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	
Printed & electronic		**Printed & electronic**	
1-year subscription	$115	1-year subscription	$145
2-year subscription	$210	2-year subscription	$270
3-year subscription	$285	3-year subscription	$375
1-year student subscription	$ 85	1-year student subscription	$115
1-year institutional subscription	$345	1-year institutional subscription	$375
2-year institutional subscription	$625	2-year institutional subscription	$685
3-year institutional subscription	$875	3-year institutional subscription	$965
Electronic only		**Electronic only**	
1-year subscription	$ 85	1-year subscription	$ 85
2-year subscription	$155	2-year subscription	$155
3-year subscription	$215	3-year subscription	$215
1-year student subscription	$ 55	1-year student subscription	$ 55

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* (ISSN 1536-867X) is a publication of Stata Press. Stata, STATA, Stata Press, Mata, MATA, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2015)
15, Number 2, pp. 574–587

Speaking Stata: Species of origin

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

Abstract. Counting and measurement scales often have natural origins that deserve respect in analysis. In other problems, we should feel free to shift or translate the origin or to rotate a periodic scale. Examples with environmental data focus on trends in time and seasonal cycles. Several small tips are bundled together on working with noncalendar years in Stata.

Keywords: st0394, calendar, noncalendar year, graphics, origin, translation, rotation, logit regression, Poisson regression

1 Introduction

Data that are counted or measured are counted or measured relative to some origin, a zero point that defines the start of a scale. If we count how many Stata mugs we own, or measure the size of a *Stata Journal* issue, the origin of zero mugs—or of zero height, width, and thickness—seems natural as well as conventional and convenient.

It is, or should be, standard in statistical science that some origins deserve respect as defining natural limiting or boundary conditions. Such origins are ignored at our scientific and statistical peril. The origins may concern both response variables (outcome or dependent variables, if you will) and predictor variables.

Consider first what can be codified as a principle of *respecting the range of a response*. Thus, if a count response is necessarily zero or positive, any model that could predict negative values is intrinsically suspect, if not downright absurd. That principle usually leads to fitting on a logarithmic scale by using a logarithmic transformation or a logarithmic link function. If a response is a proportion, then any predictions for that response arguably should respect the bounds of zero and one. That principle is satisfied by many suitable link functions, of which logit is currently the most popular in many fields.

The argument is already likely to seem contentious. Essentially, any application of mathematics entails some assumptions that are ridiculous if taken literally. Many geographical calculations depend on the Earth being taken locally as a plane, an approximation known widely and for a long time to be quite wrong. Nevertheless, the approximation is often close enough to the truth to be practical. Questions in classical mechanics often have facetious flavor underlining the point: "An elephant whose mass may be neglected [...]" (Wilkes 1966, 2) conveys the tone. If the other object is the Earth, the neglect is indeed negligible.

In statistical science, we often explain to students that a Gaussian distribution being defined for the entire real line is neither here nor there if the probability of impossible values is too trivial to matter. More generally, any assumption can be adopted with abandon if it eases analysis; we just need to watch out that it does not bite. Thus linear probability models can seem simple and practical, so long as we are careful not to extrapolate too far.

But if some origins seem fixed and natural, others are just conventional and sometimes not even convenient. In particular, often the standard Western calendar is just too arbitrary to be helpful. That is hardly surprising considering its bizarre history as a result of several arbitrary choices.

This column explores such themes with examples posed and poised in time. The examples consider how best to model simple trends (section 2) and seasonal periodicities (section 3). Shifting the origin of the calendar is an easy subtraction, but defining noncalendar years is a little more tricky, so we also gather together the small Stata details (section 4). Some readers might prefer to take the examples as read and skip to the last section if their concern is to get code for their own problem.

2 Trends in time: The case of Atlantic hurricanes

Plotting time series and looking for simple trends is a standard descriptive or exploratory exercise. In many fields, ranging from, say, economics to environmental science, we might have a few years, decades, or centuries of recent data.

