Description: the reverse cumulative (upper tail or survivor) F distribution with df1 numerator and df2 denominator degrees of freedom; 1 if f < 0
Ftail(df1, df2, f) = 1 – F(df1, df2, f).
Domain df1: 2e–10 to 2e+17 (may be nonintegral)
Domain df2: 2e–10 to 2e+17 (may be nonintegral)
Domain f: −8e+307 to 8e+307; interesting domain is f ≥ 0
Range: 0 to 1

invF(df1, df2, p)
Description: the inverse cumulative F distribution: if F(df1, df2, f) = p, then
invF(df1, df2, p) = f
Domain df1: 2e–10 to 2e+17 (may be nonintegral)
Domain df2: 2e–10 to 2e+17 (may be nonintegral)
Domain p: 0 to 1
Range: 0 to 8e+307

invFtail(df1, df2, p)
Description: the inverse reverse cumulative (upper tail or survivor) F distribution:
if Ftail(df1, df2, f) = p, then invFtail(df1, df2, p) = f
Domain df1: 2e–10 to 2e+17 (may be nonintegral)
Domain df2: 2e–10 to 2e+17 (may be nonintegral)
Domain p: 0 to 1
Range: 0 to 8e+307
\( \text{nFden}(df_1, df_2, np, f) \)

**Description:** the probability density function of the noncentral \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom and noncentrality parameter \( np \); 0 if \( f < 0 \)

\( \text{nFden}(df_1, df_2, 0, f) = \text{Fden}(df_1, df_2, f) \), but \( \text{Fden}() \) is the preferred function to use for the central \( F \) distribution.

Also, if \( F \) follows the noncentral \( F \) distribution with \( df_1 \) and \( df_2 \) degrees of freedom and noncentrality parameter \( np \), then

\[
\frac{df_1 F}{df_2 + df_1 F}
\]

follows a noncentral beta distribution with shape parameters \( a = df_1 / 2, b = df_2 / 2, \) and noncentrality parameter \( np \), as given in \( \text{nbetaden}() \). \( \text{nFden}() \) is computed based on this relationship.

**Domain**
- \( df_1 \): 1e–323 to 8e+307 (may be nonintegral)
- \( df_2 \): 1e–323 to 8e+307 (may be nonintegral)
- \( np \): 0 to 1,000
- \( f \): \( -8e+307 \) to \( 8e+307 \); interesting domain is \( f \geq 0 \)
- **Range**: 0 to 8e+307

\( \text{nF}(df_1, df_2, np, f) \)

**Description:** the cumulative noncentral \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom and noncentrality parameter \( np \); 0 if \( f < 0 \)

\( \text{nF}(df_1, df_2, 0, f) = \text{F}(df_1, df_2, f) \)

\( \text{nF}() \) is computed using \( \text{nibeta}() \) based on the relationship between the noncentral beta and noncentral \( F \) distributions: \( \text{nF}(df_1, df_2, np, f) = \text{nibeta}(df_1/2, df_2/2, np, df_1 \times f / \{(df_1 \times f) + df_2\}) \).

**Domain**
- \( df_1 \): 2e–10 to 1e+8
- \( df_2 \): 2e–10 to 1e+8
- \( np \): 0 to 10,000
- \( f \): \( -8e+307 \) to \( 8e+307 \)
- **Range**: 0 to 1

\( \text{nFtail}(df_1, df_2, np, f) \)

**Description:** the reverse cumulative (upper tail or survivor) noncentral \( F \) distribution with \( df_1 \) numerator and \( df_2 \) denominator degrees of freedom and noncentrality parameter \( np \); 1 if \( f < 0 \)

\( \text{nFtail}() \) is computed using \( \text{nibeta}() \) based on the relationship between the noncentral beta and \( F \) distributions. See Johnson, Kotz, and Balakrishnan (1995) for more details.

**Domain**
- \( df_1 \): 1e–323 to 8e+307 (may be nonintegral)
- \( df_2 \): 1e–323 to 8e+307 (may be nonintegral)
- \( np \): 0 to 1,000
- \( f \): \( -8e+307 \) to \( 8e+307 \); interesting domain is \( f \geq 0 \)
- **Range**: 0 to 1
invnF\((df_1, df_2, np, p)\)

Description: the inverse cumulative noncentral \(F\) distribution: if 
\[ nF(df_1, df_2, np, f) = p, \text{ then } invnF(df_1, df_2, np, p) = f \]

Domain \(df_1\): 1e–6 to 1e+6 (may be nonintegral)
Domain \(df_2\): 1e–6 to 1e+6 (may be nonintegral)
Domain \(np\): 0 to 10,000
Domain \(p\): 0 to 1
Range: 0 to 8e+307

invnFtail\((df_1, df_2, np, p)\)

Description: the inverse reverse cumulative (upper tail or survivor) noncentral \(F\) distribution: if 
\[ nFtail(df_1, df_2, np, f) = p, \text{ then } invnFtail(df_1, df_2, np, p) = f \]

Domain \(df_1\): 1e–323 to 8e+307 (may be nonintegral)
Domain \(df_2\): 1e–323 to 8e+307 (may be nonintegral)
Domain \(np\): 0 to 1,000
Domain \(p\): 0 to 1
Range: 0 to 8e+307

npnF\((df_1, df_2, f, p)\)

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

Domain \(df_1\): 2e–10 to 1e+6 (may be nonintegral)
Domain \(df_2\): 2e–10 to 1e+6 (may be nonintegral)
Domain \(f\): 0 to 8e+307
Domain \(p\): 0 to 1
Range: 0 to 1,000

Gamma distribution

gammaden\((a, b, g, x)\)

Description: the probability density function of the gamma distribution; 0 if \(x < g\)

The probability density function of the gamma distribution is defined by

\[ \frac{1}{\Gamma(a)b^a} (x-g)^{a-1} e^{-\frac{(x-g)}{b}} \]

where \(a\) is the shape parameter, \(b\) is the scale parameter, and \(g\) is the location parameter.

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(g\): –8e+307 to 8e+307
Domain \(x\): –8e+307 to 8e+307; interesting domain is \(x \geq g\)
Range: 0 to 8e+307
\texttt{gammap}(a,x)

Description: the cumulative gamma distribution with shape parameter \(a\); 0 if \(x < 0\)

The cumulative gamma distribution with shape parameter \(a\) is defined by

\[
\frac{1}{\Gamma(a)} \int_{0}^{x} e^{-t} t^{a-1} \, dt
\]

The cumulative Poisson (the probability of observing \(k\) or fewer events if the expected is \(x\)) can be evaluated as \(1 - \text{gammap}(k+1, x)\). The reverse cumulative (the probability of observing \(k\) or more events) can be evaluated as \(\text{gammap}(k, x)\). See Press et al. (2007, 259–266) for a more complete description and for suggested uses for this function.

\texttt{gammap()} is also known as the incomplete gamma function (ratio).

Probabilities for the three-parameter gamma distribution (see \texttt{gammaden()}) can be calculated by shifting and scaling \(x\); that is, \(\text{gammap}(a, (x - g)/b)\).

Domain \(a\): \(1\times10^{-10}\) to \(1\times10^{17}\)
Domain \(x\): \(-8\times10^{307}\) to \(8\times10^{307}\); interesting domain is \(x \geq 0\)
Range: 0 to 1

\texttt{gammaptail}(a,x)

Description: the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter \(a\); 1 if \(x < 0\)

The reverse cumulative (upper tail or survivor) gamma distribution with shape parameter \(a\) is defined by

\[
\text{gammap}(a, x) = 1 - \text{gammap}(a, x) = \int_{x}^{\infty} \text{gammaden}(a, t) \, dt
\]

\texttt{gammaptail()} is also known as the complement to the incomplete gamma function (ratio).

Domain \(a\): \(1\times10^{-10}\) to \(1\times10^{17}\)
Domain \(x\): \(-8\times10^{307}\) to \(8\times10^{307}\); interesting domain is \(x \geq 0\)
Range: 0 to 1

\texttt{invgammap}(a,p)

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

Domain \(a\): \(1\times10^{-10}\) to \(1\times10^{17}\)
Domain \(p\): 0 to 1
Range: 0 to \(8\times10^{307}\)
invgammaptail\( (a,p) \)  
Description: the inverse reverse cumulative (upper tail or survivor) gamma distribution: if \( \text{gammap}(a,x) = p \), then \( \text{invgammap}(a,p) = x \)  
Domain \( a \): \( 1e^{-10} \) to \( 1e^{+17} \)  
Domain \( p \): 0 to 1  
Range: 0 to \( 8e^{+307} \)

dgammapda\( (a,x) \)  
Description: \( \frac{\partial P(a,x)}{\partial a} \), where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)  
Domain \( a \): \( 1e^{-7} \) to \( 1e^{+17} \)  
Domain \( x \): \( -8e^{+307} \) to \( 8e^{+307} \); interesting domain is \( x \geq 0 \)  
Range: \(-16 \) to 0

dgammapdada\( (a,x) \)  
Description: \( \frac{\partial^2 P(a,x)}{\partial a^2} \), where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)  
Domain \( a \): \( 1e^{-7} \) to \( 1e^{+17} \)  
Domain \( x \): \( -8e^{+307} \) to \( 8e^{+307} \); interesting domain is \( x \geq 0 \)  
Range: \(-0.02 \) to \( 4.77e^{+5} \)

dgammapdadx\( (a,x) \)  
Description: \( \frac{\partial^2 P(a,x)}{\partial a \partial x} \), where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)  
Domain \( a \): \( 1e^{-7} \) to \( 1e^{+17} \)  
Domain \( x \): \( -8e^{+307} \) to \( 8e^{+307} \); interesting domain is \( x \geq 0 \)  
Range: \(-0.04 \) to \( 8e^{+307} \)

dgammapdx\( (a,x) \)  
Description: \( \frac{\partial^2 P(a,x)}{\partial x^2} \), where \( P(a,x) = \text{gammap}(a,x) \); 0 if \( x < 0 \)  
Domain \( a \): \( 1e^{-10} \) to \( 1e^{+17} \)  
Domain \( x \): \( -8e^{+307} \) to \( 8e^{+307} \); interesting domain is \( x \geq 0 \)  
Range: \( 0 \) to \( 1e^{+40} \)

lnigammaden\( (a,b,x) \)  
Description: the natural logarithm of the inverse gamma density, where \( a \) is the shape parameter and \( b \) is the scale parameter  
Domain \( a \): \( 1e^{-300} \) to \( 1e^{+300} \)  
Domain \( b \): \( 1e^{-300} \) to \( 1e^{+300} \)  
Domain \( x \): \( 1e^{-300} \) to \( 8e^{+307} \)  
Range: \(-8e^{+307} \) to \( 8e^{+307} \)
Hypergeometric distribution

\texttt{hypergeometricp}(N,K,n,k)

Description: 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. Success is obtaining an element with the attribute of interest.

Domain \( N \): 2 to 1e+5
Domain \( K \): 1 to \( N-1 \)
Domain \( n \): 1 to \( N-1 \)
Domain \( k \): \( \max(0,n-N+K) \) to \( \min(K,n) \)
Range: 0 to 1 (right exclusive)

\texttt{hypergeometric}(N,K,n,k)

Description: the cumulative probability of the hypergeometric distribution

\( N \) is the population size, \( K \) is the number of elements in the population that have the attribute of interest, and \( n \) is the sample size. Returned is the probability of observing \( k \) or fewer elements from a sample of size \( n \) that have the attribute of interest.

Domain \( N \): 2 to 1e+5
Domain \( K \): 1 to \( N-1 \)
Domain \( n \): 1 to \( N-1 \)
Domain \( k \): \( \max(0,n-N+K) \) to \( \min(K,n) \)
Range: 0 to 1

Inverse Gaussian distribution

\texttt{igaussianden}(m,a,x)

Description: the probability density of the inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 0 if \( x \leq 0 \)

Domain \( m \): 1e–323 to 8e+307
Domain \( a \): 1e–323 to 8e+307
Domain \( x \): \(-8e+307 \) to \( 8e+307 \)
Range: 0 to \( 8e+307 \)

\texttt{igaussian}(m,a,x)

Description: the cumulative inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 0 if \( x \leq 0 \)

Domain \( m \): 1e–323 to 8e+307
Domain \( a \): 1e–323 to 8e+307
Domain \( x \): \(-8e+307 \) to \( 8e+307 \)
Range: 0 to 1

\texttt{igaussiantail}(m,a,x)

Description: the reverse cumulative (upper tail or survivor) inverse Gaussian distribution with mean \( m \) and shape parameter \( a \); 1 if \( x \leq 0 \)

\texttt{igaussiantail}(m,a,x) = 1 - \texttt{igaussian}(m,a,x)

Domain \( m \): 1e–323 to 8e+307
Domain \( a \): 1e–323 to 8e+307
Domain \( x \): \(-8e+307 \) to \( 8e+307 \)
Range: 0 to 1
invigaussian($m,a,p$)
Description: the inverse of igaussian(): if
\[ \text{igaussian}(m,a,x) = p, \text{ then } \text{invigaussian}(m,a,p) = x \]
Domain $m$: 1e–323 to 8e+307
Domain $a$: 1e–323 to 1e+8
Domain $p$: 0 to 1 (exclusive)
Range: 0 to 8e+307

invgaussiantail($m,a,p$)
Description: the inverse of igaussiantail(): if
\[ \text{igaussiantail}(m,a,x) = p, \text{ then } \text{invgaussiantail}(m,a,p) = x \]
Domain $m$: 1e–323 to 8e+307
Domain $a$: 1e–323 to 1e+8
Domain $p$: 0 to 1 (exclusive)
Range: 0 to 8e+307

lnigaussianden($m,a,x$)
Description: the natural logarithm of the inverse Gaussian density with mean $m$ and shape parameter $a$
Domain $m$: 1e–323 to 8e+307
Domain $a$: 1e–323 to 8e+307
Domain $x$: 1e–323 to 8e+307
Range: $-8e+307$ to $8e+307$

