SUGUK 2004 invited lecture: Topics in time series modeling with Stata

Christopher F Baum
Boston College*

July 21, 2004

1 Stata: the language of choice for time series analysis?

Over time, Stata has come to incorporate more and more features for effective analysis of time series data: whether pure time series, or panel data with emphasis placed on the time series dimension of the panel. In this context, I must stress that 'time series modeling' should be taken in the broad sense, referring to (possibly multivariate) models built with data organized as time series, rather than the narrow sense of "Box–Jenkins" or ARIMA univariate time series models. I only consider modeling performed in the time domain (rather than the frequency domain).

Prior to Stata 7, support for time series analysis was weak, since Stata lacked the notion of a time series calendar. In working with time series data, the researcher wants to refer to observations in terms of calendar time, and see dates displayed on the statistical output and in graphs. Stata 7 incorporated a time series calendar and added a bewildering array of date functions which, if properly employed, can decompose dates into their components (e.g., the calendar month associated with a particular date) and translate dates between the several supported data frequencies.

At the same time, the time series operators (`L., D., F.`) were introduced, bringing a tremendous simplification to any programming involving time series, and a built–in mechanism ensuring that inappropriate computations are not made. For instance, `x[_n-1]` will always refer to the previous observation on `x`, which may or may not belong to the prior period. `L.x` will unambiguously refer to the prior period's value. One may succinctly refer to a set of lagged (or led) values with a numlist: `L(1/4).x` to specify that four lags are to be included in a regressor list. One may even combine the operators: e.g., the lagged second difference in `x` is denoted `LD2.x`, while the second lag of Δx is `L2D.x`.

Reliance on this housekeeping becomes overwhelmingly important in working with panel data, in

*I am very grateful to Dr Mustafa Caglayan and the Department of Economics at the University of Leicester for their hospitality. Most of this lecture was prepared in Leicester during a visit to the department in May and June 2004.

which one must always be concerned with staying within bounds of an individual unit's time series. In the context of an unbalanced panel, Stata's approach to housekeeping is far superior to that of a matrix language such as GAUSS or MATLAB, and places much less of a burden on the researcher's keeping track of those details. Since Stata refers to observations by their associated date (after a time series calendar has been established by `tsset`) rather than by their number, users' references to the data may be in the more natural context of specifying calendar dates rather than calculating observation numbers: e.g., `regress inv L(1/4).gdp if tin(1968q4,1979q3)` will restrict the sample to that range of dates.

A second advantage, from the programmer's standpoint, is that the nature of Stata's data transformation commands (`generate, replace, egen`) make it feasible in many instances to perform a transformation over the individual time series of a panel with little overhead. In a number of routines discussed below this feature has been used to advantage to greatly simplify the code, and make a routine more generally useful.

Nevertheless, the facilities now present in Stata for management of time series data have their weaknesses. It is inordinately difficult for many users to transform dates generated in some other software (e.g., Excel) to the Stata date format without mucking around with substrings, wrangling with two–digit vs. four–digit years, etc. Stata does not support business–daily data, and for those of us in economics and finance, that is a pity. It is most unfortunate to give up the advantageous features of Stata's calendar and time series operators when working with this common data format. Some of Stata's most common commands lack support for time series operators: e.g. `xtreg`, which unlike most `xt` commands, requires the generation of lagged or differenced series. And my pet peeve: many time series statistical commands cannot be coerced to operate on a single time series of a panel, although that should be a perfectly legitimate computation. I will discuss and illustrate some of the routines that I have authored or coauthored to deal with some of these issues.

In this talk, I will discuss a number of Stata's capabilities in the area of time series modeling, including data management and graphics (although graphics is far from my area of expertise; I'll leave that to Vince Wiggins and Nick Cox, both experts in the field). I will focus on a number of user–contributed routines, some of which have found their way into official Stata, with others likely to follow. Given time limitations, there are certain areas I will not cover in this discussion: vector autoregressions and structural VARs (a laudable and very ambitious addition to Stata 8); ARCH and GARCH modeling (another very well–implemented and powerful modeling tool in Stata 8); cointegration tests (available via Patrick Joly's `johans` and `vececm` routines, and soon perhaps from official Stata) nor panel unit root tests (several of which I have (co–)authored).

