Rolling regressions with Stata

Christopher F Baum
Boston College*

August 11, 2004

1 Introduction

In this paper, we consider the creation of a Stata time–series routine to compute rolling or moving–window regression estimates. Although Stata contains a command to compute statistics for subsamples—`tabstat`—it cannot be coerced to deal with overlapping subsamples. That is, `tabstat` works like any Stata command prefixed with `by:`, so that if one defines each twelve months of a monthly series as one element of a by–group, `tabstat` will handle the task of computing annual statistics very nicely. But it will not deal with computing statistics from a sequence of by–groups that are formed by a "moving window" with, e.g., eleven months overlap. By the same token, one may indeed compute regression estimates for non–overlapping subsamples via official Stata's `statsby`; but that command is not capable of dealing with overlapping subsamples, as that would correspond to the same observation being a member of several by–groups.

2 Design considerations

The challenge in devising a rolling or moving–window regression routine is not in the necessary computations, nor even in the programming: it lies in providing a user interface that will allow the researcher to specify, in some comprehensible form, what she would like calculated for each crank of the window. The new routine `rollreg` [`findit rollreg`] provides that functionality with logic borrowed heavily from a RATS routine originally authored by

*I am also indebted to Nicholas J. Cox for editorial suggestions, and participants in the 10th UK Stata Users Group meetings for their comments on an earlier version of the subject of this paper.

Simon van Norden at the Bank of Canada (and available from the web–based SSC archive in that format).

The first concern with a moving–window estimation routine: how should the window be designed? One obvious scheme would mimic Baum and Cox' `mvsumm` and `mvcorr` routines [moving summarize and moving correlation: `findit mvsumm` and `findit mvcorr`], and allow for a window of fixed width that is to be passed through the sample, one period at a time: the `move(#)` option. (One could imagine something like a 12–month window that is to be advanced to end–of–quarter months, but that could be achieved by merely discarding the intermediate window estimates). But there are also applications in which an "expanding window" is desired: that is, starting with the first τ periods, compute a set of estimates that consider observations $1 \ldots (\tau + 1)$, $1 \ldots (\tau + 2)$, and so on. This sort of window corresponds to the notion of the information set available to an economic agent at a point in time (and to the scheme used to generate instruments in a dynamic panel data model (cf. `xtabond`). Thus, `rollreg` also offers that functionality via its `add(`τ`)` option. For completeness, the routine also offers the `drop(`τ`)` option, which implements a window that initially takes into account the last τ periods, and then expands the window back toward the beginning of the sample. This sort of moving–window estimate is useful in considering the usefulness of past information in generating an *ex ante* forecast, using a greater or lesser amount of that information in the computation. One of these three options must be provided when executing `rollreg`.

A further choice need be made: a moving–window regression will generate sequences of results corresponding to each estimation period. From the design standpoint, should those sequences be stored in columns of a matrix (which perhaps make them easier to present in tabular format) or as additional variables in the current dataset (which perhaps make them easier to include in computations, or in graphical presentations à la `tsgraph` or `tsline`)? The latter, on balance, seems handier, and is implemented in `rollreg` via the mandatory `stub(`*string*`)` option, which specifies that new variables should be created with names beginning with *string*.

3 Examples

As an illustration:

```
* rollreg_X.do    24jun2004 CFBaum
* Program illustrating use of rollreg
webuse wpi1, clear
g t2 = t^2
rollreg D.wpi t t2, move(24) stub(wpiM) graph(summary)
rollreg D.wpi t t2, add(24) stub(wpiA) graph(summary)
rollreg D2.wpi LD.wpi LD2.wpi t, move(48) stub(wpiM2) robust graph(full)
```

