The Stata Journal (2001) **1**, Number 1, pp. 1–16

# Stata: The language of choice for time series analysis?

Christopher F. Baum
Boston College

## 1 Introduction

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

Prior to Stata 6, support for time series analysis was weak, as 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 6 incorporated a time series calendar and added a lengthy list 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 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 only appropriate computations are 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 and may even refer to a different individual's observation in a panel context. In contrast, `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 $\Delta x$ is `L2D.x`.

Reliance on these housekeeping details becomes overwhelmingly important in working with panel data, in 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. Stata refers to observations by their associated date (once a time series calendar has been established by `tsset`) rather than by their number; therefore, 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`

© 2001 StataCorp LP 

`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`) makes 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.

Although a number of useful features have been added, Stata's current facilities for the management of time series data have their limitations. It appears to be difficult for many users to transform dates generated in some other software (e.g., Excel) to the Stata date format without dealing with substrings, wrangling with two–digit vs. four–digit years, et cetera. Stata does not support business–daily data, and for those in economics and finance, 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.

In this paper, I will discuss a number of Stata's capabilities in the area of time series modeling, including data management and graphics. 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. For brevity, there are some areas I will not cover in this discussion: vector autoregressions and structural VARs, ARCH and GARCH modeling, cointegration tests (now available in official Stata's July 2004 update) and panel unit root tests. I concentrate on a number of features and capabilities that may not be so well known, and present some new methodologies for time–series data analysis.

# 2 Data management and graphics

## 2.1 tsmktim

First, let us consider some useful data management features. One often imports a time series, perhaps with a spreadsheet–formatted date variable and wants to establish a time series calendar for these data. One must work with the existing date to bring it into a proper Stata date variable using the `date()` or `mdy()` functions, assign a proper format to the variable (e.g., `%ty` for annual, `%tm` for monthly, etc.) and then use `tsset` to define the time series calendar with that nicely formatted date variable. Some time ago, Vince Wiggins and the author (2000a) wrote a utility that handles those three steps in one straightforward command: `tsmktim`, in which one need only specify the name of a new time series calendar variable and its start date:

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

This 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 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` in June 2004 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 discussed here, `tsmktim` is available from the SSC archive via official Stata's `ssc` command and may be located with `findit`.

## 2.2 egen functions for time series

There are also a number of `egen` functions which prove very useful with time series data. Official Stata's `egen` contains the `ma()` function, which computes $k$–period centered moving averages (where $k$ must be odd). This is of little use if one wants a one–sided moving average. For example, we might want a weighted moving average of four prior values, with arithmetic weights `0.4(0.1)0.1`. That construct can be viewed as a filter applied to a series in the time domain, and computed with `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 Nick Cox's `egenmore` package provide useful housekeeping tools: `eom()`, for instance, will generate a new variable with the date of the End of Month for a given month and year (which may be specified to be a weekday) and `bom()` provides the same functionality for the Beginning of Month. Both functions allow specification of lags and leads: e.g., adding `lag(3)` to the `eom()` function will return the Stata date for the last day of the third month prior. These functions are often very useful in working with financial data, and analogues could readily be constructed to provide similar functionality with quarterly or weekly data. Another function often useful in working with individual panel data is the `record` function; i.e.

```
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's `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 when 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 tsgraph

Let us now consider a graphics tool of primary interest to those still relying on Stata version 7. In the days of Stata 7, it seemed overly tedious to produce a "time-series line graph": a simple line plot of one or more time series on the $y$ axis versus time, appropriately labeled, on the $x$ axis. Those of us who work with time series data find this to be a rather common task, and Nick Cox and the author wrote `tsgraph` as an answer to that need. The routine will automatically produce a line graph with connected points, no markers, and by default (for use on the printed page) will select up to four different line styles, enabling the legend to be more useful than the default multi–colored graph. Considerable effort was also made in this routine to generate "nice" time–axis labels; that is, those which correspond to natural units, such as quinquennia for annual data, first quarters for quarterly data, etc. The routine will also intelligently consider that if the data are `tsset` as a panel, up to four units' timeseries of a single *varname* will be automatically plotted when that single *varname* is specified as the $y$ variable.

The new graphics command `tsline`, added to Stata after the release of version 8.0, is capable of doing all this and no doubt a great deal more. Indeed, using the Stata Manual Dataset `invest2.dta`, one can argue that `tsline investment income consumption` eventually produces a more stylish graph than `tsgraph investment income consumption`. The graph produced by `tsline` will contain dashed line types if a monochrome style is chosen, and it appears to construct "nice" date labels automatically. If one uses Stata 8, `tsline` is probably a more useful tool than `tsgraph`. For those using Stata 7, and those interested in dissecting a simple Stata program to determine how the code has

been written, `tsgraph` might be of some interest. As an illustration:

