The Stata Journal (2024)
24, Number 4, pp. 777–783

DOI: 10.1177/1536867X241297950

Stata tip 158: The devil is in the delta

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

1 Introduction: Why use tsset or xtset?

Many researchers deal with data that are single or multiple time series, either for a single entity or for several entities. Data for several entities are often called *panel data* or *longitudinal data*, particularly in economic, social, and health or medical sciences, where the entities may be employees, companies, countries, patients, or pigs (for example, Diggle et al. [2002]; Wooldridge [2010]). Many datasets in other sciences have the same form even if neither emphasized term is used alongside them. In meteorology or hydrology, we might have temperature, precipitation, or river discharge data for multiple stations.

Declaring your Stata dataset as time-series or panel or longitudinal data with a `tsset` or `xtset` command is usually a good idea and indeed essential for many purposes. Time-series operators such as lag or difference operators cannot be used without such a declaration. Various data management, graphical, and statistical commands also require that declaration.

There is a kind of symmetry between `tsset` and `xtset`.

You may use `tsset` *timevar* if you have no panel identifier or at least need only to declare a time variable *timevar* for a dataset that consists of time series. You should use `xtset` *idvar* if you wish to declare a panel identifier variable *idvar* but nevertheless do not wish to declare a time variable. This situation will often arise because observations occur at irregularly (unequally) spaced times, so time-series operators would be of little or no use. See Lazzaro (2023) for detailed advice on that case.

Either `tsset` or `xtset` may be used for declaring both a panel (or some other) identifier and a time variable. So `tsset` *idvar timevar* and `xtset` *idvar timevar* have the same effects.

Behind the problems discussed here lie the complications of the calendar and of time-of-day recording. Many will have been familiar to you since childhood. The details, often bizarre or recondite, may variously fascinate or frustrate, depending on whether you are reading for pleasure or wrestling a dataset into a shape that matches their quirks and is fit for later analysis. Informative references include Richards (1998), Blackburn and Holford-Strevens (1999), Holford-Strevens (2005), and Reingold and Dershowitz (2018).

2 An option that may be essential

The point of this tip is to underline that the `delta()` option of `tsset` or `xtset` is essential in many cases, even though as the term implies, it is optional as a matter of syntax. Use of the `delta()` option is documented in detail in the help files for these commands and the corresponding manual entries, but the need for this option and the surrounding pitfalls are often misunderstood, so another version of key facts and advice may prove helpful.

3 Dealing with regularly spaced data: delta() may help

It is quite common to have data regularly, meaning equally, spaced in time, yet the interval concerned is not 1 in the time units you are using. Stata does not inspect your data to determine the typical or still less the correct spacing that should obtain. The responsibility is yours: as the researcher, you should know about your data and about how your data should be treated.

If you do not specify the `delta()` option, each command, `tsset` or `xtset`, assumes `delta(1)`. Your syntax may well be legal without the `delta()` option, but the consequences of omitting it may be utterly wrong for your analysis. Typically, your dataset might appear to Stata to be mostly holes or gaps. That may manifest itself as difficulty with low lags or even a puzzling failure to find any observations to work with.

To make the point concrete, here are some common examples.

3.1 Data for years but multiple years apart

Olympic years or US presidential election years are typically 4 years apart. National population censuses are often once every 10 years. Typically, an option call such as `delta(4)` or `delta(10)` will help with such data. The examples incidentally underline that your data may be untidier than the main principle implies. Many events were postponed or canceled in 2020 because of the ongoing COVID-19 pandemic, but let's leave that on one side.

Consider what happens if you specify `tsset` or `xtset` but do not specify the correct argument to `delta()`. If data are for years like 2024, 2020, 2016, and so on, and `delta(1)` is implied, then for lag 1, commands will look for values for years 2023, 2019, 2015, and so on. For lag 2, commands will look for years 2022, 2018, 2014, and so on. Either those years are just absent from the data or those years are present in the data, yet the values for those years for key variables are missing. Either way, the result will not be what you want. Similar problems will arise with lag 3. For lag 4, values will be found, but they really should be for lag 1. This illustration stands for all others where time spacing is not matched by `delta()`.

3.2 Data for months or quarters but indexed by daily dates

Ideally, under this heading, the daily dates are already Stata numeric daily dates counted from 0 as 1 January 1960 so that later daily dates are indexed by positive integers and earlier daily dates by negative integers. If your daily dates do not match this form, `help datetime` gives guidance on how to achieve it. Sometimes, the daily dates are notional but with a detectable convention, such as using the first or last date of a month or of a quarter.