Laplace distribution

laplaceden($m,b,x$)
Description: the probability density of the Laplace distribution with mean $m$ and scale parameter $b$
Domain $m$: $-8e+307$ to $8e+307$
Domain $b$: 1e–307 to 8e+307
Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 8e+307

laplace($m,b,x$)
Description: the cumulative Laplace distribution with mean $m$ and scale parameter $b$
Domain $m$: $-8e+307$ to $8e+307$
Domain $b$: 1e–307 to 8e+307
Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 1

laplacetail($m,b,x$)
Description: the reverse cumulative (upper tail or survivor) Laplace distribution with mean $m$ and scale parameter $b$
\[ \text{laplacetail}(m,b,x) = 1 - \text{laplace}(m,b,x) \]
Domain $m$: $-8e+307$ to $8e+307$
Domain $b$: 1e–307 to 8e+307
Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 1
invlaplace(m, b, p)
Description: the inverse of laplace(): if laplace(m, b, x) = p, then
invlaplace(m, b, p) = x
Domain m: \(-8e+307 \text{ to } 8e+307\)
Domain b: \(1e^{-307} \text{ to } 8e+307\)
Domain p: 0 to 1 (exclusive)
Range: \(-8e+307 \text{ to } 8e+307\)

invlaplacetail(m, b, p)
Description: the inverse of laplacetail(): if laplacetail(m, b, x) = p,
then invlaplacetail(m, b, p) = x
Domain m: \(-8e+307 \text{ to } 8e+307\)
Domain b: \(1e^{-307} \text{ to } 8e+307\)
Domain p: 0 to 1 (exclusive)
Range: \(-8e+307 \text{ to } 8e+307\)

lnlaplaceden(m, b, x)
Description: the natural logarithm of the density of the Laplace distribution with mean m and
scale parameter b
Domain m: \(-8e+307 \text{ to } 8e+307\)
Domain b: \(1e^{-307} \text{ to } 8e+307\)
Domain x: \(-8e+307 \text{ to } 8e+307\)
Range: \(-8e+307 \text{ to } 707\)

Logistic distribution

logisticden(x)
Description: the density of the logistic distribution with mean 0 and standard deviation \(\pi/\sqrt{3}\)
logisticden(x) = logisticden(1, x) = logisticden(0, 1, x), where x is the value of a logistic random variable.
Domain x: \(-8e+307 \text{ to } 8e+307\)
Range: 0 to 0.25

logisticden(s, x)
Description: the density of the logistic distribution with mean 0, scale s, and standard deviation \(s\pi/\sqrt{3}\)
logisticden(s, x) = logisticden(0, s, x), where s is the scale and x is the value of a logistic random variable.
Domain s: \(1e^{-323} \text{ to } 8e+307\)
Domain x: \(-8e+307 \text{ to } 8e+307\)
Range: 0 to 8e+307
logisticden\((m, s, x)\)
Description: the density of the logistic distribution with mean \(m\), scale \(s\), and standard deviation \(s\pi/\sqrt{3}\)

The density of the logistic distribution is defined as

\[
\frac{\exp\left\{-(x - m)/s\right\}}{s\left[1 + \exp\left\{-(x - m)/s\right\}\right]^2}
\]

where \(m\) is the mean, \(s\) is the scale, and \(x\) is the value of a logistic random variable.

Domain \(m\): \(-8e+307\) to \(8e+307\)
Domain \(s\): \(1e-323\) to \(8e+307\)
Domain \(x\): \(-8e+307\) to \(8e+307\)
Range: 0 to \(8e+307\)

logistic\((x)\)
Description: the cumulative logistic distribution with mean 0 and standard deviation \(\pi/\sqrt{3}\)

\[
\text{logistic}(x) = \text{logistic}(1, x) = \text{logistic}(0, 1, x), \text{ where } x \text{ is the value of a logistic random variable.}
\]

Domain \(x\): \(-8e+307\) to \(8e+307\)
Range: 0 to 1

logistic\((s, x)\)
Description: the cumulative logistic distribution with mean 0, scale \(s\), and standard deviation \(s\pi/\sqrt{3}\)

\[
\text{logistic}(s, x) = \text{logistic}(0, s, x), \text{ where } s \text{ is the scale and } x \text{ is the value of a logistic random variable.}
\]

Domain \(s\): \(1e-323\) to \(8e+307\)
Domain \(x\): \(-8e+307\) to \(8e+307\)
Range: \(0\) to 1

logistic\((m, s, x)\)
Description: the cumulative logistic distribution with mean \(m\), scale \(s\), and standard deviation \(s\pi/\sqrt{3}\)

The cumulative logistic distribution is defined as

\[
[1 + \exp\{-(x - m)/s\}]^{-1}
\]

where \(m\) is the mean, \(s\) is the scale, and \(x\) is the value of a logistic random variable.

Domain \(m\): \(-8e+307\) to \(8e+307\)
Domain \(s\): \(1e-323\) to \(8e+307\)
Domain \(x\): \(-8e+307\) to \(8e+307\)
Range: 0 to 1
logistictail($x$)
Description: the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$

$logistictail(x) = logistictail(1,x) = logistictail(0,1,x)$, where $x$ is the value of a logistic random variable.

Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 1

logistictail($s,x$)
Description: the reverse cumulative logistic distribution with mean 0, scale $s$, and standard deviation $s\pi/\sqrt{3}$

$logistictail(s,x) = logistictail(0,s,x)$, where $s$ is the scale and $x$ is the value of a logistic random variable.

Domain $s$: $1e-323$ to $8e+307$
Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 1

logistictail($m,s,x$)
Description: the reverse cumulative logistic distribution with mean $m$, scale $s$, and standard deviation $s\pi/\sqrt{3}$

The reverse cumulative logistic distribution is defined as

$$[1 + \exp\{(x - m)/s\}]^{-1}$$

where $m$ is the mean, $s$ is the scale, and $x$ is the value of a logistic random variable.

Domain $m$: $-8e+307$ to $8e+307$
Domain $s$: $1e-323$ to $8e+307$
Domain $x$: $-8e+307$ to $8e+307$
Range: 0 to 1

invlogistic($p$)
Description: the inverse cumulative logistic distribution: if logistic($x$) = $p$, then $invlogistic(p) = x$

Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$

invlogistic($s,p$)
Description: the inverse cumulative logistic distribution: if logistic($s,x$) = $p$, then $invlogistic(s,p) = x$

Domain $s$: $1e-323$ to $8e+307$
Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$
invlogistic($m, s, p$)
Description: the inverse cumulative logistic distribution: if $\text{logistic}(m, s, x) = p$, then $\text{invlogistic}(m, s, p) = x$

Domain $m$: $-8e+307$ to $8e+307$
Domain $s$: $1e-323$ to $8e+307$
Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$

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

Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$

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

Domain $s$: $1e-323$ to $8e+307$
Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$

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

Domain $m$: $-8e+307$ to $8e+307$
Domain $s$: $1e-323$ to $8e+307$
Domain $p$: 0 to 1
Range: $-8e+307$ to $8e+307$

Negative binomial distribution

nbinomialp($n, k, p$)
Description: the negative binomial probability

When $n$ is an integer, nbinomialp() returns the probability of observing exactly $\text{floor}(k)$ failures before the $n$th success when the probability of a success on one trial is $p$.

Domain $n$: $1e-10$ to $1e+6$ (can be nonintegral)
Domain $k$: 0 to $1e+10$
Domain $p$: 0 to 1 (left exclusive)
Range: 0 to 1
nbinomial\((n,k,p)\)
Description: the cumulative probability of the negative binomial distribution

\[ n \] can be nonintegral. When \( n \) is an integer, \( \text{nbinomial}() \) returns the probability of observing \( k \) or fewer failures before the \( n \)th success, when the probability of a success on one trial is \( p \).

The negative binomial distribution function is evaluated using \( \text{ibeta}() \).

Domain \( n \): \( 1e^{-10} \) to \( 1e+17 \) (can be nonintegral)
Domain \( k \): \( 0 \) to \( 2^{53} - 1 \)
Domain \( p \): \( 0 \) to 1 (left exclusive)
Range: \( 0 \) to 1

nbinomialtail\((n,k,p)\)
Description: the reverse cumulative probability of the negative binomial distribution

When \( n \) is an integer, \( \text{nbinomialtail}() \) returns the probability of observing \( k \) or more failures before the \( n \)th success, when the probability of a success on one trial is \( p \).

The reverse negative binomial distribution function is evaluated using \( \text{ibetatail}() \).

Domain \( n \): \( 1e^{-10} \) to \( 1e+17 \) (can be nonintegral)
Domain \( k \): \( 0 \) to \( 2^{53} - 1 \)
Domain \( p \): \( 0 \) to 1 (left exclusive)
Range: \( 0 \) to 1

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

\( \text{invnbinomial}() \) is evaluated using \( \text{invibeta}() \).

Domain \( n \): \( 1e^{-10} \) to \( 1e+17 \) (can be nonintegral)
Domain \( k \): \( 0 \) to \( 2^{53} - 1 \)
Domain \( q \): \( 0 \) to 1 (exclusive)
Range: \( 0 \) to 1

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

\( \text{invnbinomialtail}() \) is evaluated using \( \text{invibetatail}() \).

Domain \( n \): \( 1e^{-10} \) to \( 1e+17 \) (can be nonintegral)
Domain \( k \): \( 1 \) to \( 2^{53} - 1 \)
Domain \( q \): \( 0 \) to 1 (exclusive)
Range: \( 0 \) to 1 (exclusive)

Normal (Gaussian), binormal, and multivariate normal distributions

normalden\((z)\)
Description: the standard normal density, \( N(0,1) \)
Domain: \( -8e+307 \) to \( 8e+307 \)
Range: \( 0 \) to 0.39894 ...
normalden\((x, \sigma)\)
Description: the normal density with mean 0 and standard deviation \(\sigma\)

\[
\text{normalden}(x, 1) = \text{normalden}(x) \quad \text{and} \quad \text{normalden}(x, \sigma) = \text{normalden}(x/\sigma)/\sigma.
\]
Domain \(x\): \(-8\times10^{307}\) to \(8\times10^{307}\)
Domain \(\sigma\): \(1\times10^{-308}\) to \(8\times10^{307}\)
Range: 0 to \(8\times10^{307}\)

normalden\((x, \mu, \sigma)\)
Description: the normal density with mean \(\mu\) and standard deviation \(\sigma\), \(N(\mu, \sigma^2)\)

\[
\text{normalden}(x, 0, s) = \text{normalden}(x, s) \quad \text{and} \quad \text{normalden}(x, \mu, \sigma) = \text{normalden}((x - \mu)/\sigma)/\sigma. \quad \text{In general,}
\]
\[
\text{normalden}(z, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left\{ \frac{(z-\mu)}{\sigma} \right\}^2}
\]
Domain \(x\): \(-8\times10^{307}\) to \(8\times10^{307}\)
Domain \(\mu\): \(-8\times10^{307}\) to \(8\times10^{307}\)
Domain \(\sigma\): \(1\times10^{-308}\) to \(8\times10^{307}\)
Range: 0 to \(8\times10^{307}\)

normal\((z)\)
Description: the cumulative standard normal distribution

\[
\text{normal}(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx
\]
Domain: \(-8\times10^{307}\) to \(8\times10^{307}\)
Range: 0 to 1

invnormal\((p)\)
Description: the inverse cumulative standard normal distribution: if \(\text{normal}(z) = p\), then \(\text{invnormal}(p) = z\)
Domain: \(1\times10^{-323}\) to \(1 - 2^{-53}\)
Range: \(-38.449394\) to \(8.2095362\)

lnnormalden\((z)\)
Description: the natural logarithm of the standard normal density, \(N(0, 1)\)
Domain: \(-1\times10^{154}\) to \(1\times10^{154}\)
Range: \(-5\times10^{307}\) to \(-0.91893853 = \ln\text{normalden}(0)\)

lnnormalden\((x, \sigma)\)
Description: the natural logarithm of the normal density with mean 0 and standard deviation \(\sigma\)

\[
\text{lnnormalden}(x, 1) = \text{lnnormalden}(x) \quad \text{and} \quad \text{lnnormalden}(x, \sigma) = \text{lnnormalden}(x/\sigma) - \ln(\sigma).
\]
Domain \(x\): \(-8\times10^{307}\) to \(8\times10^{307}\)
Domain \(\sigma\): \(1\times10^{-323}\) to \(8\times10^{307}\)
Range: \(-5\times10^{307}\) to \(742.82799\)
\textbf{lnnormalden}(x, \mu, \sigma)

Description: the natural logarithm of the normal density with mean \(\mu\) and standard deviation \(\sigma\), \(N(\mu, \sigma^2)\)

\[
\text{lnnormalden}(x, 0, s) = \text{lnnormalden}(x, s) \quad \text{and} \\
\text{lnnormalden}(x, \mu, \sigma) = \text{lnnormalden}((x - \mu)/\sigma) - \ln(\sigma).
\]

In general,

\[
\text{lnnormalden}(z, \mu, \sigma) = \ln \left[ \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left\{ \frac{(z-\mu)^2}{\sigma^2} \right\}} \right]
\]

Domain \(x\): \(-8e+307\) to \(8e+307\)
Domain \(\mu\): \(-8e+307\) to \(8e+307\)
Domain \(\sigma\): \(1e–323\) to \(8e+307\)
Range: \(1e–323\) to \(8e+307\)

\textbf{lnnormal}(z)

Description: the natural logarithm of the cumulative standard normal distribution

\[
\text{lnnormal}(z) = \ln \left( \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \right)
\]

Domain: \(-1e+99\) to \(8e+307\)
Range: \(-5e+197\) to \(0\)

\textbf{binormal}(h, k, \rho)

Description: the joint cumulative distribution \(\Phi(h, k, \rho)\) of bivariate normal with correlation \(\rho\)

Cumulative over \((-\infty, h] \times (-\infty, k]\):

\[
\Phi(h, k, \rho) = \frac{1}{2\pi \sqrt{1-\rho^2}} \int_{-\infty}^{h} \int_{-\infty}^{k} \exp \left\{ -\frac{1}{2(1-\rho^2)} \left( x_1^2 - 2\rho x_1 x_2 + x_2^2 \right) \right\} dx_1 dx_2
\]

Domain \(h\): \(-8e+307\) to \(8e+307\)
Domain \(k\): \(-8e+307\) to \(8e+307\)
Domain \(\rho\): \(-1\) to \(1\)
Range: \(0\) to \(1\)

\textbf{lnmvnormalden}(M, V, X)

Description: the natural logarithm of the multivariate normal density

\(M\) is the mean vector, \(V\) is the covariance matrix, and \(X\) is the random vector.