The website http://weather.unisys.com is rich in interesting climatic datasets. We first read in one dataset on Atlantic hurricanes. The small tactic here, which I often find highly practical, is to read in a dataset as a single string variable and then extract what is needed using string commands and functions. Here 244 characters happen to be long enough to hold the data comfortably and short enough to fit into a string variable over several versions of Stata.

```
. infix str data 1-244 using
> http://weather.unisys.com/hurricane/atlantic/tracks.atl, clear
(14,348 observations read)
```

Inspection shows that each hurricane has a distinct SNBR, so that lets us reduce the dataset to one observation per hurricane. The other observations give data on each storm as it moves.

```
. keep if strpos(data, "SNBR")
(12,872 observations deleted)
```

The calendar year can be extracted using string functions and finally using `real()` to insist to Stata that it should go into a numeric variable.

```
. generate year = real(substr(word(data, 2), -4, 4))
```

A further reduction to yearly counts then follows.

```
. contract year, freq(frequency)
```

Because the data source may change, the dataset at this point is included in the media for this issue of the *Stata Journal* as `hurricane.dta`.

Informal examination of the data suggests an upward trend, combined, unsurprisingly, with considerable irregularity. The simplest plausible model is provided by Poisson regression, given that the response is counted.

```
. poisson freq year, vce(robust)
Iteration 0:   log pseudolikelihood = -420.99233
Iteration 1:   log pseudolikelihood = -420.99233
Poisson regression                              Number of obs     =        161
                                                Wald chi2(1)      =      44.07
                                                Prob > chi2       =     0.0000
Log pseudolikelihood = -420.99233               Pseudo R2         =     0.0659
```

frequency	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
year	.0043418	.0006541	6.64	0.000	.0030599	.0056237
_cons	-6.188661	1.264591	-4.89	0.000	-8.667213	-3.710109

You can check for yourself that, over the range of the data, negative binomial regression gives a very similar fit. That is also true for a naive linear regression, but the qualification "over the range of the data" is in that case vital: a linear regression predicts an unphysical negative number of storms before about 1698. It would be extremely foolish to use such an extrapolation. For that reason and others, a linear regression is in principle not a good approximation for such data. We proceed with the Poisson for our purposes.

Following a command like `poisson` with a single predictor (here `year`), we can use the convenience command `regplot` (Cox 2004, 2010a) to show data and predicted values. Here we copy the coefficients to the subtitle of the graph:

```
. regplot, subtitle(exp(-6.189 + 0.00434 year)) bands(200)
```

The result is shown in figure 1.

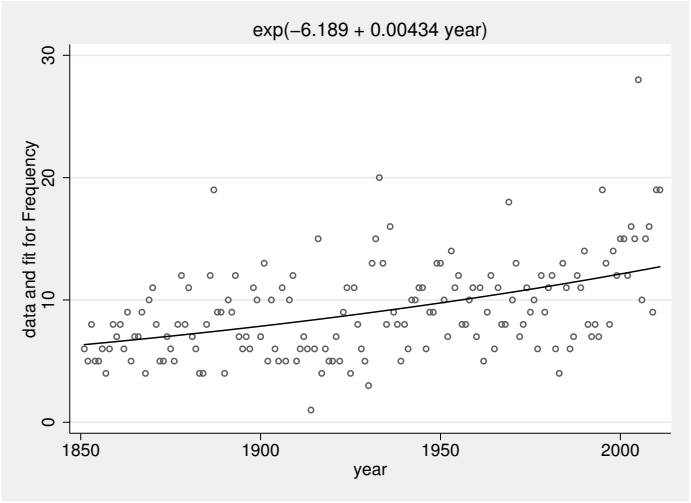

Figure 1. Data and Poisson regression fit for yearly counts of Atlantic hurricanes

The option `bands(200)` encourages the underlying `twoway mspline` commands to stretch the predicted curve to a wider range.

The fit looks helpful, but it is awkward. The intercept for the regression can be explained as $\exp(-6.188661) = 0.00205257$ or, more reasonably, 0.002 hurricanes per year in a notional year 0. This too is an extrapolation. Although many researchers just ignore intercepts when the origin is far outside the range of the data, there is a simpler alternative: translate the origin to a more convenient year. Suppose that we decide on 2000.

```
. generate yM2000 = year - 2000
. poisson freq yM2000, vce(robust)
Iteration 0:   log pseudolikelihood = -420.99233
Iteration 1:   log pseudolikelihood = -420.99233
Poisson regression                              Number of obs     =        161
                                                Wald chi2(1)      =      44.07
                                                Prob > chi2       =     0.0000
Log pseudolikelihood = -420.99233               Pseudo R2         =     0.0659

------------------------------------------------------------------------------
             |               Robust
   frequency |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      yM2000 |   .0043418   .0006541     6.64   0.000     .0030599    .0056237
       _cons |   2.494995   .0529787    47.09   0.000     2.391159    2.598832
------------------------------------------------------------------------------
```

The slope coefficient remains the same, but the intercept is now for (year − 2000) = 0 or, more plainly, year = 2000. A detail of `regplot` is that we can specify a different variable for the x axis, one not used as a predictor in the previous model. (We should do that only when the plot makes sense, but there is nothing incorrect about the syntax for such a choice.)