In this case, `delta()` cannot help immediately, because the intervals in terms of daily dates are not constant in length. Month lengths in the conventional Western calendar vary sufficiently enough to produce a mess. That is, month lengths vary from 28 to 31 even with a rigid convention about using month starts or month ends to index monthly data. Correspondingly, quarter lengths vary too in terms of number of days included. The solution here is first to map to monthly or quarterly dates using function `mofd()` or function `qofd()`.

Consider some daily dates, first input as strings:

```
. input str9 sdate
        sdate
  1. "31/1/2024"
  2. "29/2/2024"
  3. "31/3/2024"
  4. "30/4/2024"
  5. "31/5/2024"
  6. end
```

A quick glance shows monthly data indexed by the last day of each month, but Stata cannot work that out by itself. To make progress, we first use `daily()` or equivalently `date()` to generate a numeric daily date variable. At the same time, we also specify a daily date display format:

```
. generate date = daily(sdate, "DMY")
. format date %td
. list
```

	sdate	date
1.	31/1/2024	31jan2024
2.	29/2/2024	29feb2024
3.	31/3/2024	31mar2024
4.	30/4/2024	30apr2024
5.	31/5/2024	31may2024

If we now `tsset` in terms of this new date variable, there is a warning about gaps, but it is understated. There are no flashing lights or blaring klaxons to signal danger ahead, yet it lurks nevertheless.

```
. tsset date
Time variable: date, 31jan2024 to 31may2024, but with gaps
        Delta: 1 day
```

The solution is to work in terms of a monthly date variable, which we need to `generate` first.

```
. generate mdate = mofd(date)
. format mdate %tm
. list
```

	sdate	date	mdate
1.	31/1/2024	31jan2024	2024m1
2.	29/2/2024	29feb2024	2024m2
3.	31/3/2024	31mar2024	2024m3
4.	30/4/2024	30apr2024	2024m4
5.	31/5/2024	31may2024	2024m5

Now the problem is soluble: use `tsset` with this new variable.

```
. tsset mdate
Time variable: mdate, 2024m1 to 2024m5
        Delta: 1 month
```

Essentially, the same kind of solution solves a similar problem with quarterly data. Imagine that we had quarterly data indexed by daily dates such as 31 March, 30 June, 30 September, and 31 December or 1 January, 1 April, 1 July, and 1 October. The need is first to map to a quarterly date using the function `qofd()` and then to `tsset` in terms of that.

In either case, if you have a panel or some other identifier as well, then specify that too as needed.

3.3 Data for weeks but indexed by daily dates

Again, ideally the daily dates are already Stata numeric daily dates. In this case, I do not recommend that you try to make use of Stata's weekly dates. For why not, here is the gist, but see, for example, Cox (2010) for more discussion if needed. For yet more advice on handling weeks in Stata if your problem is not explained here, see Cox (2012a,b, 2019, 2022a,b).

Weeks are easy for people to understand informally but often awkward to use in statistically-based research. Essentially, there are several definitions of weeks in use, and almost never do weeks nest neatly into months, quarters, half-years, or years.

Stata's own definition of weeks is idiosyncratic. Week 1 always starts on 1 January of any year, week 2 on 8 January, and so on. Week 52 always finishes on 31 December and is thus 9 or 8 days long, depending on whether the year is or is not leap: in a leap year, there are 29 days in February and 366 days in total; otherwise, there are 28 days in February and 365 days in total. With Stata's definition, there is never a week 53 in any year. With this definition, Stata weeks always nest within years, which is tidy in itself. Unfortunately, weekly data usually arrive using some other definition.

There are two common versions of the problem of weeks being indexed by daily dates. In the simpler version, observations are typically once per week. For this case, my suggestion is that daily dates will work well to index each week.

As an example with real data, I downloaded data on financial stress from https://fred.stlouisfed.org/series/STLFSI4/ on 23 June 2024 and got 1,990 observations. Here are the last 5 for the St. Louis Fed Financial Stress Index.

```
. clear
. input str10 date float stlfsi4
          date    stlfsi4
  1. "17/05/2024" -.8975
  2. "24/05/2024" -.9651
  3. "31/05/2024" -.8585
  4. "07/06/2024" -.8039
  5. "14/06/2024" -.8508
  6. end
```

So each week is indexed by a daily date, Friday, in each week. All we need to do is to use that daily date to index each week. The only twist is the need to declare `delta(7)` in the `tsset` command.

```
. generate wdate = daily(date, "DMY")
. format wdate %td
. tsset wdate, delta(7)
Time variable: wdate, 17may2024 to 14jun2024
        Delta: 7 days
```

With this solution, graphical and tabular output can make direct use of dates that should make sense to both researchers and readers.

