

date() — Date and time manipulation[Description](#)[Syntax](#)[Conformability](#)[Diagnostics](#)[Also see](#)

Description

These functions mirror Stata's date functions; see [D] [Datetime](#).

Syntax

```
tc = clock(strdatetime, pattern [, year])
```

```
tc = mdyhms(month, day, year, hour, minute, second)
```

```
tc = dhms(td, hour, minute, second)
```

```
tc = hms(hour, minute, second)
```

```
hour = hh(tc)
```

```
minute = mm(tc)
```

```
second = ss(tc)
```

```
td = dof(tc)
```

```
tC = Cofc(tc)
```

```
tC = Clock(strdatetime, pattern [, year])
```

```
tC = Cmdyhms(month, day, year, hour, minute, second)
```

```
tC = Cdhms(td, hour, minute, second)
```

```
tC = Chms(hour, minute, second)
```

```
hour = hhC(tC)
```

```
minute = mmC(tC)
```

```
second = ssC(tC)
```

```
td = dofC(tC)
```

```
tc = cofC(tC)
```

```
td = date(strdate, dpattern [, year])
```

```
td = mdy(month, day, year)
```

```
tw = yw(year, week)
```

```
tm = ym(year, month)
```

```
tq = yq(year, quarter)
```

```
th = yh(year, half)
```

```
tc = cofd(td)
```

```
tC = Cofd(td)
```

```
td = dofb(tb, "calendar")
```

```
tb = bofd("calendar", td)
```

```
month = month(td)
```

```
day = day(td)
```

```
year = year(td)
```

```
dow = dow(td)
```

```
week = week(td)
```

```
quarter = quarter(td)
```

```
half = halfyear(td)
```

```
doy = doy(td)
```

```
ty = yearly(strydate, ypattern [, year])
```

```
ty = yofd(td)
```

```
td = dofy(ty)
```

```
th = halfyearly(strhdate, hpattern [, year])
```

```
th = hofd(td)
```

```
td = dofh(th)
```

```
tq = quarterly(strqdate, qpattern [, year])
```

```
tq = qofd(td)
```

```
td = dofq(tq)
```

```
tm = monthly(strmdate, mpattern [, year])
```

```
tm = mofd(td)
```

```
td = dofM(tm)
```

```
tw = weekly(strwdate, wpattern [, year])
```

```
tw = wofd(td)
```

```
td = dof(tw)
```

```
hours = hours(ms)
```

```
minutes = minutes(ms)
```

```
seconds = seconds(ms)
```

```
ms = msofhours(hours)
```

```
ms = msofminutes(minutes)
```

```
ms = msofseconds(seconds)
```

```
int = age(td, td [, strleapday])
```

```
real = age_frac(td, td [, strleapday])
```

```
int = Clockdiff(tC, tC, strtimeunit)
```

```
int = clockdiff(tc, tc, strtimeunit)
```

```
real = Clockdiff_frac(tC, tC, strtimeunit)
```

```
real = clockdiff_frac(tc, tc, strtimeunit)
```

```
int = datediff(td, td, strtimeunit [, strleapday])
```

```
real = datediff_frac(td, td, strtimeunit [, strleapday])
```

```
td = birthday(td, year [, strleapday])
```

```
td = previousbirthday(td, td [, strleapday])
```

```
td = nextbirthday(td, td [, strleapday])
```

```
bool = isleapyear(year)
```

```
year = previousleapyear(year)
```

```
year = nextleapyear(year)
```

```
int = daysinmonth(td)
```

```
td = firstdayofmonth(td)
```

```
td = lastdayofmonth(td)
```

```
int = datepart(td, strtimeunit)
```

```
int = clockpart(tc, strtimeunit)
```

```
int = Clockpart(tC, strtimeunit)
```

```
bool = isleapsecond(tC)
```

```
td = today()
```

```
tc = now()
```

where

<i>tc</i> :	number of milliseconds from 01jan1960 00:00:00.000, unadjusted for leap seconds
<i>tC</i> :	number of milliseconds from 01jan1960 00:00:00.000, adjusted for leap seconds
<i>strdatetime</i> :	string-format date, time, or date/time, for example, "15jan1992", "1/15/1992", "15-1-1992", "January 15, 1992", "9:15", "13:42", "1:42 p.m.", "1:42:15.002 pm", "15jan1992 9:15", "1/15/1992 13:42", "15-1-1992 1:42 p.m.", "January 15, 1992 1:42:15.002 pm"
<i>pattern</i> :	order of month, day, year, hour, minute, and seconds in <i>strdatetime</i> , plus optional default century, for example, "DMYhms" (meaning day, month, year, hour, minute, second), "DMYhm", "MDYhm", "hmMDY", "hms", "hm", "MDY", "MD19Y", "MDY20Yhm"
<i>td</i> :	number of days from 01jan1960
<i>tb</i> :	business date (days)
<i>calendar</i> :	string scalar containing calendar name or %tb format
<i>strdate</i> :	string-format date, for example, "15jan1992", "1/15/1992", "15-1-1992", "January 15, 1992"
<i>dpattern</i> :	order of month, day, and year in <i>strdate</i> , plus optional default century, for example, "DMY" (meaning day, month, year), "MDY", "MD19Y"