```
* tsgraph_X.do    07sep2004 CFBaum
* Program illustrating use of tsgraph v tsline
use http://www.stata-press.com/data/r8/invest2.dta, clear
* In Stata 8, could do
* webuse invest2, clear
tsset company time
drop if company>4
tsgraph invest market stock if company==1
more
tsline invest market stock if company==1
more
* illustrate automatic use on panel
tsgraph invest, ti("Investment expenditures by firm")
```

I turn now to discussion of a number of statistical / econometric capabilities for time series modeling. Some additional examples of time series modeling are provided in Baum (2004).

# 3 Statistics/econometrics

## 3.1 Capabilities of arima

A frequent participant in the Statalist listserv will appreciate that **Reading the Fine Manuals** is a rarely practiced art. One of those gems that can only be gleaned by **RTFM**, though, is a proper appreciation of one of Stata's most powerful time series commands, `arima`. The difficulty here is actually semantic in nature. Given Stata's late arrival in the domain of econometric software for time series analysis (vs. TSP, RATS, eViews, PC-GIVE, and the like), researchers in this area imagine that a command named `arima` does exactly that, providing for the 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. Naturally, Stata's `arima` command performs that estimation. However, it is not widely appreciated that Stata's `arima` command does this and much more. I can appreciate the difficulty of documenting such a wide array of features (`arima` is perhaps one of Stata's most complex commands in terms of its user interface), and the desire to have a single command that opens the door to estimating a wide range of models rather than a long list of alternative commands. This logic is the same rationale underlying many of the `xt` commands, such as `xtreg` or `xtgee`. It should be noted that the use of the term `arima` for this command seems to result in many users overlooking its potential usefulness for a wide variety of time series modeling tasks.

For instance, the `arima` command 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 only 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.

Also worthy of note is that `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, $\hat{y_\tau} = X_\tau \hat{\beta}$, where one of the columns of $X$ is $y_{\tau-1}$. For many purposes, a sequence of static forecasts is appropriate. For instance, if we want to mimic 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, we should forecast $y_{T+\tau+1}$. However, 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 wish to specify an *ARMA(p,q)* error structure—and use the `dynamic()` option on the subsequent `predict` command. To compare the two forecasting strategies, consider:

```
* arima_X.do  10sep2004 CFBaum
use http://www.stata-press.com/data/r8/friedman2.dta, clear
* in Stata 8, could do
webuse friedman2, clear
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))
label var pc92 "Actual"
label var consump_st "one-step forecast"
label var consump_dyn "dynamic forecast"
* graphics could be produced in Stata 7 via tsgraph
tsline pc92 consump_st consump_dyn if tin(1982q1,1986q4), ///
 ti("Actual and Predicted Real Consumption")
graph display, xsize(4) ysize(3) scheme(s2mono)
```

yielding the estimated equation and figure:

```
. arima pc92 L.pc92 L(0/1).m2 if tin(,1981q4)
(setting optimization to BHHH)
Iteration 0:   log likelihood = -392.09822
Iteration 1:   log likelihood = -392.09822
ARIMA regression
Sample:  1959q2 to 1981q4                       Number of obs      =        91
                                                Wald chi2(3)       =  58210.57
Log likelihood = -392.0982                      Prob > chi2        =    0.0000
```

| pc92 | Coef. | OPG Std. Err. | z | P>\|z\| | [95% Conf. | Interval] |
|---|---|---|---|---|---|---|
| pc92 | | | | | | |
| pc92 | | | | | | |
| L1 | 1.01849 | .0134459 | 75.75 | 0.000 | .992137 | 1.044844 |
| m2 | | | | | | |
| -- | 1.022222 | .378699 | 2.70 | 0.007 | .2799852 | 1.764458 |
| L1 | -1.074798 | .3888382 | -2.76 | 0.006 | -1.836906 | -.3126889 |
| _cons | .8652995 | 20.18578 | 0.04 | 0.966 | -38.6981 | 40.4287 |
| /sigma | 17.99031 | .9378816 | 19.18 | 0.000 | 16.15209 | 19.82852 |

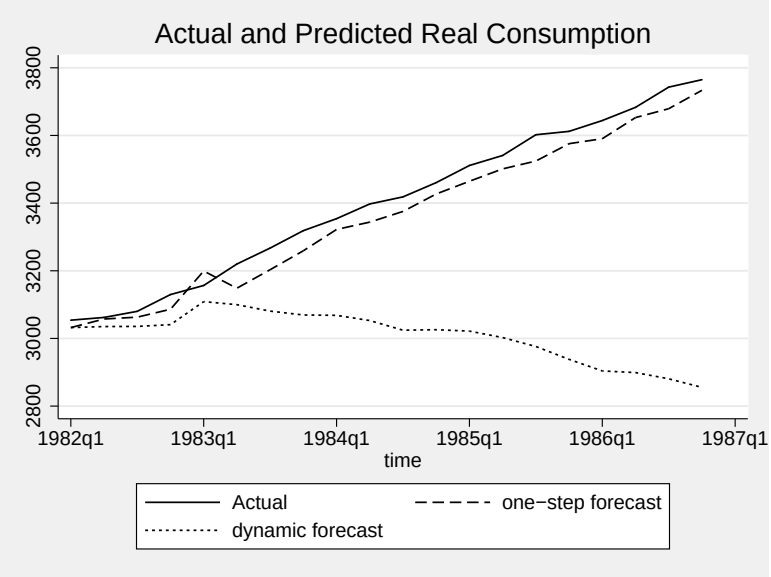


Figure 1: Static and dynamic forecasts with arima.

In this example, it may be seen that although the model generates reasonable one–step–ahead forecasts (underpredicting slightly), the dynamic, or recursive form of the model is quite unsatisfactory, leading over a 20–quarter forecast horizon to an unreasonably low level of predicted real personal consumption expenditure.

## 3.2 ivreg2

There are a number of features in the latest version of `ivreg2` (Baum, Schaffer and Stillman, 2003, 2004) that are relevant for the analysis of time series and panel data models. The version of `ivreg2` published in the *Stata Journal* and presented by Mark Schaffer at the 2003 UK Stata Users Group meetings implemented instrumental variables and a Generalized Method of Moments (GMM) estimator that efficiently estimated IV–style models with arbitrary heteroskedasticity. In the latest update of the software (available from SSC), we have added a number of features: notably, the ability to utilize 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. In fact, since `ivreg2` can estimate models without any instrumental variables, it may be used to reproduce any model estimable with official Stata's `newey`.

Two aspects of these extended `ivreg2` capabilities are worth mention. In computing a robust covariance matrix, one may want the robust–to–AC correction without applying the robust–to–H correction. The computation of the "White piece" of a robust covariance matrix involves estimation of fourth moments of the estimated residuals, and if there is no cause to question the assumption of homoskedasticity, one should apply the "Newey–West" component without the "White" or "sandwich" correction for arbitrary heteroskedasticity. The new version of `ivreg2` allows for that by decoupling those as separate options, which is something that `newey` cannot do. Indeed, although "HAC standard errors" computed by the Newey–West formula are routinely reported in the literature, it must be noted that the choice of the Bartlett kernel in that formula is an arbitrary smoothing of the empirical autocorrelation function. All that is required is an empirical autocovariance matrix that is guaranteed to be positive definite, and a number of kernels can achieve that goal. Just as with any application of kernel estimation, one may want to choose a different kernel. The new version of `ivreg2` allows for the specification of the kernel from a list of eight choices (with Bartlett, à la Newey–West, as default). Because some of those kernels do not involve an integer truncation point, `ivreg2` uses a bandwidth (`bw()`) option rather than the `lag()` option of `newey`. For kernels with an integer truncation point (such as Bartlett), the bandwidth is one greater than the number of lags (thus `bw(1)` specifies that no lagged values of the residuals are to be included; `bw(2)` = `lag(1)`, et cetera). Of course, the use of `kernel()` and `bw()` options allows `ivreg2` to be used to estimate instrumental variables models with AC or HAC errors, which cannot be achieved with `newey`.