2 Data management and graphics

2.1 tsmktim

Properly defining a time–series date variable, or the time–series component of a panel date variable

```
tsmktim datevar, start(1970)
```

```
tsmktim datevar, start(1970q2)
tsmktim datevar, start(1970m5)
tsmktim datevar, start(1jul1970)
tsmktim datevar, start(1970q2) seq(ind)
```

This quite simple routine will extract the date from the `start` argument, classify the data frequency, generate the appropriate series, assign that frequency's format and perform `tsset datevar`. The last example above handles the case when there are some non–consecutive observations, as identified by the `ind` series, which will then be used to place the proper gaps in the data. But what if we have a panel, and want to identify each unit's timeseries as beginning in 1970? A revision of `tsmktim` earlier this month brought that capability: one may now command

```
tsmktim datevar, start(1970) i(country)
```

in order to achieve that goal, having the result of `tsset country datevar` to define both a panel variable and a time variable. Like most of the routines I will discuss, `tsmktim` is available from the SSC archive which I maintain via official Stata's `ssc` command (in versions 7 and 8.x).

2.2 egen functions useful for time series data

- `egen, filter` from Nick Cox's `egenmore` package. That routine has the flexibility to compute any linear filter (including two–sided filters), with the option of automatically scaling the weights to unity. For instance,

  ```
  egen filty = filter(y), l(-2/2) c(1 4 6 4 1) n
  ```

  specifies that a two–sided centered/centred moving average be computed, with weights 1/16, 4/16, 6/16, 4/16, 1/16. The `n` option specifies that the weights are to be normalized/normalised (dividing by their sum of 16). As an illustration of Stata's flexibility with time series data, note that `egen, filter` may readily be applied to panel data (those which have been defined as a panel to Stata via `tsset panelvar datevar`). This same `egen` command could be employed in that context, and the filter would then be automatically applied separately to each timeseries within the panel.

  Several other functions in the `egenmore` package provide useful housekeeping tools:

- `eom()` will generate a new variable with the date of the End of Month for a given month and year

- `bom()` provides the same functionality for the Beginning of Month.

- the `record()` function: e.g.

  ```
  egen maxtodate = record(wage), by(id) order(year)
  ```

  ```
  egen hiprice = record(share_price), by(firm) order(quote_date)
  ```

  where the first example identifies the highest wage to date in a worker's career (related, perhaps, to her "reservation wage"), while the second identifies the highest price received to date for each firm's shares.

2.3 tsspell

A last utility routine that might be just the trick for many users' needs is Nick Cox' `tsspell`. Beyond the notion of spells of illness, spells of unemployment, etc. we often wish to identify spells in which some characteristic is present: e.g., a period during which a share price does not decline. Spells may be used to advantage defined on the presence of changes in a variable (e.g., the Bank of England changing the base rate); by the occurrence of a particular event (such as a general election, or a natural phenomenon such as an earthquake); or by the presence of some condition (e.g., the period during which Labour forms the government, or those quarters in which the sign of the change in real GDP is positive). Like the current version of `tsmktim`, `tsspell` will automatically handle data which have been defined as a panel, generating spells for each unit in the panel.

2.4 Modified time series routines

Trivial modifications of official Stata commands that permit those commands to operate on single time series within a panel: `panelauto`, providing various autocorrelation routines, and `panelunit`, providing a number of unit root tests.

Researchers often work with "long–format" panel data involving time series of non–trivial length: e.g., quarterly GDP for the G–8 economies, or a set of monthly exchange rates between USD and a set of other currencies. In this context, we wish to evaluate the time series properties of each unit's time series, either in raw data form or in terms of regression residuals (from, for instance, a fixed–effects model). Annoyingly, most of official Stata's diagnostic commands will not work in this format even when an `if` or `in` qualifier is used to restrict the scope of the command to a single time series. These packages rectify that limitation. With these routines, one might readily generate a set of tests for each time series in a panel and array the test results for evaluation and presentation. For example:

```
webuse grunfeld, clear
mat def comp_stat = J(10,5,0)
qui forvalues i=1/10 {
reg invest L.invest mvalue time if company==‘i’
mat comp_stat[‘i’,1] = e(r2)
archlm2
mat comp_stat[‘i’,2] = r(arch)
mat comp_stat[‘i’,3] = r(p)
bgodfrey2, lag(2)
mat comp_stat[‘i’,4] = r(chi2)
mat comp_stat[‘i’,5] = r(p)
local rn "‘rn’ comp_‘i’"
 }
mat rownames comp_stat = ‘rn’
mat colnames comp_stat = r^2 ARCH p-value BG(2) p-value
mat list comp_stat, format(%9.3f) ///
 ti("Regression statistics for Grunfeld data") noheader
```

2.5 tsgraph

Provides analogous capability to Stata 8.2's `tsline` for Stata 7 users (plus a bit more, in terms of automatically handling up to four time series from a panel).