<i>hour:</i>	hour, 0–23
<i>minute:</i>	minute, 0–59
<i>second:</i>	second, 0.000–59.999 (maximum 60.999 in case of leap second)
<i>month:</i>	month number, 1–12
<i>day:</i>	day-of-month number, 1–31
<i>year:</i>	year, for example, 1942, 1995, 2008
<i>week:</i>	week within year, 1–52
<i>quarter:</i>	quarter within year, 1–4
<i>half:</i>	half within year, 1–2
<i>dow:</i>	day of week, 0–6, where 0 = Sunday
<i>doj:</i>	day within year, 1–366
<i>ty:</i>	calendar year
<i>strydate:</i>	string-format calendar year, for example, "1980", "80"
<i>ypattern:</i>	pattern of <i>strydate</i> , for example, "Y", "19Y"
<i>th:</i>	number of halves from 1960h1
<i>strhdate:</i>	string-format <i>hdate</i> , for example, "1982-1", "1982h2", "2 1982"
<i>hpattern:</i>	pattern of <i>strhdate</i> , for example, "YH", "19YH", "HY"
<i>tq:</i>	number of quarters from 1960q1
<i>strqdate:</i>	string-format <i>qdate</i> , for example, "1982-3", "1982q2", "3 1982"
<i>qpattern:</i>	pattern of <i>strqdate</i> , for example, "YQ", "19YQ", "QY"
<i>tm:</i>	number of months from 1960m1
<i>strmdate:</i>	string-format <i>mdate</i> , for example, "1982-3", "1982m2", "3/1982"
<i>mpattern:</i>	pattern of <i>strmdate</i> , for example, "YM", "19YM", "MR"
<i>tw:</i>	number of weeks from 1960w1
<i>strwdate:</i>	string-format <i>wdate</i> , for example, "1982-3", "1982w2", "1982-15"
<i>wpattern:</i>	pattern of <i>strwdate</i> , for example, "YW", "19YW", "WY"
<i>hours:</i>	interval of time in hours (positive or negative, real)
<i>minutes:</i>	interval of time in minutes (positive or negative, real)
<i>seconds:</i>	interval of time in seconds (positive or negative, real)
<i>ms:</i>	interval of time in milliseconds (positive or negative, integer)

<i>int</i> :	integer value
<i>real</i> :	real value
<i>bool</i> :	Boolean logic value, 0 (false) or 1 (true)
<i>strleapday</i> :	string-format nonleap-year anniversary for 29feb, for example, "feb28", "28feb", "mar01", "mar1", "1mar"
<i>strtimeunit</i> :	string-format unit of time, for example, "year", "y", "month", "mon", "day", "d", "hour", "h", "minute", "min", "second", "s", "millisecond", "ms"

Functions return an element-by-element result. Functions are usually used with scalars.

All variables are *real matrix* except the *str** and **pattern* variables, which are *string matrix*.

Conformability

`clock(strdatetime, pattern, year)`, `Clock(strdatetime, pattern, year)`:

<i>strdatetime</i> :	$r_1 \times c_1$
<i>pattern</i> :	$r_2 \times c_2$ (c-conformable with <i>strdatetime</i>)
<i>year</i> :	$r_3 \times c_3$ (optional, c-conformable)
<i>result</i> :	$\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`mdyhms(month, day, year, hour, minute, second)`,
`Cmdyhms(month, day, year, hour, minute, second)`:

<i>month</i> :	$r_1 \times c_1$
<i>day</i> :	$r_2 \times c_2$
<i>year</i> :	$r_3 \times c_3$
<i>hour</i> :	$r_4 \times c_4$
<i>minute</i> :	$r_5 \times c_5$
<i>second</i> :	$r_6 \times c_6$ (all variables c-conformable)
<i>result</i> :	$\max(r_1, r_2, r_3, r_4, r_5, r_6) \times \max(c_1, c_2, c_3, c_4, c_5, c_6)$

`hms(hour, minute, second)`, `Chms(hour, minute, second)`:

<i>hour</i> :	$r_1 \times c_1$
<i>minute</i> :	$r_2 \times c_2$
<i>second</i> :	$r_3 \times c_3$
<i>result</i> :	$\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`dhms(td, hour, minute, second)`, `Cdhms(td, hour, minute, second)`:

<i>td</i> :	$r_1 \times c_1$
<i>hour</i> :	$r_2 \times c_2$
<i>minute</i> :	$r_3 \times c_3$
<i>second</i> :	$r_4 \times c_4$ (all variables c-conformable)
<i>result</i> :	$\max(r_1, r_2, r_3, r_4) \times \max(c_1, c_2, c_3, c_4)$

`hh(x)`, `mm(x)`, `ss(x)`, `hhC(x)`, `mmC(x)`, `ssC(x)`:

<i>x</i> :	$r \times c$
<i>result</i> :	$r \times c$

date(*strdate*, *dpattern*, *year*):

strdate: $r_1 \times c_1$
dpattern: $r_2 \times c_2$ (c-conformable with *strdate*)
year: $r_3 \times c_3$ (optional, c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

mdy(*month*, *day*, *year*):

month: $r_1 \times c_1$
day: $r_2 \times c_2$
year: $r_3 \times c_3$ (all variables c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

yw(*year*, *detail*), ym(*year*, *detail*), yq(*year*, *detail*), yh(*year*, *detail*):

year: $r_1 \times c_1$
detail: $r_2 \times c_2$ (c-conformable with *year*)
result: $\max(r_1, r_2) \times \max(c_1, c_2)$

month(*td*), day(*td*), year(*td*), dow(*td*), week(*td*), quarter(*td*), halfyear(*td*), doy(*td*):

td: $r \times c$
result: $r \times c$

yearly(*str*, *pat*, *year*), halfyearly(*str*, *pat*, *year*), quarterly(*str*, *pat*, *year*),
 monthly(*str*, *pat*, *year*), weekly(*str*, *pat*, *year*):

str: $r_1 \times c_1$
pat: $r_2 \times c_2$ (c-conformable with *str*)
year: $r_3 \times c_3$ (optional, c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

Cofc(*x*), cofC(*x*), dofC(*x*), dofC(*x*), cofd(*x*), Cofd(*x*), yofd(*x*), dofy(*x*), hofd(*x*),
 dofh(*x*), qofd(*x*), dofq(*x*), mofd(*x*), dofM(*x*), wofd(*x*), dofW(*x*):

x: $r \times c$
result: $r \times c$

dofb(*tb*, "calendar"):

tb: $r \times c$
calendar: 1×1
result: $r \times c$

bofd("calendar", *td*):

calendar: 1×1
td: $r \times c$
result: $r \times c$

hours(*x*), minutes(*x*), seconds(*x*), msOfhours(*x*), msOfminutes(*x*), msOfseconds(*x*):

x: $r \times c$
result: $r \times c$

age(*td1*, *td2*, *str*), age_frac(*td1*, *td2*, *str*):

td1: $r_1 \times c_1$
td2: $r_2 \times c_2$ (c-conformable with *td1*)
str: $r_3 \times c_3$ (optional, c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`datediff(td1, td2, str1, str2)`, `datediff_frac(td1, td2, str1, str2)`:

td1: $r_1 \times c_1$
td2: $r_2 \times c_2$
str1: $r_3 \times c_3$ (all variables c-conformable)
str2: $r_4 \times c_4$ (optional, c-conformable)
result: $\max(r_1, r_2, r_3, r_4) \times \max(c_1, c_2, c_3, c_4)$

`Clockdiff(tC1, tC2, str)`, `Clockdiff_frac(tC1, tC2, str)`:

tC1: $r_1 \times c_1$
tC2: $r_2 \times c_2$
str: $r_3 \times c_3$ (all variables c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`clockdiff(tc1, tc2, str)`, `clockdiff_frac(tc1, tc2, str)`:

tc1: $r_1 \times c_1$
tc2: $r_2 \times c_2$
str: $r_3 \times c_3$ (all variables c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`birthday(td, detail, str)`, `previousbirthday(td, detail, str)`,
`nextbirthday(td, detail, str)`:

td: $r_1 \times c_1$
detail: $r_2 \times c_2$ (c-conformable with *td*)
str: $r_3 \times c_3$ (optional, c-conformable)
result: $\max(r_1, r_2, r_3) \times \max(c_1, c_2, c_3)$

`isleapyear(year)`, `previousleapyear(year)`, `nextleapyear(year)`:

year: $r \times c$
result: $r \times c$

`daysinmonth(td)`, `firstdayofmonth(td)`, `lastdayofmonth(td)`:

td: $r \times c$
result: $r \times c$

`Clockpart(detail, str)`, `clockpart(detail, str)`, `datepart(detail, str)`:

detail: $r_1 \times c_1$
str: $r_2 \times c_2$
result: $\max(r_1, r_2) \times \max(c_1, c_2)$

`isleapsecond(tC)`:

tC: $r \times c$
result: $r \times c$

Diagnostics

None.

Also see

[M-4] [Dates](#) — Date and time functions