Time series estimation and forecasting

Christopher F Baum

Boston College and DIW Berlin

IMF Institute, Spring 2011

Stata's time series calendar

To take full advantage of Stata's time series capabilities, you should be familiar with its time series *calendar* and *operators*. The time series calendar allows you to specify, via the `tsset` command, that data are time series at an annual, half-yearly, quarterly, monthly, weekly or daily frequency. You may also specify intraday frequencies (as `clocktime`), as Stata's calendar variable has microsecond accuracy. The frequency may also be specified as `generic`.

For instance, `tsset year, yearly` will specify that the integer variable `year` in your dataset is the calendar variable, and the data frequency is annual. You may also use `tsset` to specify that the data are panel data: e.g., `tsset country qtr, quarterly` would indicate that your data are a (possibly unbalanced) panel of country-level quarterly data.

For all but annual data, you must construct a calendar variable according to Stata's definition. Stata, like Unix, assumes that time 0 is 1 January 1960 AD. Thus, `display daily("13feb2011","DMY")` yields 18671, as that is how many days have elapsed since 1/1/1960.

`display daily("13aug1951","DMY")` yields −3063, as that date is that many days prior to 1/1/1960.

Likewise, `display quarterly("2011Q1","YQ")` yields 204, as we are 204 calendar quarters beyond 1960q1.

There are a set of functions, described at `help dates and times`, that allow you to convert one calendar variable into another frequency, or convert string data (such as 13/02/2011 or 2001Q3) into Stata dates.

The delta option

The `tsset` command also has an optional argument, `delta( )`, which allows you to specify that data are defined at one frequency but recorded at another. For instance, US Census data are produced every decade. You could define a time series of Census data as `tsset year, yearly delta(10)` to indicate that the data are aligned with particular years, but only recorded every ten years. The use of the `delta(10)` option will cause Stata to consider the lagged value of 2000 to be 1990, for instance, rather than 1999, which would be missing.

tsmktim

If you have a time series that is complete (with no gaps), starting in a given time period, it may be easiest to establish the calendar variable with my `tsmktim` routine. This utility, available from SSC, allows you to issue a command like `tsmktim yq, start(1973q3)` which not only creates the variable `yq` as a quarterly calendar variable, starting in 1973q3, but gives it the proper `%tq` format, so that dates display as dates rather than as integers.

Gaps in time series

A problem arises, though, in that many daily time series contain gaps for weekends and holidays. Stata does not have a business-daily data concept, and even if weekends are excluded, holidays are problematic. Many of Stata's time series commands do not tolerate gaps, and we normally want to consider Friday to be followed by Monday in Western financial market data.

A way to circumvent this problem is described in *Stata Tip 40: Taking care of business*, included in your materials. Briefly, the solution involves creating two time series calendar variables: one with proper dates, which contains gaps, and a second that does not. The second may be created by `generate t = _n` where `_n` refers to the sequential observation number.

With these two calendar variables (say, `ymd` and `t`) defined, you may `tsset t` when you want to use data management or statistical commands that are sensitive to the presence of gaps: for instance, creating a first difference, or referring to a lagged value. After completing estimation and producing forecasts, you may want to tabulate or graph the forecasts with proper calendar dates attached. You may then `tsset ymd` to attach the proper calendar variable for those operations.

This technique may also be used in panel data that contains gaps, as under the control of a `by:` prefix the observation number (`_n`) refers to the observation within the by-group rather than within the entire data set. Thus, using `by country:`, for instance, you could produce the sequential calendar variable for the entire panel with one command.

Stata's time series operators

Stata has several time series *operators*, described at `help tsvarlist`, which allow you to refer to lags, leads, differences and seasonal differences for a data set that has been `tsset`. These are prefixes of the variable names, such as `L.gnp`, `F.gdp`, `D.tb3mo`, or `S.tb3mo`, respectively. To specify higher lags or leads, you may use `L4.gnp` or `F2.gdp`.

Keep in mind that `D2.tb3mo` is the difference of the difference; if you want to specify the difference between that variable at t and $(t-2)$, use the 'seasonal difference.' That is particularly useful for quarterly data, where `S4.sales` will refer to quarter-over-quarter sales, comparing the observation to that of the same quarter in the previous year.

The operators may also be combined, so that you can use `L2D.x` to refer to the second lag of the first difference of x, which could also be formed as `DL2.x`. In either case, the operators are applied from the dot leftward.

A major advantage of the time series operator syntax is that you need not create the lagged, led, differenced variables. Like factor variables in Stata 11, they will be instantiated on the fly, and will not be permanently added to the data set in memory.

Time series operators ensure validity

The most important argument for using time series operators is that they enforce validity of any time series expressions. If there is a gap in the data: for instance, if we have data for 1971–1975 and 1977–2000, referring to the prior observation using an observation subscript `[_n - 1]` will improperly consider the lagged value of 1977 to be that of 1975. If the data are `tsset`, the lagged value or first difference of 1977 will properly be flagged as missing.

This is even more important in the case of panel data, where we do not want the lagged value of one panel unit to refer to the last value of the previous unit. The time series operators, under a panel `tsset`, will respect the data set's organization and avoid such errors. Thus, you should always use the time series operators, on a single time series or in a panel.

The operators may also be used in a Stata `numlist`, so that

```
regress y L(-4/4),x
```

will run a Sims test for Granger causality, including four leads of x, current x, and four lags of x in the regression. Following that regression, to jointly test the coefficients of future x for significance,

```
testparm L(-4/-1).x
```

will do so, and provide the proper *F*-test and p-value.

The time series operators may also be used with a parenthesized varlist: for instance,

```
regress gdp L(1/4).(govtexp  money)
```

will regress `gdp` on four lags of `govtexp` and four lags of `money`.

Forecast accuracy statistics

To compare in-sample forecast accuracy, it may be useful to use `estat ic` after estimating a regression model, which will produce the AIC and BIC statistics.

For instance, using the `usmacro1` data set, let us fit models with differing number of lags on the regressor and store the estimates so that they may be compared with `estimates stats`. We hold the sample fixed with `if e(sample)`.

```
 . use usmacro1
. eststo clear
. eststo eight: qui regress tr10yr L(1/8).rmbase
. eststo six: qui regress tr10yr L(1/6).rmbase if e(sample)
. eststo four: qui regress tr10yr L(1/4).rmbase if e(sample)
. est stat eight six four
```

```
─────────────┬───────────────────────────────────────────────────────────────
       Model │    Obs    ll(null)   ll(model)     df          AIC         BIC
─────────────┼───────────────────────────────────────────────────────────────
       eight │    199   -471.4506   -447.3226      9     912.6452    942.2849
         six │    199   -471.4506   -448.5081      7     911.0163    934.0694
        four │    199   -471.4506   -449.5158      5     909.0316    925.4981
─────────────┴───────────────────────────────────────────────────────────────
               Note:  N=Obs used in calculating BIC; see [R] BIC note
```

Both AIC and BIC indicate that the model with four lags is preferred.

the tin( ) function

A useful function for time series data that have been declared as such by `tsset` is the `tin) ( )` function, which should be read tee-in. We can specify calendar dates using this function to restrict the estimation sample:

```
. tsset
        time variable:  yq, 1959q1 to 2010q3
                delta:  1 quarter
. qui regress tr10yr L(1/4).rmbase if tin( , 2008q1)
. qui regress tr10yr L(1/4).rmbase if tin(1973q4, 1987q2)
. qui regress tr10yr L(1/4).rmbase if tin(1973q4, )
```

The `tin( )` function is also useful if you need to produce out-of-sample forecasts. Recall that the `predict` command will generate predicted values, residuals, and other series for the entire data set. You might want to run a regression over a subperiod, with a holdout sample of more recent observations, and then forecast through the holdout sample period. That is readily specified with `tin()`:

```
. qui regress tr10yr L(1/4).rmbase if tin( , 2008q1)

. predict double tr10yrhat if tin(2008q2, 2009q4), xb
(200 missing values generated)
```

In this example, we produce predicted values only for the out-of-sample period.

Rolling-window estimation

Stata provides a prefix, `rolling:`, which can be used to automate various types of rolling-window estimation for the evaluation of a model's structural stability. These include *fixed-width* windows, specified with the `window( )` option; *expanding* windows, specified with the `recursive` option; and *contracting* windows, specified with the `rrecursive`, or reverse recursive option.

The fixed-width window executes a statistical command for the specified number of calendar periods, then moves both the beginning and ending calendar period forward by one period and repeats it, and so on, until the last period of the sample is reached. With the `stepsize( )` option, you may move the window by more than one period.

The expanding window executes the command for the number of calendar periods specified in `window( )`, then repeats for a sample with one more calendar period, and so on. The left side of the window is held fixed while the right side expands.

The contracting window executes the command for the number of calendar periods specified in `window( )`, then repeats for a sample excluding the earliest calendar period, and so on. The left side of the window moves while the right side is held fixed at the last period.

For all uses of `rolling:`, a new data set is created with the results of the statistical command. This behavior is similar to that of other prefix commands such as `simulate:`, `jackknife:` and `bootstrap`. The dataset will contain two new variables, `start` and `end`, which identify the ‘edges’ of the window for each observation. One of those variables may be used to merge the new data set back on the original data set.

Although the most common use of `rolling:` may involve estimation (e-class) commands such as `regress`, the prefix may also be used with r-class statistical commands such as `summarize`. However, if your only interest is in producing moving-window descriptive statistics, you might find Baum and Cox's `mvsumm` command easier to use. Along those lines, their `mvcorr` routine, which produces moving-window correlations of two time series, should be noted. Both routines will automatically operate on a panel data set that has been properly `tsset`.

The `rolling:` prefix works with the concept of an *exp_list*, or list of expressions, that are to be computed for each window. For an e-class command such as `regress`, the default *exp_list* is `_b`, the vector of estimated coefficients (that is, `e(b)`. For a r-class command such as `summarize`, the default *exp_list* is all the scalars stored in `r( )`. You may override this behavior by specifying particular expressions in the *exp_list*.

For instance, to add the standard errors of the estimated coefficients to the *exp_list*, you may specify `_se`. To add the R^2 or RMS Error statistics from a regression, specify `r2=e(r2) rmse=e(rmse)` in the *exp_list*.

We will illustrate in a later talk how statistics not available in `e( )` or `r( )` may be collected.

You generally will want to specify the `saving` *filename*`, replace` option to `rolling:`, so that a new data set will be constructed, leaving the current data set in memory. Otherwise, `rolling:` will replace the current data set in memory with its results.

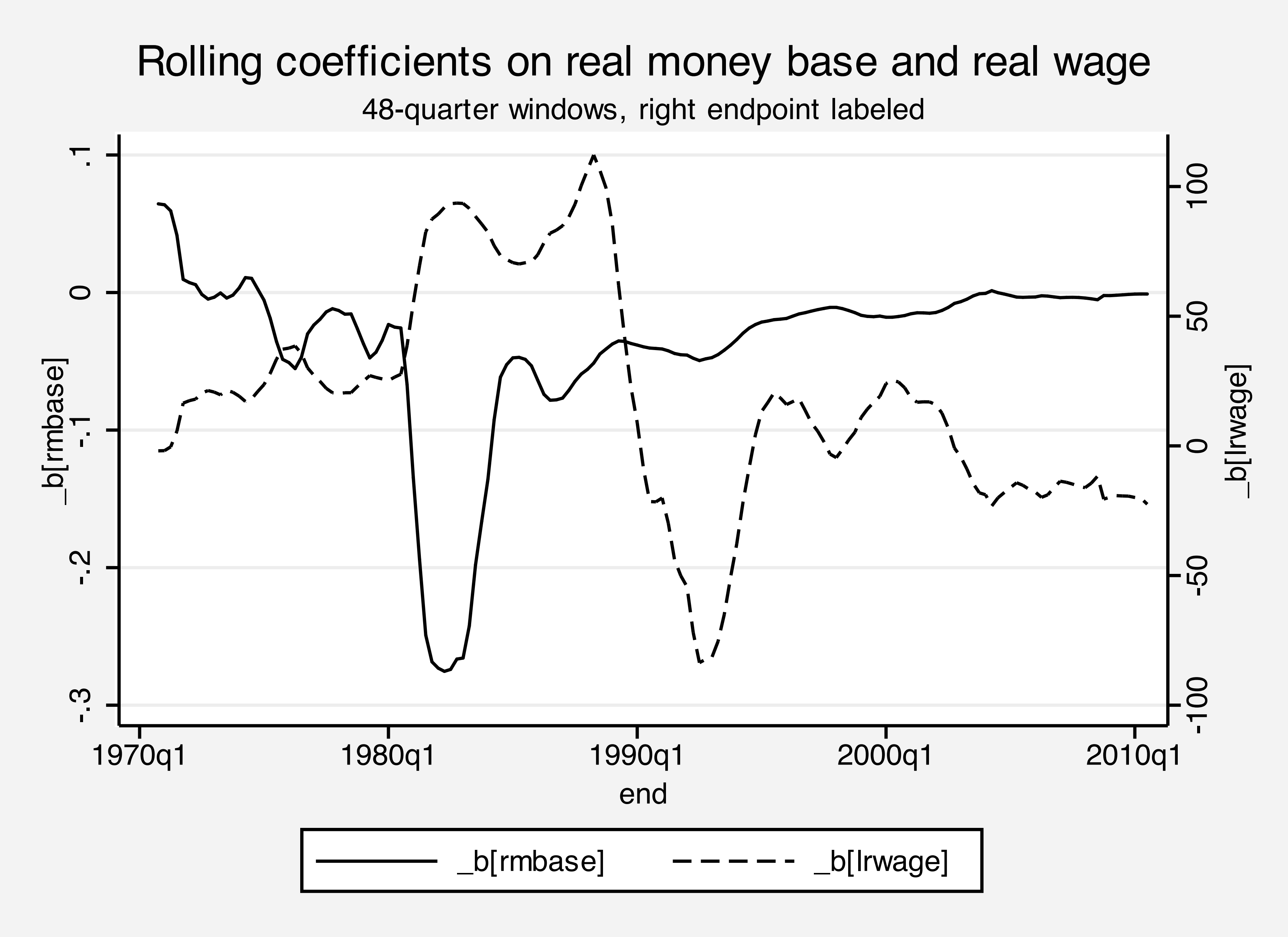
For example, say that we want to produce moving-window regression estimates from a window containing 48 quarterly observations:

```
 . rolling _b _se r2=e(r2) rmse=e(rmse), window(48)  ///
> saving(rolltr10, replace) nodots: regress tr10yr rmbase lrwage, robust
file rolltr10.dta saved
. use rolltr10, clear
(rolling: regress)
. describe
Contains data from rolltr10.dta
  obs:           160                       rolling: regress
 vars:            10                       15 Feb 2011 14:50
 size:         7,680 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
start           float  %tq
end             float  %tq
_b_rmbase       float  %9.0g                  _b[rmbase]
_b_lrwage       float  %9.0g                  _b[lrwage]
_b_cons         float  %9.0g                  _b[_cons]
_se_rmbase      float  %9.0g                  _se[rmbase]
_se_lrwage      float  %9.0g                  _se[lrwage]
_se_cons        float  %9.0g                  _se[_cons]
_eq2_r2         float  %9.0g                  e(r2)
_eq2_rmse       float  %9.0g                  e(rmse)
-------------------------------------------------------------------------------
Sorted by:
```

Notice that all options to `rolling:` appear before the colon.

