The Stata Journal (2010)
10, Number 4, pp. 682–685

Stata tip 68: Week assumptions

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

1 Introduction

Stata's handling of dates and times is centered on daily dates. Days are aggregated into weeks, months, quarters, half-years, and years. Days are divided into hours, minutes, and seconds. This may all sound simple in principle, leaving just the matter of identifying the syntax for converting from one form of representation to another. For an introduction, see [U] **24 Working with dates and times**. For more comprehensive treatments, see [D] **dates and times** and [D] **functions**. As a matter of history, know that specific date functions were introduced in Stata 4 in 1995, replacing an earlier system based on ado-files. These date functions were much enhanced in Stata 6 in 1999 and again in Stata 10 in 2007.

However, matters are not quite as simple as this description implies. The occasional addition of leap seconds lengthening the year is a complication arising with some datasets. A little thought shows that weeks are also awkward—whatever the precise definition of a week, weeks are not guaranteed to nest neatly and evenly into months, quarters, half-years, or years. This tip focuses on Stata's solution for weeks and on how to set up your own alternatives given different definitions of the week. A conventional Western calendar with seven-day weeks is assumed throughout this tip. For much richer historical, cultural, and computational context, see Richards (1998), Holford-Strevens (2005), and Dershowitz and Reingold (2008).

Gabriel Rossman is thanked for providing stimulating comments.

2 Stata's definition of weeks

Stata's definition of weeks matches one desideratum and violates others, as would any other definition. For Stata, week 1 of any year starts on 1 January, whatever day of the week that is (Sunday through Saturday).

```
. display %tw wofd(mdy(1,1,2010))
 2010w1
. display %tw wofd(mdy(1,7,2010))
 2010w1
. display %tw wofd(mdy(1,8,2010))
 2010w2
```

In these examples, two date functions are used: `mdy()` yields daily dates from month, day, and year components, and `wofd()` converts such dates to the corresponding weeks. Most users prefer to see dates shown intelligibly with date display formats such as `%tw`, thus hiding the underlying Stata machinery, which pivots on a convention that 1 January 1960 is date origin or day 0.

At the end of the year, for Stata the 52nd week always lasts 8 days in nonleap years and 9 days in leap years. (Recall that $52 \times 7 = 364$, so a calendar year always has either 1 day or 2 days more than that.)

This definition ensures that weeks nest within years, meaning that no week ever starts in one calendar year and finishes in the next. Also by this definition, a year has precisely 52 weeks, even though the last week is never 7 days long. Naturally, this solution (and indeed any other) cannot ensure that weeks always nest within months, quarters, or half-years. (February is sometimes an exception, but necessarily the only one.)

3 Alternative assumptions

Users who deal with weeks may wish to work with other definitions. The most obvious alternatives arise from regarding particular days of the week as defining either the start or the end of the week. With these definitions, the relation of weeks to the days of the week is fixed, and all weeks last precisely 7 days. However, other desiderata are violated: notably, weeks may now span two calendar years.

Such definitions are most likely to appeal whenever weekly cycles are part of what is being investigated and particular days have meaning. Examples from major religions will be familiar. Particular days of the week are often key for financial transactions. More parochially, Durham University teaching weeks start on Thursdays in the first term of each academic year and on Mondays in the other two terms.

With such alternatives, there is no need to set up a new numbering system for weeks, at least as far as working within Stata is concerned. Each week can just be identified by its start or end date, whichever is desired. If you want to map the weeks that do occur in your dataset to a simple numbering scheme, `egen, group()` does this easily.

Classifying weeks by their start days is an exercise in rounding down, and classifying them by their end days is one in rounding up, so solutions using `floor()` and `ceil()` are possible; see Cox (2003). However, a solution using a date is likely to seem more attractive.

