The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
1-year subscription	$ 79	1-year subscription	$115
2-year subscription	$155	2-year subscription	$225
3-year subscription	$225	3-year subscription	$329
3-year subscription (electronic only)	$210	3-year subscription (electronic only)	$210
1-year student subscription	$ 48	1-year student subscription	$ 79
1-year university library subscription	$ 99	1-year university library subscription	$135
2-year university library subscription	$195	2-year university library subscription	$265
3-year university library subscription	$289	3-year university library subscription	$395
1-year institutional subscription	$225	1-year institutional subscription	$259
2-year institutional subscription	$445	2-year institutional subscription	$510
3-year institutional subscription	$650	3-year institutional subscription	$750

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The Stata Journal (2012)
12, Number 3, pp. 565–569

Stata tip 111: More on working with weeks

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

1 Introduction

Researchers in several fields work with weekly data. The most prominent examples as far as Stata is concerned come from business, economics, and finance. In particular, the Federal Reserve Bank of St. Louis offers a repository of many economic time series, some of which are weekly. See http://research.stlouisfed.org/fred2/ and documentation on the Stata command `freduse`. The M2 money stock for weeks ending on Monday and the St. Louis Financial Stress Index for weeks ending on Friday are specific examples.

Like any kind of time-series data, weekly data vary in character. In some cases, the data are totals, such as sales totals. It seems typical in such cases that any within-week variations (including zeros for days when a business is closed and no transactions occur) are not considered interesting or important. In other cases, the data are weekly averages, but weekly cycles are not reported on similar grounds. In yet other cases, weekly values are point values: the underlying variable is continuously or discretely varying but just happens to be reported once a week. These differences do not affect how weekly data are held within Stata, but they do affect interpretations of analyses and how weekly data should be aggregated to longer intervals.

Some advice on dealing with such weekly data was given in a previous Stata tip (Cox 2010). The main purposes of this tip are to further discuss how to convert data presented in yearly and weekly form to daily dates and how to aggregate such data to months or longer intervals.

Kit Baum, Kevin Crow, and David Drukker made helpful comments on the issues discussed here.

2 Previous tip

The previous tip emphasized that

1. Stata's own definition of weekly dates is idiosyncratic, hinging on the idea that January 1 is always the start of the first week in any given year and that there are always precisely 52 weeks in any year. Thus week 52 is either 8 or 9 days long, depending on whether a year is a leap year. Defined this way, weeks always fit into years.

 dm0065

2. any week defined in the more common way as being always 7 days long and so starting, or equivalently finishing, on a particular day of the week can always be defined by that particular day of the week.

3. Stata's `dow()` function is useful for manipulations. It returns 0 for Sundays, 1 for Mondays, and so forth until 6 for Saturdays.

3 Converting year–week data to Stata daily dates

Data for the St. Louis Financial Stress Index can be read into Stata by typing

```
. freduse STLFSI
```

This is an example of easy-to-use weekly data referred to daily dates that are 7 days apart. Such data may be declared time-series data by specifying `delta(7)` with `tsset` or `xtset`. It would not be a good idea to try to declare the dates as Stata weekly dates or even to try to convert them to Stata weekly dates. Weeks defined by beginning or ending days may span two different years, and it is entirely possible for 53 weeks to occur in some years. However, Stata's weekly dates are defined too rigidly to allow either variation.

Yet users often report that their weekly data arrive in years and weeks, such as year 2012, week 31. We need a way of converting that kind of data to equivalent daily dates. We need only the day of the week considered to characterize each week. Whether that day is at the beginning or end of the week (or even in the middle) is for the moment immaterial.

Here is an example. We have a composite variable `yearweek` specifying year and week. We will use Friday as the key day as a first example.

```
. list yearweek

     +----------+
     | yearweek |
     |----------|
  1. |   201148 |
  2. |   201149 |
  3. |   201150 |
  4. |   201151 |
  5. |   201152 |
     |----------|
  6. |   201201 |
  7. |   201202 |
  8. |   201203 |
  9. |   201204 |
 10. |   201205 |
     +----------+
```

If such a variable were a string variable, we could split the components by typing

```
. generate year = real(substr(yearweek, 1, 4))
. generate week = real(substr(yearweek, -2, 2))
```

If such a variable were a numeric variable, we could do it this way:

```
. generate year = floor(yearweek/100)
. generate week = mod(yearweek, 100)
```

Naturally, if we started with separate variables for year and week, neither conversion would be necessary.