We can now present the coefficient estimates graphically (optionally, with interval estimates):

```
 . tsset end, quarterly
        time variable:  end, 1970q4 to 2010q3
                delta:  1 quarter
. tw (tsline _b_rmbase) (tsline _b_lrwage, yaxis(2)),  ///
> scheme(s2mono) ti("Rolling coefficients on real money base and real wage") //
> /
> t2("48-quarter windows, right endpoint labeled")
```


Rolling coefficients on real money base and real wage
48-quarter windows, right endpoint labeled
_b[rmbase]
.1
0
-.1
-.2
-.3
_b[lrwage]
100
50
0
-50
-100
1970q1
1980q1
1990q1
2000q1
2010q1
end
_b[rmbase]
_b[lrwage]

Structural break models

Rolling-window regression estimates are one way of evaluating structural stability, or the lack thereof, in a time series regression. Many other approaches have been developed: for instance, the cusums and cusums-squared tests of Brown, Durbin, Evans (*JRSS*, 1975).

With the widespread use of unit root tests, whose power is severely degraded by the presence of structural breaks, a number of tests allowing for one or more structural breaks have been devised (Zivot & Andrews, *JBES*, 1992; Perron & Vogelsang, *JBES*, 1992; Clemente, Montanes, Reyes, *Ec. Let.*, 1998). Stata routines for these tests are available from SSC as packages `zandrews` and `clemao_io`, respectively.

The Elliott–Müller approach

However, tests for structural stability or structural change are not only relevant in the case of potentially nonstationary processes. We may be equally concerned with the temporal stability of a regression model in which there is no evidence of nonstationarity. To that end, Elliott and Müller (EM) proposed a novel strategy (*REStud*, 2006) that they claim is an encompassing approach to the problem of testing for temporal stability.

They consider “tests of the null hypothesis of a stable linear model

$$y_t = X_t' \bar{\beta} + Z_t' \gamma + \epsilon$$

against the alternative of a partially unstable model

$$y_t = X_t' \beta_t + Z_t' \gamma + \epsilon$$

where the variation in β_t is of the strong form” (p. 907), or nontrivial.

Consideration of this alternative has led to a huge literature based on the “diversity of possible ways $\{\beta_t\}$ can be non-constant.” EM point out that optimal tests and their asymptotic distributions have not been derived for many particular models of the alternative.

Their approach develops a single unified framework, noting that the “seemingly different approaches of ‘structural breaks’ and ‘random coefficients’ are in fact equivalent.” (p.908) EM unify the approaches that describe a breaking process with a number of non-random parameters with tests that specify stochastic processes for $\{\beta_t\}$ without requiring to specify its exact evolution.

The processes considered include breaks that occur in a random fashion, serial correlation in the changes of the coefficients, a clustering of break dates, and so on.

Under a normality assumption on the disturbances, “small sample efficient tests in this broad set are asymptotically equivalent” and “leaving the exact breaking process unspecified (apart from a scaling parameter) does not result in a loss of power in large samples.” (p. 908)

The consequences of this approach to the problem of structural stability are profound. “The equivalence of power over many models means that there is little point in deriving further optimal tests for particular processes in our set” (p. 908) and the researcher can carry out (almost) efficient inference without specifying the exact path of the breaking process.

Furthermore, the computation of EM’s Quasi-Local Level (q_{LL}) test statistic is straightforward, and it remains valid for very general specifications of the error term and covariates. The computation requires no more than $(k + 1)$ OLS regressions for a model with k covariates, in contrast to many approaches which require T or T^2 regressions. No arbitrary trimming of the data is required.

In the *structural break* literature, a fixed number of N breaks at $\tau_1, \ldots, \tau_N$ are assumed. Much of the literature addresses $N = 1$: e.g. the “Chow test”, cusums tests of Brown–Durbin–Evans, Bai and Perron, Andrews and Ploberger, etc.

In contrast, the *time-varying parameter* literature considers a random process generating β_t: often considered as a random walk process. The approaches of Leybourne and McCabe, Nyblom, and Saikkonen and Luukonen are based on classical statistics, while Koop and Potter and Giordani et al. consider a Bayesian approach. All of these approaches are very analytically challenging.

EM argue that tests for one of these phenomena will have power against the other, and *vice versa*. Therefore a single approach will suffice.

EM raise the interesting question: why do we test for parameter constancy? They consider three motivations:

1. Stability relates to theoretical constructs such as the Lucas critique of economic policymaking
2. Forecasting will depend crucially on a stable relationship
3. Standard inference on $\bar{\beta}$ will be useless if $\{\beta_t\}$ varies in a permanent fashion; persistent changes will render a fixed model misleading

"The more pervasive these three motivations are, the more persistent the changes in $\{\beta\}$." (p. 912) Therefore, EM propose that a useful test should maximize its power against persistent changes in $\{\beta_t\}$.

The conditions underlying the EM test allow for diverse breaking models, from relatively rare (including a single break) to very frequent small breaks (such as breaks every period with probability p). Breaks can also occur with a regular pattern, such as every 16 quarters following U.S. presidential elections.

Computation of the q_{LL} test statistic is straightforward, relying only on OLS regressions and construction of an estimate of the long-run covariance matrix of $\{X_t \epsilon_t\}$. For uncorrelated ϵ_t, a robust covariance matrix will suffice. For possibly autocorrelated ϵ_t, a HAC (Newey–West) covariance matrix is appropriate.

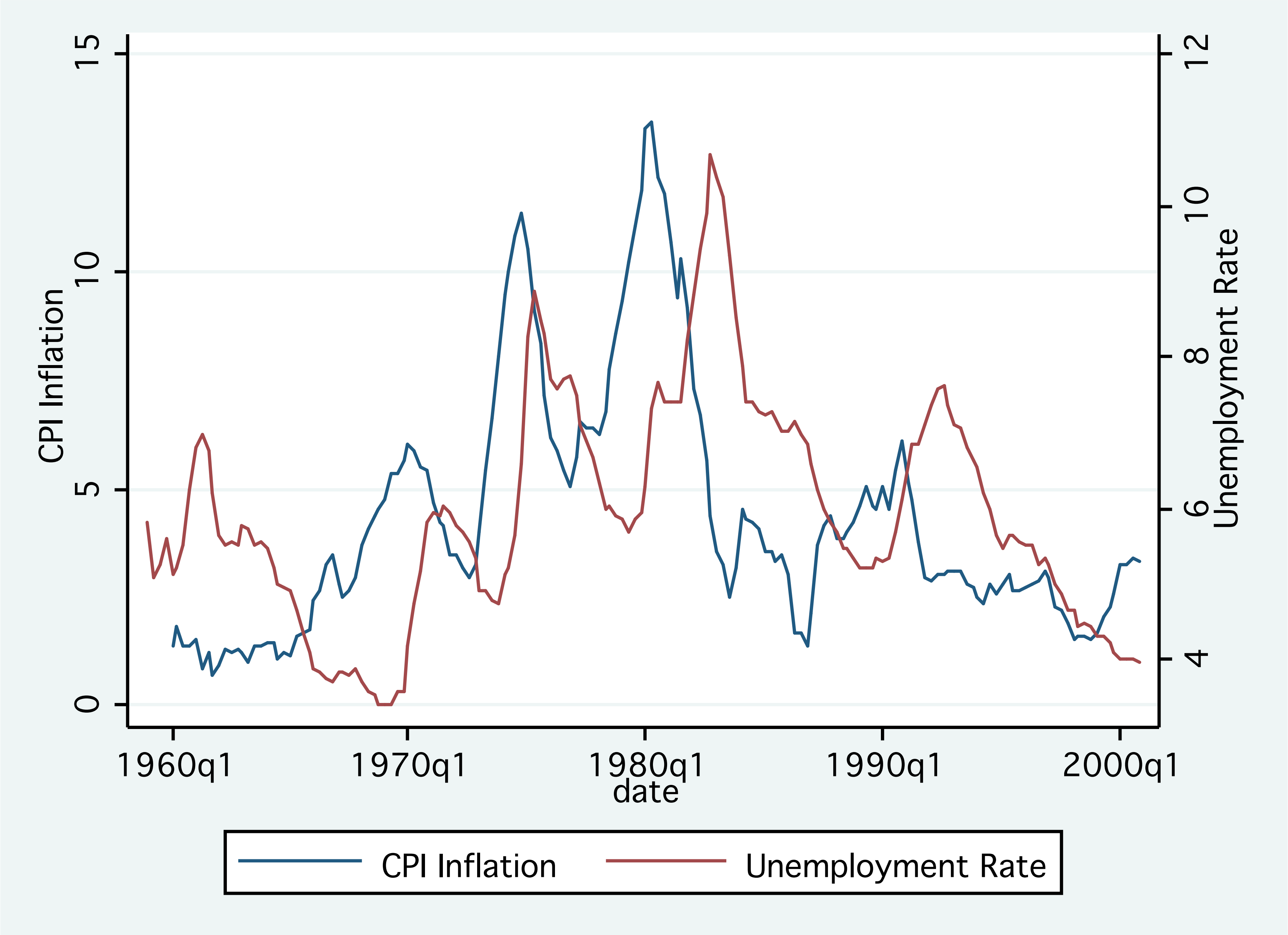
The null hypothesis of parameter stability is rejected for *small* values of $\widehat{q_{LL}}$: that is, values more negative than the critical values. Asymptotic critical values are provided by EM for $k = 1, \ldots, 10$ and are independent of the dimension of Z_t (the set of covariates assumed to have stable coefficients).

The `qll` Stata command implements the EM q_{LL} test using Mata to produce the test statistic. EM's Table 1 of asymptotic critical values is stored in the program and used to produce 10%, 5% and 1% critical values corresponding to number of regressors with potentially unstable parameters. The command syntax:

qll *depvar varlist* [if *exp*] [in *range*] [, (*zvarlist*) rlag(*#*)]

where the parenthesized `zvarlist` optionally specifies the list of covariates assumed to have stable coefficients (none are required). The `rlag` option specifies the number of lags to be used in computing the long-run covariance matrix of $\{X_t \epsilon_t\}$. If a negative value is given, the optimal lag order is chosen by the BIC criterion. The `qll` command is available from SSC.

To illustrate, we consider a regression of inflation on the lagged unemployment rate, the Treasury bill rate and the Treasury bond rate. We assume the latter two coefficients are stable over the period. We test over the full sample and a 1990–2000 subsample.


CPI Inflation
0
5
10
15
Unemployment Rate
4
6
8
10
12
1960q1
1970q1
1980q1
1990q1
2000q1
date
CPI Inflation
Unemployment Rate

```
 . qll inf L.UR (TBILL TBON), rlag(8)
Elliott-Muller qLL test statistic for time varying coefficients
in the model of inf, 1960q1 - 2000q4
Allowing for time variation in 1 regressors
H0: all regression coefficients fixed over the sample period (N = 164)
Test stat.    1% Crit.Val.   5% Crit.Val.   10% Crit.Val.
   -2.260        -11.05         -8.36          -7.14
Long-run variance computed with 8 lags.


. qll inf L.UR (TBILL TBON) if tin(1990q1,), rlag(8)
Elliott-Muller qLL test statistic for time varying coefficients
in the model of inf, 1990q1 - 2000q4
Allowing for time variation in 1 regressors
H0: all regression coefficients fixed over the sample period (N = 44)
Test stat.    1% Crit.Val.   5% Crit.Val.   10% Crit.Val.
   -6.647        -11.05         -8.36          -7.14
Long-run variance computed with 8 lags.
```

In both samples, using eight lags to calculate the long-run covariance matrix, the null hypothesis that the coefficients on the lagged unemployment rate (`L.UR`) are stable cannot be rejected at the 10% level of confidence. The Elliott–Müller q_{LL} test indicates that the stability of this regression model, allowing for instability in the coefficient of the unemployment rate only, cannot be rejected by the data.

Time series filtering

Official Stata contains a number of commands for time series filtering in the `tssmooth` suite, including single and double exponential smoothing; Holt–Winters seasonal and nonseasonal smoothing; moving-average filtering; and nonlinear filtering.

A number of user-written routines provide Stata commands for time series filtering. My `hprescott` command provides the Hodrick–Prescott filter, which may be applied to multiple time series as well as to the time series within a panel using the `by:` prefix. The somewhat similar Butterworth high-pass filter is also available from SSC as `butterworth`.

A recent addition to SSC is Jorge Pérez' implementation of the Corbae–Ouliaris filter, `couliari`. That routine improves upon the Baxter–King filter (my `bking` routine) in its handling of endpoints of the series.

Interpolation

Official Stata provides some facilities for interpolation and extrapolation of time series, such as the `ipolate` command. A nonparametric locally weighted regression interpolation can also be performed by the `lowess` command. Kernel-weighted local polynomial smoothing is available using the `lpoly` command.

In some instances a time series is needed at a higher frequency, but must obey the accounting constraints that the higher-frequency observations sum to the lower-frequency observed series. I have programmed the proportional Denton method, as described in the IMF's *Quarterly National Accounts Manual*, 2001.

The `denton` command, available from SSC, can interpolate annual data to quarterly, subject to adding-up constraints. The associated `dentonmq` command can interpolate quarterly data to monthly frequency. These routines are in the process of being translated into Mata to remove constraints on the length of time series.

A good illustration of how the Denton procedure may be used, by Victor H. Aguiar, is included in your materials.

Aggregation

In some cases, you may want to go the other way, and aggregate higher-frequency data to a lower frequency for presentation or combination with other lower-frequency data. In general terms, this sort of aggregation can be performed with Stata's `collapse` command, which is capable of producing a new data set of 'collapsed' means, counts, standard deviations, or other statistics.

The particular needs of time series modelers suggest that you may want to sum some series, average others over the longer period, and pick beginning-of-period or end-of-period values for others. My `tscollap` routine performs these functions, as well as computing geometric means for growth rates. It can also be applied to panel data, operating on each time series within a panel.

