Time series estimation and forecasting

Christopher F Baum

Boston College and DIW Berlin

IMF Institute for Capacity Development, October 2018

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("10sep2013","DMY")` yields 19611, 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("2013Q3","YQ")` yields 214, as that is 214 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.

Business calendars

In Stata version 12 onward, you can define a business-daily calendar that properly treats time series observed in 'business time'. This avoids the problems that arise if you have a set of market quotations, organized by calendar date, and end up with gaps for weekends and holidays. See `help bcal` for details.

In Stata 13 onward, you can use the `bcal create` command to generate a business calendar from the current dataset, based on the dates listed in a specified *varname*. Gaps between the dates available in that *varname* are taken as business holidays. For example:

```
 . sysuse sp500, clear
(S&P 500)
. bcal create sp500, from(date) generate(bdate)
  Business calendar sp500 (format %tbsp500):
    purpose:
      range:  02jan2001  31dec2001
                 14977      15340   in %td units
                     0        247   in %tbsp500 units
     center:  02jan2001
                 14977              in %td units
                     0              in %tbsp500 units
    omitted:       116              days
                   116.4            approx. days/year
   included:       248              days
                   248.9            approx. days/year
  REMARKS:
    business calendar file sp500.stbcal saved
    variable bdate created; it contains business dates in %tbsp500 format
```

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 onward, they will be instantiated on the fly, and will not be permanently added to the data set in memory.

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. 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.

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)`.

Try it out:

```
. bcuse 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, as it has the smallest values of the criteria.

A number of measures of in-sample forecast accuracy have been devised. These include:

- Root mean squared error of forecast (RMSE)
- Mean absolute error of forecast (MAE)
- Mean absolute percentage error of forecast (MAPE)
- Theil's U

These measures can be computed by the `fcstats` routine, available from the SSC Archive. It can compare the actual series to one or two forecast series, with an option to graph the actual and forecast series.

Theil's U statistic (*Applied Economic Forecasting*, 1966), sometimes termed U2, is based on the concept of information entropy. It can be considered as the RMSE of the forecast divided by the RMSE of the naive model, $y_{t+1} = y_t$.

Theil's U takes on a value of 1 if the forecasting method is no more accurate than a naive forecast. If it is less than 1, the forecasting method is more accurate than a naive forecast, and vice versa.

To illustrate, consider that lags of the one- and two-month futures rates of the Japanese yen can be considered as forecasts of the expected future spot rate. **Try it out:**

```
. bcuse barclaymonthly, nodesc clear
        time variable:  month, 1984m12 to 1998m12
                delta:  1 month

. // consider one-month and two-month futures as predictors of spot rate
. fcstats bbjpysp L.bbjpy1f L2.bbjpy2f

Forecast accuracy statistics for bbjpysp, N = 167

              L.bbjpy1f      L2.bbjpy2f
RMSE          5.2500629      7.5411856
MAE           3.8609281      5.9017074
MAPE          .0287943       .04398991
Theil´s U     1.0030129      1.46186
```

The Theil's U statistics show that the one-month futures rate is about as accurate as a naïve forecast, while the two-month futures rate is considerably worse.

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. This corresponds to estimating the model using the information set available to modelers at the time.

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`.

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*.

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.

Say that we want to produce moving-window regression estimates from a window containing 48 quarterly observations:

```
. bcuse usmacro1
. 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)
```

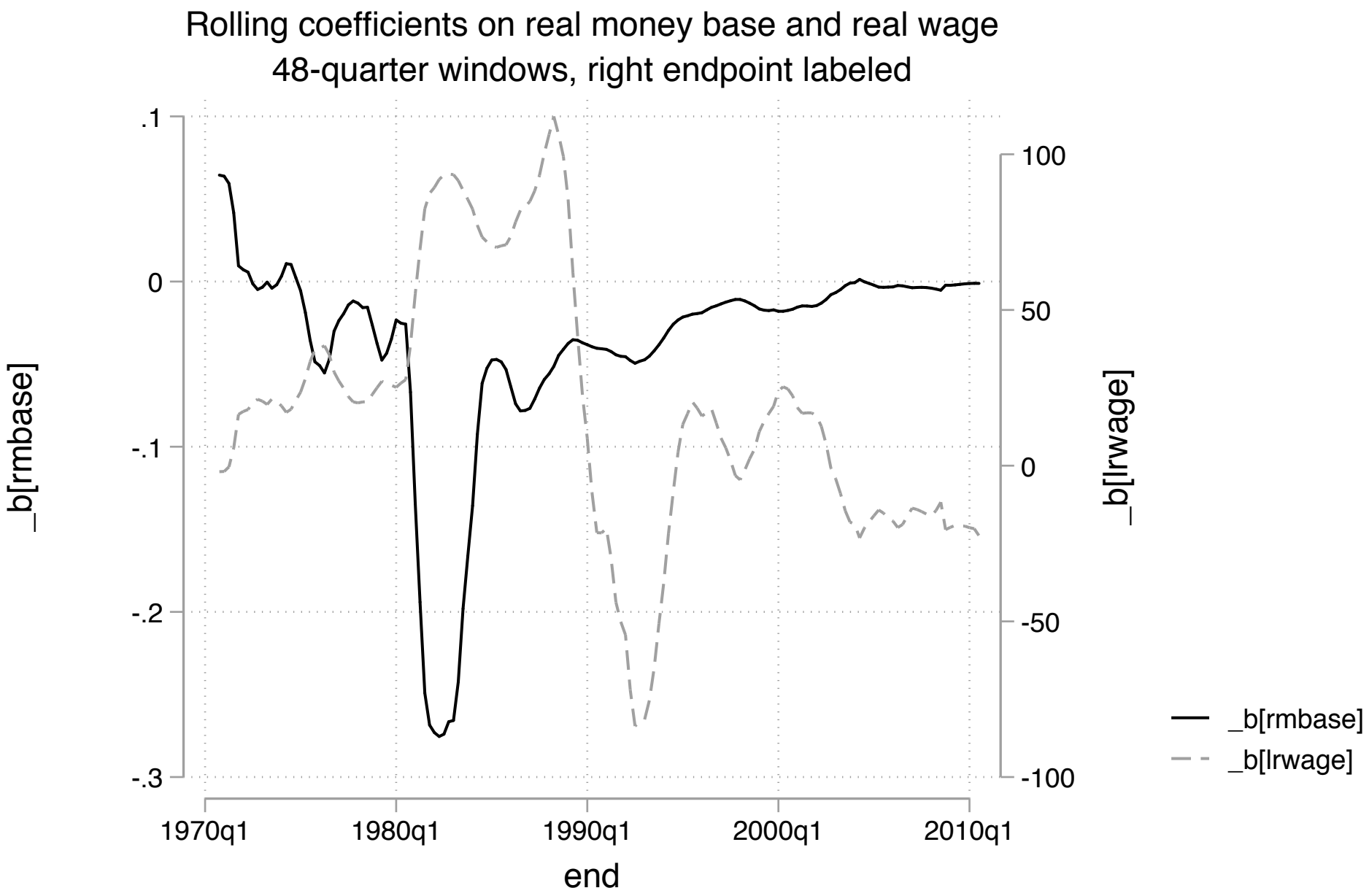
variable name	storage type	display format	value label	variable label
start	float	%tq		
end	float	%tq		
_b_rmbase	float	%9.0g		_b[rmbase]
_b_lrwage	float	%9.0g		_b[lrwage]
_b_cons	float	%9.0g		_b[_cons]
_se_rmbase	float	%9.0g		_se[rmbase]
_se_lrwage	float	%9.0g		_se[lrwage]
_se_cons	float	%9.0g		_se[_cons]
_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(plotplainblind) 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]
_b[lrwage]
.1
0
-.1
-.2
-.3
100
50
0
-50
-100
1970q1
1980q1
1990q1
2000q1
2010q1
end
_b[rmbase]
_b[lrwage]

Time series smoothing and 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.

The `tsfilter` command provides the Baxter–King, Butterworth, Christiano–Fitzgerald and Hodrick–Prescott filters. Jorge Pérez' implementation of the Corbae–Ouliaris filter is also available from the SSC archive as `couliari`. That routine improves upon the Baxter–King filter 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 low-frequency series (annual or quarterly) to a higher frequency (quarterly or monthly) using an indicator series observed at the higher frequency. The routine has recently been rewritten to take advantage of Mata.

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.