Domain \(M\): \(1 \times n\) and \(n \times 1\) vectors
Domain \(V\): \(n \times n\), positive-definite, symmetric matrices
Domain \(X\): \(1 \times n\) and \(n \times 1\) vectors
Range: \(-8e+307\) to \(8e+307\)
Poisson distribution

`poissonp(m, k)`
Description: the probability of observing floor(k) outcomes that are distributed as Poisson with mean m
The Poisson probability function is evaluated using `gammad()`. 
Domain `m`: 1e–10 to 1e+8  
Domain `k`: 0 to 1e+9  
Range: 0 to 1

`poisson(m, k)`
Description: the probability of observing floor(k) or fewer outcomes that are distributed as Poisson with mean m
The Poisson distribution function is evaluated using `gammap()`. 
Domain `m`: 1e–10 to 2^{53} – 1  
Domain `k`: 0 to 2^{53} – 1  
Range: 0 to 1

`poissontail(m, k)`
Description: the probability of observing floor(k) or more outcomes that are distributed as Poisson with mean m
The reverse cumulative Poisson distribution function is evaluated using `gamm()`. 
Domain `m`: 1e–10 to 2^{53} – 1  
Domain `k`: 0 to 2^{53} – 1  
Range: 0 to 1

`invpoisson(k, p)`
Description: the Poisson mean such that the cumulative Poisson distribution evaluated at k is p: if poisson(m,k) = p, then invpoisson(k,p) = m 
The inverse Poisson distribution function is evaluated using `invgammap()`. 
Domain `k`: 0 to 2^{53} – 1  
Domain `p`: 0 to 1 (exclusive)  
Range: 1.110e–16 to 2^{53}

`invpoissontail(k, q)`
Description: the Poisson mean such that the reverse cumulative Poisson distribution evaluated at k is q: if poissontail(m,k) = q, then invpoissontail(k,q) = m 
The inverse of the reverse cumulative Poisson distribution function is evaluated using `invgamma()`.
Domain `k`: 0 to 2^{53} – 1  
Domain `q`: 0 to 1 (exclusive)  
Range: 0 to 2^{53} (left exclusive)
Student’s t and noncentral Student’s t distributions

\[ t_{den}(df, t) \]
Description: the probability density function of Student’s t distribution

\[ t_{den}(df, t) = \frac{\Gamma\{((df + 1)/2)\}}{\sqrt{\pi df} \Gamma(df/2)} \cdot (1 + t^2/df)^{-(df+1)/2} \]

Domain \( df \): 1e–323 to 8e+307 (may be nonintegral)
Domain \( t \): -8e+307 to 8e+307
Range: 0 to 0.39894 ...

\[ t(df, t) \]
Description: the cumulative Student’s t distribution with \( df \) degrees of freedom
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( t \): -8e+307 to 8e+307
Range: 0 to 1

\[ t_{tail}(df, t) \]
Description: the reverse cumulative (upper tail or survivor) Student’s t distribution; the probability \( T > t \)

\[ t_{tail}(df, t) = \int_{t}^{\infty} \frac{\Gamma\{((df + 1)/2)\}}{\sqrt{\pi df} \Gamma(df/2)} \cdot (1 + x^2/df)^{-(df+1)/2} \, dx \]

Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( t \): -8e+307 to 8e+307
Range: 0 to 1

\[ invt(df, p) \]
Description: the inverse cumulative Student’s t distribution: if \( t(df, t) = p \), then \( invt(df, p) = t \)
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( p \): 0 to 1
Range: -8e+307 to 8e+307

\[ invttail(df, p) \]
Description: the inverse reverse cumulative (upper tail or survivor) Student’s t distribution: if \( t_{tail}(df, t) = p \), then \( invttail(df, p) = t \)
Domain \( df \): 2e–10 to 2e+17 (may be nonintegral)
Domain \( p \): 0 to 1
Range: -8e+307 to 8e+307

\[ invnt(df, np, p) \]
Description: the inverse cumulative noncentral Student’s t distribution: if \( nt(df, np, t) = p \), then \( invnt(df, np, p) = t \)
Domain \( df \): 1 to 1e+6 (may be nonintegral)
Domain \( np \): -1,000 to 1,000
Domain \( p \): 0 to 1
Range: -8e+307 to 8e+307
invnttail(df, np, p)
Description: the inverse reverse cumulative (upper tail or survivor) noncentral Student’s t distribution: if nttail(df, np, t) = p, then invnttail(df, np, p) = t
Domain df: 1 to 1e+6 (may be nonintegral)
Domain np: −1,000 to 1,000
Domain p: 0 to 1
Range: −8e+10 to 8e+10

ntden(df, np, t)
Description: the probability density function of the noncentral Student’s t distribution with df degrees of freedom and noncentrality parameter np
Domain df: 1e–100 to 1e+10 (may be nonintegral)
Domain np: −1,000 to 1,000
Domain t: −8e+307 to 8e+307
Range: 0 to 0.39894 ...

nt(df, np, t)
Description: the cumulative noncentral Student’s t distribution with df degrees of freedom and noncentrality parameter np
   nt(df, 0, t) = t(df, t).
Domain df: 1e–100 to 1e+10 (may be nonintegral)
Domain np: −1,000 to 1,000
Domain t: −8e+307 to 8e+307
Range: 0 to 1

nttail(df, np, t)
Description: the reverse cumulative (upper tail or survivor) noncentral Student’s t distribution with df degrees of freedom and noncentrality parameter np
Domain df: 1e–100 to 1e+10 (may be nonintegral)
Domain np: −1,000 to 1,000
Domain t: −8e+307 to 8e+307
Range: 0 to 1

npnt(df, t, p)
Description: the noncentrality parameter, np, for the noncentral Student’s t distribution: if nt(df, np, t) = p, then npnt(df, t, p) = np
Domain df: 1e–100 to 1e+8 (may be nonintegral)
Domain t: −8e+307 to 8e+307
Domain p: 0 to 1
Range: −1,000 to 1,000
Tukey’s Studentized range distribution

tukeyprob\left(k, df, x\right)
Description: the cumulative Tukey’s Studentized range distribution with \(k\) ranges and \(df\) degrees of freedom; 0 if \(x < 0\)
If \(df\) is a missing value, then the normal distribution is used instead of Student’s \(t\).
tukeyprob() is computed using an algorithm described in Miller (1981).

Domain \(k\): 2 to 1e+6
Domain \(df\): 2 to 1e+6
Domain \(x\): \(-8e+307\) to \(8e+307\)
Range: 0 to 1

invtukeyprob\left(k, df, p\right)
Description: the inverse cumulative Tukey’s Studentized range distribution with \(k\) ranges and \(df\) degrees of freedom
If \(df\) is a missing value, then the normal distribution is used instead of Student’s \(t\).
If \(\text{tukeyprob}(k, df, x) = p\), then \(\text{invtukeyprob}(k, df, p) = x\).
invtukeyprob() is computed using an algorithm described in Miller (1981).

Domain \(k\): 2 to 1e+6
Domain \(df\): 2 to 1e+6
Domain \(p\): 0 to 1
Range: 0 to 8e+307

Weibull distribution

weibullden\left(a, b, x\right)
Description: the probability density function of the Weibull distribution with shape \(a\) and scale \(b\)
\[\text{weibullden}(a,b,x) = \text{weibullden}(a, b, 0, x),\] where \(a\) is the shape, \(b\) is the scale, and \(x\) is the value of Weibull random variable.

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(x\): 1e–323 to 8e+307
Range: 0 to 8e+307
weibullden(a, b, g, x)
Description: the probability density function of the Weibull distribution with shape a, scale b, and location g.

The probability density function of the generalized Weibull distribution is defined as

\[ \frac{a}{b} \left( \frac{x - g}{b} \right)^{a-1} \exp \left\{ - \left( \frac{x - g}{b} \right)^a \right\} \]

for \( x \geq g \) and 0 for \( x < g \), where \( a \) is the shape, \( b \) is the scale, \( g \) is the location parameter, and \( x \) is the value of a generalized Weibull random variable.

Domain:
- \( a \): 1e–323 to 8e+307
- \( b \): 1e–323 to 8e+307
- \( g \): \(-8e+307\) to \(8e+307\)
- \( x \): \(-8e+307\) to \(8e+307\); interesting domain is \( x \geq g \)

Range: 0 to 8e+307

weibull(a, b, x)
Description: the cumulative Weibull distribution with shape a and scale b.

weibull(a, b, x) = weibull(a, b, 0, x), where \( a \) is the shape, \( b \) is the scale, and \( x \) is the value of a Weibull random variable.

Domain:
- \( a \): 1e–323 to 8e+307
- \( b \): 1e–323 to 8e+307
- \( x \): 1e–323 to 8e+307

Range: 0 to 1

weibull(a, b, g, x)
Description: the cumulative Weibull distribution with shape a, scale b, and location g.

The cumulative Weibull distribution is defined as

\[ 1 - \exp \left[ - \left( \frac{x - g}{b} \right)^a \right] \]

for \( x \geq g \) and 0 for \( x < g \), where \( a \) is the shape, \( b \) is the scale, \( g \) is the location parameter, and \( x \) is the value of a Weibull random variable.

The mean of the Weibull distribution is \( g + b \Gamma((a + 1)/a) \) and its variance is \( b^2 \left( \Gamma((a + 2)/a) - \left[ \Gamma((a + 1)/a) \right]^2 \right) \) where \( \Gamma() \) is the gamma function described in lngamma().

Domain:
- \( a \): 1e–323 to 8e+307
- \( b \): 1e–323 to 8e+307
- \( g \): \(-8e+307\) to \(8e+307\)
- \( x \): \(-8e+307\) to \(8e+307\); interesting domain is \( x \geq g \)

Range: 0 to 1
\textbf{weibulltail}(a,b,x) \\
Description: the reverse cumulative Weibull distribution with shape \(a\) and scale \(b\)

\[
\text{weibulltail}(a,b,x) = \text{weibulltail}(a,b,0,x), \text{where } a \text{ is the shape, } b \text{ is the scale, and } x \text{ is the value of a Weibull random variable.}
\]

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(x\): 1e–323 to 8e+307
Range: 0 to 1

\textbf{weibulltail}(a,b,g,x) \\
Description: the reverse cumulative Weibull distribution with shape \(a\), scale \(b\), and location \(g\)

The reverse cumulative Weibull distribution is defined as

\[
\exp \left\{ -\left( \frac{x-g}{b} \right)^a \right\}
\]

for \(x \geq g\) and 0 if \(x < g\), where \(a\) is the shape, \(b\) is the scale, \(g\) is the location parameter, and \(x\) is the value of a generalized Weibull random variable.

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(g\): \(-8e+307\) to \(8e+307\)
Domain \(x\): \(-8e+307\) to \(8e+307\); interesting domain is \(x \geq g\)
Range: 0 to 1

\textbf{invweibull}(a,b,p) \\
Description: the inverse cumulative Weibull distribution with shape \(a\) and scale \(b\): if 
\[
\text{weibull}(a,b,x) = p, \text{ then } \text{invweibull}(a,b,p) = x
\]

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(p\): 0 to 1
Range: 1e–323 to 8e+307

\textbf{invweibull}(a,b,g,p) \\
Description: the inverse cumulative Weibull distribution with shape \(a\), scale \(b\), and location \(g\): if 
\[
\text{weibull}(a,b,g,x) = p, \text{ then } \text{invweibull}(a,b,g,p) = x
\]

Domain \(a\): 1e–323 to 8e+307
Domain \(b\): 1e–323 to 8e+307
Domain \(g\): \(-8e+307\) to \(8e+307\)
Domain \(p\): 0 to 1
Range: \(g + c(\text{epsdouble})\) to 8e+307
**invweibulltail**(a, b, p)

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

Domain a: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain b: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain p: 0 to 1
Range: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)

**invweibulltail**(a, b, g, p)

Description: the inverse reverse cumulative Weibull distribution with shape a, scale b, and location g: if \( \text{weibulltail}(a, b, g, x) = p \), then \( \text{invweibulltail}(a, b, g, p) = x \)

Domain a: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain b: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain g: \( -8 \times 10^{307} \) to \( 8 \times 10^{307} \)
Domain p: 0 to 1
Range: \( g + c(\text{epsdouble}) \) to \( 8 \times 10^{307} \)

**Weibull (proportional hazards) distribution**

**weibullphden**(a, b, x)

Description: the probability density function of the Weibull (proportional hazards) distribution with shape a and scale b

\[
\text{weibullphden}(a, b, x) = \text{weibullphden}(a, b, 0, x), \text{ where } a \text{ is the shape, } b \text{ is the scale, and } x \text{ is the value of Weibull (proportional hazards) random variable.}
\]

Domain a: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain b: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain x: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Range: 0 to \( 8 \times 10^{307} \)

**weibullphden**(a, b, g, x)

Description: the probability density function of the Weibull (proportional hazards) distribution with shape a, scale b, and location g

The probability density function of the Weibull (proportional hazards) distribution is defined as

\[
ba(x - g)^{a-1}\exp\{-b(x - g)^a\}
\]

for \( x \geq g \) and 0 for \( x < g \), where a is the shape, b is the scale, g is the location parameter, and x is the value of a Weibull (proportional hazards) random variable.

Domain a: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain b: \( 1 \times 10^{-323} \) to \( 8 \times 10^{307} \)
Domain g: \( -8 \times 10^{307} \) to \( 8 \times 10^{307} \)
Domain x: \( -8 \times 10^{307} \) to \( 8 \times 10^{307} \); interesting domain is \( x \geq g \)
Range: 0 to \( 8 \times 10^{307} \)
weibullph\((a, b, x)\)
Description: the cumulative Weibull (proportional hazards) distribution with shape \(a\) and scale \(b\)

\[
weibullph(a, b, x) = weibullph(a, b, 0, x),
\]
where \(a\) is the shape, \(b\) is the scale, and \(x\) is the value of a Weibull random variable.

<table>
<thead>
<tr>
<th>Domain (a):</th>
<th>1e–323 to 8e+307</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain (b):</td>
<td>1e–323 to 8e+307</td>
</tr>
<tr>
<td>Domain (x):</td>
<td>1e–323 to 8e+307</td>
</tr>
<tr>
<td>Range:</td>
<td>0 to 1</td>
</tr>
</tbody>
</table>

weibullph\((a, b, g, x)\)
Description: the cumulative Weibull (proportional hazards) distribution with shape \(a\), scale \(b\), and location \(g\)

The cumulative Weibull (proportional hazards) distribution is defined as

\[
1 - \exp\left\{-b(x - g)^a\right\}
\]

for \(x \geq g\) and 0 if \(x < g\), where \(a\) is the shape, \(b\) is the scale, \(g\) is the location parameter, and \(x\) is the value of a Weibull (proportional hazards) random variable. The mean of the Weibull (proportional hazards) distribution is

\[
g + b^{-\frac{2}{a}} \Gamma\{(a + 1)/a\}
\]

and its variance is

\[
b^{-\frac{2}{a}} \left(\Gamma\{(a + 2)/a\} - \left[\Gamma\{(a + 1)/a\}\right]^2\right)
\]

where \(\Gamma()\) is the gamma function described in lngamma\((x)\).

