# Title

stata.com

**datetime business calendars —** Business calendars

Description Syntax Remarks and examples Also see

## Description

Stata provides user-definable business calendars.

## Syntax

*Apply business calendar format*

`format` *varlist* `%tb`*calname*

*Apply detailed date format with business calendar format*

`format` *varlist* `%tb`*calname*[ `:`*datetime-specifiers* ]

*Convert between business dates and regular dates*

{ `generate` | `replace` } *bdate* `= bofd("`*calname*`",` *regulardate*`)`

{ `generate` | `replace` } *regulardate* `= dofb(`*bdate*`, "`*calname*`")`

File *calname*`.stbcal` contains the business calendar definition.

Details of the syntax follow:

1. Definition.
   Business calendars are regular calendars with some dates crossed out:

November 2011

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | X |
| X | 7 | 8 | 9 | 10 | 11 | X |
| X | 14 | 15 | 16 | 17 | 18 | X |
| X | 21 | 22 | 23 | X | X | X |
| X | 28 | 29 | 30 | | | |

   A date that appears on the business calendar is called a business date. 11nov2011 is a business date. 12nov2011 is not a business date with respect to this calendar.

   Crossed-out dates are literally omitted. That is,

$$18\text{nov}2011 + 1 = 21\text{nov}2011$$

$$28\text{nov}2011 - 1 = 23\text{nov}2011$$

   Stata's lead and lag operators work the same way.

2. Business calendars are named.
   Assume that the above business calendar is named `simple`.

3. Business calendars are defined in files named *calname*`.stbcal`, such as `simple.stbcal`. Calendars may be supplied by StataCorp and already installed, obtained from other users directly or via the SSC, or written yourself. Calendars can also be created automatically from the current dataset with the `bcal create` command; see [D] **bcal**. Stbcal-files are treated in the same way as ado-files.

   You can obtain a list of all business calendars installed on your computer by typing `bcal dir`; see [D] **bcal**.

4. Datetime format.
   The date format associated with the business calendar named `simple` is `%tbsimple`, which is to say `%` + `t` + `b` + *calname*.

   | | |
   |---|---|
   | `%` | it is a format |
   | `t` | it is a datetime |
   | `b` | it is based on a business calendar |
   | *calname* | the calendar's name |

5. Format variables the usual way.
   You format variables to have business calendar formats just as you format any variable, using the `format` command.

   ```
   . format mydate %tbsimple
   ```

   specifies that existing variable `mydate` contains values according to the business calendar named `simple`. See [D] **format**.

   You may format variables `%tb`*calname* regardless of whether the corresponding stbcal-file exists. If it does not exist, the underlying numeric values will be displayed in a `%g` format.

6. Detailed date formats.
   You may include detailed datetime format specifiers by placing a colon and the detail specifiers after the calendar's name.

   ```
   . format mydate %tbsimple:CCYY.NN.DD
   ```

   would display 21nov2011 as 2011.11.21. See [D] **datetime display formats** for detailed datetime format specifiers.

7. Reading business dates.
   To read files containing business dates, ignore the business date aspect and read the files as if they contained regular dates. Convert and format those dates as `%td`; see *HRF-to-SIF conversion functions* in [D] **datetime**. Then convert the regular dates to `%tb` business dates:

   ```
   . generate mydate = bofd("simple", regulardate)
   . format mydate %tbsimple
   . assert mydate!=. if regulardate!=.
   ```

   The first statement performs the conversion.

   The second statement attaches the `%tbsimple` date format to the new variable `mydate` so that it will display correctly.

   The third statement verifies that all dates recorded in `regulardate` fit onto the business calendar. For instance, 12nov2011 does not appear on the `simple` calendar but, of course, it does appear on the regular calendar. If the data contained 12nov2011, that would be an error. Function `bofd()` returns missing when the date does not appear on the specified calendar.