Consider this pretend dataset of 8 days in 2010.

```
. clear
. set obs 8
obs was 0, now 8
. gen day = _n + mdy(9,25,2010)
. gen day2 = day
. format day %tdDay
. format day2 %tdd_m
. gen dow = dow(day)
. list

     ┌──────────────────────┐
     │ day     day2    dow  │
     ├──────────────────────┤
  1. │ Sun   26 Sep      0  │
  2. │ Mon   27 Sep      1  │
  3. │ Tue   28 Sep      2  │
  4. │ Wed   29 Sep      3  │
  5. │ Thu   30 Sep      4  │
     ├──────────────────────┤
  6. │ Fri    1 Oct      5  │
  7. │ Sat    2 Oct      6  │
  8. │ Sun    3 Oct      0  │
     └──────────────────────┘
```

The Stata function `dow()` returns day of the week coded `0` for Sunday through `6` for Saturday. The dates 26 September 2010 and 3 October 2010 were Sundays. Now we have within reach one-line solutions for various problems, given here in terms of the example daily date variable `day`.

The simplest case is classifying days in each week by the Sundays that start them. Glancing at the listing above shows that we just need to subtract `dow(day)` from `day`:

```
. gen sunstart = day - dow(day)
```

Working with other days can be thought of as rotating the days of the weeks to an origin other than Sunday. One good way to do that is using the versatile function `mod()` (Cox 2007). If d is one of 1, …, 6, then `mod(dow(day) -` d`, 7)` rotates the results of `dow()` so that day d is the new origin. $d = 0$ leaves the days as they were.

Hence if we wish to classify days by starting Fridays, subtract `mod(dow(day) - 5, 7)` from `day`:

```
. gen fristart = day - mod(dow(day) - 5, 7)
```

This is consistent with the simpler rule for Sunday. As just implied, `mod(dow(day) - 0, 7)` is identical to `mod(dow(day), 7)` and in turn to `dow(day)`.

Now consider classifying weeks by the days that end them. We now need to add an appropriate number between 0 and 6 to the date. That number will cycle from 6 to 0 to 6, rather than from 0 to 6 to 0; it is given by `mod(`d` - dow(day), 7)`. So, Friday-ending weeks are given by

```
. gen friend = day + mod(5 - dow(day), 7)
```

This trick also offers a good way to generate ending Sundays:

```
. gen sunend = day + mod(0 - dow(day), 7)
```

The zero in the line just above is not necessary but is left in to emphasize the resemblance to the previous line.

Finally, let's show the results of these calculations. There are several other ways to compute them, but using `mod()` has its attractions. This method could also be extended to weeks that are not 7 days long, which do arise in certain research problems.

```
. format su* fr* %tdd_m

. list day* *start *end

     ┌──────────────────────────────────────────────────────────┐
     │ day     day2   sunstart   fristart   friend   sunend    │
     ├──────────────────────────────────────────────────────────┤
  1. │ Sun   26 Sep     26 Sep     24 Sep    1 Oct   26 Sep    │
  2. │ Mon   27 Sep     26 Sep     24 Sep    1 Oct    3 Oct    │
  3. │ Tue   28 Sep     26 Sep     24 Sep    1 Oct    3 Oct    │
  4. │ Wed   29 Sep     26 Sep     24 Sep    1 Oct    3 Oct    │
  5. │ Thu   30 Sep     26 Sep     24 Sep    1 Oct    3 Oct    │
     ├──────────────────────────────────────────────────────────┤
  6. │ Fri    1 Oct     26 Sep      1 Oct    1 Oct    3 Oct    │
  7. │ Sat    2 Oct     26 Sep      1 Oct    8 Oct    3 Oct    │
  8. │ Sun    3 Oct      3 Oct      1 Oct    8 Oct    3 Oct    │
     └──────────────────────────────────────────────────────────┘
```

References

Cox, N. J. 2003. Stata tip 2: Building with floors and ceilings. *Stata Journal* 3: 446–447.

———. 2007. Stata tip 43: Remainders, selections, sequences, extractions: Uses of the modulus. *Stata Journal* 7: 143–145.

Dershowitz, N., and E. M. Reingold. 2008. *Calendrical Calculations*. 3rd ed. Cambridge: Cambridge University Press.

Holford-Strevens, L. 2005. *The History of Time: A Very Short Introduction*. Oxford: Oxford University Press.

Richards, E. G. 1998. *Mapping Time: The Calendar and Its History*. Oxford: Oxford University Press.

Editors' note. Some alert readers have noted that there was no Stata tip 68; we moved from 67 to 69 without explanation. This was pure oversight on the part of the *Stata Journal*. In Stata tradition, we now fix this bug with an apology, particularly so that any future compilations of tips include precisely the number advertised.