<table>
<thead>
<tr>
<th>Domain (a):</th>
<th>1e–323 to 8e+307</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain (b):</td>
<td>1e–323 to 8e+307</td>
</tr>
<tr>
<td>Domain (g):</td>
<td>-8e+307 to 8e+307</td>
</tr>
<tr>
<td>Domain (x):</td>
<td>-8e+307 to 8e+307; interesting domain is (x \geq g)</td>
</tr>
<tr>
<td>Range:</td>
<td>0 to 1</td>
</tr>
</tbody>
</table>

weibullphtail\((a, b, x)\)
Description: the reverse cumulative Weibull (proportional hazards) distribution with shape \(a\) and scale \(b\)

\[
weibullphtail(a, b, x) = weibullphtail(a, b, 0, x),
\]
where \(a\) is the shape, \(b\) is the scale, and \(x\) is the value of a Weibull (proportional hazards) random variable.

<table>
<thead>
<tr>
<th>Domain (a):</th>
<th>1e–323 to 8e+307</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain (b):</td>
<td>1e–323 to 8e+307</td>
</tr>
<tr>
<td>Domain (x):</td>
<td>1e–323 to 8e+307</td>
</tr>
<tr>
<td>Range:</td>
<td>0 to 1</td>
</tr>
</tbody>
</table>
weibullphtail(a,b,g,x)
Description: the reverse cumulative Weibull (proportional hazards) distribution with shape a, scale b, and location g

The reverse cumulative Weibull (proportional hazards) distribution is defined as

\[
\exp \left\{ -b(x - g)^a \right\}
\]

for \( x \geq g \) and 0 of \( x < g \), where \( a \) is the shape, \( b \) is the scale, \( g \) is the location parameter, and \( x \) is the value of a Weibull (proportional hazards) random variable.

Range: 0 to 1

Domain a: \( 1e^{-323} \) to \( 8e+307 \)
Domain b: \( 1e^{-323} \) to \( 8e+307 \)
Domain g: \(-8e+307 \) to \( 8e+307 \)
Domain x: \(-8e+307 \) to \( 8e+307 \); interesting domain is \( x \geq g \)

invweibullph(a,b,p)
Description: 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

Domain a: \( 1e^{-323} \) to \( 8e+307 \)
Domain b: \( 1e^{-323} \) to \( 8e+307 \)
Domain p: 0 to 1

Range: \( 1e^{-323} \) to \( 8e+307 \)

invweibullph(a,b,g,p)
Description: 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

Domain a: \( 1e^{-323} \) to \( 8e+307 \)
Domain b: \( 1e^{-323} \) to \( 8e+307 \)
Domain g: \(-8e+307 \) to \( 8e+307 \)
Domain p: 0 to 1

Range: \( g + c(\text{epsdouble}) \) to \( 8e+307 \)

invweibullphptail(a,b,p)
Description: the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b: if weibullphptail(a,b,x) = p, then invweibullphptail(a,b,p) = x

Domain a: \( 1e^{-323} \) to \( 8e+307 \)
Domain b: \( 1e^{-323} \) to \( 8e+307 \)
Domain p: 0 to 1

Range: \( 1e^{-323} \) to \( 8e+307 \)
invweibullphtail(a, b, g, p)
Description: 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
Domain a: 1e–323 to 8e+307
Domain b: 1e–323 to 8e+307
Domain g: −8e+307 to 8e+307
Domain p: 0 to 1
Range: g + c(epsdouble) to 8e+307

Wishart distribution
lnwishartden(df, V, X)
Description: the natural logarithm of the density of the Wishart distribution; missing if df ≤ n − 1
df denotes the degrees of freedom, V is the scale matrix, and X is the Wishart random matrix.
Domain df: 1 to 1e+100 (may be nonintegral)
Domain V: n × n, positive-definite, symmetric matrices
Domain X: n × n, positive-definite, symmetric matrices
Range: −8e+307 to 8e+307

lniwishartden(df, V, X)
Description: the natural logarithm of the density of the inverse Wishart distribution; missing if df ≤ n − 1
df denotes the degrees of freedom, V is the scale matrix, and X is the inverse Wishart random matrix.
Domain df: 1 to 1e+100 (may be nonintegral)
Domain V: n × n, positive-definite, symmetric matrices
Domain X: n × n, positive-definite, symmetric matrices
Range: −8e+307 to 8e+307

John Wishart (1898–1956) was born in Montrose, Scotland. He obtained a degree in mathematics
and physics from the University of Edinburgh. He learned mathematics from E. T. Whittaker, upon
whose recommendation he became Karl Pearson’s research assistant. During his apprenticeship,
he worked on approximations to the incomplete beta function and published multiple papers on
this topic. He is best known for deriving the generalized product moment distribution, which
was consequently named the Wishart distribution. This distribution is a critical component in the
calculation of covariance matrices and Bayesian statistics.
Wishart served in both world wars, fighting with the Black Watch regiment in the first and working
for the Intelligence Corps in the second. Upon his return from World War II, he resumed his
involvement with the Royal Statistical Society, becoming chairman of the Research Section in
1945. A few years later, he also served as Associate Editor for the journal Biometrika.
He taught courses in statistics and agriculture at Cambridge and became the Head of the Statistical
Laboratory. He published multiple papers applying statistical methods to agricultural research
and was involved with the United Nations Food and Agriculture Organization. He was in Mexico
to establish an agricultural research center on behalf of this organization when he died.
References


Also see

[FN] Functions by category

[D] egen — Extensions to generate

[D] generate — Create or change contents of variable

[M-4] Statistical — Statistical functions

[U] 13.3 Functions
### String functions

#### Contents

*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*(\(s_1, s_2\))

the position in ASCII string \(s_1\) of the first character of \(s_1\) not found in ASCII string \(s_2\), or 0 if all characters of \(s_1\) are found in \(s_2\)

*plural*(\(n, s\))

the plural of \(s\) if \(n \neq \pm 1\)

*plural*(\(n, s_1, s_2\))

the plural of \(s_1\), as modified by or replaced with \(s_2\), if \(n \neq \pm 1\)

*real*(\(s\))

\(s\) converted to numeric or missing

*regexcapture*(\(n\))

subexpression \(n\) from a previous *regexm()* or *regexmatch()* match

*regexcapturenamed*(\(grp\))

subexpression corresponding to matching group named \(grp\) in regular expression from a previous *regexm()* or *regexmatch()* match

*regexm*(\(s, re\))

a match of a regular expression, which evaluates to 1 if regular expression \(re\) is satisfied by the ASCII string \(s\); otherwise, 0

*regexmatch*(\(s, re[ , noc\[ , std\[ , nlalt\] ] ]\) )

a match of a regular expression, which evaluates to 1 if regular expression \(re\) is satisfied by the ASCII string \(s\); otherwise, 0

*regexr*(\(s_1, re, s_2\))

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

*regexreplace*(\(s_1, re, s_2[ , noc\[ , fmt\[ , std\[ , nlalt\] ] ] ]\) )

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

*regexreplaceall*(\(s_1, re, s_2[ , noc\[ , fmt\[ , std\[ , nlalt\] ] ] ]\) )

replaces all substrings within ASCII string \(s_1\) that match \(re\) with ASCII string \(s_2\) and returns the resulting string

*regexs*(\(n\))

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

*soundex*(\(s\))

the soundex code for a string, \(s\)

*soundex_nara*(\(s\))

the U.S. Census soundex code for a string, \(s\)

*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*(\(n\))

---

149
The string functions in Stata include:

- `string(n,s)`: a synonym for `strofreal(n,s)`
- `strtrim(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(s1,s2)`: 1 if `s1` matches the pattern `s2`; otherwise, 0
- `strofreal(n)`: `n` converted to a string
- `strofreal(n,s)`: `n` converted to a string using the specified display format
- `strpos(s1,s2)`: the position in `s1` at which `s2` is first found, 0 if `s2` does not occur, and 1 if `s2` is empty
- `strproper(s)`: a string with the first ASCII letter and any other letters immediately following characters that are not letters capitalized; all other ASCII letters converted to lowercase
- `strreverse(s)`: the reverse of ASCII string `s`
- `strrpos(s1,s2)`: the position in `s1` at which `s2` is last found, 0 if `s2` does not occur, and 1 if `s2` is empty
- `strrtrim(s)`: `s` without trailing blanks (ASCII space character `char(32)`)
- `strtoname(s[,p])`: `s` translated into a Stata 13 compatible name
- `strtrim(s)`: `s` without leading and trailing blanks (ASCII space character `char(32)`); equivalent to `strltrim(strrtrim(s))`
- `strupper(s)`: uppercase ASCII characters in string `s`
- `subinword(s1,s2,s3,n)`: `s1`, where the first `n` occurrences in `s1` of `s2` have been replaced with `s3`
- `subinword(s1,s2,s3,n)`: `s1`, where the first `n` occurrences in `s1` of `s2` as a word have been replaced with `s3`
- `substr(s,n1,n2)`: the substring of `s`, starting at `n1`, for a length of `n2`
- `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,n1,n2)`: the Unicode substring of `s`, starting at character `n1`, for `n2` 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(s1,s2[,loc])`: compares two Unicode strings
- `ustrcompareex(s1,s2,case,cslv,norm,num,alt,fr)`: compares two Unicode strings
- `ustrfix(s[,rep])`: replaces each invalid UTF-8 sequence with a Unicode character
- `ustrfrom(s,enc,mode)`: converts the string `s` in encoding `enc` to a UTF-8 encoded Unicode string
- `ustrinvalidcnt(s)`: the number of invalid UTF-8 sequences in `s`
- `ustrleft(s,n)`: the first `n` Unicode characters of the Unicode string `s`
ustrlen(s)
the number of characters in the Unicode string s
ustrlower(s[,loc])
lowercase all characters of Unicode string s under the given locale loc
ustrltrim(s)
removes the leading Unicode whitespace characters and blanks from the Unicode string s
ustrnormalize(s,norm)
normalizes Unicode string s to one of the five normalization forms specified by norm
ustrpos(s1,s2[,n])
the position in s1 at which s2 is first found; otherwise, 0
ustrregexm(s,re[,noc])
performs a match of a regular expression and evaluates to 1 if regular expression re is satisfied by the Unicode string s; otherwise, 0
ustrregexra(s1,re,s2[,noc])
replaces all substrings within the Unicode string s1 that match re with s2 and returns the resulting string
ustrregexrf(s1,re,s2[,noc])
replaces the first substring within the Unicode string s1 that matches re with s2 and returns the resulting string
ustrregexs(n)
subexpression n from a previous ustrregexm() match
ustrreverse(s)
the reverse of Unicode string s
ustrright(s,n)
the last n Unicode characters of the Unicode string s
ustrrpos(s1,s2[,n])
the position in s1 at which s2 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,sc,case,cs,slv,case,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])
upercase all characters in string s under the given locale loc
ustrword(s,n[,loc])
the nth Unicode word in the Unicode string s
ustrwordcount(s[,loc])
the number of nonempty Unicode words in the Unicode string s
usubinstr(s1,s2,s3,n)
replaces the first n occurrences of the Unicode string s2 with the Unicode string s3 in s1
usubstr(s,n1,n2)
the Unicode substring of s, starting at n1, for a length of n2
word(s,n)
the nth word in s; missing (""") if n is missing
wordbreaklocale(loc,type) the most closely related locale supported by ICU from loc if type is 1, the actual locale where the word-boundary analysis data come from if type is 2; or an empty string is returned for any other type

wordcount(s) the number of words in s

Functions

In the display below, s indicates a string subexpression (a string literal, a string variable, or another string expression) and n indicates a numeric subexpression (a number, a numeric variable, or another numeric expression).

If your strings contain Unicode characters or you are writing programs that will be used by others who might use Unicode strings, read [U] 12.4.2 Handling Unicode strings.

abbrev(s,n)
Description: name s, abbreviated to a length of n
Length is measured in the number of display columns, not in the number of characters. For most users, the number of display columns equals the number of characters. For a detailed discussion of display columns, see [U] 12.4.2.2 Displaying Unicode characters.
If any of the characters of s are a period, “.”, and n < 8, then the value of n defaults to a value of 8. Otherwise, if n < 5, then n defaults to a value of 5. If n is missing, abbrev() will return the entire string s. abbrev() is typically used with variable names and variable names with factor-variable or time-series operators (the period case).

abbrev("displacement",8) is displa~t.

Domain s: strings
Domain n: integers 5 to 32
Range: strings

char(n)
Description: the character corresponding to ASCII or extended ASCII code n; "" if n is not in the domain
Note: ASCII codes are from 0 to 127; extended ASCII codes are from 128 to 255. Prior to Stata 14, the display of extended ASCII characters was encoding dependent. For example, char(128) on Microsoft Windows using Windows-1252 encoding displayed the Euro symbol, but on Linux using ISO-Latin-1 encoding, char(128) displayed an invalid character symbol. Beginning with Stata 14, Stata’s display encoding is UTF-8 on all platforms. The char(128) function is an invalid UTF-8 sequence and thus will display a question mark. There are two Unicode functions corresponding to char(): uchar() and ustrunescape(). You can use uchar(8364) or ustrunescape("\u20AC") to display a Euro sign on all platforms.