In the more challenging version, observations may be for two or more days of each week, say, Mondays to Fridays. We seek reduction of those data to weeks in some fashion, say, to weekly totals or means. The easiest reduction starts with declaring that weeks start on Sundays. In the last week of the previous example, these days are Monday to Friday:

```
. clear
. input str9 sdate
        sdate
  1. "10/6/2024"
  2. "11/6/2024"
  3. "12/6/2024"
  4. "13/6/2024"
  5. "14/6/2024"
  6. end
. generate date = daily(sdate, "DMY")
. format date %td
```

The function `dow()` returns the day of the week as 0 for Sunday to 6 for Saturday. If we subtract its result, we will get the Sunday that starts the week.

```
. generate wdate = date - dow(date)
. format wdate %td
. list
```

	sdate	date	wdate
1.	10/6/2024	10jun2024	09jun2024
2.	11/6/2024	11jun2024	09jun2024
3.	12/6/2024	12jun2024	09jun2024
4.	13/6/2024	13jun2024	09jun2024
5.	14/6/2024	14jun2024	09jun2024

If we want weeks to be indexed as ending on Fridays, just add 5 to that weekly date. We could now proceed to `collapse` or `contract` data for other variables, presumed offstage in this example, to weekly summaries.

The community-contributed command `myweeks` (Cox 2018) from Statistical Software Components Archive gives code for implementing either idea, or, indeed, for numbering weeks by successive integers. With the latter solution, `delta(1)` is then fine.

3.4 Data for datetimes but with an interval not 1 ms

Stata datetimes are, or should be, implemented using `double` variables holding integers with time units in milliseconds (ms). This is driven by one fact: financial data analyzed by many Stata users have timestamps with such a resolution. Any physical or other scientists in need of, say, a nanosecond resolution are presumably either using other software or entirely capable of setting up their own code holding times for their own purposes.

It would be a surprise, however, to find many (or even any) Stata datasets with equally spaced data in which observations were once every ms. More commonly, the time step or resolution is much longer. Documentation for both `xtset` and `tsset` shows the flexibility allowed in specifying time steps other than 1 ms.

References

Blackburn, B., and L. Holford-Strevens. 1999. *The Oxford Companion to the Year*. Oxford: Oxford University Press.

Cox, N. J. 2010. Stata tip 68: Week assumptions. *Stata Journal* 10: 682–685. https://doi.org/10.1177/1536867X1101000409.

———. 2012a. Stata tip 111: More on working with weeks. *Stata Journal* 12: 565–569. https://doi.org/10.1177/1536867X1201200316.

———. 2012b. Stata tip 111: More on working with weeks, erratum. *Stata Journal* 12: 765. https://doi.org/10.1177/1536867X1201200416.

———. 2018. myweeks: Stata module for generating numbered weeks from daily date variables. Statistical Software Components S459340, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s459340.html.

———. 2019. Speaking Stata: The last day of the month. *Stata Journal* 19: 719–728. https://doi.org/10.1177/1536867X19874247.

———. 2022a. Stata tip 145: Numbering weeks within months. *Stata Journal* 22: 224–230. https://doi.org/10.1177/1536867X221083928.

———. 2022b. Stata tip 145: Numbering weeks within months, erratum. *Stata Journal* 22: 465–466. https://doi.org/10.1177/1536867X221106438.

Diggle, P. J., P. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Holford-Strevens, L. 2005. *The History of Time: A Very Short Introduction*. Oxford: Oxford University Press. https://doi.org/10.1093/actrade/9780192804990.001.0001.

Lazzaro, C. 2023. Stata tip 150: When is it appropriate to xtset a panel dataset with panelvar only? *Stata Journal* 23: 281–292. https://doi.org/10.1177/1536867X231162020.

Reingold, N. M., and N. Dershowitz. 2018. *Calendrical Calculations: The Ultimate Edition*. Cambridge: Cambridge University Press.

Richards, E. G. 1998. *Mapping Time: The Calendar and Its History*. Oxford: Oxford University Press. https://doi.org/10.1093/oso/9780198504139.001.0001.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.