The calculation pivots on finding the first Friday in each year. Once we have done that, every other Friday in that year is just a multiple of seven days later. It is usually convenient, if a little wasteful in storage, to start by defining a variable that is the day of the week for each January 1. (If, as an exception, data all lie within a single year, then there is a single value to deal with, and we need not use a variable.)

```
. generate jan1 = dow(mdy(1,1,year))
```

Let's use a local macro to hold the day of the week we want, if only to expose the generality of what we are doing. `dow()` returns 5 for Fridays.

```
. local wanted = 5
```

We can be sure that the first Friday of the year is within the first 7 days of the year. If the first day of the year is in fact Sunday to Friday, then the first Friday is respectively January 6, 5, 4, 3, 2, or 1. If the first day of the year is in fact Saturday, we have to wait until January 7. If we were identifying Mondays, the sequence would be another rotation of the integers 7, ..., 1 (namely, January 2, 1, 7, 6, 5, 4, 3). Experiment or deduction yields the general rule for the first day of interest within the year as

```
. generate weekdates =
> cond(jan1 <= `wanted´, 1 -  jan1 + `wanted´, 8 - jan1 + `wanted´)
```

That is, for another day of the week other than Friday, the definition of `wanted` would change, but the command above would still be valid. See Kantor and Cox (2005) for a tutorial on the `cond()` function.

For the weeks actually present in our data, we need to do the following:

```
. replace weekdates = weekdates +  7 * (week - 1)
(9 real changes made)
```

These are, or rather should be, days of the year somewhere between 1 and 365 or 366, depending on whether each year in question is a leap year. We do not have to agonize about identifying leap years because there is an easy way to identify whether December 31 is day 365 or 366. If the day of the year calculated is more than that, there is an error in the dataset, which we should want to know about.

```
. replace weekdates = . if weekdates > doy(mdy(12, 31, year))
(0 real changes made)
```

Then the conversion to daily dates is straightforward:

```
. generate date = mdy(1,1,year) + weekdates - 1
. format date %td
```

These are daily dates, but as I explained earlier, we can get the best of both worlds by specifying `delta(7)` with `tsset` or `xtset`.

We could now type

```
. drop jan1
```

Many problems are now easier. For example, suppose we wanted to change the characterization of weeks by tagging in terms of first days rather than last days or even by tagging by a middle day (which might seem appropriate for graphics). All such changes are just adding or subtracting a fixed number of days. In what follows, we will take the dates in our example as defining the end of each week.

4 Aggregating weeks to longer intervals

Suppose we want to aggregate weekly values to longer intervals such as months. We will focus on months, because further aggregation of months to quarters, half-years, or years is easy. In practice, aggregation usually entails averaging or summing values, according to the nature of the variable. Aggregation may follow creation of a monthly date by using the standard function `mofd()`, but how to treat weeks that span months remains an issue.

A partial week ends on or before the 6th of any month or begins 6 or fewer days before the end of any month, which means 7 or fewer days before the beginning of any month. If we work with weeks defined by their beginning days, then the end of a month could be day 28, 29, 30, or 31, depending on the month and whether a year is a leap year. In practice, it can be easier to consider the end of the month as being also the day before the first day of the next month, as explained in Samuels and Cox (2012). However, in practice, working with weeks defined by their ending days is even easier, and we just explained how to calculate such a variable if one does not already exist.

Let's fake some data for our example:

```
. set seed 2803
. generate y = ceil(42 * sqrt(runiform()))
```

Then `mofd()` gives us the monthly date corresponding to each daily date:

```
. generate month = mofd(date)
. format month %tm
```

If the data were spot or point data for the dates specified, it is simplest, and may even be best, just to average all data for each month. How might we deal otherwise with the problem of spanning weeks?

First, we will generate the lesser of the day of the month and 7. This will be between 1 and 6 for weeks that end on those days of the month and 7 otherwise. Thus we have a variable that tags those weeks spanning two months and indicates how many days fall in each of the two months (because if x days fall in the second month, $7 - x$ days fall in the first).

```
. generate length = min(day(date), 7)
```

Keeping track of the number of observations in the original dataset, we now need to split each spanning week between two months. The trick is to expand the data first with the `expand` command and then collapse the data with the `collapse` command.

```
. local N = _N
. expand 2 if length < 7
(3 observations created)
. replace month = month - 1 if _n > `N´
(3 real changes made)
. replace length = 7 - length if _n > `N´
(3 real changes made)
. collapse (mean) y (count) days=length [fw=length], by(month)
. list
```

	month	y	days
1.	2011m11	32	5
2.	2011m12	23.09678	31
3.	2012m1	31.12903	31
4.	2012m2	42	3

In this example, the weighted mean of the outcome and the number of the days with observations are produced. `collapse` can produce many other reductions of raw data. `contract` is an alternative for some problems.

References

Cox, N. J. 2010. Stata tip 68: Week assumptions. *Stata Journal* 10: 682–685.

Kantor, D., and N. J. Cox. 2005. Depending on conditions: A tutorial on the cond() function. *Stata Journal* 5: 413–420.

Samuels, S. J., and N. J. Cox. 2012. Stata tip 105: Daily dates with missing days. *Stata Journal* 12: 159–161.