Domain n: integers 0 to 255
Range: ASCII characters
uchar(n)
Description: the Unicode character corresponding to Unicode code point n or an empty string if n is beyond the Unicode code-point range
Note that uchar() takes the decimal value of the Unicode code point. ustrunescape() takes an escaped hex digit string of the Unicode code point. For example, both uchar(8364) and ustrunescape("\u20ac") produce the Euro sign.
Domain n: integers ≥ 0
Range: Unicode characters

collatorlocale(loc, type)
Description: 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
For any other type, loc is returned in a canonicalized form.
collatorlocale("en_us_texas", 0) = en_US_Texas
collatorlocale("en_us_texas", 1) = en_US
collatorlocale("en_us_texas", 2) = root
Domain loc: strings of locale name
Domain type: integers
Range: strings

collatorversion(loc)
Description: the version string of a collator based on locale loc
The Unicode standard is constantly adding more characters and the sort key format may change as well. This can cause ustrsortkey() and ustrsortkeyex() to produce incompatible sort keys between different versions of International Components for Unicode. The version string can be used for versioning the sort keys to indicate when saved sort keys must be regenerated.
Range: strings

indexnot(s1, s2)
Description: 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
indexnot() is intended for use only with plain ASCII strings. For Unicode characters beyond the plain ASCII range, the position and character are given in bytes, not characters.
Domain s1: ASCII strings (to be searched)
Domain s2: ASCII strings (to search for)
Range: integers ≥ 0
plural($n, s$)
Description: the plural of $s$ if $n \neq \pm 1$

The plural is formed by adding “s” to $s$.

 plural(1, "horse") = "horse"
 plural(2, "horse") = "horses"

Domain $n$: real numbers
Domain $s$: strings
Range: strings

plural($n, s_1, s_2$)
Description: the plural of $s_1$, as modified by or replaced with $s_2$, if $n \neq \pm 1$

If $s_2$ begins with the character “+”, the plural is formed by adding the remainder of $s_2$ to $s_1$. If $s_2$ begins with the character “-”, the plural is formed by subtracting the remainder of $s_2$ from $s_1$. If $s_2$ begins with neither “+” nor “-”, then the plural is formed by returning $s_2$.

 plural(2, "glass", "+es") = "glasses"
 plural(1, "mouse", "mice") = "mouse"
 plural(2, "mouse", "mice") = "mice"
 plural(2, "abcdefg", "-efg") = "abcd"

Domain $n$: real numbers
Domain $s_1$: strings
Domain $s_2$: strings
Range: strings

real($s$)
Description: $s$ converted to numeric or missing

Also see strofreal().

 real("5.2")+1 = 6.2
 real("hello") = .

Domain $s$: strings
Range: $-8e+307$ to $8e+307$ or missing

regexcapture($n$)
Description: subexpression $n$ from a previous regexm() or regexmatch() match

regexcapture(0) returns the entire string that satisfied the regular expression.

Domain $n$: integers
Range: ASCII strings or missing

regexcapturenamed($grp$)
Description: subexpression corresponding to matching group named $grp$ in regular expression from a previous regexm() or regexmatch() match

Domain $grp$: ASCII strings
Range: ASCII strings or missing
regexm(s,re)
Description: a match of a regular expression, which evaluates to 1 if regular expression re is satisfied by the ASCII string s; otherwise, 0

Regular expression syntax is based on Henry Spencer’s NFA algorithm, and this is nearly identical to the POSIX.2 standard. s and re may not contain binary 0 (\0).

regexm() is intended for use only with plain ASCII characters. For Unicode characters beyond the plain ASCII range, the match is based on bytes. For a character-based match, see ustrregexm().

For more advanced regular expression matching, see regexmatch().

Domain s: ASCII strings
Domain re: regular expressions
Range: 0, 1, or missing

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

regexmatch() is intended for use only with plain ASCII characters. For Unicode characters beyond the plain ASCII range, the match is based on bytes. For a character-based match, see ustrregexm().

If noc is specified and is not 0, a case-insensitive match is performed; otherwise, a case-sensitive match is performed.

std specifies the regular expression standard: 1 for POSIX Extended Regular, 2 for POSIX Basic Regular, 3 for Emacs, 4 for AWK, 5 for grep, 6 for egrep, or any other number for Perl, the default.

If nlalt is specified and is 0, the newline character, char(10), is not treated like alternation operator |; otherwise, newline has the same effect as |.

s and re may not contain binary 0 (\0).

Domain s: ASCII strings
Domain re: regular expression
Domain noc: integers
Domain std: integers
Domain nlalt: integers
Range: 0, 1, or missing
\textbf{regexr}(s_1, re, s_2)

Description: replaces the first substring within ASCII string \( s_1 \) that matches \( re \) with ASCII string \( s_2 \) and returns the resulting string.

If \( s_1 \) contains no substring that matches \( re \), the unaltered \( s_1 \) is returned. \( s_1 \) and the result of \( \text{regexr}() \) may be at most 1,100,000 characters long. \( s_1, re, \) and \( s_2 \) may not contain binary 0 (\( \backslash 0 \)).

\( \text{regexr}() \) is intended for use only with plain ASCII characters. For Unicode characters beyond the plain ASCII range, the match is based on bytes, and the result is restricted to 1,100,000 bytes. For a character-based match, see \( \text{ustrregexrf}() \) or \( \text{ustrregexra}() \).

For more advanced regular expression replacement, see \( \text{regexreplace}() \) and \( \text{regexreplaceall}() \).

\begin{itemize}
  \item Domain \( s_1 \): ASCII strings
  \item Domain \( re \): regular expressions
  \item Domain \( s_2 \): ASCII strings
  \item Range: ASCII strings
\end{itemize}

\textbf{regexreplace}(s_1, re, s_2[, noc[, fmt[, std[, nlalt]]]])

Description: replaces the first substring within ASCII string \( s_1 \) that matches \( re \) with ASCII string \( s_2 \) and returns the resulting string.

If \( noc \) is specified and is not 0, a case-insensitive match is performed; otherwise, a case-sensitive match is performed.

\( fmt \) specifies the format string syntax supported in \( s_2 \): 1 for literal, where \( s_2 \) is treated as a string literal (no special character substitution), 2 for sed, or any other number for Perl, the default.

\( std \) specifies the regular expression standard: 1 for POSIX Extended Regular, 2 for POSIX Basic Regular, 3 for Emacs, 4 for AWK, 5 for grep, 6 for egrep, or any other number for Perl, the default.

If \( nlalt \) is specified and is 0, the newline character, \texttt{char(10)}{}, is not treated like alternation operator |; otherwise, newline has the same effect as |.

If \( s_1 \) contains no substring that matches \( re \), the unaltered \( s_1 \) is returned. \( s_1, s_2, \) and \( re \) may not contain binary 0 (\( \backslash 0 \)).

\begin{itemize}
  \item Domain \( s_1 \): ASCII strings
  \item Domain \( re \): regular expression
  \item Domain \( s_2 \): ASCII strings
  \item Domain \( noc \): integers
  \item Domain \( fmt \): integers
  \item Domain \( std \): integers
  \item Domain \( nlalt \): integers
  \item Range: ASCII strings
\end{itemize}
Description: replaces all substrings within ASCII string $s_1$ that match $re$ with ASCII string $s_2$ and returns the resulting string

If $noc$ is specified and is not 0, a case-insensitive match is performed; otherwise, a case-sensitive match is performed.

$fmt$ specifies the format string syntax supported in $s_2$: 1 for literal, where $s_2$ is treated as a string literal (no special character substitution), 2 for sed, or any other number for Perl, the default.

$std$ specifies the regular expression standard: 1 for POSIX Extended Regular, 2 for POSIX Basic Regular, 3 for Emacs, 4 for AWK, 5 for grep, 6 for egrep, or any other number for Perl, the default.

If $nlalt$ is specified and is 0, the newline character, char(10), is not treated like alternation operator |; otherwise, newline has the same effect as |.

If $s_1$ contains no substring that matches $re$, the unaltered $s_1$ is returned. $s_1$, $s_2$, and $re$ may not contain binary 0 (\0).

Domain $s_1$: ASCII strings
Domain $re$: regular expression
Domain $s_2$: ASCII strings
Domain $noc$: integers
Domain $fmt$: integers
Domain $std$: integers
Domain $nlalt$: integers
Range: ASCII strings

regesx($n$)
Description: subexpression $n$ from a previous regexm() or regexmatch() match, where $0 \leq n < 10$

Subexpression 0 is reserved for the entire string that satisfied the regular expression. The returned subexpression may be at most 1,100,000 characters (bytes) long.

For more options to return matching substrings, see regexcapture() and regexcapturenamed().

Domain $n$: 0 to 9
Range: ASCII strings

Description: performs a match of a regular expression and evaluates to 1 if regular expression $re$ is satisfied by the Unicode string $s$; otherwise, 0

If $noc$ is specified and not 0, a case-insensitive match is performed. The function may return a negative integer if an error occurs.

ustrregexm("12345", "([0-9])\{5\}") = 1
ustrregexm("de TRÈS près", "rè\$") = 1
ustrregexm("de TRÈS près", "Rè\$") = 0
ustrregexm("de TRÈS près", "Rè\$", 1) = 1

Domain $s$: Unicode strings
Domain $re$: Unicode regular expressions
Domain $noc$: integers
Range: integers
ustrregexrf($s_1, re, s_2[, , noc])
Description: replaces the first substring within the Unicode string $s_1$ that matches $re$ with $s_2$ and returns the resulting string
If $noc$ is specified and not 0, a case-insensitive match is performed. The function may return an empty string if an error occurs.
`ustrregexrf("très près", "rès", "X") = "tX près"
`ustrregexrf("TRÈS près", "Rès", "X") = "TRÈS près"
`ustrregexrf("TRÈS près", "Rès", "X", 1) = "TX près"

Domain $s_1$: Unicode strings
Domain $re$: Unicode regular expressions
Domain $s_2$: Unicode strings
Domain $noc$: integers
Range: Unicode strings

ustrregexra($s_1, re, s_2[, , noc])
Description: replaces all substrings within the Unicode string $s_1$ that match $re$ with $s_2$ and returns the resulting string
If $noc$ is specified and not 0, a case-insensitive match is performed. The function may return an empty string if an error occurs.
`ustrregexra("très près", "rès", "X") = "tX pX"
`ustrregexra("TRÈS près", "Rès", "X") = "TRÈS près"
`ustrregexra("TRÈS près", "Rès", "X", 1) = "TX pX"

Domain $s_1$: Unicode strings
Domain $re$: Unicode regular expressions
Domain $s_2$: Unicode strings
Domain $noc$: integers
Range: Unicode strings

ustrregexs($n$)
Description: subexpression $n$ from a previous `ustrregexm()` match
Subexpression 0 is reserved for the entire string that satisfied the regular expression. The function may return an empty string if $n$ is larger than the maximum count of subexpressions from the previous match or if an error occurs.

Domain $n$: integers $\geq 0$
Range: strings
soundex(s)
Description: the soundex code for a string, s

The soundex code consists of a letter followed by three numbers: the letter is the first ASCII letter of the name and the numbers encode the remaining consonants. Similar sounding consonants are encoded by the same number. Unicode characters beyond the plain ASCII range are ignored.

soundex("Ashcraft") = "A226"
soundex("Robert") = "R163"
soundex("Rupert") = "R163"

Domain s: strings
Range: strings

soundex_nara(s)
Description: the U.S. Census soundex code for a string, s

The soundex code consists of a letter followed by three numbers: the letter is the first ASCII letter of the name and the numbers encode the remaining consonants. Similar sounding consonants are encoded by the same number. Unicode characters beyond the plain ASCII range are ignored.

soundex_nara("Ashcraft") = "A261"

Domain s: strings
Range: strings

strcat(s₁,s₂)
Description: there is no strcat() function; instead the addition operator is used to concatenate strings

"hello " + "world" = "hello world"
"a" + "b" = "ab"
"Café " + "de Flore" = "Café de Flore"

Domain s₁: strings
Domain s₂: strings
Range: strings

strdup(s₁,n)
Description: there is no strdup() function; instead the multiplication operator is used to create multiple copies of strings

"hello" * 3 = "hellohellohello"
3 * "hello" = "hellohellohello"
0 * "hello" = ""
"hello" * 1 = "hello"
"Здравствуйте " * 2 = "Здравствуйте Здравствуйте "

Domain s₁: strings
Domain n: nonnegative integers 0, 1, 2, ...
Range: strings
160  String functions

string($n$)
Description: a synonym for strofreal($n$)

string($n,s$)
Description: a synonym for strofreal($n,s$)

strtrim($s$)
Description: $s$ with multiple, consecutive internal blanks (ASCII space character char(32)) collapsed to one blank

\[
\text{strtrim("hello there") = "hello there"}
\]
Domain $s$: strings
Range: strings with no multiple, consecutive internal blanks

strlen($s$)
Description: the number of characters in ASCII $s$ or length in bytes

strlen() is intended for use only with plain ASCII characters and for use by programmers who want to obtain the byte-length of a string. Note that any Unicode character beyond ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, é takes 2 bytes.

For the number of characters in a Unicode string, see ustrlen().

\[
\begin{align*}
\text{strlen("ab")} &= 2 \\
\text{strlen("é")} &= 2
\end{align*}
\]
Domain $s$: strings
Range: integers $\geq 0$

ustrlen($s$)
Description: the number of characters in the Unicode string $s$

An invalid UTF-8 sequence is counted as one Unicode character. An invalid UTF-8 sequence may contain one byte or multiple bytes. Note that any Unicode character beyond the plain ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, é takes 2 bytes.

\[
\begin{align*}
\text{ustrlen("médiiane")} &= 7 \\
\text{strlen("médiiane")} &= 8
\end{align*}
\]
Domain $s$: Unicode strings
Range: integers $\geq 0$
String functions

udstrlen(s)
Description: the number of display columns needed to display the Unicode string \( s \) in the Stata Results window
A Unicode character in the CJK (Chinese, Japanese, and Korean) encoding usually requires two display columns; a Latin character usually requires one column. Any invalid UTF-8 sequence requires one column.

\[
\text{udstrlen("中值")} = 4 \\
\text{ustrlen("中值")} = 2 \\
\text{strlen("中值")} = 6
\]
Domain \( s \): Unicode strings
Range: integers \( \geq 0 \)

strlower(s)
Description: lowercase ASCII characters in string \( s \)
Unicode characters beyond the plain ASCII range are ignored.

\[
\text{strlower("THIS")} = \text{"this"} \\
\text{strlower("CAFÉ")} = \text{"cafÉ"}
\]
Domain \( s \): strings
Range: strings with lowercased characters

ustrlower(s[,loc])
Description: lowercase all characters of Unicode string \( s \) under the given locale \( loc \)
If \( loc \) is not specified, the default locale is used. The same \( s \) but different \( loc \) may produce different results; for example, the lowercase letter of “I” is “i” in English but a dotless “i” in Turkish. The same Unicode character can be mapped to different Unicode characters based on its surrounding characters; for example, Greek capital letter sigma \( \Sigma \) has two lowercases: \( \varsigma \), if it is the final character of a word, or \( \sigma \). The result can be longer or shorter than the input Unicode string in bytes.

\[
\text{ustrlower("MÉDIANE","fr")} = \text{"médiane"} \\
\text{ustrlower("İSTANBUL","tr")} = \text{"istanbul"} \\
\text{ustrlower("ΟΔΥΣΣΕΥΣΣ")} = \text{"όδυσσεύς"}
\]
Domain \( s \): Unicode strings
Domain \( loc \): locale name
Range: Unicode strings

strltrim(s)
Description: \( s \) without leading blanks (ASCII space character \( \text{char}(32) \))

\[
\text{strltrim(" this")} = \text{"this"}
\]
Domain \( s \): strings
Range: strings without leading blanks
ustrltrim($x$)
Description: removes the leading Unicode whitespace characters and blanks from the Unicode string $s$

Note that, in addition to $\text{char}(32)$, ASCII characters $\text{char}(9)$, $\text{char}(10)$, $\text{char}(11)$, $\text{char}(12)$, and $\text{char}(13)$ are whitespace characters in Unicode standard.

ustrltrim(" this") = "this"
ustrltrim(char(9)+"this") = "this"
ustrltrim(unstrunescape("\u1680")+" this") = "this"

Domain $s$: Unicode strings
Range: Unicode strings

strmatch($s_1, s_2$)
Description: 1 if $s_1$ matches the pattern $s_2$; otherwise, 0

strmatch("17.4","1??4") returns 1. In $s_2$, "?" means that one character goes here, and "*" means that zero or more bytes go here. Note that a Unicode character may contain multiple bytes; thus, using "*" with Unicode characters can infrequently result in matches that do not occur at a character boundary.