3 Statistics/econometrics

3.1 Capabilities of arima

Stata's `arima` command performs estimation of univariate *ARIMA(p,d,q)* models, or "Box–Jenkins" models with an *AR(p)* autoregressive component, a *MA(q)* moving average component, and requiring d^{th} order differencing to achieve covariance stationarity—and much, much more.The use of the term `arima` for this command seems to lead to many users overlooking its potential usefulness for a wide variety of time series modeling tasks.

`arima` may be used to fit an ordinary regression model—of y on a set of regressors X—in which the disturbances are modeled as an *ARMA(p,q)* process. Unlike `prais`, which fits a regression model with *AR(1)* errors, `arima` is capable of fitting a general error structure to any regression model, including those containing one or more lags of the dependent variable, via maximum likelihood and the Kalman filter.

`arima` provides the ability to compute dynamic forecasts of a regression model that contains a lagged dependent variable. In the context of a regression model with strictly exogenous regressors (that is, those whose distributions are independent of the error process), *ex ante* or out–of–sample predictions may be calculated via `predict` for any post–sample period for which the regressors are available. If the regression model contains a lagged dependent variable, an *ex ante* prediction may only be made for the period for which the dependent variable is available, and `predict` will compute a one–step–ahead static forecast: that is, $\hat{y}_t = X_t\hat{\beta}$, where one of the columns of X is y_{t-1}. For many purposes, a sequence of static forecasts is appropriate; for instance, mimicking the decisions of economic agents at each point in time over the forecast horizon, where their information set contains all variables dated $T + \tau$ and earlier, and they seek to forecast $y_{T+\tau+1}$. But if we are building a dynamic model for y, we may want to simulate the performance of that model over a horizon $(T + 1) \ldots (T + \kappa)$, where the initial conditions include information dated T and earlier, and the further evolution of y over the forecast period is purely determined by the model (possibly inclusive of stochastic shocks, in a stochastic simulation). To construct the dynamic or recursive forecasts of y implied by this mechanism, we must estimate the model with `arima`—even if we do not specify an *ARMA(p,q)* error structure—and use the `dynamic()` option on the subsequent `predict` command. To compare the two forecasting strategies, consider:

```
webuse friedman2, clear
label var pc92 "Real Personal Consumption"
arima pc92 L.pc92 L(0/1).m2 if tin(,1981q4)
* static (one-step-ahead) 20-quarter forecast
```

```
predict consump_st if tin(1982q1,1986q4)
* dynamic (recursive) 20-quarter forecast
predict consump_dyn if tin(1982q1,1986q4), dynamic(q(1982q1))
tsline pc92 consump_st consump_dyn if tin(1982q1,1986q4)
```

3.2 ivreg2

There are a number of features in the latest version of `ivreg2` (Baum, Schaffer and Stillman, 2003) that are relevant for the analysis of time series and panel data models. `ivreg2` can now perform GMM estimation to deal with arbitrary departures from independence of the errors (e.g., serial correlation), both in conjunction with the heteroskedasticity–robust ("White") component and in stand–alone form. The former capability allows `ivreg2` to efficiently estimate models with HAC (heteroskedasticity– and autocorrelation–consistent) standard errors, and for the specification of the kernel from a list of eight choices (with Bartlett, à la Newey–West, as default).

3.3 Unit root tests

3.3.1 dfgls

The "DF–GLS" (Dickey–Fuller Generalised Least Squares) approach of Elliott, Rothenberg, Stock is preferred by many time series econometricians to the "first–generation", more widely known tests of Dickey and Fuller (`dfuller`) or Phillips and Perron (`pperron`). Inferences drawn from the DF–GLS test are likely to be more robust than those based on the first–generation tests.

3.3.2 kpss and other fractional integration tests

In contrast, the KPSS (Kwiatkowski, Phillips, Schmidt and Shin, 1992) test utilizes the (perhaps more natural) null hypothesis of stationarity, or $I(0)$, rather than the Dickey–Fuller style null hypothesis of $I(1)$ or nonstationarity in levels (difference stationarity). The KPSS test (command `kpss`) is also often used (in conjunction with, e.g., `dfgls`) to detect "long memory" or fractional integration: a non–integer value of the integration parameter which implies that the series is neither $I(0)$ nor $I(1)$, but $I(d), 0 < d < 1$. A number of other user–written routines examine this prospect for single (and in some cases multiple) time series: e.g., `gphudak`, `modlpr` and `roblpr` (Baum and Wiggins, 2000b).