At the same time, we also flag graphically what the intercept implies.

```
. regplot year, subtitle(exp(2.495 + 0.00434 (year - 2000)))
> xline(2000, lcolor(gs12)) yline(`=exp(_b[_cons])´, lcolor(gs12)) bands(200)
```

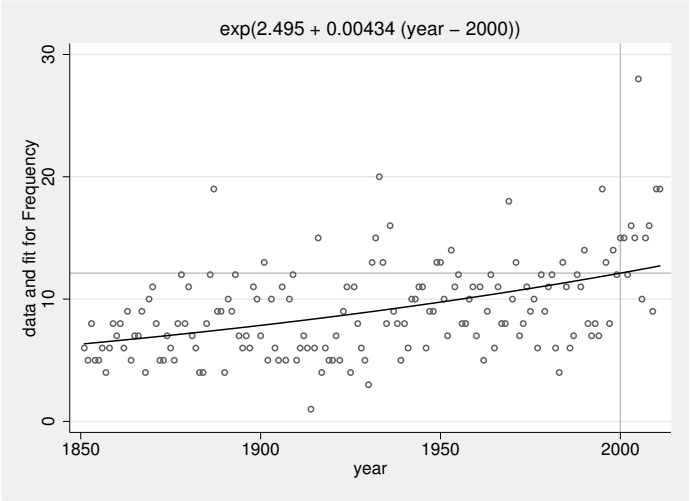

Figure 2. Data and Poisson regression fit for yearly counts of Atlantic hurricanes, but with intercept defined for an origin at year 2000

Figure 2 shows the results. The intersection of the added lines shows the intercept as exp(2.494995), about 12.12, in 2000. Such translation can be useful in teaching of all kinds, including explaining to your researcher colleagues what Poisson regression does.

3 Seasonality of outcome: The case of rainfalls at Shahhat, Libya

We turn to a dataset for Shahhat in Libya containing daily rainfalls for the years 1981–1993 (Ali 1995). Close to Shahhat are the remains of the classical city of Cyrene, birthplace of Eratosthenes. The focus here is on seasonality of rainfall. Although the story will be that Shahhat can be very dry over much of the year, it is also one of the wettest places in Cyrenaica. Over 2,400 years ago, Herodotus commented that the land around Cyrene (and implicitly the climate) allows three harvests annually (*Histories* 4.199; for a modern version, see, for example, Herodotus [2013, 334]).

The rainfall data are included in the media for this issue of the *Stata Journal* as `shahhat_rainfall.dta`.

```
. use shahhat_rainfall, clear
```

One of the variables is a binary indicator `rainday` with value 1 if there was rain and 0 otherwise. Another key variable is day of the year `doy` with value 1 for January 1 through to 366 or 365 for December 31, the latter depending on whether a year was a leap year or not. The variable name here echoes the useful `doy()` function.

We can look at a modest reduction of the raw data to see the general structure of seasonality.

```
. bysort doy: egen prain = mean(rainday)
. scatter prain doy, ytitle(probability of rain) xlabel(1(50)351) msymbol(Oh)
```