Also see regexm(), regexr(), and regexs().

strmatch("céf", "caf?") = 1

Domain $s_1$: strings
Domain $s_2$: strings
Range: integers 0 or 1

strofreal($n$)
Description: $n$ converted to a string

Also see real().

strofreal(4)+"F" = "4F"
strofreal(1234567) = "1234567"
strofreal(12345678) = "1.23e+07"
strofreal(.) = "."

Domain $n$: $-8e+307$ to $8e+307$ or missing
Range: strings
strofreal\(n, s\)
Description: \(n\) converted to a string using the specified display format

Also see `real()`.

\[
\begin{align*}
\text{strofreal}(4, "\%9.2f") &= "4.00" \\
\text{strofreal}(123456789, "\%11.0g") &= "123456789" \\
\text{strofreal}(123456789, "\%13.0gc") &= "123,456,789" \\
\text{strofreal}(0, "\%td") &= "01jan1960" \\
\text{strofreal}(225, "\%tq") &= "2016q2" \\
\text{strofreal}(225, "not a format") &= ""
\end{align*}
\]

Domain \(n\): \(-8e+307\) to \(8e+307\) or missing

Domain \(s\): strings containing \(\%fmt\) numeric display format

Range: strings

strpos\(s_1, s_2\)
Description: the position in \(s_1\) at which \(s_2\) is first found, 0 if \(s_2\) does not occur, and 1 if \(s_2\) is empty

`strpos()` is intended for use only with plain ASCII characters and for use by programmers who want to obtain the byte-position of \(s_2\). Note that any Unicode character beyond ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, é takes 2 bytes.

To find the character position of \(s_2\) in a Unicode string, see `ustrpos()`.

\[
\begin{align*}
\text{strpos}("this", "is") &= 3 \\
\text{strpos}("this", "it") &= 0 \\
\text{strpos}("this", ") &= 1
\end{align*}
\]

Domain \(s_1\): strings (to be searched)

Domain \(s_2\): strings (to search for)

Range: integers \(\geq 0\)

ustrpos\(s_1, s_2[n]\)
Description: the position in \(s_1\) at which \(s_2\) is first found; otherwise, 0

If \(n\) is specified and is greater than 0, the search starts at the \(n\)th Unicode character of \(s_1\). An invalid UTF-8 sequence in either \(s_1\) or \(s_2\) is replaced with a Unicode replacement character \(\ufffd\) before the search is performed.

\[
\begin{align*}
\text{ustrpos}("médiane", "édi") &= 2 \\
\text{ustrpos}("médiane", "édi", 3) &= 0 \\
\text{ustrpos}("médiane", ") &= 0
\end{align*}
\]

Domain \(s_1\): Unicode strings (to be searched)

Domain \(s_2\): Unicode strings (to search for)

Domain \(n\): integers

Range: integers
strproper(s)
Description: 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

strproper() implements a form of titlecasing and is intended for use only with plain ASCII strings. Unicode characters beyond ASCII are treated as characters that are not letters. To titlecase strings with Unicode characters beyond the plain ASCII range or to implement language-sensitive rules for titlecasing, see ustrtitle().

strproper("mR. joHn a. sMitH") = "Mr. John A. Smith"
strproper("jack o’reilly") = "Jack O’Reilly"
strproper("2-cent’s worth") = "2-Cent’S Worth"
strproper("vous êtes") = "Vous Êtes"

Domain s: strings
Range: strings

ustrtitle(s[,loc])
Description: a string with the first characters of Unicode words titlecased and other characters lowercased

If loc is not specified, the default locale is used. Note that a Unicode word is different from a Stata word produced by function word(). The Stata word is a space-separated token. A Unicode word is a language unit based on either a set of word-boundary rules or dictionaries for some languages (Chinese, Japanese, and Thai). The titlecase is also locale dependent and context sensitive; for example, lowercase “ij” is considered a digraph in Dutch. Its titlecase is “IJ”.

ustrtitle("vous êtes", "fr") = "Vous Êtes"
ustrtitle("mR. joHn a. sMitH") = "Mr. John A. Smith"
ustrtitle("ijmuiden", "en") = "Ijmuiden"
ustrtitle("ijmuiden", "nl") = "IJmuiden"

Domain s: Unicode strings
Domain loc: Unicode strings
Range: Unicode strings

strreverse(s)
Description: the reverse of ASCII string s

strreverse() is intended for use only with plain ASCII characters. For Unicode characters beyond ASCII range (code point greater than 127), the encoded bytes are reversed.

To reverse the characters of Unicode string, see ustrreverse().

strreverse("hello") = "olleh"

Domain s: ASCII strings
Range: ASCII reversed strings
ustrreverse(s)
Description: the reverse of Unicode string s

The function does not take Unicode character equivalence into consideration. Hence, a Unicode character in a decomposed form will not be reversed as one unit. An invalid UTF-8 sequence is replaced with a Unicode replacement character \ufffd.

ustrreverse("médiane") = "enaidém"

Domain s: Unicode strings
Range: reversed Unicode strings

strrpos(s1, s2)
Description: the position in s1 at which s2 is last found, 0 if s2 does not occur, and 1 if s2 is empty

strrpos() is intended for use only with plain ASCII characters and for use by programmers who want to obtain the last byte-position of s2. Note that any Unicode character beyond ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, é takes 2 bytes.

To find the last character position of s2 in a Unicode string, see ustrrpos().

strrpos("this","is") = 3
strrpos("this","is") = 6
strrpos("this","it") = 0
strrpos("this",""") = 1

Domain s1: strings (to be searched)
Domain s2: strings (to search for)
Range: integers ≥ 0

ustrrpos(s1, s2[, n])
Description: the position in s1 at which s2 is last found; otherwise, 0

If n is specified and is greater than 0, only the part between the first Unicode character and the nth Unicode character of s1 is searched. An invalid UTF-8 sequence in either s1 or s2 is replaced with a Unicode replacement character \ufffd before the search is performed.

ustrrpos("enchânté", "n") = 6
ustrrpos("enchânté", "n", 5) = 2
ustrrpos("enchânté", "n", 6) = 6
ustrrpos("enchânté", "ne") = 0

Domain s1: Unicode strings (to be searched)
Domain s2: Unicode strings (to search for)
Domain n: integers
Range: integers

strrtrim(s)
Description: s without trailing blanks (ASCII space character char(32))

strrtrim("this ") = "this"

Domain s: strings
Range: strings without trailing blanks
ustrrtrim(s)
Description: remove trailing Unicode whitespace characters and blanks from the Unicode string s
Note that, in addition to char(32), ASCII characters char(9), char(10), char(11), char(12), and char(13) are considered whitespace characters in the Unicode standard.
ustrrtrim("this ") = "this"
ustrrtrim("this"+char(10)) = "this"
ustrrtrim("this "+ustrunescape("\u2000")) = "this"
Domain s: Unicode strings
Range: Unicode strings

strtoname(s[ ,p ])
Description: s translated into a Stata 13 compatible name
strtoname() results in a name that is truncated to 32 bytes. Each character in s that is not allowed in a Stata name is converted to an underscore character, _. If the first character in s is a numeric character and p is not 0, then the result is prefixed with an underscore. Stata 14 names may be 32 characters; see [U] 11.3 Naming conventions.
strtoname("name") = "name"
strtoname("a name") = "a_name"
strtoname("5",1) = "_5"
strtoname("5:30",1) = "_5_30"
strtoname("5",0) = "5"
strtoname("5:30",0) = "5_30"
Domain s: strings
Domain p: integers 0 or 1
Range: strings

ustrtoname(s[ ,p ])
Description: string s translated into a Stata name
ustrtoname() results in a name that is truncated to 32 characters. Each character in s that is not allowed in a Stata name is converted to an underscore character, _. If the first character in s is a numeric character and p is not 0, then the result is prefixed with an underscore.
ustrtoname("name",1) = "name"
ustrtoname("the médiane") = "the_médiane"
ustrtoname("0média\nne") = "_0médiane"
ustrtoname("0média\nne", 1) = "_0médiane"
ustrtoname("0média\nne", 0) = "0médiane"
Domain s: Unicode strings
Domain p: integers 0 or 1
Range: Unicode strings
**strtrim(s)**

Description: $s$ without leading and trailing blanks (ASCII space character `char(32)`); equivalent to `strltrim(strrtrim(s))`

```
strtrim(" this ") = "this"
```

Domain $s$: strings
Range: strings without leading or trailing blanks

**ustrtrim(s)**

Description: removes leading and trailing Unicode whitespace characters and blanks from the Unicode string $s$

Note that, in addition to `char(32)`, ASCII characters `char(9)`, `char(10)`, `char(11)`, `char(12)`, and `char(13)` are considered whitespace characters in the Unicode standard.

```
ustrtrim(" this ") = "this"
ustrtrim(char(11)+" this "+char(13)) = "this"
ustrtrim(" this "+ustrunescape("\u2000")) = "this"
```

Domain $s$: Unicode strings
Range: Unicode strings

**strupper(s)**

Description: uppercase ASCII characters in string $s$

Unicode characters beyond the plain ASCII range are ignored.

```
strupper("this") = "THIS"
strupper("café") = "CAFé"
```

Domain $s$: strings
Range: strings with uppercased characters

**ustrupper(s[,loc])**

Description: uppercase all characters in string $s$ under the given locale $loc$

If $loc$ is not specified, the default locale is used. The same $s$ but a different $loc$ may produce different results; for example, the uppercase letter of “i” is “I” in English, but “ı” with a dot in Turkish. The result can be longer or shorter than the input string in bytes; for example, the uppercase form of the German letter ß (code point \u00df) is two capital letters “SS”.

```
ustrupper("médiáne","fr") = "MÉDIANE"
ustrupper("Rußland", "de") = "RUSSLAND"
ustrupper("İstanbul", "tr") = "İSTANBUL"
```

Domain $s$: Unicode strings
Domain $loc$: locale name
Range: Unicode strings
subinstr($s_1, s_2, s_3, n$)
Description: $s_1$, where the first $n$ occurrences in $s_1$ of $s_2$ have been replaced with $s_3$

$subinstr$ is intended for use only with plain ASCII characters and for use by programmers who want to perform byte-based substitution. Note that any Unicode character beyond ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, ´e takes 2 bytes.

To perform character-based replacement in Unicode strings, see $usubinstr$.

If $n$ is missing, all occurrences are replaced.

Also see $regex()$, $regexr()$, and $regexs()$.

$subinstr(\"this is the day\",\"is\",\"X\",1) = \"th\<no-break/\>X is the day\"
$subinstr(\"this is the hour\",\"is\",\"X\",2) = \"th\<no-break/\>X X the hour\"
$subinstr(\"this is this\",\"is\",\"X\",.) = \"th\<no-break/\>X \th\<no-break/\>X\"

Domain $s_1$: strings (to be substituted into)
Domain $s_2$: strings (to be substituted from)
Domain $s_3$: strings (to be substituted with)
Domain $n$: integers ≥ 0 or missing
Range: strings

usubinstr($s_1, s_2, s_3, n$)
Description: replaces the first $n$ occurrences of the Unicode string $s_2$ with the Unicode string $s_3$ in $s_1$

If $n$ is missing, all occurrences are replaced. An invalid UTF-8 sequence in $s_1$, $s_2$, or $s_3$ is replaced with a Unicode replacement character \ufffd before replacement is performed.