3.3.3 Tests allowing for structural change

A well–known weakness of the "Dickey–Fuller" style unit root test, with $I(1)$ as a null hypothesis, is their potential confusion of structural breaks in the series as evidence of nonstationarity. Many econometricians have attempted to deal with this confusion by devising unit root tests that allow

for some sort of structural instability in an otherwise deterministic model. One test of this nature, devised by Zivot and Andrews (1992), allows for a single structural break in the intercept and/or the trend of the series, as determined by a grid search over possible breakpoints, and then conducts a Dickey–Fuller style unit root test conditional on the series inclusive of the estimated optimal break(s). I have made the `zandrews` test, translated from RATS code, available in Stata.

One obvious weakness of the Zivot–Andrews strategy, as well as similar tests proposed by Perron and Vogelsang (1992), is their inability to deal with more than one break in a time series. Clemente, Montañés and Reyes (1998) proposed tests that would allow for two events within the observed history of a time series: either additive outliers (the AO model, which captures a sudden change in a series) or innovational outliers (the IO model, allowing for a gradual shift in the mean of the series). This taxonomy of structural breaks follows from Perron and Vogelsang's work (1992), but in this paper those authors dealt with series including a single AO or IO event. The single–break additive outlier model may be written as (Baum, Barkoulas and Caglayan (1999)):

$$y_t = \delta DT_{bt} + y_{t-1} + w_t$$

with $DT_{bt} = 1$ for $t = T_b + 1$ and 0 otherwise, under the null hypothesis of a unit root. T_b is the breakpoint, to be located by grid search. Under the alternative hypothesis,

$$y_t = c + \delta DU_t + v_t$$

where $DU_t = 1$ for $t > T_b$ and 0 otherwise. This specification nests the null hypothesis in the case that the distribution of v_t may be factored into a unit root and a stationary ARMA process. The testing strategy is then to estimate the regression

$$y_t = \mu + \delta DU_t + \tilde{y_t}$$

the residuals of which are then regressed on their lagged values, a number of lagged differences and a set of dummy variables needed to make the distribution of the test statistic tractable:

$$\tilde{y_t} = \sum_{i=1}^{k} \omega_i DT_{bt-i} + \alpha \tilde{y_{t-i}} + \sum_{i=1}^{k} \theta_i \Delta \tilde{y_{t-i}} + e_t$$

The coefficient on the lagged value is analogous to the Augmented Dickey–Fuller coefficient, and will be significantly less than unity if the null may be rejected.

The equivalent model for the innovational outlier (gradual change) model expresses the shock (the effect of δ above) as having the same ARMA representation as other shocks to the model, leading to the formulation

$$y_t = \mu + \delta DU_t + \phi DT_{bt} + \alpha y_{t-1} + \sum_{i=1}^{k} \theta_i \Delta y_{t-i} + e_t$$

where again an α coefficient significantly less than unity will provide evidence against the $I(1)$ null hypothesis.

In each of these models, the breakpoint T_b and the appropriate lag order k are unknown. T_b is located via a grid search, while k is determined by a set of sequential F–tests. These models are the Perron–Vogelsang single–break forms of the AO and IO schemes; they have been generalized

by Clemente et al. to double breaks by introducing DU_{1t} and DU_{2t}, which by analogy are defined as step functions for two breakpoints, T_{b1} and T_{b2}.

I have constructed a draft set of Stata routines (`clemao1, clemao2, clemio1, clemio2`) that implement the AO and IO models for one and two structural breaks.

It might be very useful to compare results from `dfuller` with those forthcoming from the `clem...` routines; if the latter routines provide evidence of significant additive or innovational outliers in the time series, then results derived from `dfgls` are placed in doubt, since this is evidence that the ADF model is clearly misspecified by the omission of relevant explanatory variables.

3.4 Calculating statistics from moving–window samples

A natural concern in the presence of structural change might be the degree to which descriptive statistics of a particular series are time–dependent. Covariance stationarity of a time series requires, in simple terms, that the mean and variance of the series are time–invariant, and that the remaining elements of the autocovariance function of the time series are also constant over time. A broader notion of stationarity requires that the entire distribution function is time–invariant: e.g., the degree of skewness or kurtosis present in the series should also be fixed over time. 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. The `mvsumm` routine of Baum and Cox deals with this task. It will compute any of the univariate statistics available from `summ, detail` and generate a time series containing that statistic over the defined time series sample (requiring prior use of `tsset`). One may specify the window width (the number of periods included in the statistic's computation) as an option, as well as the alignment of the resulting statistic with the original series. This routine is especially handy for many tasks in financial research, in which some measure of recent performance–the average share price over the last twelve months, or the standard deviation (volatility) of the share price over that interval–is often needed as a regressor. As an ilustration of Stata's programming flexibility, the `mvsumm` routine will automatically operate separately on each time series of a panel if it detects that a panel calendar has been established by `tsset`. That added flexibility was very readily incorporated in its logic.