In this example, using the quarterly US macro data set, we create the (geometric) average inflation rate, the end-of-year monetary base, the average oil price over the year and the first quarter's oil price as new series. The data are now `tsset` by the new `year` variable.

```
 . use usmacro1, clear
. tscollap dcpi (gmean) mbase (last) oilprice (mean) foilpr=oilprice (first), /
> *
> */ to(y) gen(yr)


Converting from Q to Y

        time variable:  yr, 1959 to 2009
                delta:  1 year
. summarize
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        dcpi |        51    4.049612    2.875048   -.342528   13.53583
       mbase |        51    313.3805    333.4971   40.81946   1779.044
    oilprice |        51    22.01961    19.83468       2.92    90.6733
      foilpr |        51    21.72855    20.27324       2.92    99.5875
          yr |        51        1984    14.86607       1959       2009
```

Ex ante forecasting

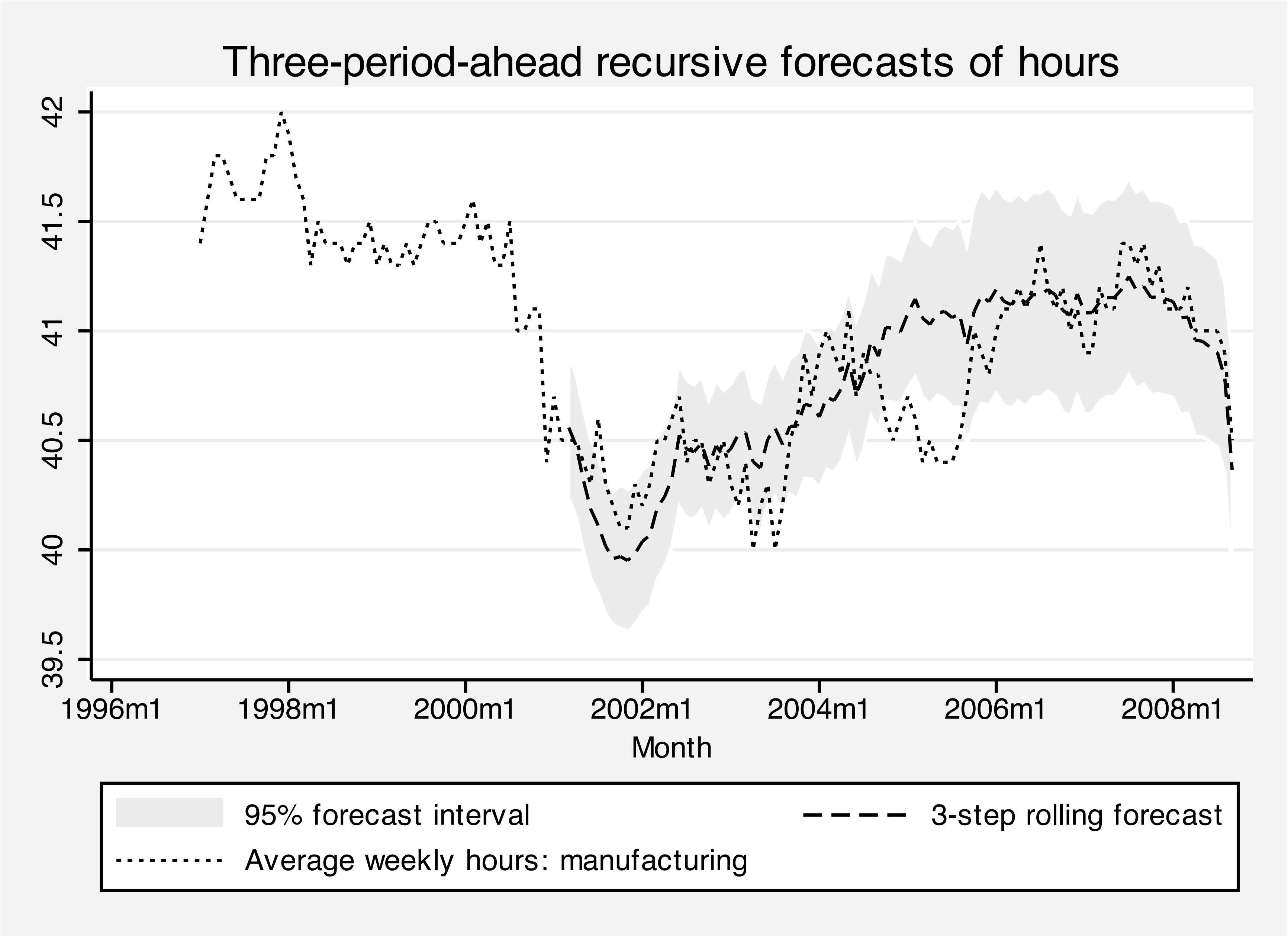
One feature not readily supported by the `rolling:` syntax is the production of a sequence of *ex ante* forecasts from moving-window estimation. This came to light in a discussion with Jim Stock last fall when he said that producing these forecasts was an important capability, but one that could not be readily achieved in the `rolling:` syntax. In response, I wrote a beta version of `staticfc`, which does just that. It is not yet posted on SSC, but is included in your materials.

The routine has a 'required option', `generate( )`, in which you specify a 'stub' from which new variables will be created. The stub itself is the name of the *ex ante* rolling forecast variable, while `stub_s` and `stub_n` will contain the standard error of forecast and number of observations used in the estimation, respectively.

At present, the routine only supports the `rolling:` option of *recursive* estimation: that is, the expanding window. You may choose the initial number of periods to be used in estimation, as well as the number of steps ahead to forecast. The routine only handles static models (lacking lagged dependent variables) at present. Optionally, you may graph the forecast series with its 95% confidence interval.

To illustrate, we use Stata's `manufac` monthly data set and estimate a model of hours as depending on a distributed lag of capital utilization and its logarithm, using 48 months of data as the initial estimation sample. We consider three-step-ahead forecasts.

```
 . webuse manufac, clear
(St. Louis Fed (FRED) manufacturing data)

. tsset
        time variable:  month, 1972m1 to 2008m12
                delta:  1 month

. staticfc hours L(1/2).caputil lncaputil if tin(1997m1,2008m12), ///
> init(48) step(3) gen(cfc4) graph(fig4) replace ///
> ti("Three-period-ahead recursive forecasts of hours")
(file fig4.gph saved)
```


Three-period-ahead recursive forecasts of hours
42
41.5
41
40.5
40
39.5
1996m1
1998m1
2000m1
2002m1
2004m1
2006m1
2008m1
Month
95% forecast interval
3-step rolling forecast
Average weekly hours: manufacturing

ARIMA and ARMAX models

Stata's capabilities to estimate ARIMA or 'Box–Jenkins' models are implemented by the `arima` command. These modeling tools include both the traditional $ARIMA(p, d, q)$ framework as well as multiplicative seasonal ARIMA components.

However, the `arima` command has features that go beyond univariate time series modeling. It also implements ARMAX models: that is, regression equations with ARMA errors. This feature generalizes the capability of Stata's `prais` command to estimate a regression with first-order autoregressive ($AR(1)$) errors. In both the ARIMA and ARMAX contexts, the `arima` command implements dynamic forecasts.

To illustrate, we fit an ARIMA(p,d,q) model to the US consumer price index (CPI):

```
 . use usmacro1
. arima cpi, arima(1, 1, 1) nolog
ARIMA regression
Sample:  1959q2 - 2010q3                        Number of obs      =       206
                                                Wald chi2(2)       =  12657.64
Log likelihood =  -105.364                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
       D.cpi |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
cpi          |
       _cons |   .4711825   .0508081     9.27   0.000     .3716004    .5707646
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |  -.3478959   .0590356    -5.89   0.000    -.4636036   -.2321882
             |
          ma |
         L1. |   .9775208   .0123013    79.46   0.000     .9534106    1.001631
-------------+----------------------------------------------------------------
      /sigma |   .4011922    .008254    48.61   0.000     .3850146    .4173697
------------------------------------------------------------------------------
. estimates store e42a
```

In this example, we use the `arima(p, d, q)` option to specify the model. The `ar( )` and `ma( )` options may also be used separately, in which case a *numlist* of lags to be included is specified. Differencing is then applied to the dependent variable using the `D.` operator. For example:

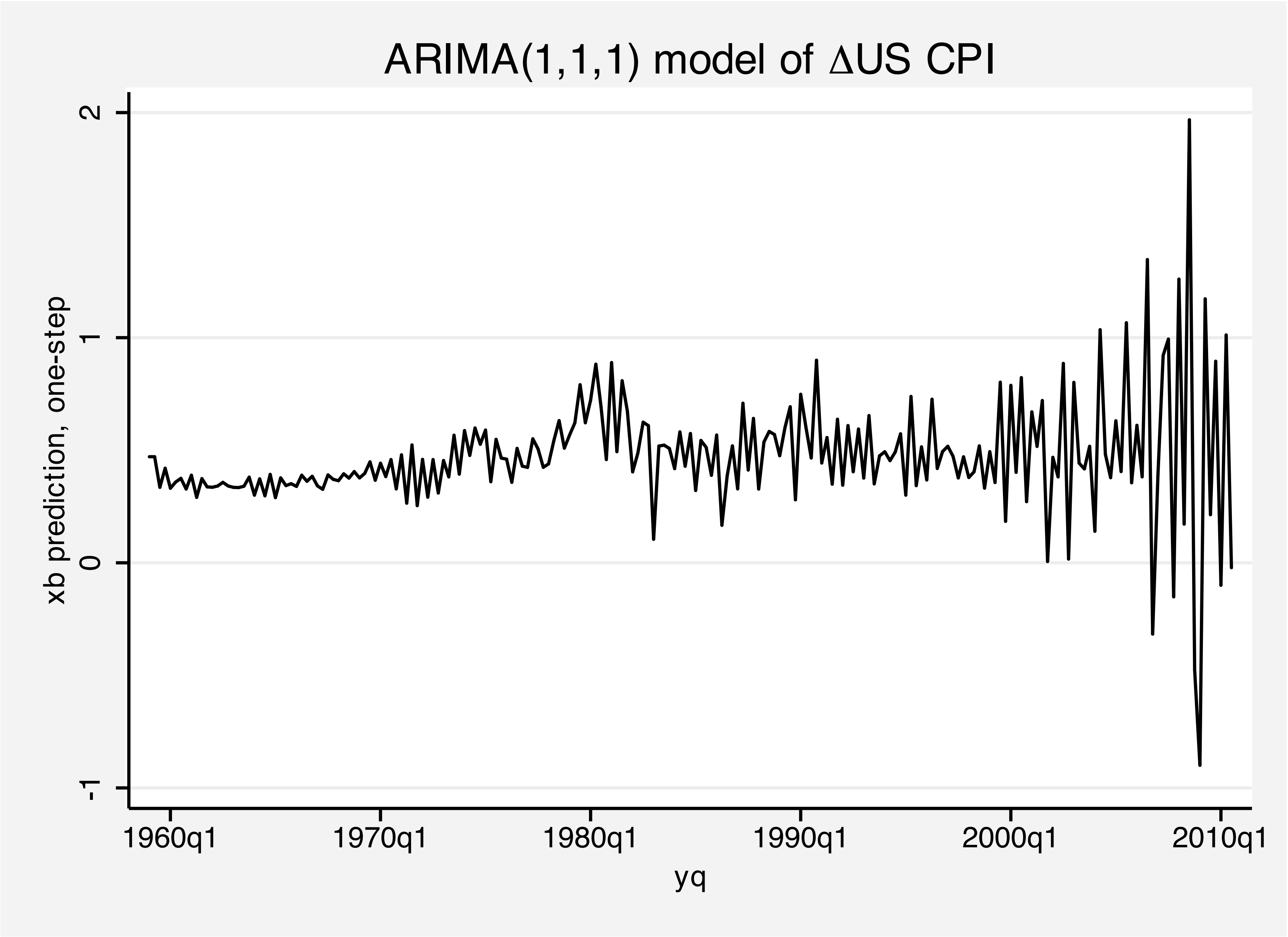
```
 . use usmacro1
. arima D.cpi, ar(1 4) nolog
ARIMA regression
Sample:  1959q2 - 2010q3                        Number of obs      =       206
                                                Wald chi2(2)       =    105.12
Log likelihood = -112.7938                      Prob > chi2        =    0.0000
------------------------------------------------------------------------------
             |                 OPG
       D.cpi |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
cpi          |
       _cons |   .4578741   .1086742     4.21   0.000     .2448766    .6708716
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .3035501   .0686132     4.42   0.000     .1690707    .4380295
         L4. |   .3342019   .0407126     8.21   0.000     .2544068     .413997
-------------+----------------------------------------------------------------
      /sigma |   .4177019   .0071104    58.75   0.000     .4037658    .4316381
------------------------------------------------------------------------------
```

Several prediction options are available after estimating an `arima` model. The default option, `xb`, predicts the actual dependent variable: so if `D.cpi` is the dependent variable, predictions are made for that variable. In contrast, the `y` option generates predictions of the original variable, in this case `cpi`.

The `mse` option calculates the mean squared error of predictions, while `yresiduals` are computed in terms of the original variable.

We recall the estimates from the first model fitted, and calculate predictions for the actual dependent variable, ΔCPI:

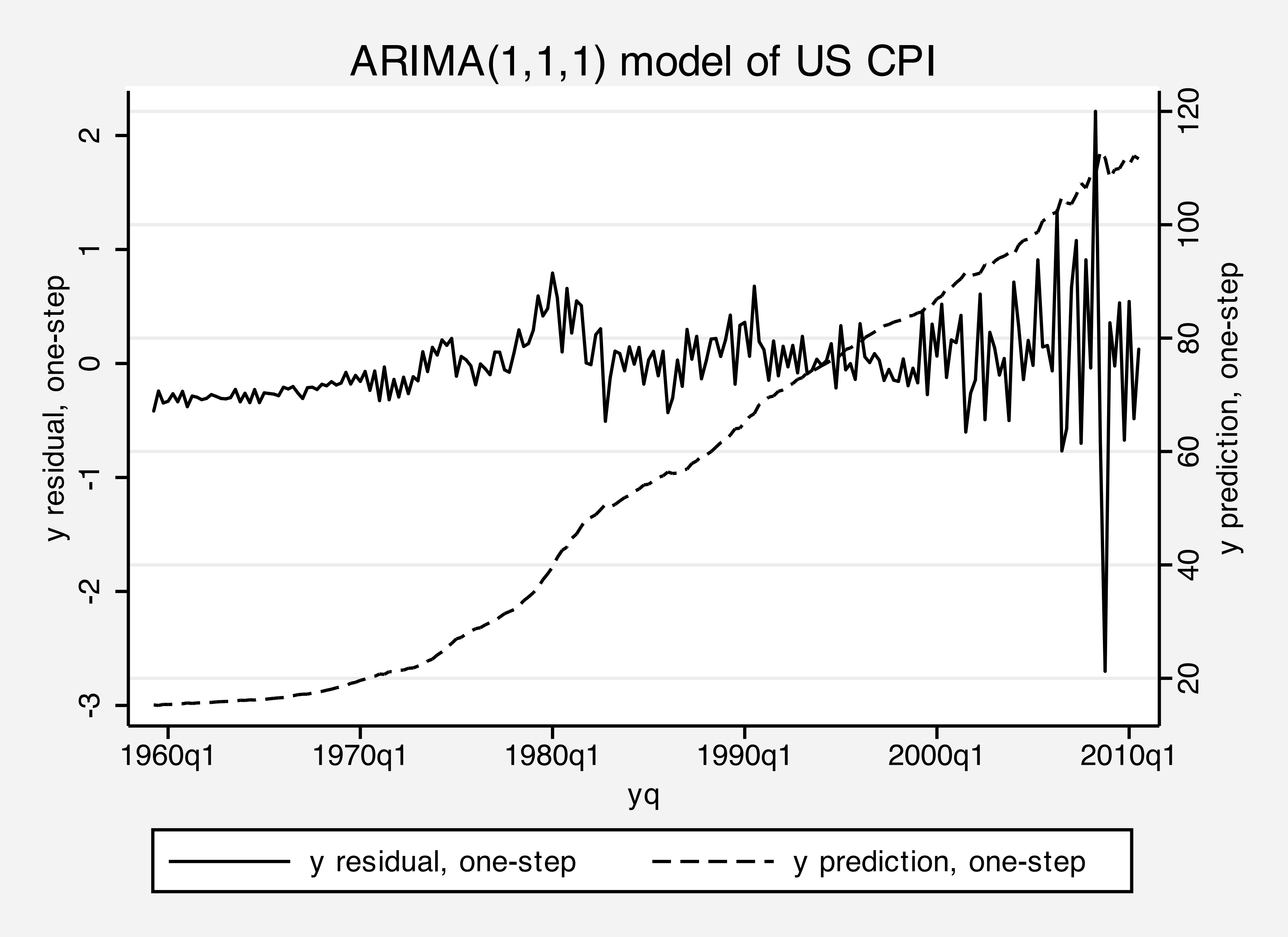
```
. estimates restore e42a
(results e42a are active now)

. predict double dcpihat, xb

. tsline dcpihat, ///
>    ti("ARIMA(1,1,1) model of {&Delta}US CPI") scheme(s2mono)
```


ARIMA(1,1,1) model of ΔUS CPI
xb prediction, one-step
2
1
0
-1
1960q1
1970q1
1980q1
1990q1
2000q1
2010q1
yq

We can see that the predictions are becoming increasingly volatile in recent years.