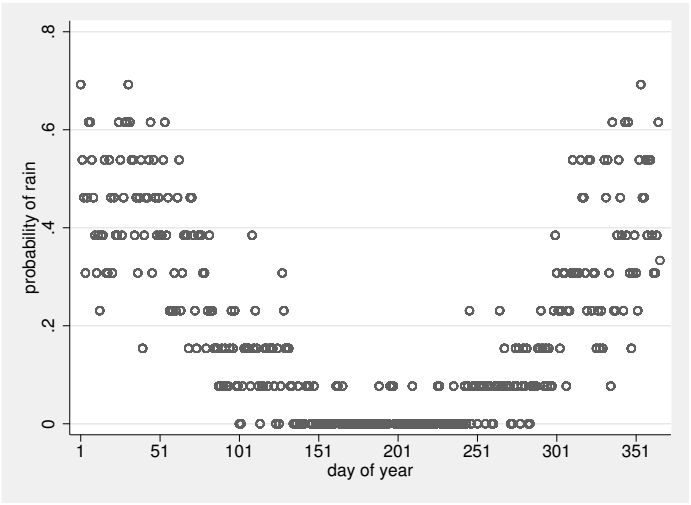

Figure 3. Probability of rain at Shahhat, Libya, for each day of the year

The evident granularity arises because the only possible values in 13 complete calendar years are the fractions 0/13, 1/13, ... The one visible exception is for day 366 with 1/3 days (1 out of 3) with rain on December 31 in leap years 1984, 1988, and 1992. Such detail aside, the clear pattern is of a dry season in summer and a wet season in winter. Once more, therefore, the data show a fairly simple pattern, but using the conventional calendar year produces an awkward graph. The wet season is split inconveniently at the end of the year. In this and other similar problems, the pragmatic advice is instead to cut the year when affairs are least interesting or important.

The solution in this case is thus not a translation but a rotation. We could use an origin for graphical analysis based on a certain day of year (for example, day 180 as start) or a certain calendar date (for example, July 1 as start). It seems clear that rainfall from above is indifferent to calendar conventions used by people down below,

but in many economic and other problems, the calendar has more meaning, so we focus on that kind of choice.

Given daily dates, we split so that daily dates before July 1 are joined to dates in the previous calendar year. Using `ncy` to indicate a noncalendar year, we see that this could be

```
. generate ncy = cond(date < mdy(7,1,year(date)), year(date) - 1, year(date))
```

In this particular dataset, there already is a variable containing the calendar year, but we just used a more general formulation in which the function `year()` is used to extract the year from a daily date variable.

Given this new variable for the noncalendar year, a variable for the new day of year, taking value 1 on July 1 and 365 or 366 on June 30, will be produced by

```
. generate doy2 = date - mdy(7,1, ncy) + 1
```

For our particular example, we need more. The seasonality appears to suggest sinusoidal variation. A previous column gave a tutorial on how to use sine and cosine terms in regression (Cox 2006). A first step is to scale years to unit length. One solution for this could be

```
. generate foy = (doy2 - 0.5) / (365 + (mdy(2, 29, year(date)) < .))
. label variable foy "fraction of year from 1 July"
```

Take this term by term. The numerator is `(doy2 - 0.5)`. Subtracting 0.5 to center each day on its middle is a minute refinement, but the same centering would be more important for (say) monthly or quarterly data. The denominator is 365 plus either 1 or 0, depending on whether we are in a noncalendar year that contains February 29. We do not have to devise code ourselves to detect leap years, but can rely on the `mdy()` call specified returning missing if the implied date did not occur.

So much for the data management. Let us turn to some modeling. Creation of sine and cosine terms is now immediate, and (although this is partly taste, partly judgment) the simplest plausible model for predicting a probability varying seasonally is arguably a logit with a single pair of sine and cosine terms.

```
. generate sine = sin(2 * _pi * foy)
. generate cosine = cos(2 * _pi * foy)
. logit rainday sine cosine
Iteration 0:   log likelihood = -2337.1439
Iteration 1:   log likelihood = -1918.4166
Iteration 2:   log likelihood = -1864.2499
Iteration 3:   log likelihood =  -1863.184
Iteration 4:   log likelihood =  -1863.184
Logistic regression                             Number of obs     =      4,748
                                                LR chi2(2)        =     947.92
                                                Prob > chi2       =     0.0000
Log likelihood =  -1863.184                     Pseudo R2         =     0.2028
```

rainday	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sine	-.4757375	.0605334	-7.86	0.000	-.5943807	-.3570943
cosine	-1.901648	.0773467	-24.59	0.000	-2.053245	-1.750051
_cons	-1.961515	.0567856	-34.54	0.000	-2.072813	-1.850217

A peculiar but unproblematic feature of this kind of model is that the sine and cosine terms should be considered yoked together. (For more on why, see the tutorial just cited.) In any case, we can plot data and predictions on a single graph because sine and cosine are both functions of fraction of the year. The default for `regplot` is to plot the original data for `rainday` as zeros and ones, but it is better to suppress that and to show instead the mean probabilities as a more informative reduction of the data.

```
. regplot foy, ms(none) addplot(scatter prain foy) bands(200)
```