$usubinstr(\"de très près\",\"ès\",\"es\",1) = \"de tres près\"
$usubinstr(\"de très près\",\"ès\",\"X\",2) = \"de tr\<no-break/\>X pr\<no-break/\>X\"

Domain $s_1$: Unicode strings (to be substituted into)
Domain $s_2$: Unicode strings (to be substituted from)
Domain $s_3$: Unicode strings (to be substituted with)
Domain $n$: integers ≥ 0 or missing
Range: Unicode strings
subinword($s_1,s_2,s_3,n$)
Description: $s_1$, where the first $n$ occurrences in $s_1$ of $s_2$ as a word have been replaced with $s_3$
A word is defined as a space-separated token. A token at the beginning or end of $s_1$ is considered space-separated. This is different from a Unicode word, which is a language unit based on either a set of word-boundary rules or dictionaries for several languages (Chinese, Japanese, and Thai). If $n$ is missing, all occurrences are replaced.
Also see regexm(), regexr(), and regexs().

\[
\begin{align*}
\text{subinword}(&"this is the day","is","X",1) = "this X the day" \\
\text{subinword}(&"this is the hour","is","X",.) = "this X the hour" \\
\text{subinword}(&"this is this","th","X",.) = "this is this"
\end{align*}
\]

Domain $s_1$: strings (to be substituted for)
Domain $s_2$: strings (to be substituted from)
Domain $s_3$: strings (to be substituted with)
Domain $n$: integers $\geq 0$ or missing
Range: strings

substr($s,n_1,n_2$)
Description: the substring of $s$, starting at $n_1$, for a length of $n_2$
substr() is intended for use only with plain ASCII characters and for use by programmers who want to extract a subset of bytes from a string. For those with plain ASCII text, $n_1$ is the starting character, and $n_2$ is the length of the string in characters. For programmers, substr() is technically a byte-based function. For plain ASCII characters, the two are equivalent but you can operate on byte values beyond that range. Note that any Unicode character beyond ASCII range (code point greater than 127) takes more than 1 byte in the UTF-8 encoding; for example, é takes 2 bytes.
To obtain substrings of Unicode strings, see usubstr().
If $n_1 < 0$, $n_1$ is interpreted as the distance from the end of the string; if $n_2 = . (missing)$, the remaining portion of the string is returned.

\[
\begin{align*}
\text{substr}(&"abcdef",2,3) = "bcd" \\
\text{substr}(&"abcdef",-3,2) = "de" \\
\text{substr}(&"abcdef",2,.) = "bcdef" \\
\text{substr}(&"abcdef",-3,.) = "def" \\
\text{substr}(&"abcdef",2,0) = "" \\
\text{substr}(&"abcdef",15,2) = ""
\end{align*}
\]

Domain $s$: strings
Domain $n_1$: integers $\geq 1$ and $\leq -1$
Domain $n_2$: integers $\geq 1$
Range: strings
usubstr\((s, n_1, n_2)\)
Description: the Unicode substring of \(s\), starting at \(n_1\), for a length of \(n_2\)

If \(n_1 < 0\), \(n_1\) is interpreted as the distance from the last character of the \(s\); if \(n_2 = .\) (missing), the remaining portion of the Unicode string is returned.

\[
\begin{align*}
\text{usubstr("m'ediane",2,3)} & = \text{"'edi"} \\
\text{usubstr("m'ediane",-3,2)} & = \text{"an"} \\
\text{usubstr("m'ediane",2,.)} & = \text{"édi"}
\end{align*}
\]

Domain \(s\): Unicode strings
Domain \(n_1\): integers \(\geq 1\) and \(\leq -1\)
Domain \(n_2\): integers \(\geq 1\)
Range: Unicode strings

udsubstr\((s, n_1, n_2)\)
Description: the Unicode substring of \(s\), starting at character \(n_1\), for \(n_2\) display columns

If \(n_2 = .\) (missing), the remaining portion of the Unicode string is returned. If \(n_2\) display columns from \(n_1\) is in the middle of a Unicode character, the substring stops at the previous Unicode character.

\[
\begin{align*}
\text{udsubstr("m'ediane",2,3)} & = \text{"'edi"} \\
\text{udsubstr("中值","1","1")} & = \text{""} \\
\text{udsubstr("中值","1","2")} & = \text{"中"}
\end{align*}
\]

Domain \(s\): Unicode strings
Domain \(n_1\): integers \(\geq 1\)
Domain \(n_2\): integers \(\geq 1\)
Range: Unicode strings

tobytes\((s[], n)\)
Description: escaped decimal or hex digit strings of up to 200 bytes of \(s\)

The escaped decimal digit string is in the form of \(\text{"dDDD}\). The escaped hex digit string is in the form of \(\text{"xhh}\). If \(n\) is not specified or is 0, the decimal form is produced. Otherwise, the hex form is produced.

\[
\begin{align*}
\text{tobytes("abc")} & = \text{"\d097\d098\d099"} \\
\text{tobytes("abc", 1)} & = \text{"\x61\x62\x63"} \\
\text{tobytes("café")} & = \text{"\d099\d097\d102\d195\d169"}
\end{align*}
\]

Domain \(s\): Unicode strings
Domain \(n\): integers
Range: strings

uisdigit\((s)\)
Description: 1 if the first Unicode character in \(s\) is a Unicode decimal digit; otherwise, 0

A Unicode decimal digit is a Unicode character with the character property \(\text{Nd}\) according to the Unicode standard. The function returns \(-1\) if the string starts with an invalid UTF-8 sequence.

Domain \(s\): Unicode strings
Range: integers
uisletter(s)

Description: 1 if the first Unicode character in s is a Unicode letter; otherwise, 0

A Unicode letter is a Unicode character with the character property L according to the Unicode standard. The function returns -1 if the string starts with an invalid UTF-8 sequence.

Domain s: Unicode strings
Range: integers

ustrcompare(s1, s2 [, loc ])

Description: compares two Unicode strings

The function returns -1, 1, or 0 if s1 is less than, greater than, or equal to s2. The function may return a negative number other than -1 if an error happens. The comparison is locale dependent. For example, z < ö in Swedish but ö < z in German. If loc is not specified, the default locale is used. The comparison is diacritic and case sensitive. If you need different behavior, for example, case-insensitive comparison, you should use the extended comparison function ustrcompareex(). Unicode string comparison compares Unicode strings in a language-sensitive manner. On the other hand, the sort command compares strings in code-point (binary) order. For example, uppercase “Z” (code-point value 90) comes before lowercase “a” (code-point value 97) in code-point order but comes after “a” in any English dictionary.

ustrcompare("z", "ō", "sv") = -1
ustrcompare("z", "ō", "de") = 1

Domain s1: Unicode strings
Domain s2: Unicode strings
Domain loc: Unicode strings
Range: integers

ustrcompareex(s1, s2, loc, case, cslv, norm, num, alt, fr)

Description: compares two Unicode strings

The function returns -1, 1, or 0 if s1 is less than, greater than, or equal to s2. The function may return a negative number other than -1 if an error occurs. The comparison is locale dependent. For example, z < ö in Swedish but ö < z in German. If loc is not specified, the default locale is used.

st controls the strength of the comparison. Possible values are 1 (primary), 2 (secondary), 3 (tertiary), 4 (quaternary), or 5 (identical). -1 means to use the default value for the locale. Any other numbers are treated as tertiary. The primary difference represents base letter differences; for example, letter “a” and letter “b” have primary differences. The secondary difference represents diacritical differences on the same base letter; for example, letters “a” and “ä” have secondary differences. The tertiary difference represents case differences of the same base letter; for example, letters “a” and “A” have tertiary differences. Quaternary strength is useful to distinguish between Katakana and Hiragana for the JIS 4061 collation standard. Identical strength is essentially the code-point order of the string, hence, is rarely useful.

ustrcompareex("café","cafe","fr", 1, -1, -1, -1, -1, -1, -1) = 0
ustrcompareex("café","cafe","fr", 2, -1, -1, -1, -1, -1, -1, -1) = 1
ustrcompareex("Café","café","fr", 3, -1, -1, -1, -1, -1, -1, -1) = 1
case controls the uppercase and lowercase letter order. Possible values are 0 (use order specified in tertiary strength), 1 (uppercase first), or 2 (lowercase first). -1 means to use the default value for the locale. Any other values are treated as 0.

`ustrcompareex("Café","café","fr", -1, 1, -1, -1, -1, -1, -1) = -1`

`ustrcomparex("Café","café","fr", -1, 2, -1, -1, -1, -1, -1) = 1`

cslv controls whether an extra case level between the secondary level and the tertiary level is generated. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. Combining this setting to be “on” and the strength setting to be primary can achieve the effect of ignoring the diacritical differences but preserving the case differences. If the setting is “on”, the result is also affected by the case setting.

`ustrcompareex("café","Cafe","fr", 1, -1, 1, -1, -1, -1, -1) = -1`

`ustrcompareex("café","Cafe","fr", 1, 1, -1, -1, -1, -1, -1) = 1`

norm controls whether the normalization check and normalizations are performed. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. Most languages do not require normalization for comparison. Normalization is needed in languages that use multiple combining characters such as Arabic, ancient Greek, or Hebrew.

num controls how contiguous digit substrings are sorted. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. If the setting is “on”, substrings consisting of digits are sorted based on the numeric value. For example, “100” is after value “20” instead of before it. Note that the digit substring is limited to 254 digits, and plus/minus signs, decimals, or exponents are not supported.

`ustrcompareex("100","20","en", -1, -1, -1, -1, 0, -1, -1) = -1`

`ustrcompareex("100","20","en", -1, -1, -1, -1, 1, -1, -1) = 1`

alt controls how spaces and punctuation characters are handled. Possible values are 0 (use primary strength) or 1 (alternative handling). Any other values are treated as 0. If the setting is 1 (alternative handling), “onsite”, “on-site”, and “on site” are considered equals.

`ustrcompareex("onsite","on-site","en", -1, -1, -1, -1, 1, -1, -1) = 0`

`ustrcompareex("onsite","on site","en", -1, -1, -1, -1, 1, -1, -1) = 1`

fr controls the direction of the secondary strength. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. All other values are treated as “off”. If the setting is “on”, the diacritical letters are sorted backward. Note that the setting is “on” by default only for Canadian French (locale fr_CA).

`ustrcompareex("coté","côte","fr_CA",-1,-1,-1,-1,-1,0) = -1`

`ustrcompareex("coté","côte","fr_CA",-1,-1,-1,-1,-1,-1) = 1`

`ustrcompareex("coté","côte","fr",-1,-1,-1,-1,-1,-1) = 1`
Domain \( s_1 \): Unicode strings
Domain \( s_2 \): Unicode strings
Domain \( \text{loc} \): Unicode strings
Domain \( \text{st} \): integers
Domain \( \text{case} \): integers
Domain \( \text{cslv} \): integers
Domain \( \text{norm} \): integers
Domain \( \text{alt} \): integers
Domain \( \text{fr} \): integers
Range: integers

\text{ustrfix}(s[, rep])

Description: replaces each invalid UTF-8 sequence with a Unicode character

In the one-argument case, the Unicode replacement character \( \ufffd \) is used. In the two-argument case, the first Unicode character of \( rep \) is used. If \( rep \) starts with an invalid UTF-8 sequence, then Unicode replacement character \( \ufffd \) is used. Note that an invalid UTF-8 sequence can contain one byte or multiple bytes.

\[
\text{ustrfix}(\text{char}(200)) = \text{ustrunescape}(\text{char}(200)) \\
\text{ustrfix}(\text{char}(229)+\text{char}(174)+\text{char}(174)+\text{char}(174)+\text{char}(174)+\text{char}(174), \text{char}(174)) = \text{ustrunescape}(\text{char}(229)+\text{char}(174)+\text{char}(174)+\text{char}(174)+\text{char}(174)+\text{char}(174) + \text{char}(174))
\]

Domain \( s \): Unicode strings
Domain \( \text{rep} \): Unicode character
Range: Unicode strings

\text{ustrfrom}(s, enc, mode)

Description: converts the string \( s \) in encoding \( enc \) to a UTF-8 encoded Unicode string

\( mode \) controls how invalid byte sequences in \( s \) are handled. The possible values are 1, which substitutes an invalid byte sequence with a Unicode replacement character \( \ufffd \); 2, which skips any invalid byte sequences; 3, which stops at the first invalid byte sequence and returns an empty string; or 4, which replaces any byte in an invalid sequence with an escaped hex digit sequence \( %Xhh \). Any other values are treated as 1. A good use of value 4 is to check what invalid bytes a Unicode string \( ust \) contains by examining the result of \text{ustrfrom}(ust, "utf-8", 4).

Also see \text{ustrto}().

\[
\text{ustrfrom}(\text{char}(233), \text{latin1}, 1) = \text{char}(233) \\
\text{ustrfrom}(\text{char}(233), \text{utf-8}, 1) = \text{char}(233) + \text{ustrunescape}(\text{char}(233)) \\
\text{ustrfrom}(\text{char}(233), \text{utf-8}, 2) = \text{char}(233) + \text{ustrunescape}(\text{char}(233)) \\
\text{ustrfrom}(\text{char}(233), \text{utf-8}, 3) = \text{char}(233) + \text{ustrunescape}(\text{char}(233)) \\
\text{ustrfrom}(\text{char}(233), \text{utf-8}, 4) = \text{char}(233) + \text{ustrunescape}(\text{char}(233))
\]

Domain \( s \): strings in encoding \( enc \)
Domain \( enc \): Unicode strings
Domain \( mode \): integers
Range: Unicode strings
ustrinvalidcnt(s)
Description: the number of invalid UTF-8 sequences in s

An invalid UTF-8 sequence may contain one byte or multiple bytes.

ustrinvalidcnt("médiane") = 0
ustrinvalidcnt("médiane"+char(229)) = 1
ustrinvalidcnt("médiane"+char(229)+char(174)) = 1
ustrinvalidcnt("médiane"+char(174)+char(158)) = 2

Domain s: Unicode strings
Range: integers

ustrleft(s,n)
Description: the first n Unicode characters of the Unicode string s

An invalid UTF-8 sequence is replaced with a Unicode replacement character \ufffd.

ustrleft("円形実験室",3) = "円形"
ustrleft("円形実験室",5) = "円形実"

Domain s: Unicode strings
Domain n: integers
Range: Unicode strings

ustrnormalize(s,norm)
Description: normalizes Unicode string s to one of the five normalization forms specified by norm

The normalization forms are nfc, nfd, nfkc, nfkd, or nkcc. The function returns an empty string for any other value of norm. Unicode normalization removes the Unicode string differences caused by Unicode character equivalence. nfc specifies Normalization Form C, which normalizes decomposed Unicode code points to a composited form. nfd specifies Normalization Form D, which normalizes composited Unicode code points to a decomposed form. nfkc and nfkd produce canonical equivalent form. nfkc and nfkd are similar to nfc and nfd but produce compatibility equivalent forms. nkcc specifies nkfc with casefolding. This normalization and casefolding implement the Unicode Character Database.

In the Unicode standard, both “ī” (\u0069 followed by a diaeresis \u0308) and the composite character \u00ef represent “i” with 2 dots as in “naïve”. Hence, the code-point sequence \u0069\u0308 and the code point \u00ef are considered Unicode equivalent. According to the Unicode standard, they should be treated as the same single character in Unicode string operations, such as in display, comparison, and selection. However, Stata does not support multiple code-point characters; each code point is considered a separate Unicode character. Hence, \u0069\u0308 is displayed as two characters in the Results window. ustrnormalize() can be used with "nfc" to normalize \u0069\u0308 to the canonical equivalent composited code point \u00ef.

ustrnormalize(ustrunescape("\u0069\u0308"), "nfc") = "ī"
The decomposed form nfd can be used to removed diacritical marks from base letters. First, normalize the Unicode string to canonical decomposed form, and then call `ustrto()` with mode `skip` to skip all non-ASCII characters.