We may also compute predicted values and residuals for the level of *CPI*:

```
. estimates restore e42a
(results e42a are active now)

. predict double cpihat, y
(1 missing value generated)

. predict double cpieps, yresiduals
(1 missing value generated)

. tw (tsline cpieps, yaxis(2)) (tsline cpihat), ///
>    ti("ARIMA(1,1,1) model of US CPI") scheme(s2mono)
```

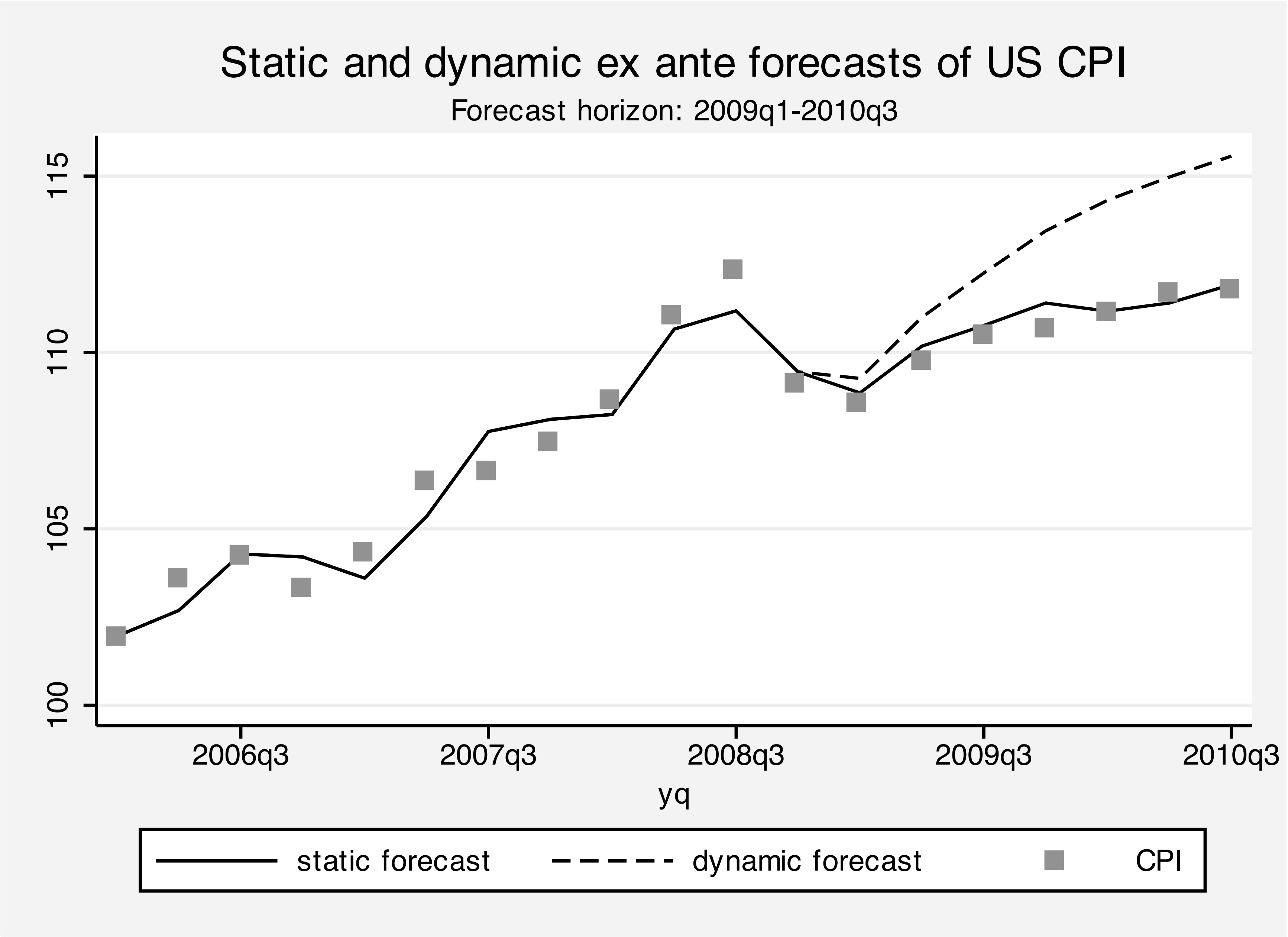

ARIMA(1,1,1) model of US CPI
y residual, one-step
-3
-2
-1
0
1
2
y prediction, one-step
20
40
60
80
100
120
1960q1
1970q1
1980q1
1990q1
2000q1
2010q1
yq
y residual, one-step
y prediction, one-step

We now illustrate the estimation of an ARMAX model of Δcpi as a function of $\Delta oilprice$ with $ARMA(1,1)$ errors. The estimation sample runs through 2008q4.

```
 . arima d.cpi d.oilprice if tin(, 2008q4), ar(1) ma(1) nolog
ARIMA regression
Sample:  1959q2 - 2008q4                        Number of obs      =       199
                                                Wald chi2(3)       =   1829.64
Log likelihood = -27.08681                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
       D.cpi |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
cpi          |
    oilprice |
         D1. |   .0602003   .0021528    27.96   0.000     .0559808    .0644198
             |
       _cons |   .4397912   .1833278     2.40   0.016     .0804753    .7991071
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .9732011   .0296099    32.87   0.000     .9151667    1.031235
             |
          ma |
         L1. |  -.7867952   .0535747   -14.69   0.000    -.8917997   -.6817906
-------------+----------------------------------------------------------------
      /sigma |   .2765534   .0091383    30.26   0.000     .2586426    .2944642
------------------------------------------------------------------------------
```

We compute static (one-period-ahead) *ex ante* forecasts and dynamic (multi-period-ahead) *ex ante* forecasts for 2009q1–2010q3. In specifying the dynamic forecast, the `dynamic( )` option indicates the period in which references to *y* should first evaluate to the prediction of the model rather than historical values. In all prior periods, references to *y* are to the actual data.

```
 . predict double cpihat_s if tin(2006q1,), y
(188 missing values generated)
. label var cpihat_s "static forecast"
. predict double cpihat_d if tin(2006q1,), dynamic(tq(2008q4)) y
(188 missing values generated)
. label var cpihat_d "dynamic forecast"
. tw (tsline cpihat_s cpihat_d if !mi(cpihat_s)) ///
>    (scatter cpi yq if !mi(cpihat_s), c(i)), scheme(s2mono) ///
>    ti("Static and dynamic ex ante forecasts of US CPI") ///
>    t2("Forecast horizon: 2009q1-2010q3") legend(rows(1))
```


Static and dynamic ex ante forecasts of US CPI
Forecast horizon: 2009q1-2010q3
100
105
110
115
2006q3
2007q3
2008q3
2009q3
2010q3
yq
static forecast
dynamic forecast
CPI

Vector autoregressive (VAR) models

Stata has a complete suite of commands for fitting and forecasting vector autoregressive (VAR) models and structural vector autoregressive (SVAR) models. Its capabilities include estimating and interpreting impulse response functions (IRFs), dynamic multipliers, and forecast error vector decompositions (FEVDs).

Subsidiary commands allow you to check the stability condition of VAR or SVAR estimates; to compute log-order selection statistics for VARs; to perform pairwise Granger causality tests for VAR estimates; and to test for residual autocorrelation and normality in the disturbances of VARs.

Dynamic forecasts may be computed and graphed after VAR or SVAR estimation.

A p-th order vector autoregression, or $VAR(p)$, with exogenous variables x can be written as:

$$\mathbf{y}_t = \mathbf{v} + \mathbf{A}_1 \mathbf{y}_{t-1} + \cdots + \mathbf{A}_p \mathbf{y}_{t-p} + \mathbf{B}_0 \mathbf{x}_t + \mathbf{B}_1 \mathbf{B}_{t-1} + \cdots + \mathbf{B}_s \mathbf{x}_{t-s} + \mathbf{u}_t$$

where $\mathbf{y}_t$ is a vector of K variables, each modeled as function of p lags of those variables and, optionally, a set of exogenous variables $\mathbf{x}_t$.

We assume that $E(\mathbf{u}_t) = 0, E(\mathbf{u}_t \mathbf{u}'_t) = \Sigma$ and $E(\mathbf{u}_t \mathbf{u}'_s) = 0 \; \forall t \neq s$.

If the VAR is stable (see command `varstable`) we can rewrite the VAR in moving average form as:

$$\mathbf{y}_t = \mu + \sum_{i=0}^{\infty} \mathbf{D}_i \mathbf{x}_{t-i} + \sum_{i=0}^{\infty} \Phi_i \mathbf{u}_{t-i}$$

which is the vector moving average (VMA) representation of the VAR, where all past values of y_t have been substituted out. The $\mathbf{D}_i$ matrices are the dynamic multiplier functions, or transfer functions. The sequence of moving average coefficients Φ_i are the simple impulse-response functions (IRFs) at horizon i.

Estimation of the parameters of the VAR requires that the variables in $\mathbf{y}_t$ and $\mathbf{x}_t$ are covariance stationary, with their first two moments finite and time-invariant. If the variables in $\mathbf{y}_t$ are not covariance stationary, but their first differences are, they may be modeled with a vector error correction model, or VECM.

In the absence of exogenous variables, the disturbance variance-covariance matrix Σ contains all relevant information about contemporaneous correlation among the variables in $\mathbf{y}_t$. VARs may be *reduced-form* VARs, which do not account for this contemporaneous correlation. They may be *recursive* VARs, where the K variables are assumed to form a recursive dynamic structural model where each variable only depends upon those above it in the vector $\mathbf{y}_t$. Or, they may be *structural* VARs, where theory is used to place restrictions on the contemporaneous correlations.

Stata's `varbasic` command allows you to fit a simple reduced-form VAR without constraints and graph the impulse-response functions (IRFs). The more general `var` command allows for constraints to be placed on the coefficients.

The `varsoc` command allows you to select the appropriate lag order for the VAR; command `varwle` computes Wald tests to determine whether certain lags can be excluded; `varlmar` checks for autocorrelation in the disturbances; and `varstable` checks whether the stability conditions needed to compute IRFs and FEVDs are satisfied.

IRFs, OIRFs and FEVDs

Impulse response functions, or IRFs, measure the effects of a shock to an endogenous variable on itself or on another endogenous variable. Stata's `irf` commands can compute five types of IRFs: *simple* IRFs, *orthogonalized* IRFs, *cumulative* IRFs, *cumulative orthogonalized* IRFs and *structural* IRFs. We defined the simple IRF in an earlier slide.

The forecast error variance decomposition (FEVD) measures the fraction of the forecast error variance of an endogenous variable that can be attributed to orthogonalized shocks to itself or to another endogenous variable.

To analyze IRFs and FEVDs in Stata, you estimate a VAR model and use `irf create` to estimate the IRFs and FEVDs and store them in a file. This step is done automatically by the `varbasic` command, but must be done explicitly after the `var` or `svar` commands. You may then use `irf graph`, `irf table` or other `irf` analysis commands to examine results.

For IRFs to be computed, the VAR must be stable. The simple IRFs shown above have a drawback: they give the effect over time of a one-time unit increase to one of the shocks, holding all else constant. But to the extent the shocks are contemporaneously correlated, the other shocks cannot be held constant, and the VMA form of the VAR cannot have a causal interpretation.

Orthogonalized innovations

We can overcome this difficulty by taking $E(u_t u_t') = \Sigma$, the covariance matrix of shocks, and finding a matrix $\mathbf{P}$ such that $\Sigma = \mathbf{P}\mathbf{P}'$ and $\mathbf{P}^{-1}\Sigma\mathbf{P}'^{-1} = \mathbf{I}_K$. The vector of shocks may then be *orthogonalized* by $\mathbf{P}^{-1}$. For a pure VAR, without exogenous variables,

$$\begin{aligned}
\mathbf{y}_t &= \mu + \sum_{i=0}^{\infty} \Phi_i u_{t-i} \\
&= \mu + \sum_{i=0}^{\infty} \Phi_i \mathbf{P}\mathbf{P}^{-1} u_{t-i} \\
&= \mu + \sum_{i=0}^{\infty} \Theta_i \mathbf{P}^{-1} u_{t-i} \\
&= \mu + \sum_{i=0}^{\infty} \Theta_i w_{t-i}
\end{aligned}$$

Sims (*Econometrica*, 1980) suggests that **P** can be written as the Cholesky decomposition of Σ^{-1}, and IRFs based on this choice are known as the *orthogonalized* IRFs. As a VAR can be considered to be the reduced form of a dynamic structural equation (DSE) model, choosing **P** is equivalent to imposing a recursive structure on the corresponding DSE model. The *ordering* of the recursive structure is that imposed in the Cholesky decomposition, which is that in which the endogenous variables appear in the VAR estimation.