4 Use in a panel context

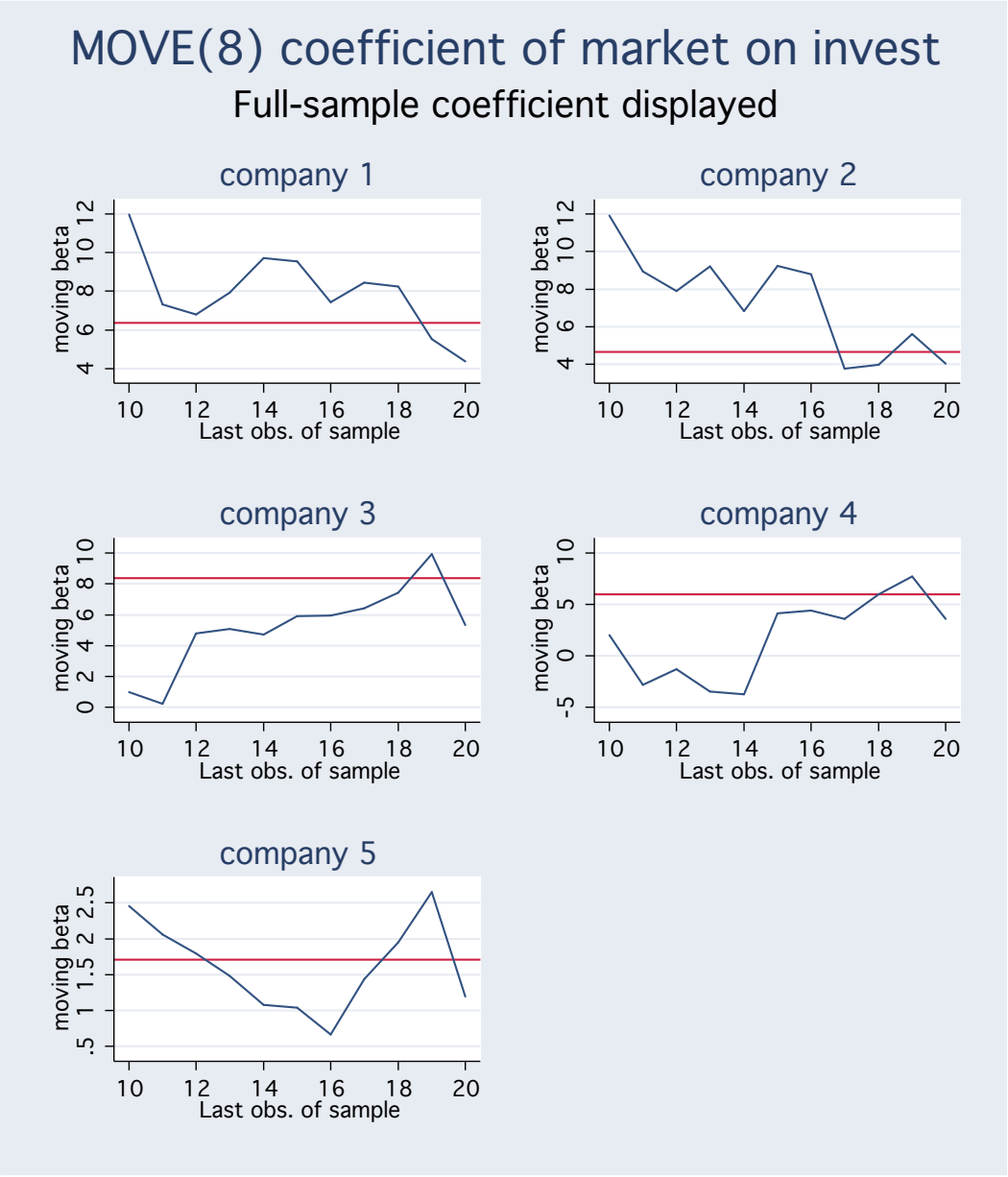
All of the features of `rollreg` are accessible in a panel–data context when applied to a single time series within the panel via an `if` or `in` qualifier. However, rolling regressions certainly have their uses in a panel context. For instance, a finance researcher may want to calculate a "CAPM beta" for each firm in a panel using a moving window of observations, simulating the information set utilized by the investor at each point in time. Therefore, `rollreg` has been enhanced to operate properly on a panel of time series, where the same sequence of rolling regressions are computed for each time series within the panel. In this context, graphical output is not available. As an illustration of `rollreg` in a panel context:

```
webuse invest2, clear
tsset company time
rollreg market L(0/1).invest time, move(8) stub(mktM)
```

Although `rollreg` does not produce graphics when multiple time series are included from a panel, it is straightforward to generate graphics using the results left behind by the routine. For instance, after running the commands above, we may use:

```
* rollreg_X2.do    11aug2004 CFBaum
* Program illustrating use of rollreg on panels
local dv `r(depvar)'
local rl `r(reglist)'
local stub `r(stub)'
local wantcoef invest
local m "`r(rolloption)'(`r(rollobs)')"
forv i=1/5 {
qui reg `dv' `rl' if company==`i'
local cinv = _b[`wantcoef']
tsline `stub'_`wantcoef' if company==`i' & `stub'_`wantcoef'<., ///
ti("company `i'") yli(`cinv') yti("moving beta") ///
name(comp`i',replace) nodraw
local all "`all' comp`i' "
}
graph combine `all', ti("`m' coefficient of `dv' on `wantcoef'") ///
ysize(7) xsize(6) col(2) ///
t1("Full-sample coefficient displayed") saving("`wantcoef'.gph",replace)
graph export rollreg.pdf, replace
```

producing the graph


MOVE(8) coefficient of market on invest
Full-sample coefficient displayed
company 1
company 2
company 3
company 4
company 5
moving beta
Last obs. of sample
10
12
14
16
18
20

5 Digression: moving correlation

It might be useful to note that one of the features of the `rollreg` routine—the ability to compute a moving correlation between two series (e.g., a regression coefficient on standardized data), or a moving autocorrelation for a single series—can actually be handled quite nicely by the `mvcorr` routine of Baum and Cox. This routine allows one to compute a moving–window correlation between two series: useful in finance, where the computation of an optimal hedge ratio involves the computation of just such a correlation—for instance, between spot and futures prices of a particular commodity. And since `mvcorr` requires `tsset`, and thus supports time–series operators, it will allow the computation of moving autocorrelations: e.g., `mvcorr invest L.invest, win(5) gen(acf) end` will specify that the first sample autocorrelation of an investment series should be computed from a five–period window, aligned with the last period of the window (via option `end`) and placed in the new variable `acf`. Like `mvsumm`, it will operate automatically on each time series of a panel. As an example of its use:

```
* mvcorr_X.do    24jun2004 CFBaum
* Program illustrating use of mvcorr
webuse grunfeld, clear
drop if company>4
mvcorr invest mvalue, win(5) gen(rho)
forv i=1/4 {
tsline rho if company==‘i’, nodraw ti("Firm ‘i’") name(comp‘i’,replace)
local g "‘g’ comp‘i’"
}
graph combine ‘g’, ti("Investment vs Market Value: Moving Correlations by Firm")
```