The revised `ivreg2` also contains a number of other new features: the ability to estimate LIML (limited–information maximum likelihood) models, perform general $k$–class estimation, and compute a number of diagnostics for "weak instruments". A `pscore()` option has been added to maintain comparability with official `ivreg`.

## 3.3 Unit root tests

### dfgls

The `dfgls` command, a user–written routine in Stata 7 (Baum and Sperling, 2000), was adopted by StataCorp developers and is now part of official Stata. Indeed, its methodology, the "DF–GLS", Dickey–Fuller Generalised Least Squares, approach of Elliott, Rothenberg, and Stock (1996) 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, and `dfgls` should be your unit root test of choice.

### kpss and other fractional integration tests

As an interesting alternative to the Dickey–Fuller style methodology underlying the first generation tests as well as `dfgls`, we might consider the KPSS test (Kwiatkowski, Phillips, Schmidt and Shin, 1992). This test, implemented some time ago for Stata (Baum, 2000), 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 `dfgls` and KPSS tests may both be applied, with hopes that the verdict of one will confirm that of the other. 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). Full details are available from the *STB* articles cited.

### Unit root 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 its 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. As an illustration, consider a timeseries driven by a deterministic trend (perhaps subject to breaks in the mean of the series, or breaks in trend) rather than following the stochastic trend of a unit root process.

One test of this nature was devised by Andrews and Zivot (1992) and presented in their article analysing the "Great Crash" of the 1930s and oil price shocks of the 1970s in terms of their effects on unit–root test behavior. This test 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. Subsequently, the procedure conducts a Dickey–Fuller style unit root test conditional on the series inclusive of the estimated optimal breaks. The author has made the `zandrews` test, translated from RATS code, available in Stata. By

default, the test allows for a break in intercept. Alternatively, a trend break or both intercept and trend breaks may be considered by employing the `break(`*string*`)` option. As in all Dickey–Fuller style tests, the degree of augmentation with additional lags of the dependent variable may have an impact on the power of the test by ensuring that the residuals are sufficiently whitened. `zandrews` provides four different methods for lag selection in the `lagmethod(`*string*`)` option. For example, one may specify the number of lags desired, rely on the *AIC* or *BIC* criteria, or allow for a sequential $t$–test to detect the optimal lag, similar to the method implemented in `dfgls`. A `graph` option is also provided to allow visual scrutiny of the unit root test statistics for alternative breakpoints. The `zandrews` routine may be applied to single time series within panel data. It requires Stata 8.0 or better.

As an illustration:

```
* zandrews_X.do  16jul2004 CFBaum
webuse turksales,clear
* contrast with Dickey-Fuller test
dfuller sales
zandrews sales, graph
zandrews sales, break(trend)
zandrews sales, break(both) trim(0.10)
zandrews sales, lagmethod(BIC)
zandrews D.sales, graph
* work with single timeseries of panel
webuse grunfeld, clear
zandrews invest if company==3, break(trend) graph
```

One obvious weakness of the Zivot–Andrews strategy, relating as well to similar tests proposed by Perron and Vogelsang (1992), is the inability to deal with more than one break in a time series. For instance, the trade–weighted value of the US dollar versus trading partners' currencies followed a $V$–shaped pattern over the 1980s and 1990s, so that a single break in intercept and trend could not have dealt satisfactorily with the evolution of this series. Addressing this problem, 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). However, in that paper the authors only dealt with series including a single $AO$ or $IO$ event. The double–break additive outlier $AO$ model as employed in Baum, Barkoulas and Caglayan (1999) involves the estimation of

$$y_t = \mu + \delta_1 DU_{1t} + \delta_2 DU_{2t} + \tilde{y}_t$$

where $DU_{mt} = 1$ for $t > T_{bm}$ and 0 otherwise, for $m = 1, 2$. $T_{b1}$ and $T_{b2}$ are the breakpoints, to be located by grid search. The residuals from this regression, $\tilde{y}_t$, are then the dependent variable in the equation to be estimated. They are 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:

$$\tilde{y_t} = \sum_{i=1}^{k} \omega_{1i} DT_{b1,t-i} + \sum_{i=1}^{k} \omega_{2i} DT_{b2,t-i} + \alpha \tilde{y}_{t-i} + \sum_{i=1}^{k} \theta_i \Delta \tilde{y}_{t-i} + e_t$$

where $DT_{bm,t} = 1$ for $t = T_{bm} + 1$ and 0 otherwise, for $m = 1, 2$. No intercept is necessary as $\tilde{y_t}$ is mean zero. This regression is then estimated over feasible pairs of $T_{b1}$ and $T_{b2}$, searching for the minimal $t$–ratio for the hypothesis $\alpha = 1$; that is, the strongest rejection of the unit root null hypothesis. The value of this minimal $t$–ratio is compared with critical values provided by Perron and Vogelsang (1992), as they do not follow the standard "Dickey–Fuller" distribution.

The equivalent model for the innovational outlier (gradual change) model expresses the shocks to the series (the effects of $\delta_1, \delta_2$ above) as having the same $ARMA$ representation as other shocks to the model, leading to the formulation

$$y_t = \mu + \delta_1 DU_{1t} + \delta_2 DU_{2t} + \phi_1 DT_{b1,t} + \phi_2 DT_{b2,t} + \alpha y_{t-1} + \sum_{i=1}^{k} \theta_i \Delta y_{t-i} + e_t$$

where again an estimate of $\alpha$ significantly less than unity will provide evidence against the $I(1)$ null hypothesis.

In each of these models, the breakpoints $T_{b1}, T_{b2}$ and the appropriate lag order $k$ are unknown. The breakpoints are located by a two–dimensional grid search for the maximal (most negative) $t$–statistic for the unit root hypothesis ($\alpha$=1), while $k$ is determined by a set of sequential $F$–tests.

The Stata routines `clemao2` and `clemio2` implement the $AO$ and $IO$ models for two structural breaks, respectively. If their estimates show that there is no evidence of a second break in the series, the original Perron–Vogelsang techniques should be used to test for a unit root in the presence of one structural break. For convenience, the single–break routines are also provided in this package as routines `clemao1` and `clemio1`. In applying Dickey–Fuller tests in time series that may exhibit structural breaks, one should consider the results forthcoming from the `clem` $AO$ or $IO$ routines. If these estimates provide evidence of significant additive or innovational outliers in the time series, then results derived from `dfuller`, `pperron` or `dfgls` are placed in doubt, as this is evidence that the model excluding structural change is clearly misspecified by the omission of relevant explanatory variables. Like `zandrews`, the `clem` $AO$ or $IO$ routines may be applied to single time series within panel data. They require Stata 8.2.

To illustrate:

```
* clem_X.do    16jul2004 CFBaum
* Program illustrating use of Clemente, Montanes, Reyes
* structural break unit root tests
webuse m1gdp, clear
```

```
label var ln_m1 "log(M1), SA"
label var t "calendar quarter"
clemao1 ln_m1, graph
more
clemio1 D.ln_m1, graph
more
clemao2 ln_m1 if tin(1959q1,2002q3), trim(0.10) maxlag(6) graph
```

We reproduce the output and graph below from the first test conducted in the program above: that for the (log) level of M1, the U.S. money supply. We note that the break detected by the test roughly corresponds to the timing of the 1987 U.S. stock-market crash. Despite the structural break, we are unable to reject the null hypothesis of a unit root in this series.

```
. clemao1 ln_m1, graph
Clemente-Montañés-Reyes unit-root test with single mean shift, AO model
ln_m1    T =  157        optimal breakpoint :   1987q3
AR( 1)               du1            (rho - 1)     const
------------------------------------------------------------------------
Coefficient:         1.32622       -0.04842       5.58624
t-statistic:          20.073         -2.530
P-value:               0.000         -3.560  (5% crit. value)
```

## 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 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 constant over time. A strict 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 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. On the other hand, it will not deal with computing statistics from a sequence of by–groups that are formed by a "moving window" with, for example, 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

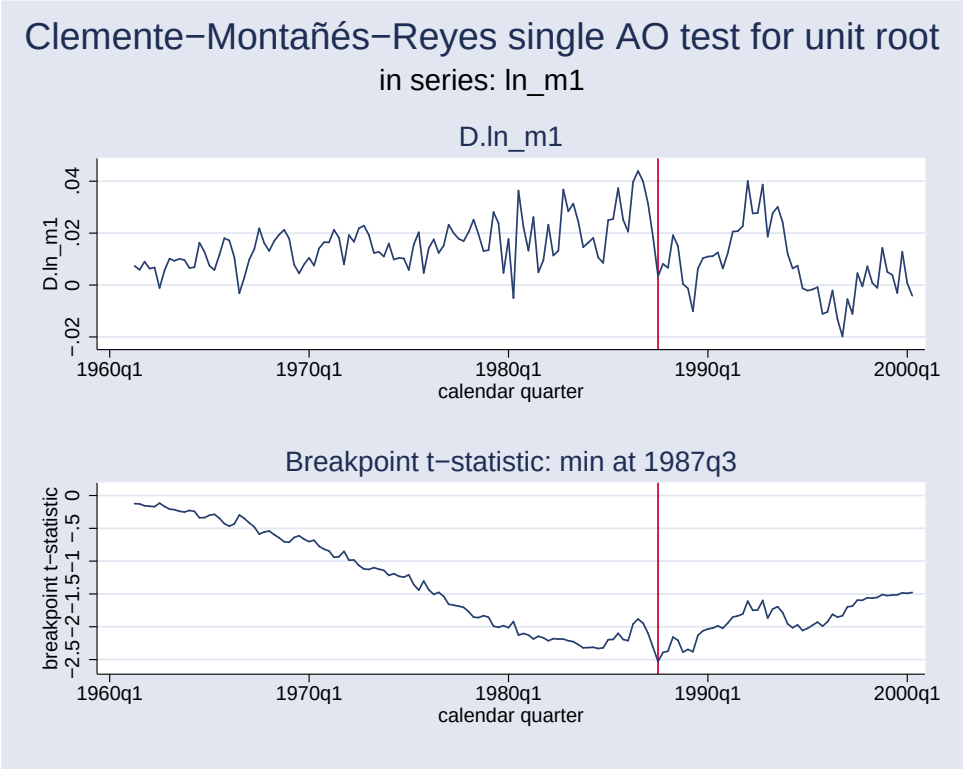


Figure 2: Unit root test with additive outliers.

that interval—is often needed as a regressor. The `mvsumm` routine automatically will operate separately on each time series of a panel if it detects that a panel calendar has been established by `tsset`. This added flexibility was incorporated in its logic in a June 2004 update.

Similar to the underlying `summarize`, `mvsumm` will handle only a single time–series. How might a moving correlation be generated? By a trivial set of modifications to `mvsumm`, producing `mvcorr`. 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, we might want to calculate the moving correlation between spot and futures prices of a particular commodity. The `mvcorr` routine requires `tsset`, thus supporting time–series operators, and it will allow the computation of moving autocorrelations. For example, `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`, `mvcorr` will operate automatically on each time series of a panel. It requires Stata 8.2.

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")
```

## 3.5 Moving–window regression estimates

Last, we consider the creation of a Stata time–series routine to 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`. However, 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.

The challenge in devising such a routine is not in the necessary computations, nor even in the programming. Rather, it lies in providing a user interface that will allow the researcher to specify, in some comprehensible form, what she or he would like calculated for each crank of the window. The new routine `rollreg` provides that functionality with logic borrowed heavily from a RATS routine originally authored by Simon van Norden at the Bank of Canada (and available from the web–based SSC archive in RATS format).

The first concern with a moving–window estimation routine: how should the window be designed? One obvious 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: 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). There are also applications in which an "expanding window" is desired: that is, starting with the first $\tau$ 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(`$\tau$`)` option. For completeness, the routine also offers the `drop(`$\tau$`)` option, which implements a window that initially takes into account the last $\tau$ 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*.

As an illustration:

```
* rollreg_X.do    07sep2004 CFBaum
* Program illustrating use of rollreg
webuse wpi1, clear
g t2 = t^2
rollreg D.wpi t t2, move(24) stub(wpiM) graph(summary)
more
rollreg D.wpi t t2, add(24) stub(wpiA) graph(summary)
more
rollreg D2.wpi LD.wpi LD2.wpi t, move(48) stub(wpiM2) robust graph(full)
```

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. 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, we may use the following code to produce Figure 3:

```
* rollreg_X2.do    09sep2004 CFBaum
* Program illustrating use of rollreg on panels
webuse invest2, clear
tsset company time
rollreg market L(0/1).invest time, move(8) stub(mktM)
local dv `r(depvar)'
local rl `r(reglist)'
local stub `r(stub)'
local wantcoef invest
local m "`r(rolloption)'(`r(rollobs)')"
forv i=1/4 {
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(4) xsize(4) col(2) ///
t1("Full-sample coefficient displayed") saving("‘wantcoef’.gph",replace)
```