As this choice is somewhat arbitrary, you may want to explore the OIRFs resulting from a different ordering. It is not necessary, using `var` and `irf create`, to reestimate the VAR with a different ordering, as the `order()` option of `irf create` will apply the Cholesky decomposition in the specified order.

Just as the OIRFs are sensitive to the ordering of variables, the FEVDs are defined in terms of a particular causal ordering.

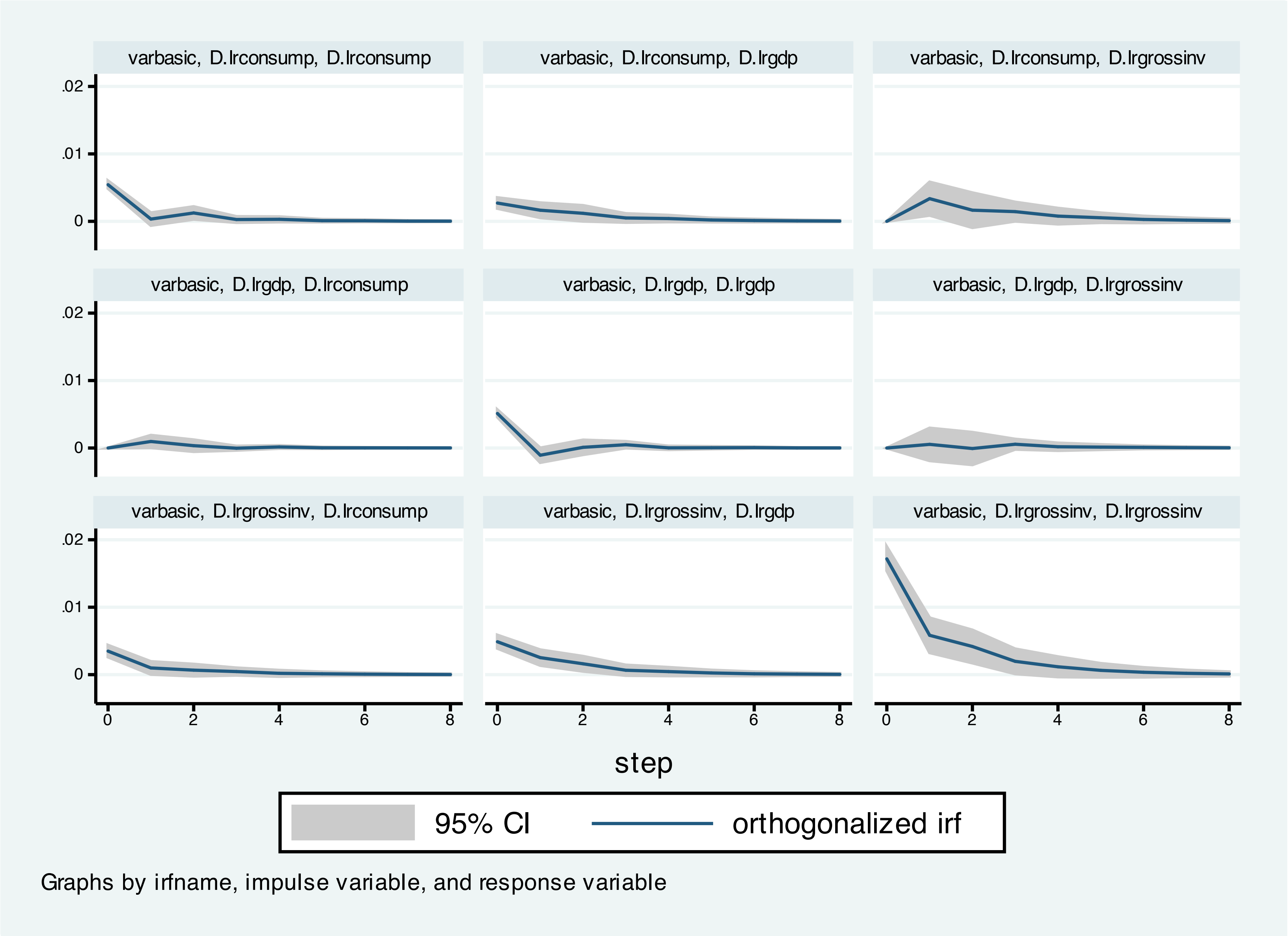
If there are additional (strictly) exogenous variables in the VAR, the dynamic multiplier functions or transfer functions can be computed. These measure the impact of a unit change in the exogenous variable on the endogenous variables over time. They are generated by `fcast compute` and graphed with `fcast graph`.

varbasic

For a simple VAR estimation, you need only specify the `varbasic` *varlist* command. The number of lags, which is given as a *numlist*, defaults to `(1 2)`. Note that you must list every lag to be included; for instance `lags(4)` would only include the fourth lag, whereas `lags(1/4)` would include the first four lags.

Using the usmacro1 dataset, let us estimate a basic VAR for the first differences of log real investment, log real consumption and log real income through 2005q4. By default, the command will produce a graph of the orthogonalized IRFs (OIRFs) for 8 steps ahead. You may choose a different horizon with the `step( )` option.

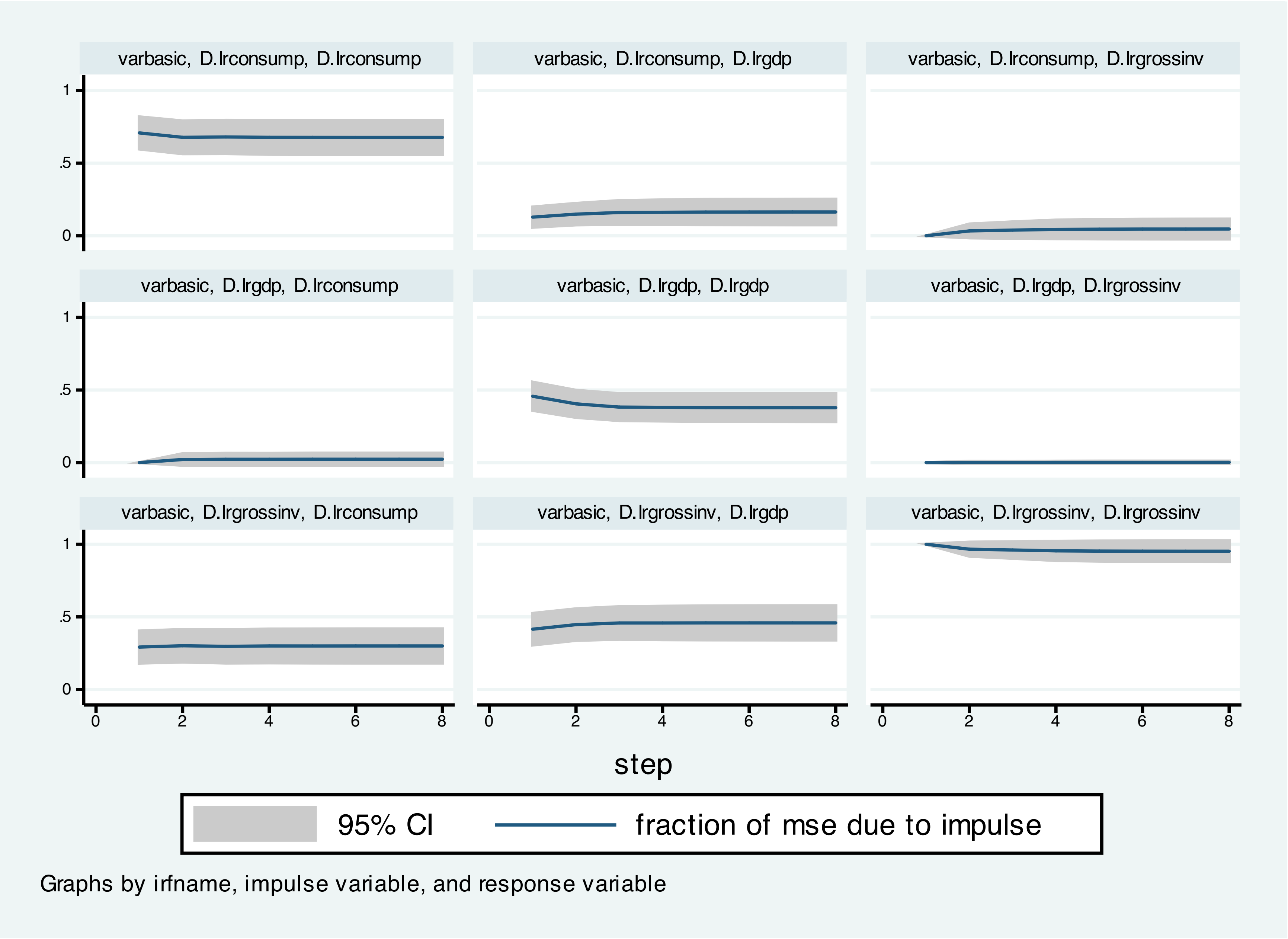
```
 . use usmacro1
. varbasic D.lrgrossinv D.lrconsump D.lrgdp if tin(,2005q4)
Vector autoregression
Sample:  1959q4 - 2005q4                        No. of obs      =       185
Log likelihood =  1905.169                      AIC             =  -20.3694
FPE            =  2.86e-13                      HQIC            = -20.22125
Det(Sigma_ml)  =  2.28e-13                      SBIC            = -20.00385
Equation           Parms      RMSE     R-sq      chi2     P>chi2
----------------------------------------------------------------
D_lrgrossinv          7     .017503   0.2030   47.12655   0.0000
D_lrconsump           7     .006579   0.0994   20.42492   0.0023
D_lrgdp               7     .007722   0.2157   50.88832   0.0000
----------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
D_lrgrossinv |
  lrgrossinv |
         LD. |   .1948761   .0977977     1.99   0.046     .0031962    .3865561
        L2D. |   .1271815   .0981167     1.30   0.195    -.0651237    .3194868
             |
   lrconsump |
         LD. |   .5667047   .2556723     2.22   0.027     .0655963    1.067813
        L2D. |   .1771756   .2567412     0.69   0.490     -.326028    .6803791
             |
       lrgdp |
         LD. |   .1051089   .2399165     0.44   0.661    -.3651189    .5753367
        L2D. |  -.1210883   .2349968    -0.52   0.606    -.5816736    .3394969
             |
```



Graphs by irfname, impulse variable, and response variable

As any of the VAR estimation commands save the estimated IRFs, OIRFs and FEVDs in an `.irf` file, you may examine the FEVDs with a graph command. These items may also be tabulated with the `irf table` and `irf ctable` commands. The latter command allows you to juxtapose tabulated values, such as the OIRF and FEVD for a particular pair of variables, while the `irf cgraph` command allows you to do the same for graphs.

```
. irf graph fevd, lstep(1)
```


varbasic, D.lrconsump, D.lrconsump
varbasic, D.lrconsump, D.lrgdp
varbasic, D.lrconsump, D.lrgrossinv
varbasic, D.lrgdp, D.lrconsump
varbasic, D.lrgdp, D.lrgdp
varbasic, D.lrgdp, D.lrgrossinv
varbasic, D.lrgrossinv, D.lrconsump
varbasic, D.lrgrossinv, D.lrgdp
varbasic, D.lrgrossinv, D.lrgrossinv
1
.5
0
0
2
4
6
8
step
95% CI
fraction of mse due to impulse
Graphs by irfname, impulse variable, and response variable

After producing any graph in Stata, you may save it in Stata's internal format using `graph save` *filename*. This will create a `.gph` file which may be accessed with `graph use`. The file contains all the information necessary to replicate the graph and modify its appearance. However, only Stata can read `.gph` files. If you want to reproduce the graph in a document, use the `graph export` *filename*.*format* command, where *format* is `.eps` (for a Windows or Linux system) or `.pdf` for Mac OS X.

We now consider a model fit with `var` to the same three variables, adding the change in the log of the real money base as an exogenous variable. We include four lags in the VAR.

```
 . var D.lrgrossinv D.lrconsump D.lrgdp if tin(,2005q4), ///
> lags(1/4) exog(D.lrmbase)
Vector autoregression
Sample:  1960q2 - 2005q4                          No. of obs      =       183
Log likelihood =  1907.061                        AIC             = -20.38318
FPE            =  2.82e-13                        HQIC            =  -20.0846
Det(Sigma_ml)  =  1.78e-13                        SBIC            = -19.64658
Equation           Parms      RMSE     R-sq      chi2     P>chi2
----------------------------------------------------------------

D_lrgrossinv         14     .017331   0.2426   58.60225   0.0000
D_lrconsump          14     .006487   0.1640   35.90802   0.0006
D_lrgdp              14     .007433   0.2989   78.02177   0.0000
----------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
D_lrgrossinv |
  lrgrossinv |
         LD. |   .2337044   .0970048     2.41   0.016     .0435785    .4238303
        L2D. |   .0746063   .0997035     0.75   0.454    -.1208089    .2700215
        L3D. |  -.1986633   .1011362    -1.96   0.049    -.3968866   -.0004401
        L4D. |   .1517106   .1004397     1.51   0.131    -.0451476    .3485688
             |
   lrconsump |
         LD. |   .4716336   .2613373     1.80   0.071     -.040578    .9838452
        L2D. |   .1322693   .2758129     0.48   0.632     -.408314    .6728527
        L3D. |   .2471462   .2697096     0.92   0.359     -.281475    .7757673
        L4D. |  -.0177416   .2558472    -0.07   0.945    -.5191928    .4837097
             |
         LD. |   .1354875   .2455182     0.55   0.581    -.3457193    .6166942
```

To evaluate whether the money base variable should be included in the VAR, we can use testparm to construct a joint test of significance of its coefficients:

```
. testparm D.lrmbase
( 1)  [D_lrgrossinv]D.lrmbase = 0
( 2)  [D_lrconsump]D.lrmbase = 0
( 3)  [D_lrgdp]D.lrmbase = 0
          chi2(  3) =    7.95
        Prob > chi2 =    0.0471
```

The variable is marginally significant in the estimated system.

A common diagnostic from a VAR are the set of block *F* tests, or Granger causality tests, that consider whether each variable plays a significant role in each of the equations. These tests may help to establish a sensible causal ordering. They can be performed by `vargranger`:

```
. vargranger

  Granger causality Wald tests
 +------------------------------------------------------------------+
 |          Equation           Excluded |   chi2     df Prob > chi2 |
 |--------------------------------------+---------------------------|
 |      D_lrgrossinv        D.lrconsump |  4.2531     4    0.373    |
 |      D_lrgrossinv           D.lrgdp  |  1.0999     4    0.894    |
 |      D_lrgrossinv               ALL  |   10.34     8    0.242    |
 |--------------------------------------+---------------------------|
 |       D_lrconsump       D.lrgrossinv |  5.8806     4    0.208    |
 |       D_lrconsump           D.lrgdp  |  8.1826     4    0.085    |
 |       D_lrconsump               ALL  |  12.647     8    0.125    |
 |--------------------------------------+---------------------------|
 |           D_lrgdp       D.lrgrossinv |  22.204     4    0.000    |
 |           D_lrgdp        D.lrconsump |  11.349     4    0.023    |
 |           D_lrgdp               ALL  |   42.98     8    0.000    |
 +------------------------------------------------------------------+
```

We may also want to compute selection order criteria to gauge whether we have included sufficient lags in the VAR. Introducing too many lags wastes degrees of freedom, while too few lags leave the equations potentially misspecified and are likely to cause autocorrelation in the residuals. The `varsoc` command will produce selection order criteria, and highlight the optimal lag.

