Title

Date and time functions

Contents

<table>
<thead>
<tr>
<th>Functions</th>
<th>Remarks and examples</th>
<th>Methods and formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>age($e_d DOB, e_d[, s_{nl}]$)</td>
<td>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</td>
<td></td>
</tr>
<tr>
<td>age_frac($e_d DOB, e_d[, s_{nl}]$)</td>
<td>the age in years, including the fractional part, on $e_d$ for date of birth $e_d DOB$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates</td>
<td></td>
</tr>
<tr>
<td>birthday($e_d DOB, Y[, s_{nl}]$)</td>
<td>the $e_d$ date of the birthday in year $Y$ for date of birth $e_d DOB$ with $s_{nl}$ the nonleap-year birthday for 29feb birthdates</td>
<td></td>
</tr>
<tr>
<td>bofd(&quot;cal&quot;, $e_d$)</td>
<td>the $e_b$ business date corresponding to $e_d$</td>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>Clockdifffrac($e_{tc1}, e_{tc2}, s_u$)</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>
<td></td>
</tr>
<tr>
<td>clockdifffrac($e_{tc1}, e_{tc2}, s_u$)</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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>cofC($e_{tc}$)</td>
<td>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)</td>
<td></td>
</tr>
</tbody>
</table>
Cofd($e_d$)  the $etC$ 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 $etC$ 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(dow,$e_d, d$)  a synonym for dayssinceweekday($e_d, d$)

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

daysuntil(dow,$e_d, d$)  a synonym for daysuntilweekday($e_d, d$)

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

dhms($e_d, h, m, s$)  the $etC$ 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 time 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 toyear; 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

lastdayofmonth($e_d$) the $e_d$ date of the last day of the month of $e_d$

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

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

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

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

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

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

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

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

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

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

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

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

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

seconds($ms$) $ms/1000$

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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e_b)</td>
<td>business calendar date (days)</td>
</tr>
<tr>
<td>(e_{tc})</td>
<td>encoded datetime (ms. since 01jan1960 00:00:00:000)</td>
</tr>
<tr>
<td>(e_{tc})</td>
<td>encoded datetime (ms. with leap seconds since 01jan1960 00:00:00:000)</td>
</tr>
<tr>
<td>(e_d)</td>
<td>encoded date (days since 01jan1960)</td>
</tr>
<tr>
<td>(e_w)</td>
<td>encoded weekly date (weeks since 1960w1)</td>
</tr>
<tr>
<td>(e_m)</td>
<td>encoded monthly date (months since 1960m1)</td>
</tr>
<tr>
<td>(e_q)</td>
<td>encoded quarterly date (quarters since 1960q1)</td>
</tr>
<tr>
<td>(e_h)</td>
<td>encoded half-yearly date (half-years since 1960h1)</td>
</tr>
<tr>
<td>(e_y)</td>
<td>encoded yearly date (years)</td>
</tr>
<tr>
<td>(M)</td>
<td>month, 1–12</td>
</tr>
<tr>
<td>(D)</td>
<td>day of month, 1–31</td>
</tr>
<tr>
<td>(Y)</td>
<td>year, 0100–9999</td>
</tr>
<tr>
<td>(h)</td>
<td>hour, 0–23</td>
</tr>
<tr>
<td>(m)</td>
<td>minute, 0–59</td>
</tr>
<tr>
<td>(s)</td>
<td>second, 0–59 or 60 if leap seconds</td>
</tr>
<tr>
<td>(ms)</td>
<td>milliseconds</td>
</tr>
<tr>
<td>(W)</td>
<td>week number, 1–52</td>
</tr>
<tr>
<td>(Q)</td>
<td>quarter number, 1–4</td>
</tr>
<tr>
<td>(H)</td>
<td>half-year number, 1 or 2</td>
</tr>
<tr>
<td>(d)</td>
<td>numeric day of the week, 0 = Sunday, 1 = Monday, . . ., 6 = Saturday</td>
</tr>
</tbody>
</table>

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.

\[
\text{age}(e_{d\text{DOB}}, e_d\left[; s_{nl}\right])
\]

**Description:** the age in integer years on \(e_d\) 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**.

When \(e_d < e_{d\text{DOB}}\), the result is missing.

**Domain**

- \(e_{d\text{DOB}}\): \(e_d\) dates 01jan0101 to 31dec9998 (integers -678,985 to 2,936,184)
- \(e_d\): \(e_d\) dates 01jan0101 to 31dec9998 (integers -678,985 to 2,936,184)
- \(s_{nl}\): strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)

**Range:** integers 0 to 9897 or missing
age(frac(edDOB, ed[ , snl ]))
Description: the age in years, including the fractional part, on ed for date of birth edDOB with snl the nonleap-year birthday for 29feb birthdates

When ed < edDOB, the result is missing.