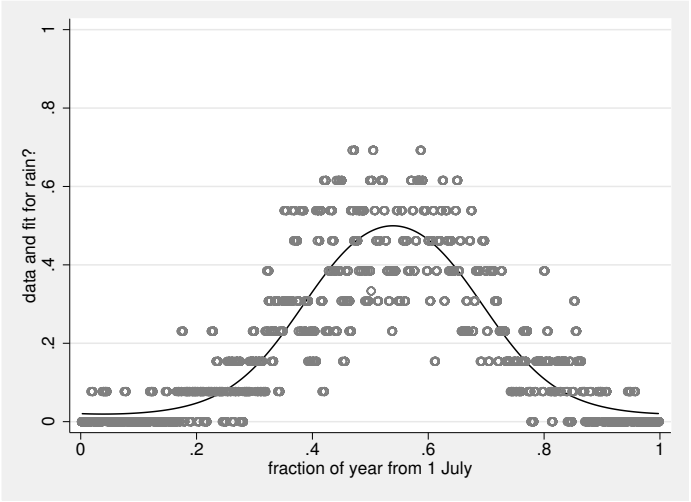

Figure 4. Logit fit giving probability of rain as a sinusoidal function of fraction of year since 1 July

Figure 4 shows that we have a good first approximation straight away. If interested, you could check for yourself that further sine and cosine terms, or even different link functions, do not help very much.

We should be clear on what has and has not been done. The modeling in terms of sines and cosines does not depend crucially on when we start each year, calendar or otherwise. The coefficients would be different, but the fit would be the same. The point of changing the origin is purely graphical—to allow better, or at least easier, visualization of the seasonal cycle.

4 Noncalendar years

As exemplified in the last section, people dealing with time-series data often want to use a framework of noncalendar years. We define a noncalendar year (hereafter NCY) as starting at some date other than the conventional beginning of the Western calendar year (January 1 for daily dates, and equivalent dates for other kinds) and ending a year later, just before the next such date.

Note that the focus here does not extend to calendars other than Western. It is entirely on questions centered on use of the Western calendar but with years starting at some particular date and ending just before. Years will therefore be 365 or 366 days long, depending on whether February 29 is observed within an NCY. We are thus excluding, for example, noncalendar years based on, variously, 52 and 53 weeks in combination.

For other problems and purposes, the large literature on calendars includes a guide to computation in Dershowitz and Reingold (2008) and a compendium of curiosities in Blackburn and Holford-Strevens (1999).

Common examples among Stata users include financial or fiscal years. Sections 2 and 3 arose from my interest in problems with environmental data, especially those directly or indirectly showing climatic or hydrological variations. Here the natural time units are often not calendar years, but years spanning Northern hemisphere winters or Southern hemisphere summers. Thus rain in Libya, snow in Scotland, and drought in Australia are illustrations of phenomena for which going from December to January is of no intrinsic importance and could indeed complicate analyses by splitting a period of real interest in two.

Noncalendar years can be imagined as aggregates of all the smaller subdivisions of time used by Stata, but the possibilities do not seem of equal practical importance. More importantly, the principles will become clear after just a few examples.

In essence, there are just two key suggestions here:

1. Noncalendar years might as well have the values of one of the years that they include. My personal convention is that they have the values of the year in which they start, so that 2010/2011 is assigned the value 2010. Users preferring the opposite or some different convention can adapt the code here. A common, but far from universal, convention in business and finance is indeed the opposite convention that years are defined by the year in which they end.

2. Noncalendar years should be labeled with value labels to indicate their identity. This follows (negatively) from the limited tunability of display formats and (positively) from the flexibility offered by value labels to show exactly what is preferred. Thus users can specify their own styles, which might be `"2010/11"`, `"2010-11"`, `"2010-2011"`, `"FY 2011"`, or whatever. Defining value labels does not obligate their use. Commonly, such value labels might make a graph too busy but be appropriate for a table, and indeed, default options usually run those ways.

If the user already has a variable (say, `year`) holding calendar years, then some calculations are made a little easier. For completeness, solutions are given both for when such a variable exists and for when it does not.

In each case, the calculation splits into code for observations in which the NCY is the same as the calendar year and code for observations in which it differs by one. The `cond()` function is natural for such branching calculations. If it is new to you, a tutorial was given by Kantor and Cox (2005).