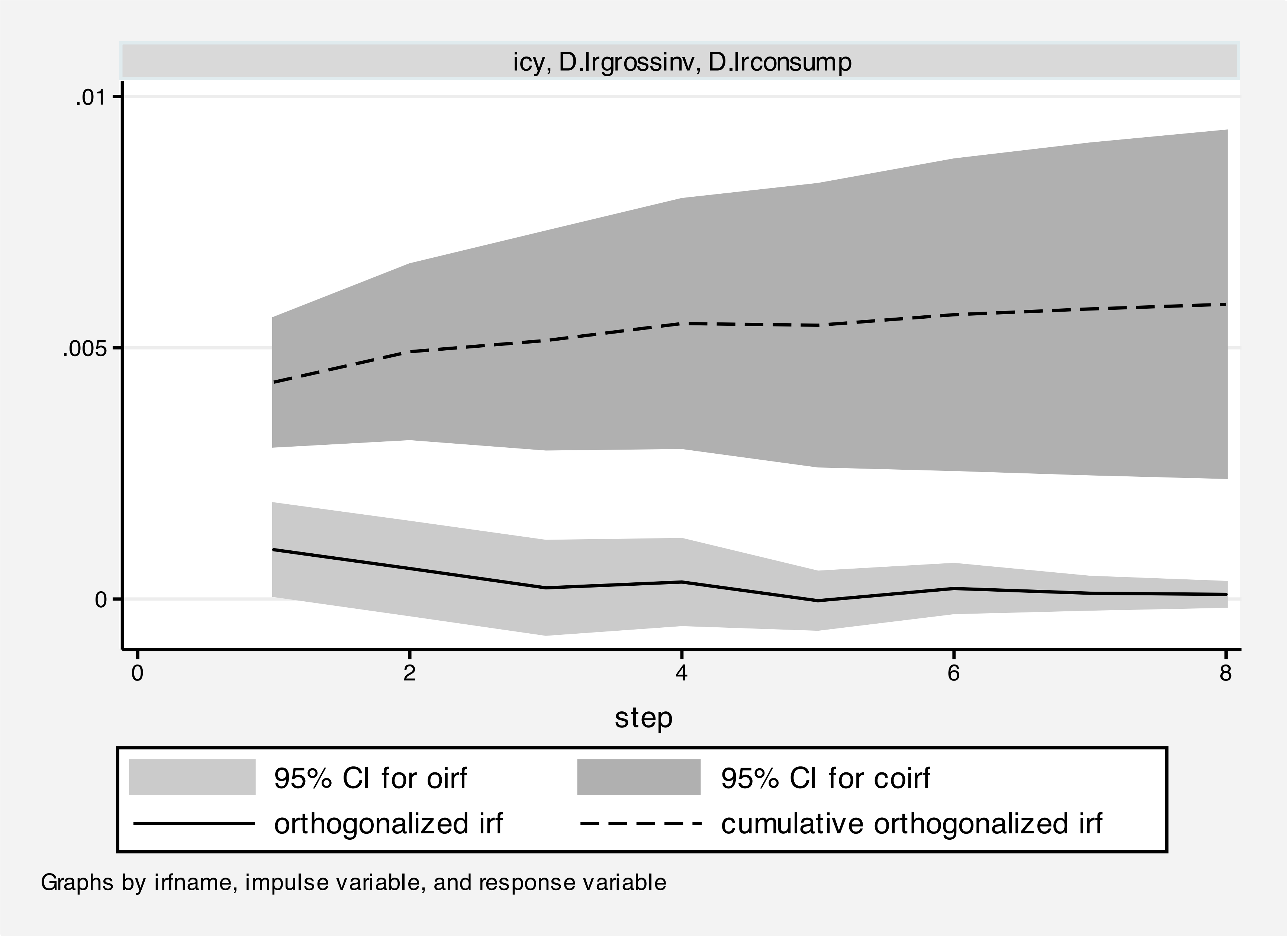
```
. varsoc
  Selection-order criteria
  Sample:  1960q2 - 2005q4                     Number of obs      =       183
 +---------------------------------------------------------------------------+
 |lag |    LL      LR      df    p      FPE       AIC       HQIC      SBIC    |
 |----+----------------------------------------------------------------------|
 |  0 |  1851.22                      3.5e-13  -20.1663  -20.1237  -20.0611   |
 |  1 |  1887.29  72.138*   9  0.000  2.6e-13* -20.4622* -20.3555* -20.1991*  |
 |  2 |  1894.14  13.716    9  0.133  2.7e-13  -20.4387  -20.2681  -20.0178   |
 |  3 |  1902.58  16.866    9  0.051  2.7e-13  -20.4325  -20.1979  -19.8538   |
 |  4 |  1907.06  8.9665    9  0.440  2.8e-13  -20.3832  -20.0846  -19.6466   |
 +---------------------------------------------------------------------------+
  Endogenous:  D.lrgrossinv D.lrconsump D.lrgdp
   Exogenous:  D.lrmbase  _cons
```

We should also be concerned with stability of the VAR, which requires the moduli of the eigenvalues of the dynamic matrix to lie within the unit circle. As there is more than one lag in the VAR we have estimated, it is likely that complex eigenvalues, leading to cycles, will be encountered.

```
. varstable

  Eigenvalue stability condition

  ┌───────────────────────────┬──────────────┐
  │        Eigenvalue         │   Modulus    │
  ├───────────────────────────┼──────────────┤
  │                           │              │
  │    .6916791               │   .691679    │
  │   -.5793137 +   .1840599i │   .607851    │
  │   -.5793137 -   .1840599i │   .607851    │
  │   -.3792302 +   .4714717i │   .605063    │
  │   -.3792302 -   .4714717i │   .605063    │
  │    .1193592 +   .5921967i │   .604106    │
  │    .1193592 -   .5921967i │   .604106    │
  │    .5317127 +   .2672997i │    .59512    │
  │    .5317127 -   .2672997i │    .59512    │
  │   -.4579249               │   .457925    │
  │    .1692559 +   .3870966i │   .422482    │
  │    .1692559 -   .3870966i │   .422482    │
  └───────────────────────────┴──────────────┘
   All the eigenvalues lie inside the unit circle.
   VAR satisfies stability condition.
```

As the estimated VAR appears stable, we can produce IRFs and FEVDs in tabular or graphical form:

```
 . irf create icy, step(8) set(res1)
(file res1.irf created)
(file res1.irf now active)
(file res1.irf updated)
. irf table oirf coirf, impulse(D.lrgrossinv) response(D.lrconsump) noci stderr
> or

                    Results from icy

          |      (1)          (1)     |      (1)          (1)
  step    |     oirf         S.E.     |     coirf        S.E.
----------+---------------------------+---------------------------
0         |   .003334      .000427    |   .003334      .000427
1         |   .000981      .000465    |   .004315      .000648
2         |   .000607      .000468    |   .004922      .000882
3         |   .000223      .000471    |   .005145      .001101
4         |   .000338      .000431    |   .005483      .001258
5         |  -.000034      .000289    |   .005449      .001428
6         |   .000209      .000244    |   .005658      .001571
7         |   .000115      .000161    |   .005773      .001674
8         |   .000092      .00012     |   .005865      .001757

(1) irfname = icy, impulse = D.lrgrossinv, and response = D.lrconsump
. irf graph oirf coirf, impulse(D.lrgrossinv) response(D.lrconsump) ///
>     lstep(1) scheme(s2mono)
```


icy, D.lrgrossinv, D.lrconsump
.01
.005
0
0
2
4
6
8
step
95% CI for oirf
95% CI for coirf
orthogonalized irf
cumulative orthogonalized irf
Graphs by irfname, impulse variable, and response variable

Structural VAR estimation

All of the capabilities we have illustrated for reduced-form VARs are also available for *structural* VARs, which are estimated with the `svar` command. In the SVAR framework, the orthogonalization matrix **P** is not constructed manually as the Cholesky decomposition of the error covariance matrix. Instead, restrictions are placed on the **P** matrix, either in terms of short-run restrictions on the contemporaneous covariances between shocks, or in terms of restrictions on the long-run accumulated effects of the shocks.

Short-run SVAR models

A short-run SVAR model without exogenous variables can be written as

$$\mathbf{A}(\mathbf{I}_K - \mathbf{A}_1 L - \mathbf{A}_2 L^2 - \cdots - \mathbf{A}_p L^p)\mathbf{y}_t = \mathbf{A}\epsilon_t = \mathbf{B}\mathbf{e}_t$$

where L is the lag operator. The vector ϵ_t refers to the original shocks in the model, with covariance matrix Σ, while the vector $\mathbf{e}_t$ are a set of orthogonalized disturbances with covariance matrix $\mathbf{I}_K$.

In a short-run SVAR, we obtain identification by placing restrictions on the matrices **A** and **B**, which are assumed to be nonsingular. The orthgonalization matrix $\mathbf{P}_{sr} = \mathbf{A}^{-1}\mathbf{B}$ is then related to the error covariance matrix by $\Sigma = \mathbf{P}_{sr}\mathbf{P}'_{sr}$.

As there are $K(K+1)/2$ free parameters in Σ, given its symmetric nature, only that many parameters may be estimated in the **A** and **B** matrices. As there are $2K^2$ parameters in **A** and **B**, the order condition for identification requires that $2K^2 - K(K+1)/2$ restrictions be placed on the elements of these matrices.

For instance, we could reproduce the effect of the Cholesky decomposition by defining matrices **A** and **B** appropriately. In the syntax of `svar`, a missing value in a matrix is a free parameter to be estimated. The form of the **A** matrix imposes the recursive structure, while the diagonal **B** orthogonalizes the effects of innovations.

```
 . matrix A = (1, 0, 0 \ ., 1, 0 \ ., ., 1)
. matrix B = (., 0, 0 \ 0, ., 0 \ 0, 0, 1)
. matrix list A
A[3,3]
    c1  c2  c3
r1   1   0   0
r2   .   1   0
r3   .   .   1
. matrix list B
symmetric B[3,3]
    c1  c2  c3
r1   .
r2   0   .
r3   0   0   1
```

```
. svar D.lrgrossinv D.lrconsump D.lrgdp if tin(,2005q4), aeq(A) beq(B) nolog
Estimating short-run parameters

Structural vector autoregression

 ( 1)  [a_1_1]_cons = 1
 ( 2)  [a_1_2]_cons = 0
 ( 3)  [a_1_3]_cons = 0
 ( 4)  [a_2_2]_cons = 1
 ( 5)  [a_2_3]_cons = 0
 ( 6)  [a_3_3]_cons = 1
 ( 7)  [b_1_2]_cons = 0
 ( 8)  [b_1_3]_cons = 0
 ( 9)  [b_2_1]_cons = 0
 (10)  [b_2_3]_cons = 0
 (11)  [b_3_1]_cons = 0
 (12)  [b_3_2]_cons = 0

Sample:  1959q4 - 2005q4                           No. of obs      =       185
Exactly identified model                           Log likelihood  =  1905.169

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      /a_1_1 |          1          .        .       .            .           .
      /a_2_1 |  -.2030461   .0232562    -8.73   0.000    -.2486274   -.1574649
      /a_3_1 |  -.1827889   .0260518    -7.02   0.000    -.2338495   -.1317283
      /a_1_2 |  (omitted)
      /a_2_2 |          1          .        .       .            .           .
      /a_3_2 |  -.4994815    .069309    -7.21   0.000    -.6353246   -.3636384
      /a_1_3 |  (omitted)
      /a_2_3 |  (omitted)
      /a_3_3 |          1          .        .       .            .           .
      /b_1_1 |   .0171686   .0008926    19.24   0.000     .0154193     .018918
```

The output from the VAR can also be displayed with the `var` option. This model is exactly identified; if we impose additional restrictions on the parameters, it would be an overidentified model, and the overidentifying restrictions could be tested.

For instance, we could impose the restriction that $\mathbf{A}_{2,1} = 0$ by placing a zero in that cell of the matrix rather than a missing value. This implies that changes in the first variable (`D.lrgrossinv`) do not contemporaneously affect the second variable, (`D.lrconsump`).

```
 . matrix Arest = (1, 0, 0 \ 0, 1, 0 \ ., ., 1)
. matrix list Arest
Arest[3,3]
    c1  c2  c3
r1   1   0   0
r2   0   1   0
r3   .   .   1
```

```
 . svar D.lrgrossinv D.lrconsump D.lrgdp if tin(,2005q4), aeq(Arest) beq(B) nolog
Estimating short-run parameters
Structural vector autoregression
...
Sample:  1959q4 - 2005q4                           No. of obs      =       185
Overidentified model                               Log likelihood  =  1873.254
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      /a_1_1 |          1          .        .       .            .           .
      /a_2_1 |  (omitted)
      /a_3_1 |  -.1827926   .0219237    -8.34   0.000    -.2257622   -.1398229
      /a_1_2 |  (omitted)
      /a_2_2 |          1          .        .       .            .           .
      /a_3_2 |   -.499383   .0583265    -8.56   0.000    -.6137008   -.3850652
      /a_1_3 |  (omitted)
      /a_2_3 |  (omitted)
      /a_3_3 |          1          .        .       .            .           .
...
-------------+----------------------------------------------------------------
LR test of identifying restrictions:  chi2(  1)=   63.83  Prob > chi2 = 0.000
```

As we would expect from the significant coefficient in the exactly identified VAR, the overidentifying restriction is clearly rejected.

Long-run SVAR models

A short-run SVAR model without exogenous variables can be written as

$$\mathbf{A}(\mathbf{I}_K - \mathbf{A}_1 L - \mathbf{A}_2 L^2 - \cdots - \mathbf{A}_p L^p)\mathbf{y}_t = \mathbf{A}\bar{\mathbf{A}}\ \mathbf{y}_t = \mathbf{B}\ \mathbf{e}_t$$

where $\bar{\mathbf{A}}$ is the parenthesized expression. If we set $\mathbf{A} = \mathbf{I}$, we can write this equation as

$$\mathbf{y}_t = \bar{\mathbf{A}}^{-1}\mathbf{B}\ \mathbf{e}_t = \mathbf{C}\ \mathbf{e}_t$$

In a long-run SVAR, constraints are placed on elements of the $\mathbf{C}$ matrix. These constraints are often exclusion restrictions. For instance, constraining $\mathbf{C}_{1,2} = 0$ forces the long-run response of variable 1 to a shock to variable 2 to zero.