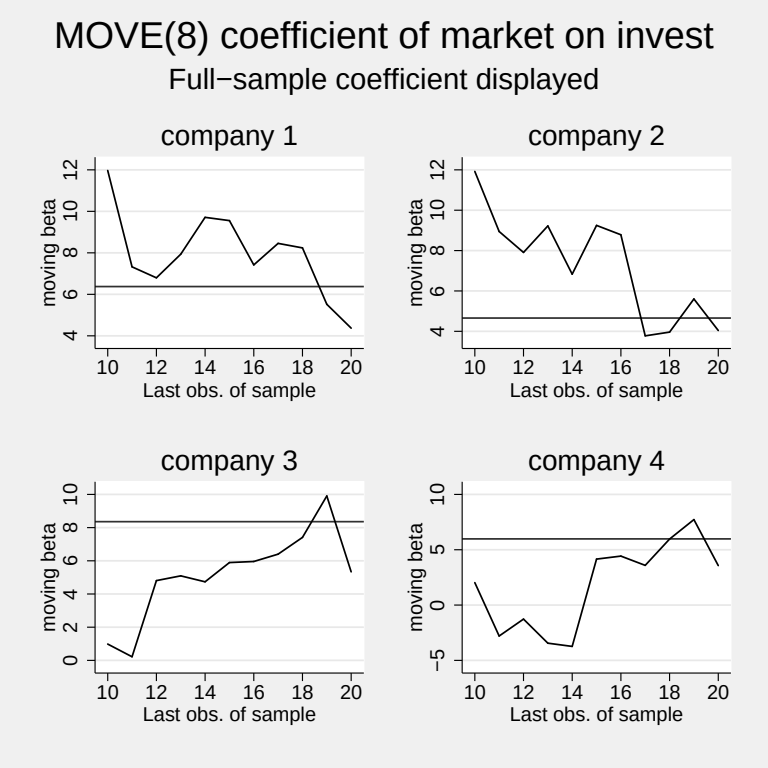


Figure 3: Rolling regression estimates.

It is interesting to note that companies 1 and 2 have broadly similar trajectories, as do companies 3 and 4: and quite different from the former pair. A clear understanding of the temporal stability of the coefficient estimates is perhaps more readily obtained graphically.

## 4 Final thoughts

The question I posed in the title of this paper, could Stata be considered the language of choice for time series analysis?, remains an open question. Relative to more specialized time series packages, Stata lacks some very useful features such as nonlinear systems estimation, multivariate GARCH and simulation of a nonlinear model, as well as some of the graphics tools useful for time series work (such as likelihood profile plots from a model estimated via maximum likelihood). On the other hand, most users find it difficult to work effectively in several packages' differing syntax, and competing packages generally do not possess Stata's flexibility in handling non–time series applications, data management, and the like. The most encouraging trend, in my mind, is that official Stata's developers have committed significant resources toward making Stata a competitive time series package, and that many user–programmers have chosen to develop and share their Stata code implementing a number of useful tools for time series tasks lacking from official Stata. The last several years of development and the rapid pace of innovation in the Stata community bode well for those who would like to rely on Stata for a very sizable fraction of their research computing in the analysis of time series and panel data.

## 5 Acknowledgements

I am very grateful to Dr Mustafa Caglayan and the Department of Economics at the University of Leicester for their hospitality. Much of the invited lecture underlying this paper was prepared in Leicester during a visit to the department in May and June 2004. I am also indebted to Paula N. Arnold and Nicholas J. Cox and an anonymous reviewer for editorial suggestions and participants in the 10th UK Stata Users Group meetings for their comments.

## 6 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., John T. Barkoulas, 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., Mark E. Schaffer, and Steven Stillman. 2004. Software updates: st0030_1. *Stata Journal*, 4, 224.

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

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

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

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

[11] Elliott, G., T. Rothenberg, and J. H. Stock. 1996. Efficient tests for an autoregressive unit root. Econometrica 64, 813-836.

[12] Kwiatkowski, D., P.C.B. Phillips, P. Schmidt, 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.

[13] 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.

**About the Author**

Christopher F. Baum is an associate professor of economics at Boston College. He is an associate editor of *Computational Economics* and *The Stata Journal*, and serves on the Advisory Council of the Society for Computational Economics. Baum founded and manages the Boston College Statistical Software Components (ssc) archive at RePEc (http://repec.org), and has developed a number of Stata commands for data management and time series analysis. His recent research has focused on the effects of uncertainty on international trade flows, bank lending behavior, and firms' cash holdings and use of leverage.