8. More on conversion.
   There are only two functions specific to business dates, `bofd()` and `dofb()`. Their definitions are

   *bdate* = `bofd("`*calname*`",` *regulardate*`)`
   *regulardate* = `dofb(`*bdate*`, "`*calname*`")`

   `bofd()` returns missing if *regulardate* is missing or does not appear on the specified business calendar. `dofb()` returns missing if *bdate* contains missing.

9. Obtaining day of week, etc.
   You obtain day of week, etc., by converting business dates to regular dates and then using the standard functions. To obtain the day of week of *bdate* on business calendar *calname*, type

   ```
   . generate dow = dow(dofb(bdate, "calname"))
   ```

   See *Extracting date components from SIFs* in [D] **datetime** for the other extraction functions.

10. Stbcal-files.
    The stbcal-file for `simple`, the calendar shown below,

November 2011

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | X |
| X | 7 | 8 | 9 | 10 | 11 | X |
| X | 14 | 15 | 16 | 17 | 18 | X |
| X | 21 | 22 | 23 | X | X | X |
| X | 28 | 29 | 30 | | | |

is

begin `simple.stbcal`

```
*! version 1.0.0
*  simple.stbcal
version 15.1
purpose "Example for manual"
dateformat dmy
range 01nov2011 30nov2011
centerdate 01nov2011
omit dayofweek (Sa Su)
omit date 24nov2011
omit date 25nov2011
```

end `simple.stbcal`

This calendar was so simple that we crossed out the Thanksgiving holidays by specifying the dates to be omitted. In a real calendar, we would change the last two lines,

```
omit date 24nov2011
omit date 25nov2011
```

to read

```
omit dowinmonth +4 Th of Nov and +1
```

which says to omit the fourth (+4) Thursday of November in every year, and omit the day after that (+1), too. See [D] **datetime business calendars creation**.

# Remarks and examples

stata.com

See [D] **datetime** for an introduction to Stata's date and time features.

Below we work through an example from start to finish.

Remarks are presented under the following headings:



## Step 1: Read the data, date as string

File `bcal_simple.raw` on our website provides data, including a date variable, that is to be interpreted according to the business calendar `simple` shown under *Syntax* above.

```
. type http://www.stata-press.com/data/r15/bcal_simple.raw
11/4/11 51
11/7/11 9
11/18/11 12
11/21/11 4
11/23/11 17
11/28/11 22
```

We begin by reading the data and then listing the result. Note that we read the date as a string variable:

```
. infile str10 sdate float x using http://www.stata-press.com/data/r15/bcal_simple
(6 observations read)
. list
```

| | sdate | x |
|---|---|---|
| 1. | 11/4/11 | 51 |
| 2. | 11/7/11 | 9 |
| 3. | 11/18/11 | 12 |
| 4. | 11/21/11 | 4 |
| 5. | 11/23/11 | 17 |
| 6. | 11/28/11 | 22 |

## Step 2: Convert date variable to %td date

Now we create a Stata internal form (SIF) `%td` format date from the string date:

```
. generate rdate = date(sdate, "MD20Y")
. format rdate %td
```

See *HRF-to-SIF conversion functions* in [D] **datetime**. We verify that the conversion went well and drop the string variable of the date:

```
. list
```

| | sdate | x | rdate |
|---|---|---|---|
| 1. | 11/4/11 | 51 | 04nov2011 |
| 2. | 11/7/11 | 9 | 07nov2011 |
| 3. | 11/18/11 | 12 | 18nov2011 |
| 4. | 11/21/11 | 4 | 21nov2011 |
| 5. | 11/23/11 | 17 | 23nov2011 |
| 6. | 11/28/11 | 22 | 28nov2011 |

```
. drop sdate
```

## Step 3: Convert %td date to %tb date

We convert the `%td` date to a `%tbsimple` date following the instructions of item 7 of *Syntax* above.

```
. generate mydate = bofd("simple", rdate)
. format mydate %tbsimple
. assert mydate!=. if rdate!=.
```