We illustrate with a two-variable SVAR in the first differences in the logs of real money and real GDP. The long-run restrictions of a diagonal **C** matrix implies that shocks to the money supply process have no long-run effects on GDP growth, and shocks to the GDP process have no long-run effects on the money supply.

```
 . matrix lr = (., 0\0, .)
. matrix list lr
symmetric lr[2,2]
    c1  c2
r1   .
r2   0   .
```

```
 . svar D.lrmbase D.lrgdp, lags(4) lreq(lr) nolog
Estimating long-run parameters
Structural vector autoregression
 ( 1)   [c_1_2]_cons = 0
 ( 2)   [c_2_1]_cons = 0
Sample:  1960q2 - 2010q3                          No. of obs      =        202
Overidentified model                              Log likelihood  =   1020.662
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      /c_1_1 |   .0524697   .0026105    20.10   0.000     .0473532    .0575861
      /c_2_1 |  (omitted)
      /c_1_2 |  (omitted)
      /c_2_2 |   .0093022   .0004628    20.10   0.000     .0083951    .0102092
------------------------------------------------------------------------------
LR test of identifying restrictions:  chi2(  1)=    1.448  Prob > chi2 = 0.229
```

The test of overidentifying restrictions cannot reject the validity of the constraints imposed on the long-run responses.

Vector error correction models (VECMs)

VECMs may be estimated by Stata's `vec` command. These models are employed because many economic time series appear to be 'first-difference stationary,' with their levels exhibiting unit root or nonstationary behavior. Conventional regression estimators, including VARs, have good properties when applied to covariance-stationary time series, but encounter difficulties when applied to nonstationary or integrated processes.

These difficulties were illustrated by Granger and Newbold (*J. Econometrics*, 1974) when they introduced the concept of *spurious regressions*. If you have two independent random walk processes, a regression of one on the other will yield a significant coefficient, even though they are not related in any way.

This insight, and Nelson and Plosser's findings (*J. Mon. Ec.*, 1982) that unit roots might be present in a wide variety of macroeconomic series in levels or logarithms, gave rise to the industry of unit root testing, and the implication that variables should be rendered stationary by differencing before they are included in an econometric model.

Further theoretical developments by Granger and Engle in their celebrated paper (*Econometrica*, 1987) raised the possibility that two or more integrated, nonstationary time series might be *cointegrated*, so that some linear combination of these series could be stationary even though each series is not.

If two series are both integrated (of order one, or $I(1)$) we could model their interrelationship by taking first differences of each series and including the differences in a VAR or a structural model.

However, this approach would be suboptimal if it was determined that these series are indeed cointegrated. In that case, the VAR would only express the short-run responses of these series to innovations in each series. This implies that the simple regression in first differences is misspecified.

If the series are cointegrated, they move together in the long run. A VAR in first differences, although properly specified in terms of covariance-stationary series, will not capture those long-run tendences.

Accordingly, the VAR concept may be extended to the vector error-correction model, or VECM, where there is evidence of cointegration among two or more series. The model is fit to the first differences of the nonstationary variables, but a lagged *error-correction term* is added to the relationship.

In the case of two variables, this term is the lagged residual from the cointegrating regression, of one of the series on the other in levels. It expresses the prior disequilibrium from the long-run relationship, in which that residual would be zero.

In the case of multiple variables, there is a vector of error-correction terms, of length equal to the number of cointegrating relationships, or *cointegrating vectors*, among the series.

In terms of economic content, we might expect that there is some long-run value of the dividend/price ratio for common equities. During market ‘bubbles’, the stock price index may be high and the ratio low, but we would expect a market correction to return the ratio to its long-run value. A similar rationale can be offered about the ratio of rents to housing prices in a housing market where there is potential to construct new rental housing as well as single-family homes.

To extend the concept to more than two variables, we might rely on the concept of purchasing power parity (PPP) in international trade, which defines a relationship between the nominal exchange rate and the price indices in the foreign and domestic economies. We might find episodes where a currency appears over- or undervalued, but in the absence of central bank intervention and effective exchange controls, we expect that the ‘law of one price’ will provide some long-run anchor to these three measures’ relationship.

Consider two series, y_t and x_t, that obey the following equations:

$$\begin{aligned} y_t + \beta x_t &= \epsilon_t, \;\; \epsilon_t = \epsilon_{t-1} + \omega_t \\ y_t + \alpha x_t &= \nu_t, \;\; \nu_t = \rho \nu_{t-1} + \zeta_t, \;\; |\rho| < 1 \end{aligned}$$

Assume that ω_t and ζ_t are *i.i.d.* disturbances, correlated with each other. The random-walk nature of ϵ_t implies that both y_t and x_t are also $I(1)$, or nonstationary, as each side of the equation must have the same order of integration. By the same token, the stationary nature of the ν_t process implies that the linear combination $(y_t + \alpha x_t)$ must also be stationary, or $I(0)$.

Thus y_t and x_t cointegrate, with a cointegrating vector $(1, \alpha)$.

We can rewrite the system as

$$\begin{aligned}\Delta y_t &= \beta\delta z_{t-1} + \eta_{1t} \\ \Delta x_t &= -\delta z_{t-1} + \eta_{2t}\end{aligned}$$

where $\delta = (1-\rho)/(\alpha-\beta),\ z_t = y_t + \alpha x_t$, and the errors (η_{1t}, η_{2t}) are stationary linear combinations of (ω_t, ζ_t).

When y_t and x_t are in equilibrium, $z_t = 0$. The coefficients on z_t indicate how the system responds to disequilibrium. A stable dynamic system must exhibit *negative feedback*: for instance, in a functioning market, excess demand must cause the price to rise to clear the market.

In the case of two nonstationary ($I(1)$) variables y_t and x_t, if there are two nonzero values (a, b) such that $ay_t + bx_t$ is stationary, or $I(0)$, then the variables are cointegrated. To identify the cointegrating vector, we set one of the values (a, b) to 1 and estimate the other. As Granger and Engle showed, this can be done by a regression in levels. If the residuals from that 'Granger–Engle' regression are stationary, cointegration is established.

In the general case of K variables, there may be 1, 2,. . . ,(K-1) cointegrating vectors representing stationary linear combinations. That is, if $\mathbf{y}_t$ is a vector of $I(1)$ variables and there exists a vector β such that $\beta\mathbf{y}_t$ is a vector of $I(0)$ variables, then the variables in $\mathbf{y}_t$ are said to be cointegrated with cointegrating vector β. In that case we need to estimate the number of cointegrating relationships, not merely whether cointegration exists among these series.

For a K-variable VAR with p lags,

$$\mathbf{y}_t = \mathbf{v} + \mathbf{A}_1 \mathbf{y}_{t-1} + \cdots + \mathbf{A}_p \mathbf{y}_{t-p} + \epsilon_t$$

let ϵ_t be *i.i.d.* normal over time with covariance matrix Σ. We may rewrite the VAR as a VECM:

$$\Delta \mathbf{y}_t = v + \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{y}_{t-i} + \epsilon_t$$

where $\Pi = \sum_{j=1}^{j=p} \mathbf{A}_j - \mathbf{I}_k$ and $\Gamma_i = -\sum_{j=i+1}^{j=p} \mathbf{A}_j$.

If all variables in $\mathbf{y}_t$ are $I(1)$, the matrix Π has rank $0 \leq r < K$, where r is the number of linearly independent cointegrating vectors. If the variables are cointegrated ($r > 0$) the VAR in first differences is misspecified as it excludes the error correction term.

If the rank of $\Pi = 0$, there is no cointegration among the nonstationary variables, and a VAR in their first differences is consistent.

If the rank of $\Pi = K$, all of the variables in $\mathbf{y}_t$ are $I(0)$ and a VAR in their levels is consistent.

If the rank of Π is $r > 0$, it may be expressed as $\Pi = \alpha\beta'$, where α and β are $(K \times r)$ matrices of rank r. We must place restrictions on these matrices' elements in order to identify the system.

Stata's implementation of VECM modeling is based on the maximum likelihood framework of Johansen (*J. Ec. Dyn. Ctrl.*, 1988 and subsequent works). In that framework, deterministic trends can appear in the means of the differenced series, or in the mean of the cointegrating relationship. The constant term in the VECM implies a linear trend in the levels of the variables. Thus, a time trend in the equation implies quadratic trends in the level data.

Writing the matrix of coefficients on the vector error correction term $\mathbf{y}_{t-1}$ as $\Pi = \alpha\beta'$, we can incorporate a trend in the cointegrating relationship and the equation itself as

$$\Delta \mathbf{y}_t = \alpha(\beta' \mathbf{y}_{t-1} + \mu + \rho t) + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{y}_{t-i} + \gamma + \tau t + \epsilon_t$$

Johansen spells out five cases for estimation of the VECM:

1. Unrestricted trend: estimated as shown, cointegrating equations are trend stationary
2. Restricted trend, $\tau = 0$: cointegrating equations are trend stationary, and trends in levels are linear but not quadratic
3. Unrestricted constant: $\tau = \rho = 0$: cointegrating equations are stationary around constant means, linear trend in levels
4. Restricted constant: $\tau = \rho = \gamma = 0$: cointegrating equations are stationary around constant means, no linear time trends in the data
5. No trend: $\tau = \rho = \gamma = \mu = 0$: cointegrating equations, levels and differences of the data have means of zero

We have not illustrated VECMs with additional (strictly) exogenous variables, but they may be added, just as in a VAR model.

To consistently test for cointegration, we must choose the appropriate lag length. The `varsoc` command is capable of making that determination, as illustrated earlier. We may then use the `vecrank` command to test for cointegration via Johansen's max-eigenvalue statistic and trace statistic.

We illustrate a simple VECM using the Penn World Tables data. In that data set, the price index is the relative price vs. the US, and the nominal exchange rate is expressed as local currency units per US dollar. If the real exchange rate is a cointegrating combination, the logs of the price index and the nominal exchange rate should be cointegrated. We test this hypothesis with respect to the UK, using Stata's default of an unrestricted constant in the taxonomy given above.

```
 . use pwt6_3, clear
(Penn World Tables 6.3, August 2009)

. keep if inlist(isocode,"GBR")
(10962 observations deleted)

. // p already defined as UK/US relative price
. g lp = log(p)

. // xrat is nominal exchange rate, GBP per USD
. g lxrat = log(xrat)

. varsoc lp lxrat if tin(,2002)

   Selection-order criteria
   Sample:  1954 - 2002                         Number of obs      =        49
  +---------------------------------------------------------------------------+
  |lag |    LL      LR      df    p      FPE       AIC      HQIC      SBIC    |
  |----+----------------------------------------------------------------------|
  |  0 |  19.4466                       .001682  -.712107  -.682811  -.63489  |
  |  1 |  173.914  308.93     4  0.000  3.6e-06  -6.85363  -6.76575  -6.62198  |
  |  2 |  206.551  65.275*    4  0.000  1.1e-06* -8.02251* -7.87603* -7.63642* |
  |  3 |  210.351  7.5993     4  0.107  1.1e-06  -8.01433  -7.80926  -7.47381  |
  |  4 |  214.265   7.827     4  0.098  1.1e-06   -8.0108  -7.74714  -7.31585  |
  +---------------------------------------------------------------------------+
   Endogenous:  lp lxrat
    Exogenous:  _cons
```

Two lags are selected by most of the criteria.

```
. vecrank lp lxrat if tin(,2002)
                       Johansen tests for cointegration
Trend: constant                                         Number of obs =      51
Sample:  1952 - 2002                                             Lags =       2
-------------------------------------------------------------------------------
                                                        5%
maximum                                      trace    critical
  rank    parms       LL        eigenvalue  statistic   value
    0       6      202.92635         .       22.9305    15.41
    1       9      213.94024     0.35074      0.9028*    3.76
    2      10      214.39162     0.01755
-------------------------------------------------------------------------------
```

We can reject the null of 0 cointegrating vectors in favor of > 0 via the trace statistic. We cannot reject the null of 1 cointegrating vector in favor of > 1. Thus, we conclude that there is one cointegrating vector. For two series, this could have also been determined by a Granger–Engle regression in levels.

```
. vec lp lxrat if tin(,2002), lags(2)
Vector error-correction model
Sample:  1952 - 2002                               No. of obs      =        51
                                                   AIC             = -8.036872
Log likelihood =  213.9402                         HQIC            =   -7.9066
Det(Sigma_ml)  =  7.79e-07                         SBIC            = -7.695962
Equation           Parms      RMSE     R-sq      chi2     P>chi2
----------------------------------------------------------------

D_lp                 4     .057538   0.4363   36.37753   0.0000
D_lxrat              4     .055753   0.4496   38.38598   0.0000
----------------------------------------------------------------

------------------------------------------------------------------------------
             |
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
D_lp         |
        _ce1 |
         L1. |    -.26966   .0536001    -5.03   0.000    -.3747143   -.1646057
             |
          lp |
         LD. |   .4083733    .324227     1.26   0.208    -.2270999    1.043847
             |
       lxrat |
         LD. |  -.1750804   .3309682    -0.53   0.597    -.8237663    .4736054
             |
       _cons |   .0027061   .0111043     0.24   0.807     -.019058    .0244702
------------------------------------------------------------------------------
             |
...
```

```
-------------+----------------------------------------------------------------
D_lxrat      |
        _ce1 |
         L1. |   .2537426   .0519368     4.89   0.000     .1519484    .3555369
             |
          lp |
         LD. |   .3566706   .3141656     1.14   0.256    -.2590827    .9724239
             |
       lxrat |
         LD. |   .8975872   .3206977     2.80   0.005     .2690313    1.526143
             |
       _cons |   .0028758   .0107597     0.27   0.789    -.0182129    .0239645
------------------------------------------------------------------------------

Cointegrating equations
Equation           Parms    chi2     P>chi2
-------------------------------------------
_ce1                 1   44.70585   0.0000
-------------------------------------------

Identification:  beta is exactly identified
                 Johansen normalization restriction imposed
------------------------------------------------------------------------------
        beta |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
_ce1         |
          lp |          1          .        .       .            .           .
       lxrat |  -.7842433   .1172921    -6.69   0.000    -1.014131   -.5543551
       _cons |  -4.982628          .        .       .            .           .
------------------------------------------------------------------------------
```

In the `lp` equation, the `L1._ce1` term is the lagged error correction term. It is significantly negative, representing the negative feedback necessary in relative prices to bring the real exchange rate back to equilibrium. The short-run coefficients in this equation are not significantly different from zero.