If we start with half-yearly dates, the only possibility offered for NCYs is that years start with half 2 and end with half 1, equivalent to running from July to June. With half-yearly dates in variable `hdate` and years in variable `year`, that could be, with an "identified by start" convention,

```
generate ncy = cond(halfyear(dofh(hdate)) == 2, year, year - 1)
```

and without a years variable,

```
generate ncy = cond(halfyear(dofh(hdate)) == 2, yofd(dofh(hdate)),  ///
     yofd(dofh(hdate)) - 1)
```

With the opposite convention of NCYs being identified by their end year, the results would be either `year + 1` or `year`, and so forth. This statement stands for all later examples too.

Quarterly dates offer 3 possibilities for NCYs, namely, starting in quarter 2, 3, or 4. The case for quarter 2 as start given quarterly date `qdate` is

```
generate ncy = cond(quarter(dofq(qdate)) >= 2, year, year - 1)
```

or

```
generate ncy = cond(quarter(dofq(qdate)) == 2, yofd(dofq(qdate)),  ///
     yofd(dofq(qdate)) - 1)
```

with similar code for 3 and 4.

A pattern is becoming evident. NCYs based on monthly dates `mdate` may start in months 2 through 12, and again, month 2 can stand proxy for the others.

```
generate ncy = cond(month(dofm(mdate)) >= 2, year, year - 1)
generate ncy = cond(month(dofm(mdate)) >= 2, yofd(dofm(mdate)),  ///
     yofd(dofm(mdate)) - 1)
```

The strong implication of detailed treatment of weeks in Cox (2010b, 2012a,b) is that users with weekly dates are best advised to work with daily dates defining weeks, say, by the days starting each week, so we skip to daily dates.

Daily dates are trickier: the occurrence of leap years is sufficient to ensure that. For that reason alone, the use of the `doy()` function, the equivalent of the `halfdate()`, `quarter()`, and `month()` functions, would afford scope for off-by-one errors and so is avoided here. It seems simpler to phrase calculations in terms of `mdy()`. (Note that the `day()` function defines day of month, not day of year.)

As in all previous cases, the calculation is split in two. The first part of the year includes days on or after the start date and in the same calendar year. The second part includes those days before the start date in any year. We choose a start date of March 1 as a messy example: it implies an end date of February 29 whenever the following calendar year is a leap year and of February 28 otherwise. Assuming as before a daily date variable `ddate`, we could use

```
generate ncy = cond(ddate >= mdy(3, 1, year(ddate)), year, year - 1)
```

or

```
generate ncy = cond(ddate >= mdy(3, 1, year(ddate)),  ///
     year(ddate), year(ddate) - 1)
```

The complication of leap years with this approach is no complication for this example because either February 29 or February 28 qualifies as earlier than March 1, regardless of which is the last day in an NCY.

This is not the messiest example that can be imagined. Anyone choosing February 29 as a start date would need to use another date instead in nonleap years, presumably March 1. That choice requires more complicated code; even so, it can still be phrased in one line. Choosing just one of the last pair of code statements, we could have

```
generate ncy = cond(ddate >= min(mdy(2, 29, year(ddate)),  ///
     mdy(3, 1, year(ddate)), year, year - 1)
```

The `min()` function here chooses the right day within each year:

1. If `mdy(2, 29, year(ddate))` is missing, then `mdy(3, 1, year(ddate))` is selected as smaller, following the usual Stata logic that numeric missings are regarded as arbitrarily large.

2. If `mdy(2, 29, year(ddate))` is not missing, then it is selected as smaller because it comes first.

However, the statements so far are not careful enough if a year date variable such as `year` is not missing, but the half-year, quarterly, monthly, or daily date is missing. Careful code would therefore need to include qualifiers such as

```
if hdate < .
```

to trap such instances. The more complex code not assuming a year date variable will not need such qualifiers.

Date-times with times within days in the simplest cases would be classified according to their daily dates, so we just need to use `dofc()` or `dofC()` first to extract that component. Robert Picard alerted me to cases in which years are deemed to end at a particular time within a day, but such more challenging examples are happily left as an exercise.