Domain edDOB: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Domain ed: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Domain snl: strings "28feb", "feb28", "01mar", "1mar", "mar01", and "mar1" (case insensitive)
Range: reals 0 to 9897.997... or missing

birthday(edDOB, Y[ , snl ])
Description: the ed date of the birthday in year Y for date of birth edDOB with snl 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 edDOB: ed 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: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549) or missing

bofd("cal", ed)
Description: the eb business date corresponding to ed
Domain cal: business calendar names and formats
Domain ed: ed as defined by business calendar named cal
Range: as defined by business calendar named cal

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

Domain ed: ed 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: etC datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999,999+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 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 $e_{tC2}$: $e_{tC}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + 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 $-312,413,759,999,999 - \text{number of leap seconds}$ to $312,413,759,999,999 + \text{number of leap seconds}$ or $\text{missing}$

Clockdiff$_\text{frac}(e_{tc1}, e_{tc2}, s_u)$

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 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: integers $-312,413,759,999,999 - \text{number of leap seconds}$ to $312,413,759,999,999 + \text{number of leap seconds}$ or $\text{missing}$

Clockdiff$_\text{frac}(e_{tC1}, e_{tC2}, s_u)$

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 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 $e_{tC2}$: $e_{tC}$ datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers $-58,695,840,000,000$ to $253,717,919,999,999$ + 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 $-312,413,759,999,999 - \text{number of leap seconds}$ to $312,413,759,999,999 + \text{number of leap seconds}$ or $\text{missing}$
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 01jan00 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 01jan00 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 01jan00 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
Cmdyhm\(s\) \((M,D,Y,h,m,s)\)

Description: the \(e_t\)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: \(e_t\)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 \( e_{tc} \) (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 \(253,717,919,999,999\))

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\( (s_{1}, s_{2}, Y) \)
**Date and time functions**

**date($s_1$, $s_2$, $Y$)**

Description: the $e_d$ 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, $\text{topyear}$, that does not exceed $Y$ is returned.

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

If neither ## nor $Y$ is specified, date() returns 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 missing

**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 $\text{datediff}(e_{d1},e_{d2},s_u,s_{nl}) = -\text{datediff}(e_{d2},e_{d1},s_u,s_{nl})$.

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 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 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

dayssincedow($e_d, d$)

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($e_d, d$)
\textbf{daysuntilweekday}(\textit{e}_d, \textit{d})

\textbf{Description}: the number of days from \textit{e}_d until next day-of-week \textit{d}

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

\textbf{Domain} \textit{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)

\textbf{Range}: integers 1 to 7 or \textit{missing}

\textbf{dhms}(\textit{e}_d, \textit{h}, \textit{m}, \textit{s})

\textbf{Description}: the \textit{etc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to \textit{e}_d, \textit{h}, \textit{m}, and \textit{s}

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

\textbf{Domain} \textit{h}: integers 0 to 23

\textbf{Domain} \textit{m}: integers 0 to 59

\textbf{Domain} \textit{s}: reals 0.000 to 59.999

\textbf{Range}: \textit{etc} 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 \textit{missing}

\textbf{dmy}(\textit{D}, \textit{M}, \textit{Y})

\textbf{Description}: the \textit{e}_d date (days since 01jan1960) corresponding to \textit{D}, \textit{M}, \textit{Y}

\textbf{Domain} \textit{D}: integers 1 to 31

\textbf{Domain} \textit{M}: integers 1 to 12

\textbf{Domain} \textit{Y}: integers 0100 to 9999 (but probably 1800 to 2100)

\textbf{Range}: \textit{e}_d dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\)) or \textit{missing}

\textbf{dofb}(\textit{e}_b, "\textit{cal}"")

\textbf{Description}: the \textit{e}_d datetime corresponding to \textit{e}_b

\textbf{Domain} \textit{e}_b: \textit{e}_b as defined by business calendar named \textit{cal}

\textbf{Domain} \textit{cal}: business calendar names and formats

\textbf{Range}: as defined by business calendar named \textit{cal}

\textbf{dofC}(\textit{etC})

\textbf{Description}: the \textit{e}_d date (days since 01jan1960) of datetime \textit{etC} (ms. with leap seconds since 01jan1960 00:00:00.000)

\textbf{Domain} \textit{etC}: \textit{etC} 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)

\textbf{Range}: \textit{e}_d dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))

\textbf{dofc}(\textit{etc})

\textbf{Description}: the \textit{e}_d date (days since 01jan1960) of datetime \textit{etc} (ms. since 01jan1960 00:00:00.000)

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

\textbf{Range}: \textit{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($M,Y,d$)
**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
- \(Y\): integers 0100 to 9999 (but probably 1800 to 2100)
- \(d\): 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 topyear; see date()