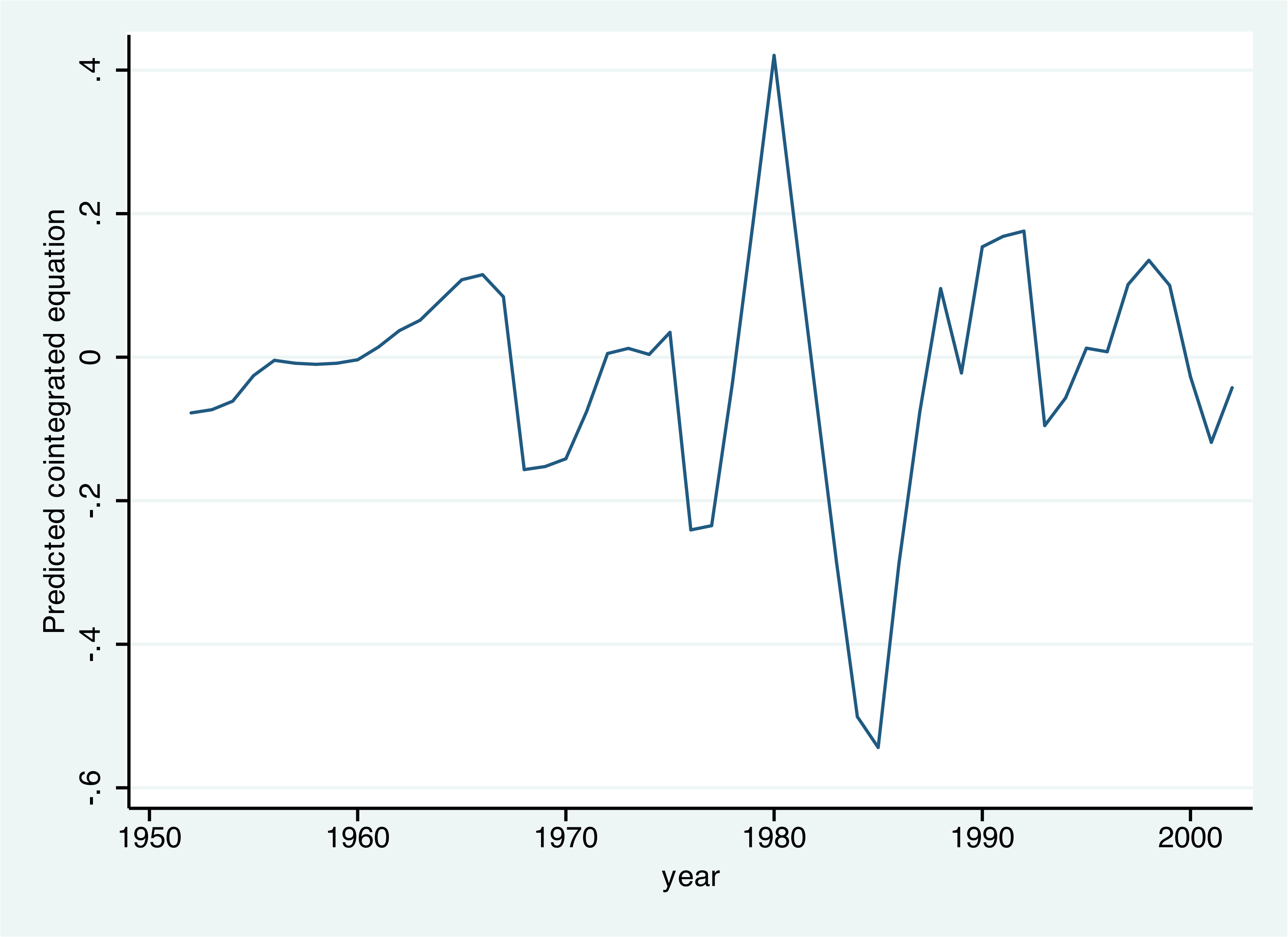
In the `lxrat` equation, the lagged error correction term is positive, as it must be for the other variable in the relationship: that is, if $(\log p - \log e)$ is above long-run equilibrium, either p must fall or e must rise. The short-run coefficient on the exchange rate is positive and significant.

The estimated cointegrating vector is listed at the foot of the output, normalized with a coefficient of unity on `lp` and an estimated coefficient of -0.78 on `lxrat`, significantly different from zero. The constant term corresponds to the μ term in the representation given above.

The significance of the lagged error correction term in this equation, and the significant coefficient estimated in the cointegrating vector, indicates that a VAR in first differences of these variables would yield inconsistent estimates due to misspecification.

We can evaluate the cointegrating equation by using `predict` to generate its in-sample values:

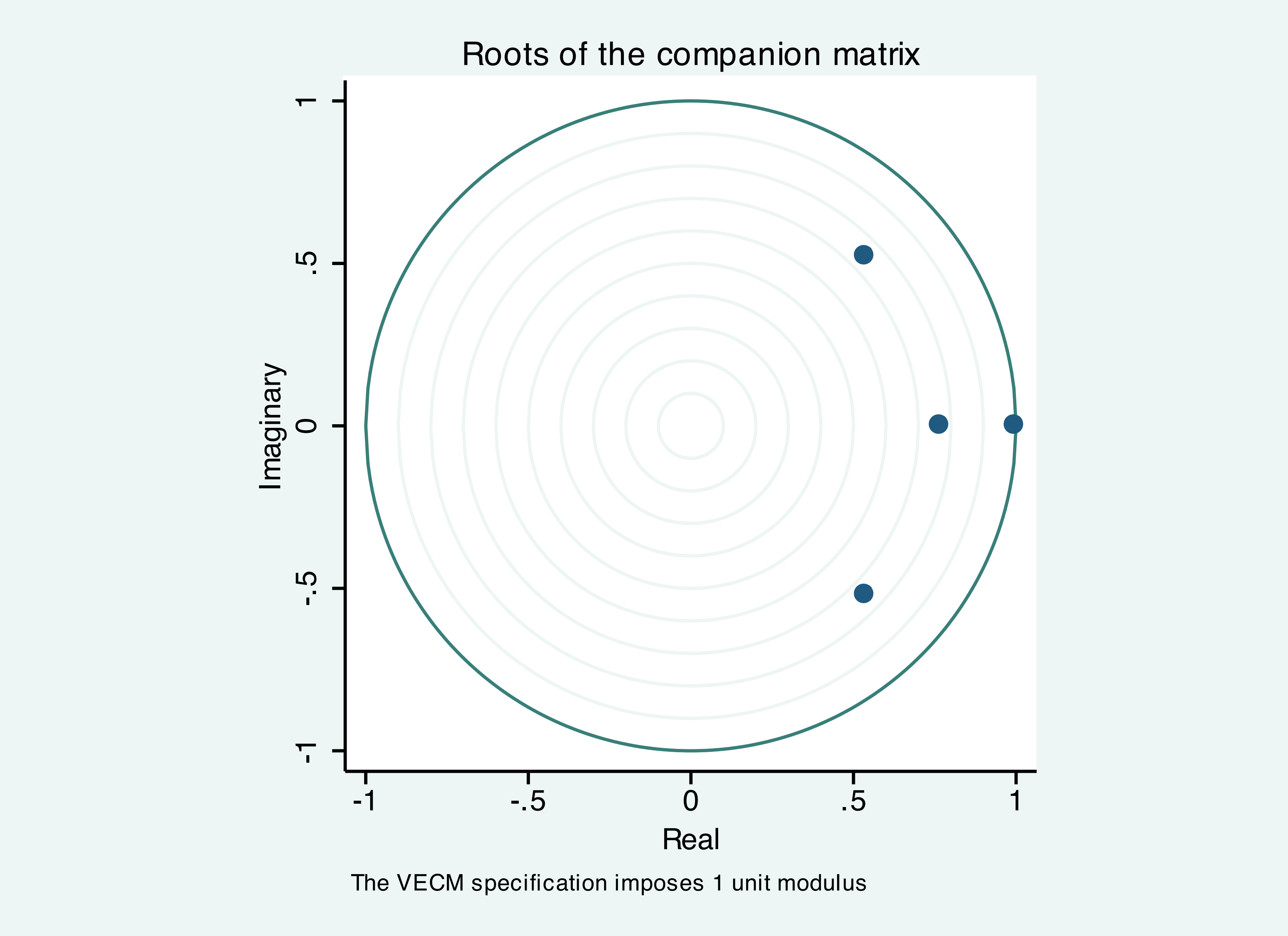
```
 . predict ce1 if e(sample), ce equ(#1)
. tsline ce1 if e(sample)
```


Predicted cointegrated equation
.4
.2
0
-.2
-.4
-.6
1950
1960
1970
1980
1990
2000
year

We should also evaluate the stability of the estimated VECM. For a K-variable model with r cointegrating relationships, the companion matrix will have $K - r$ unit eigenvalues. For stability, the moduli of the remaining r eigenvalues should be strictly less than unity.

```
. vecstable, graph
  Eigenvalue stability condition

        Eigenvalue                 Modulus

              1                          1
     .7660493                     .766049
     .5356276 +    .522604i       .748339
     .5356276 -    .522604i       .748339

  The VECM specification imposes a unit modulus.
```

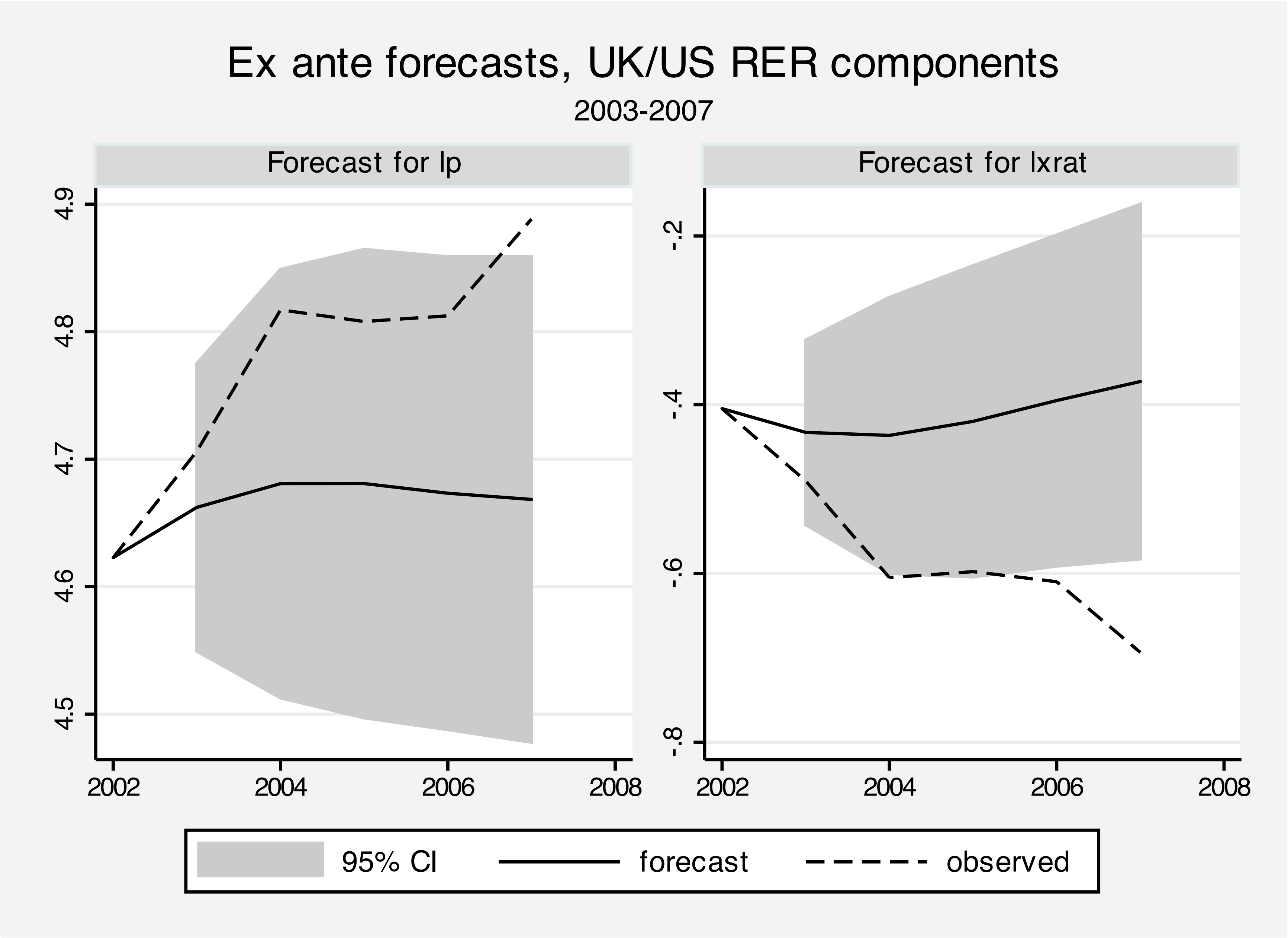
The eigenvalues meet the stability condition.


Roots of the companion matrix
Imaginary
1
.5
0
-.5
-1
-1
-.5
0
.5
1
Real
The VECM specification imposes 1 unit modulus

We can use much of the same post-estimation apparatus as developed for VARs for VECMs. Impulse response functions, orthogonalized IRFs, FEVDs, and the like can be constructed for VECMs. However, the presence of the integrated variables (and unit moduli) in the VECM representation implies that shocks may be permanent as well as transitory.

We illustrate here one feature of Stata’s `vec` suite: the capability to compute dynamic forecasts from a VECM. We estimated the model on annual data through 2002, and now forecast through the end of available data in 2007:

```
 . tsset year
        time variable:  year, 1950 to 2007
                delta:  1 year
. fcast compute ppp_, step(5)
. fcast graph ppp_lp ppp_lxrat, observed scheme(s2mono) legend(rows(1)) ///
> byopts(ti("Ex ante forecasts, UK/US RER components") t2("2003-2007"))
```


Ex ante forecasts, UK/US RER components
2003-2007
Forecast for lp
4.5
4.6
4.7
4.8
4.9
2002
2004
2006
2008
Forecast for lxrat
-.8
-.6
-.4
-.2
2002
2004
2006
2008
95% CI
forecast
observed

We see that the model's predicted log relative price was considerably lower than that observed, while the predicted log nominal exchange rate was considerably higher than that observed over this out-of-sample period.

Consult the Stata *Time Series* manual for much greater detail on Stata's VECM capabilities, applications to multiple-variable systems and alternative treatments of deterministic trends in the VECM context.

Although we have discussed a number of Stata's time series capabilities relevant for macroeconometrics in this talk, you should be aware that there are many additional Stata features that may be useful in your work. A number of them were added in Stata version 11 (2009), as significant development efforts in the time series econometrics field were a large part of that release.

Some of these additional capabilities (and their command names):

- univariate ARCH and GARCH models, including 18 alternative specifications, with Gaussian, *t* or GED errors (`arch`)
- diagonal *vech* multivariate ARCH models (`dvech`)
- linear state-space models via the Kalman filter (`sspace`)
- dynamic-factor multivariate time series models (`dfactor`)

For each of these commands, see the version 11 *Time Series* manual, available in PDF on your machine.