Also see `ustrfrom()`.

```lisp
ustrto(ustrnormalize("café", "nfd"), "ascii", 2) = "cafe"
```

**Domain s:** Unicode strings  
**Domain norm:** Unicode strings  
**Range:** Unicode strings

### `ustrright(s,n)`

**Description:** the last $n$ Unicode characters of the Unicode string $s$

An invalid UTF-8 sequence is replaced with a Unicode replacement character \ufffd.

```lisp
ustrright("Экспериментальные",3) = "ные"
ustrright("Экспериментальные",5) = "льные"
```

**Domain s:** Unicode strings  
**Domain n:** integers  
**Range:** Unicode strings

### `ustrsortkey(s[loc])`

**Description:** generates a null-terminated byte array that can be used by the `sort` command to produce the same order as `ustrcompare()`

The function may return an empty array if an error occurs. The result is locale dependent. If `loc` is not specified, the default locale is used. The result is also diacritic and case sensitive. If you need different behavior, for example, case-insensitive results, you should use the extended function `ustrsortkeyex()`.

See [U] 12.4.2.5 Sorting strings containing Unicode characters for details and examples.

**Domain s:** Unicode strings  
**Domain loc:** Unicode strings  
**Range:** null-terminated byte array
ustrsortkeyex(s, loc, case, cslv, norm, num, alt, fr)

Description: generates a null-terminated byte array that can be used by the `sort` command to produce the same order as `ustrcompare()`.

The function may return an empty array if an error occurs. The result is locale dependent. If `loc` is not specified, the default locale is used. See [U] 12.4.2.5 Sorting strings containing Unicode characters for details and examples.

**st** controls the strength of the comparison. Possible values are 1 (primary), 2 (secondary), 3 (tertiary), 4 (quaternary), or 5 (identical). -1 means to use the default value for the locale. Any other numbers are treated as tertiary. The primary difference represents base letter differences; for example, letter “a” and letter “b” have primary differences. The secondary difference represents diacritical differences on the same base letter; for example, letters “a” and “¨a” have secondary differences. The tertiary difference represents case differences of the same base letters; for example, letters “a” and “A” have tertiary differences. Quaternary strength is useful to distinguish between Katakana and Hiragana for the JIS 4061 collation standard. Identical strength is essentially the code-point order of the string and, hence, is rarely useful.

**case** controls the uppercase and lowercase letter order. Possible values are 0 (use order specified in tertiary strength), 1 (uppercase first), or 2 (lowercase first). -1 means to use the default value for the locale. Any other values are treated as 0.

**cslv** controls if an extra case level between the secondary level and the tertiary level is generated. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. Combining this setting to be “on” and the strength setting to be primary can achieve the effect of ignoring the diacritical differences but preserving the case differences. If the setting is “on”, the result is also affected by the **case** setting.

**norm** controls whether the normalization check and normalizations are performed. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. Most languages do not require normalization for comparison. Normalization is needed in languages that use multiple combining characters such as Arabic, ancient Greek, or Hebrew.

**num** controls how contiguous digit substrings are sorted. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. Any other values are treated as 0. If the setting is “on”, substrings consisting of digits are sorted based on the numeric value. For example, “100” is after “20” instead of before it. Note that the digit substring is limited to 254 digits, and plus/minus signs, decimals, or exponents are not supported.
alt controls how spaces and punctuation characters are handled. Possible values are 0 (use primary strength) or 1 (alternative handling). Any other values are treated as 0. If the setting is 1 (alternative handling), “onsite”, “on-site”, and “on site” are considered equals.

fr controls the direction of the secondary strength. Possible values are 0 (off) or 1 (on). -1 means to use the default value for the locale. All other values are treated as “off”. If the setting is “on”, the diacritical letters are sorted backward. Note that the setting is “on” by default only for Canadian French (locale fr_CA).

ustrto(s, enc, mode)
Description: converts the Unicode string s in UTF-8 encoding to a string in encoding enc

See [D] unicode encoding for details on available encodings. Any invalid sequence in s is replaced with a Unicode replacement character \ufffd. mode controls how unsupported Unicode characters in the encoding enc are handled. The possible values are 1, which substitutes any unsupported characters with the enc’s substitution strings (the substitution character for both ascii and latin1 is char(26)); 2, which skips any unsupported characters; 3, which stops at the first unsupported character and returns an empty string; or 4, which replaces any unsupported character with an escaped hex digit sequence \uhhhh or \Uhhhhhhhh. The hex digit sequence contains either 4 or 8 hex digits, depending if the Unicode character’s code-point value is less than or greater than \uffff. Any other values are treated as 1.

ustrto("café", "ascii", 1) = "caf"+char(26)
ustrto("café", "ascii", 2) = "caf"
ustrto("café", "ascii", 3) = ""
ustrto("café", "ascii", 4) = "caf\u00E9"

ustrto() can be used to removed diacritical marks from base letters. First, normalize the Unicode string to NFD form using ustrnormalize(), and then call ustrto() with value 2 to skip all non-ASCII characters.

Also see ustrfrom().

ustrto(ustrnormalize("café", "nfd"), "ascii", 2) = "cafe"
ustrtohex(s[n])
Description: escaped hex digit string of s up to 200 Unicode characters

The escaped hex digit string is in the form of \uhhhh for code points less than \uffff or \Uhhhhhhhh for code points greater than \uffff. The function starts at the nth Unicode character of s if n is specified and larger than 0. Any invalid UTF-8 sequence is replaced with a Unicode replacement character \ufffd. Note that the null terminator char(0) is a valid Unicode character. Function ustrunescape() can be applied on the result to get back the original Unicode string s if s does not contain any invalid UTF-8 sequences.

Also see ustrunescape().

ustrtohex("нулю") = "\u043d\u0443\u043b\u044e"
ustrtohex("нулю", 2) = "\u0443\u043b\u044e"
ustrtohex("i"+char(200)+char(0)+"s") = "\u0069\ufffd\u0000\u0073"

Domain s: Unicode strings
Domain n: integers ≥ 1
Range: strings

ustrunescape(s)
Description: the Unicode string corresponding to the escaped sequences of s

The following escape sequences are recognized: 4 hex digit form \uhhhh; 8 hex digit form \Uhhhhhhhh; 1–2 hex digit form \xhh; and 1–3 octal digit form \ooo, where h is [0–9A–Fa–f] and o is [0–7]. The standard ANSI C escapes \a, \b, \t, \n, \v, \f, \r, \e, \", \', \?, \\ are recognized as well. The function returns an empty string if an escape sequence is badly formed. Note that the 8 hex digit form \Uhhhhhhhh begins with a capital letter “U”.

Also see ustrtohex().

ustrunescape("\u043d\u0443\u043b\u044e") = "нулю"

Domain s: strings of escaped hex values
Range: Unicode strings

word(s,n)
Description: the nth word in s; missing ("") if n is missing

Positive numbers count words from the beginning of s, and negative numbers count words from the end of s. (1 is the first word in s, and -1 is the last word in s.) A word is a set of characters that start and terminate with spaces. This is different from a Unicode word, which is a language unit based on either a set of word-boundary rules or dictionaries for several languages (Chinese, Japanese, and Thai).

Domain s: strings
Domain n: integers
Range: strings
ustrword(s,n[,loc])

Description: the $n$th Unicode word in the Unicode string $s$

Positive $n$ counts Unicode words from the beginning of $s$, and negative $n$ counts Unicode words from the end of $s$. For examples, $n$ equal to 1 returns the first word in $s$, and $n$ equal to $-1$ returns the last word in $s$. If $loc$ is not specified, the default locale is used. A Unicode word is different from a Stata word produced by the word() function. A Stata word is a space-separated token. A Unicode word is a language unit based on either a set of word-boundary rules or dictionaries for some languages (Chinese, Japanese, and Thai). The function returns missing ("") if $n$ is greater than $cnt$ or less than $-cnt$, where $cnt$ is the number of words $s$ contains. $cnt$ can be obtained from ustrwordcount(). The function also returns missing ("") if an error occurs.

ustrword("Parlez-vous français", 1, "fr") = "Parlez"
ustrword("Parlez-vous français", 2, "fr") = "-"
ustrword("Parlez-vous français",-1, "fr") = "français"
ustrword("Parlez-vous français",-2, "fr") = "vous"

Domain $s$: Unicode strings
Domain $loc$: Unicode strings
Domain $n$: integers
Range: Unicode strings

wordbreaklocale(loc,type)

Description: 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 $type$

wordbreaklocale("en_us_texas", 1) = en_US
wordbreaklocale("en_us_texas", 2) = root

Domain $loc$: strings of locale name
Domain $type$: integers
Range: strings

wordcount(s)

Description: the number of words in $s$

A word is a set of characters that starts and terminates with spaces, starts with the beginning of the string, or terminates with the end of the string. This is different from a Unicode word, which is a language unit based on either a set of word-boundary rules or dictionaries for several languages (Chinese, Japanese, and Thai).

Domain $s$: strings
Range: nonnegative integers 0, 1, 2, ...
ustrwordcount(s[,loc])

Description: the number of nonempty Unicode words in the Unicode string s

An empty Unicode word is a Unicode word consisting of only Unicode whitespace characters. If loc is not specified, the default locale is used. A Unicode word is different from a Stata word produced by the word() function. A Stata word is a space-separated token. A Unicode word is a language unit based on either a set of word-boundary rules or dictionaries for some languages (Chinese, Japanese, and Thai). The function may return a negative number if an error occurs.

ustrwordcount("Parlez-vous français", "fr") = 4

Domain s: Unicode strings
Domain loc: Unicode strings
Range: integers

References


Also see

[FN] Functions by category
[D] egen — Extensions to generate
[D] generate — Create or change contents of variable
[M-4] String — String manipulation functions
[U] 12.4.2 Handling Unicode strings
[U] 13.2.2 String operators
[U] 13.3 Functions
Trigonometric functions

Contents

- \( \text{acos}(x) \)
  - the radian value of the arccosine of \( x \)
  - Domain: \(-1\) to 1
  - Range: 0 to \( \pi \)

- \( \text{acosh}(x) \)
  - the inverse hyperbolic cosine of \( x \)
  - Domain: 1 to 8.9e+307
  - Range: 0 to 709.77

- \( \text{asin}(x) \)
  - the radian value of the arcsine of \( x \)
  - Domain: \(-1\) to 1
  - Range: \(-\pi/2\) to \( \pi/2 \)

- \( \text{asinh}(x) \)
  - the inverse hyperbolic sine of \( x \)
  - Domain: \(-8.9e+307\) to 8.9e+307
  - Range: \(-709.77\) to 709.77

- \( \text{atan}(x) \)
  - the radian value of the arctangent of \( x \)

- \( \text{atan2}(y, x) \)
  - 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

- \( \text{atanh}(x) \)
  - the inverse hyperbolic tangent of \( x \)

- \( \text{cos}(x) \)
  - the cosine of \( x \), where \( x \) is in radians

- \( \text{cosh}(x) \)
  - the hyperbolic cosine of \( x \)

- \( \text{sin}(x) \)
  - the sine of \( x \), where \( x \) is in radians

- \( \text{sinh}(x) \)
  - the hyperbolic sine of \( x \)

- \( \text{tan}(x) \)
  - the tangent of \( x \), where \( x \) is in radians

- \( \text{tanh}(x) \)
  - the hyperbolic tangent of \( x \)
Trigonometric functions

atan\( (x) \)
Description: the radian value of the arctangent of \( x \)
Domain: \(-8e+307\) to \(8e+307\)
Range: \(-\pi/2\) to \(\pi/2\)

atan2\( (y, x) \)
Description: 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
Domain \( y \): \(-8e+307\) to \(8e+307\)
Domain \( x \): \(-8e+307\) to \(8e+307\)
Range: \(-\pi\) to \(\pi\)

atanh\( (x) \)
Description: the inverse hyperbolic tangent of \( x \)
\[ \text{atanh}(x) = \frac{1}{2}\{\ln(1 + x) - \ln(1 - x)\} \]
Domain: \(-1\) to \(1\)
Range: \(-8e+307\) to \(8e+307\)

cos\( (x) \)
Description: the cosine of \( x \), where \( x \) is in radians
Domain: \(-1e+18\) to \(1e+18\)
Range: \(-1\) to \(1\)

cosh\( (x) \)
Description: the hyperbolic cosine of \( x \)
\[ \cosh(x) = \{\exp(x) + \exp(-x)\}/2 \]
Domain: \(-709\) to \(709\)
Range: \(1\) to \(4.11e+307\)

sin\( (x) \)
Description: the sine of \( x \), where \( x \) is in radians
Domain: \(-1e+18\) to \(1e+18\)
Range: \(-1\) to \(1\)

sinh\( (x) \)
Description: the hyperbolic sine of \( x \)
\[ \sinh(x) = \{\exp(x) - \exp(-x)\}/2 \]
Domain: \(-709\) to \(709\)
Range: \(-4.11e+307\) to \(4.11e+307\)

tan\( (x) \)
Description: the tangent of \( x \), where \( x \) is in radians
Domain: \(-1e+18\) to \(1e+18\)
Range: \(-1e+17\) to \(1e+17\) or missing

tanh\( (x) \)
Description: the hyperbolic tangent of \( x \)
\[ \text{tanh}(x) = \{\exp(x) - \exp(-x)\}/\{\exp(x) + \exp(-x)\} \]
Domain: \(-8e+307\) to \(8e+307\)
Range: \(-1\) to \(1\) or missing
Technical note

The trigonometric functions are defined in terms of radians. There are $2\pi$ radians in a circle. If you prefer to think in terms of degrees, because there are also 360 degrees in a circle, you may convert degrees into radians by using the formula $r = d\pi/180$, where $d$ represents degrees and $r$ represents radians. Stata includes the built-in constant `_pi`, equal to $\pi$ to machine precision. Thus, to calculate the sine of $\theta$, where $\theta$ is measured in degrees, you could type

$$\text{sin}(\theta*\_\text{pi}/180)$$

`atan()` similarly returns radians, not degrees. The arccotangent can be obtained as

$$\text{acot}(x) = \frac{\pi}{2} - \text{atan}(x)$$

References


Also see

[FN] Functions by category
[D] `egen` — Extensions to generate
[D] `generate` — Create or change contents of variable
[M-5] `sin()` — Trigonometric and hyperbolic functions
[U] 13.3 Functions
Subject and author index

See the combined subject index and the combined author index in the *Stata Index*. 