**Domain**
- \(s_1\): strings
- \(s_2\): strings "HY" and "YH"; \(Y\) may be prefixed with ##
- \(Y\): 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
- \(m\): integers 0 to 59
- \(s\): 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)
hofd(\textit{e}_d)
Description: the \textit{e}_h half-yearly date (half years since 1960\textit{h}1) containing date \textit{e}_d
Domain \textit{e}_d: \textit{e}_d dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))
Range: \textit{e}_h dates 0100\textit{h}1 to 9999\textit{h}2 (integers \(-3,720\) to \(16,079\))

\texttt{hours}(\textit{ms})
Description: \textit{ms}/3,600,000
Domain \textit{ms}: real; milliseconds
Range: real or missing

\texttt{isleapsecond}(\textit{e}_tC)
Description: 1 if \textit{e}_tC is a leap second; otherwise, 0
Domain \textit{e}_tC: \textit{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

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

\texttt{lastdayofmonth}(\textit{e}_d)
Description: the \textit{e}_d date of the last day of the month of \textit{e}_d
Domain \textit{e}_d: \textit{e}_d dates 01jan0100 to 31dec9999 (integers \(-679,350\) to \(2,936,549\))
Range: \textit{e}_d dates 31jan0100 to 31dec9999 (integers \(-679,320\) to \(2,936,549\)) or missing

\texttt{lastdowofmonth}(\textit{M}, \textit{Y}, \textit{d})
Description: a synonym for \texttt{lastweekdayofmonth}(\textit{M}, \textit{Y}, \textit{d})

\texttt{lastweekdayofmonth}(\textit{M}, \textit{Y}, \textit{d})
Description: the \textit{e}_d date of the last day-of-week \textit{d} in month \textit{M} of year \textit{Y}
Domain \textit{M}: integers 1 to 12
Domain \textit{Y}: integers 0100 to 9999 (but probably 1800 to 2100)
Domain \textit{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: \textit{e}_d dates 25jan0100 to 31dec9999 (integers \(-679,326\) to \(2,936,549\)) or missing

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

Description: the UTC datetime (ms. 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 59.999

Range: UTC 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 \textit{missing}

\textbf{minutes}(ms)

Description: \( ms/60,000 \)

Domain \( ms \): real; milliseconds

Range: real or \textit{missing}

\textbf{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 \textit{missing}

\textbf{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 \textit{missing}

\textbf{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 \(96,479\))

\textbf{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 \textit{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 topyear; 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
\textbf{nextweekday}(\texttt{e}, \texttt{d})

Description: the \texttt{e} date of the first day-of-week \texttt{d} after \texttt{e}

Domain \texttt{e}: \texttt{e} dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)

Domain \texttt{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: \texttt{e} dates 02jan0100 to 31dec9999 (integers $-679,349$ to $2,936,549$) or missing

\textbf{now()}

Description: the current \texttt{etc} datetime

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

\textbf{previousbirthday}(\texttt{edDOB}, \texttt{e}[s_{nl}])

Description: the \texttt{e} date of the birthday immediately before \texttt{e} for date of birth \texttt{edDOB} with \texttt{s_{nl}} the nonleap-year birthday for 29feb birthdates

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

Domain \texttt{edDOB}: \texttt{e} dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)

Domain \texttt{e}: \texttt{e} dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)

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

Range: \texttt{e} dates 01jan0100 to 31dec9998 (integers $-679,350$ to $2,936,184$) or missing

\textbf{previousdow}(\texttt{ed}, \texttt{d})

Description: a synonym for \textbf{previousweekday}(\texttt{ed}, \texttt{d})

\textbf{previousleapyear}(\texttt{Y})

Description: the leap year immediately before year \texttt{Y}

Domain \texttt{Y}: integers 0100 to 9999 (but probably 1800 to 2100)

Range: integers 1584 to 9996 or missing

\textbf{previousweekday}(\texttt{ed}, \texttt{d})

Description: the \texttt{ed} date of the last day-of-week \texttt{d} before \texttt{ed}

Domain \texttt{ed}: \texttt{ed} dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)

Domain \texttt{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: \texttt{ed} dates 01jan0100 to 30dec9999 (integers $-679,350$ to $2,936,548$) or missing

\textbf{qofd}(\texttt{ed})

Description: the \texttt{eq} quarterly date (quarters since 1960q1) containing date \texttt{ed}

Domain \texttt{ed}: \texttt{ed} dates 01jan0100 to 31dec9999 (integers $-679,350$ to $2,936,549$)

Range: \texttt{eq} 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 toyear; 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)
22 Date and time functions

\( \text{tc}(l) \)

Description: convenience function to make typing dates and times in expressions easier

For example, typing \( \text{tc}(2\text{jan}1960 \ 13:42) \) is equivalent to typing \( 135720000 \); 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; \( \text{tc}(11:02) \) is equivalent to typing \( 39720000 \).