As useful as `mvsumm` might be, like the underlying `summarize` it will only handle a single time–series. I was recently contacted by an IMF researcher who wondered how a moving correlation might be generated. After only a small amount of head–scratching it became apparent to me that this could be provided by a trivial set of modifications to `mvsumm`, producing `mvcorr`. This routine, using largely identical syntax, allows one to compute a moving–window correlation between two series. Why might this be useful? Well, in finance, 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`.

3.5 Moving–window regression estimates

As a last topic, I would like to discuss some work in progress: the creation of a time–series routine, sorely lacking from Stata, that would compute moving–window regression estimates. Parallel to the rationale for `mvsumm`, one may indeed compute regression estimates for non–overlapping subsamples via official Stata's `statsby`; but that command is not really capable of dealing with overlapping subsamples, as that would correspond to the same observation being a member of several by–groups.

The challenge in devising such a 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. That is, can we define a command's syntax so that it can be more or less intuitively employed by the Stata user, and generate the statistics that one might wish to track as time series over the estimation sample? I have made a stab at producing such a routine, as `rollreg`, with the logic borrowed heavily from a RATS routine originally authored by Simon van Norden at the Bank of Canada.

A first concern with a moving–window estimation routine: how should the window be designed? One simple scheme would mimic `mvsumm`, and allow for a window of fixed width that is to be passed through the sample, one period at a time. (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, our routine must have that facility, and for completeness we also might want to allow for a window that takes into account the last τ periods, and 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, e.g., an *ex ante* forecast, using a greater or lesser amount of that information in the computation. So that must be included as well.

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, despite the necessity to avoid namespace collisions with new variable names (and Stata will ensure that any potential collisions are prevented).

Although I cannot present a full–fledged Stata command ready for distribution at this time, I have a pre–beta version of `rollreg` which illustrates some of these features, and points to the potential of having such a command readily available (through `ssc`) in the Stata environment. I will be working further to develop this facility, and welcome suggestions on features and appropriate syntax that

would make a moving–window regression estimator most useful. (There is also no reason to confine such a facility to `regress`, of course, and one possible extension would be to make `rollreg` (or perhaps `rollest`) usable with a variety of estimation commands).

References

[1] Andrews, Donald and Eric Zivot, 1992. Further evidence on the Great Crash, the oil price shock, and the unit-root hypothesis. *Journal of Business and Economic Statistics* 10, 251–70.

[2] Baum, Christopher F., Barkoulas, John T. and Mustafa Caglayan, 1999. Long memory or structural breaks: Can either explain nonstationary exchange rates under the current float? *Journal of International Financial Markets, Institutions and Money*, 9, 359–76. Available as Boston College Economics Working Paper No. 380, http://fmwww.bc.edu/ec-p/wp380.pdf.

[3] Baum, Christopher F., 2000. Tests for stationarity of a time series. *Stata Technical Bulletin* 57, sts15.

[4] Baum, Christopher F., 2004. A review of Stata 8.1 and its time series capabilities. *International Journal of Forecasting*, 20, 151–161. Available as Boston College Economics Working Paper No. 581, http://fmwww.bc.edu/ec-p/wp581.pdf.

[5] Baum, Christopher F., Mark E. Schaffer, and Steven Stillman, 2003. Instrumental variables and GMM: Estimation and testing. *Stata Journal*, 3, 1–31. Available as Boston College Economics Working Paper No. 545, http://fmwww.bc.edu/ec-p/wp545.pdf.

[6] Baum, Christopher F. and Richard Sperling, 2000. Tests for stationarity of a time series. *Stata Technical Bulletin* 58, sts15.1.

[7] Baum, Christopher F. and Vince Wiggins, 2000a. Tests for long memory in a time series. *Stata Technical Bulletin* 57, sts16.

[8] Baum, Christopher F. and Vince Wiggins, 2000b. Utility for time series data. *Stata Technical Bulletin* 57, dm81.

[9] Clemente, J., Montañés, A. and M. Reyes, 1998. Testing for a unit root in variables with a double change in the mean. *Economics Letters* 59, 175–182.

[10] Kwiatkowski, D., Phillips, P.C.B., Schmidt, P. and Y. Shin, 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54, 159-178.

[11] Perron, Pierre and Tim Vogelsang, 1992. Nonstationarity and level shifts with an application to purchasing power parity. *Journal of Business and Economic Statistics* 10, 301–20.