Let us move now to value labels. Now that we have a variable `ncy`, we can label its values. In practice, we can set up a loop over the range of values present in the data. Here are two examples, both to follow

```
summarize ncy, meanonly
```

and to precede

```
label values ncy ncy
```

The use of the `meanonly` option was discussed in Cox (2007): despite its name, it returns other results as well as the mean, notably here the minimum and maximum. In the first example, we want value labels like `"2010/2011"` and can range from the minimum to the maximum year observed:

```
forvalues y = `r(min)´/`r(max)´ {
     local Y = `y´ + 1
     label def ncy `y´ "`y´/`Y´", modify
}
```

In the second example, we want value labels like `"2010/11"` (but also `"2008/09"`). Wanting to see `"01"`, ..., `"09"`, `"10"`, and so forth, was discussed in Cox (2010c).

```
forvalues y = `r(min)´/`r(max)´ {
     local Y: display %02.0f mod(`y´ + 1, 100)
     label def ncy `y´ "`y´/`Y´", modify
}
```

The versatile function `mod()` was discussed in Cox (2007). Such a loop would also produce `"1999/00"`, which looks odd, and could be trapped in turn:

```
forvalues y = `r(min)´/`r(max)´ {
     local Y = `y´ + 1
     if mod(`Y´, 100) local Y: display %02.0f mod(`Y´, 100)
     label def ncy `y´ "`y´/`Y´", modify
}
```

Here `mod(‘Y’, 100)` is true (nonzero) when and only when a year with value given by `‘Y’` is not divisible by 100.

If there are gaps in the series of NCYs between the minimum and the maximum observed, then some value labels will not be used, which in practice should be a trivial storage problem. In fact, we might need such value labels for display on graph axes, which is reason enough to define them. That points up a small problem: graph axes

might show noncalendar years beyond the range of the data, for which value labels or equivalent text could be defined ad hoc.

Note, however, that in some cases, a specified display format will suffice. For example, a prefix `"FY "` is easy to add.

```
. display %ty!F!Y_CCYY 2011
FY 2011
```

The last word belongs to Clyde Schechter, who independently posted on Statalist a neat solution to the main problem here after this column was first submitted. Although these transformations are in effect a set of rotations, there is, for half-yearly, quarterly, and monthly problems, an equivalent simple translation. Consider starting the year in half-year 2, quarter 2, or month 2. We can have

```
. generate ncy = yofd(dofh(hdate  - 1))
. generate ncy = yofd(dofq(qdate  - 1))
. generate ncy = yofd(dofm(mdate  - 1))
```

and more generally the number to subtract is just the number of the starting period minus 1. (For June, month 6, it would be 5, and so forth.) The idea does not carry over so simply to daily dates, because of leap years, but is more appealing than most of the previous solutions.

5 Conclusion

Species of origin include those that should be considered fixed, and so should be respected in analysis, and those that may be considered mutable according to convenience. We have seen examples of both kinds.

For a counted response, Poisson regression is a natural first approximation, as is logit regression for a binary response that we use to predict probabilities. We therefore respect the origin of zero for counts and the limits of zero and one for probabilities using such models.

For trends dependent on calendar year, year 0 will often be so far outside the data range as to be irrelevant. This problem is usually ignored, but it can be solved by translation. For seasonality dependent on time of year, the conventional calendar can cut the interesting season in twain. That problem lends itself to rotation.

6 Acknowledgments

Interesting conversations with Robert Picard and Patrick Royston have some echoes within. A neat Statalist solution by Clyde Schechter came just in time to add value to section 4.

7 References

Ali, G. M. 1995. "Water erosion on the northern slope of Al-Jabal Al-Akhdar of Libya". PhD thesis, University of Durham. http://etheses.dur.ac.uk/1035/1/1035.pdf.

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

Cox, N. J. 2004. Speaking Stata: Graphing model diagnostics. *Stata Journal* 4: 449–475.

———. 2006. Speaking Stata: In praise of trigonometric predictors. *Stata Journal* 6: 561–579.

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

———. 2010a. Software update: gr0009_1: Speaking Stata: Graphing model diagnostics. *Stata Journal* 10: 164.

———. 2010b. Stata tip 68: Week assumptions. *Stata Journal* 10: 682–685.

———. 2010c. Stata tip 85: Looping over nonintegers. *Stata Journal* 10: 160–163.

———. 2012a. Stata tip 111: More on working with weeks. *Stata Journal* 12: 565–569.

———. 2012b. Stata tip 111: More on working with weeks, erratum. *Stata Journal* 12: 765.

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

Herodotus. 2013. *The Histories*. Translated by Tom Holland. Introduction and notes by Paul Cartledge. London: Penguin.

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

Wilkes, M. V. 1966. *A Short Introduction to Numerical Analysis*. London: Cambridge University Press.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He was an author of several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*. His Speaking Stata articles on graphics from 2004 to 2013 have been collected as *Speaking Stata Graphics* (College Station, TX: Stata Press, 2014).