Domain \( l \): datetime literal strings 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
Range: \( e_t \) datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers \(-58,695,840,000,000\) to \(253,717,919,999,999\))

\( \text{td}(l) \)

Description: convenience function to make typing dates in expressions easier

For example, typing \( \text{td}(2\text{jan}1960) \) is equivalent to typing \( 1 \).

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

\( \text{th}(l) \)

Description: convenience function to make typing half-yearly dates in expressions easier

For example, typing \( \text{th}(1960\text{h2}) \) is equivalent to typing \( 1 \).

Domain \( l \): half-year literal strings 0100h1 to 9999h2
Range: \( e_h \) dates 0100h1 to 9999h2 (integers \(-3,720\) to \(16,079\))

\( \text{tm}(l) \)

Description: convenience function to make typing monthly dates in expressions easier

For example, typing \( \text{tm}(1960\text{m2}) \) is equivalent to typing \( 1 \).

Domain \( l \): month literal strings 0100m1 to 9999m12
Range: \( e_m \) dates 0100m1 to 9999m12 (integers \(-22,320\) to \(96,479\))

\( \text{today]() } \)

Description: today’s \( e_d \) date

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

\( \text{tq}(l) \)

Description: convenience function to make typing quarterly dates in expressions easier

For example, typing \( \text{tq}(1960\text{q2}) \) is equivalent to typing \( 1 \).

Domain \( l \): quarter literal strings 0100q1 to 9999q4
Range: \( e_q \) dates 0100q1 to 9999q4 (integers \(-7,440\) to \(32,159\))
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: ew dates 0100w1 to 9999w52 (integers −96,720 to 418,079)

week(ed)
Description: the numeric week of the year corresponding to date ed, the %td encoded date (days since 01jan1960)
Note: The first week of a year is the first 7-day period of the year.
Domain ed: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 52 or missing

weekly(s1,s2[,Y])
Description: the ew weekly date (weeks since 1960w1) corresponding to s1 based on s2 and Y; Y specifies toyear; see date()
Domain s1: strings
Domain s2: strings "YW" and "YW"; Y may be prefixed with ##
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: ew dates 0100w1 to 9999w52 (integers −96,720 to 418,079) or missing

wofd(ed)
Description: the ew weekly date (weeks since 1960w1) containing date ed
Domain ed: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: ew dates 0100w1 to 9999w52 (integers −96,720 to 418,079)

year(ed)
Description: the numeric year corresponding to date ed
Domain ed: ed dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 0100 to 9999 (but probably 1800 to 2100)

yearly(s1,s2[,Y])
Description: the ey yearly date (year) corresponding to s1 based on s2 and Y; Y specifies toyear; see date()
Domain s1: strings
Domain s2: string "Y"; Y may be prefixed with ##
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: ey dates 0100 to 9999 (integers 0100 to 9999) or missing
**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 `age()` and `age_frac()` are based on `datediff()` and `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
\[
geq \frac{e_d}{e_d \text{DOB}} = \text{datediff}_\text{frac} \left( e_d \text{DOB}, e_d, "year", s_{nl} \right)
\]
when \( e_d \geq e_d \text{DOB} \). When \( e_d < e_d \text{DOB} \), \text{age()}\) and \text{age}_\text{frac()}\) return \textit{missing} (.)

\text{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 \text{datediff}_\text{frac()}.

\text{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, \text{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, \text{datediff}(..., "year", ...)\) increases by one in leap years when \( e_{d2} \) goes to 29feb.

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

The fractional part of \text{datediff}_\text{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 \text{datediff}_\text{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 \text{datediff}_\text{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.

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

\text{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 \text{datediff}(..., "month", ...)\) increases by one when \( e_{d2} \) goes to 30sep. If \( e_{d1} \) is on 31aug, then \text{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 \text{datediff}(..., "month", ...)\) increases by one when \( e_{d1} \) is on 29feb.

\text{datediff}_\text{frac}(..., "month", ...)\) is defined like \text{datediff}_\text{frac}(..., "year", ...). Days on which \text{datediff}_\text{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_{t1}, e_{t2}, "day", s_{nl}) and datediff_frac(e_{t1}, e_{t2}, "day", s_{nl}) have no such complications. Both are equal to e_{t2} - e_{t1} 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 * 60 * 60 * 1000 milliseconds), hours (60 * 60 * 1000 milliseconds), minutes (60 * 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 * 60 + 1 = 3,601 seconds, and a day of 24 * 60 * 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 Clockdiff_frac(e_{tC1}, e_{tC2}, "minute") = 59/61 = 0.9672 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 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

Clockdiff_frac(e_{tC1}, e_{tC2}, "milliseconds") = e_{tC2} - e_{tC1}

and

Clockdiff_frac(e_{tC1}, e_{tC2}, "seconds") = (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**

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.