Had there been any dates that could not be converted from regular dates to `simple` business dates, `assert` would have responded, "assertion is false". Nonetheless, we will list the data to show you that the conversion went well. We would usually drop the `%td` encoding of the date, but we want it to demonstrate a feature below.

```
. list
```

| | x | rdate | mydate |
|---|---|---|---|
| 1. | 51 | 04nov2011 | 04nov2011 |
| 2. | 9 | 07nov2011 | 07nov2011 |
| 3. | 12 | 18nov2011 | 18nov2011 |
| 4. | 4 | 21nov2011 | 21nov2011 |
| 5. | 17 | 23nov2011 | 23nov2011 |
| 6. | 22 | 28nov2011 | 28nov2011 |

## Key feature: Each business calendar has its own encoding

In the listing above, `rdate` and `mydate` appear to be equal. They are not:

```
. format rdate mydate %9.0g           // remove date formats
. list
```

| | x | rdate | mydate |
|---|---|---|---|
| 1. | 51 | 18935 | 3 |
| 2. | 9 | 18938 | 4 |
| 3. | 12 | 18949 | 13 |
| 4. | 4 | 18952 | 14 |
| 5. | 17 | 18954 | 16 |
| 6. | 22 | 18959 | 17 |

%tb dates each have their own encoding, and those encodings differ from the encoding used by %td dates. It does not matter. Neither encoding is better than the other. Neither do you need to concern yourself with the encoding. If you were curious, you could learn more about the encoding used by %tbsimple by typing bcal describe simple; see [D] **bcal**.

We will drop variable rdate and put the %tbsimple format back on variable mydate:

```
. drop rdate
. format mydate %tbsimple
```

## Key feature: Omitted dates really are omitted

In *Syntax*, we mentioned that for the simple business calendar

$$18\text{nov}2011 + 1 = 21\text{nov}2011$$

$$28\text{nov}2011 - 1 = 23\text{nov}2011$$

That is true:

```
. generate tomorrow = mydate + 1
. generate yesterday = mydate - 1
. format tomorrow yesterday %tbsimple
. list
```

| | x | mydate | tomorrow | yesterday |
|---|---|---|---|---|
| 1. | 51 | 04nov2011 | 07nov2011 | 03nov2011 |
| 2. | 9 | 07nov2011 | 08nov2011 | 04nov2011 |
| 3. | 12 | 18nov2011 | 21nov2011 | 17nov2011 |
| 4. | 4 | 21nov2011 | 22nov2011 | 18nov2011 |
| 5. | 17 | 23nov2011 | 28nov2011 | 22nov2011 |
| 6. | 22 | 28nov2011 | 29nov2011 | 23nov2011 |

```
. drop tomorrow yesterday
```

Stata's lag and lead operators L.*varname* and F.*varname* work similarly.

## Key feature: Extracting components from %tb dates

You extract components such as day of week, month, day, and year from business dates using the same extraction functions you use with Stata's regular %td dates, namely, dow(), month(), day(), and year(), and you use function dofb() to convert business dates to regular dates. Below we add day of week to our data, list the data, and then drop the new variable:

```
. generate dow = dow(dofb(mydate, "simple"))
. list
```

| | x | mydate | dow |
|---|---|---|---|
| 1. | 51 | 04nov2011 | 5 |
| 2. | 9 | 07nov2011 | 1 |
| 3. | 12 | 18nov2011 | 5 |
| 4. | 4 | 21nov2011 | 1 |
| 5. | 17 | 23nov2011 | 3 |
| 6. | 22 | 28nov2011 | 1 |

```
. drop dow
```

See *Extracting date components from SIFs* in [D] **datetime**.

## Key feature: Merging on dates

It may happen that you have one dataset containing business dates and a second dataset containing regular dates, say, on economic conditions, and you want to merge them. To do that, you create a regular date variable in your first dataset and merge on that:

```
. generate rdate = dofb(mydate, "simple")
. merge 1:1 rdate using econditions, keep(match)
. drop rdate
```

# Also see

[D] **bcal** — Business calendar file manipulation

[D] **datetime business calendars creation** — Business calendars creation

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