```
. bcuse 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

Unobserved components models

An important addition to Stata version 12 is the capability to estimate *unobserved components models*, or UCMs, that decompose a time series into trend, seasonal, cyclical and idiosyncratic components, allowing for exogenous variables. They may be written in the most general form as

$$y_t = \tau_t + \gamma_t + \phi_t + \beta x_t + \epsilon_t$$

where τ is the trend component, γ is the seasonal component, ϕ is the cyclical component, β is a vector of fixed parameters, x is a vector of exogenous variables, and ϵ is the idiosyncratic component.

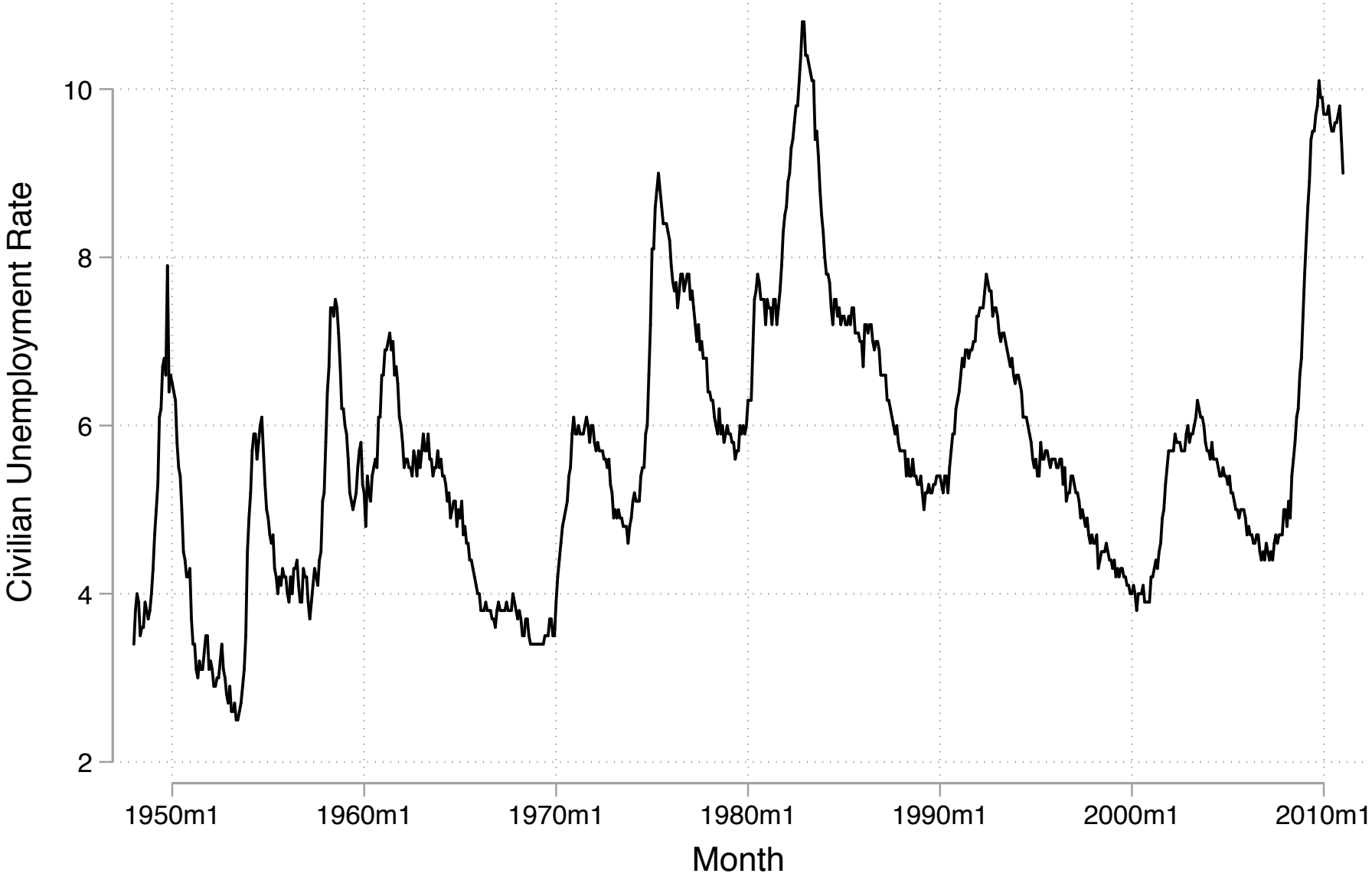
Stata's UCMs, based on the work of Harvey (1989), parameterize the trend, seasonal and cyclical components and estimate their parameters via the state-space formulation of the model. After estimation, the `predict` command can produce in-sample or out-of-sample forecasts. The default model specified by `ucm` is the random walk:

$$\begin{aligned} y_t &= \mu_t \\ \mu_t &= \mu_{t-1} + \eta_t \end{aligned}$$

and for a particular time series, it can be estimated with merely `ucm` *varname*. Note that this model is equivalent to ARIMA(0,1,0).

A richer model can be specified as containing a stationary cyclical component that produces serially correlated shocks around the random-walk trend. Harvey's parameterization of this model has three parameters: the frequency at which the random components are centered, a damping factor, and the variance of the stochastic-cycle process.

Load monthly data on the US civilian unemployment rate, `webuse unrate`, and fit a simple UCM:



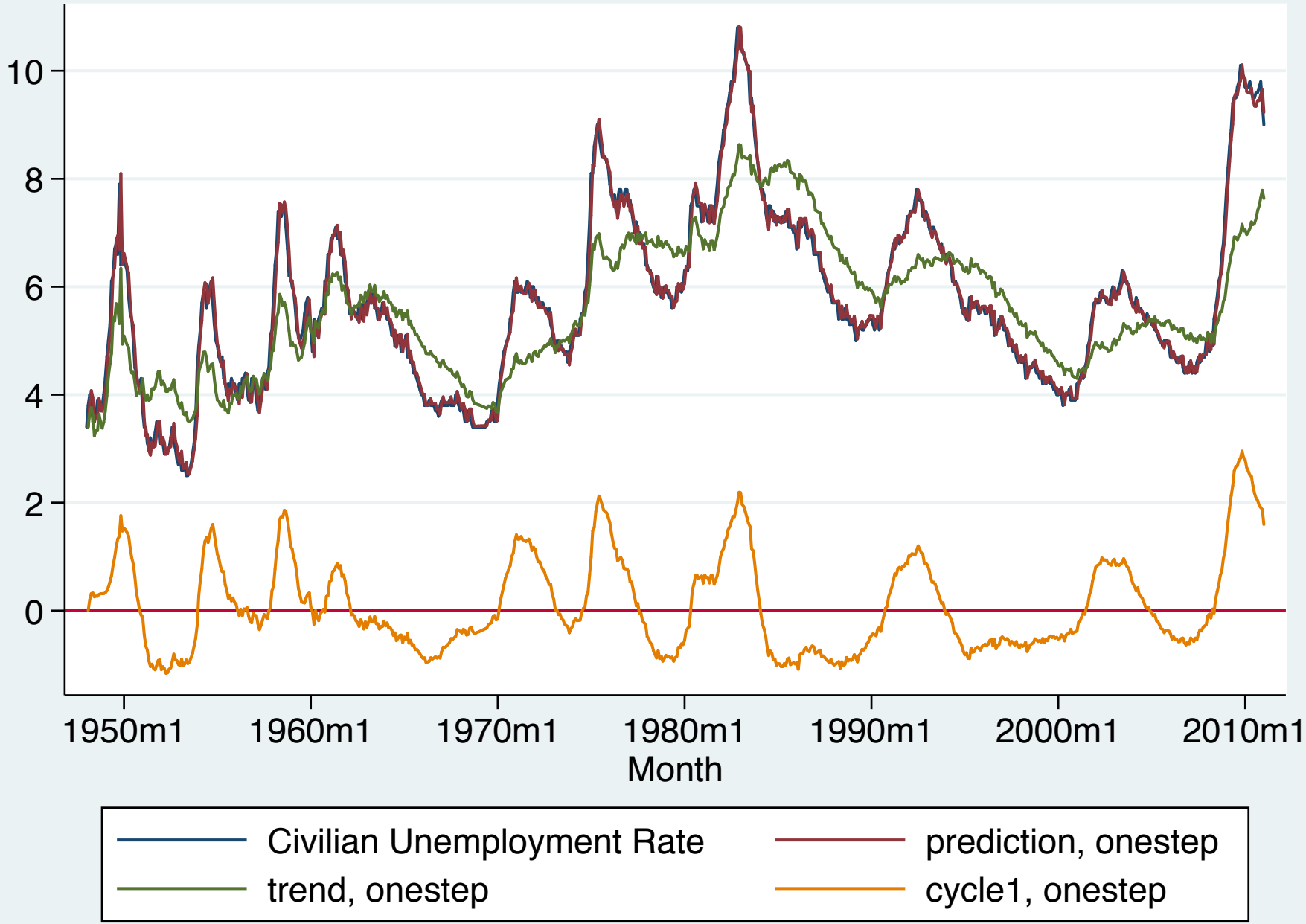
Try it out:

```
. webuse unrate
. ucm unrate, cycle(1) nolog
Unobserved-components model
Components: random walk, order 1 cycle
Sample: 1948m1 - 2011m1                         Number of obs    =       757
                                                Wald chi2(2)     =  26650.81
Log likelihood =  118.88421                     Prob > chi2      =    0.0000
```

unrate	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
frequency	.0933466	.0103609	9.01	0.000	.0730397	.1136535
damping	.9820003	.0061121	160.66	0.000	.9700207	.9939798
Variance						
level	.0143786	.0051392	2.80	0.003	.004306	.0244511
cycle1	.0270339	.0054343	4.97	0.000	.0163829	.0376848

```
Note: Model is not stationary.
Note: Tests of variances against zero are one sided, and the two-sided
    confidence intervals are truncated at zero.
```

After estimation, `estat period` reveals that the estimated cycle has a periodicity of 67.3 months, or between five and six years. We use `predict` to recover the predictions of the series, as well as their trend and cyclical components. Dynamic forecasts are also available.


10
8
6
4
2
0
1950m1
1960m1
1970m1
1980m1
1990m1
2000m1
2010m1
Month
Civilian Unemployment Rate
prediction, onestep
trend, onestep
cycle1, onestep

A variety of specifications for the trend and idiosyncratic components may be used, including a local level model (containing an added shock term vs. the pure random walk), a deterministic trend, a local level with deterministic trend, random walk with drift, local linear trend, smooth trend and random trend.

Most of these models may be directly applied to nonstationary time series. Seasonal components may also be included for applications to non-seasonally-adjusted series.

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). **Try it out!**

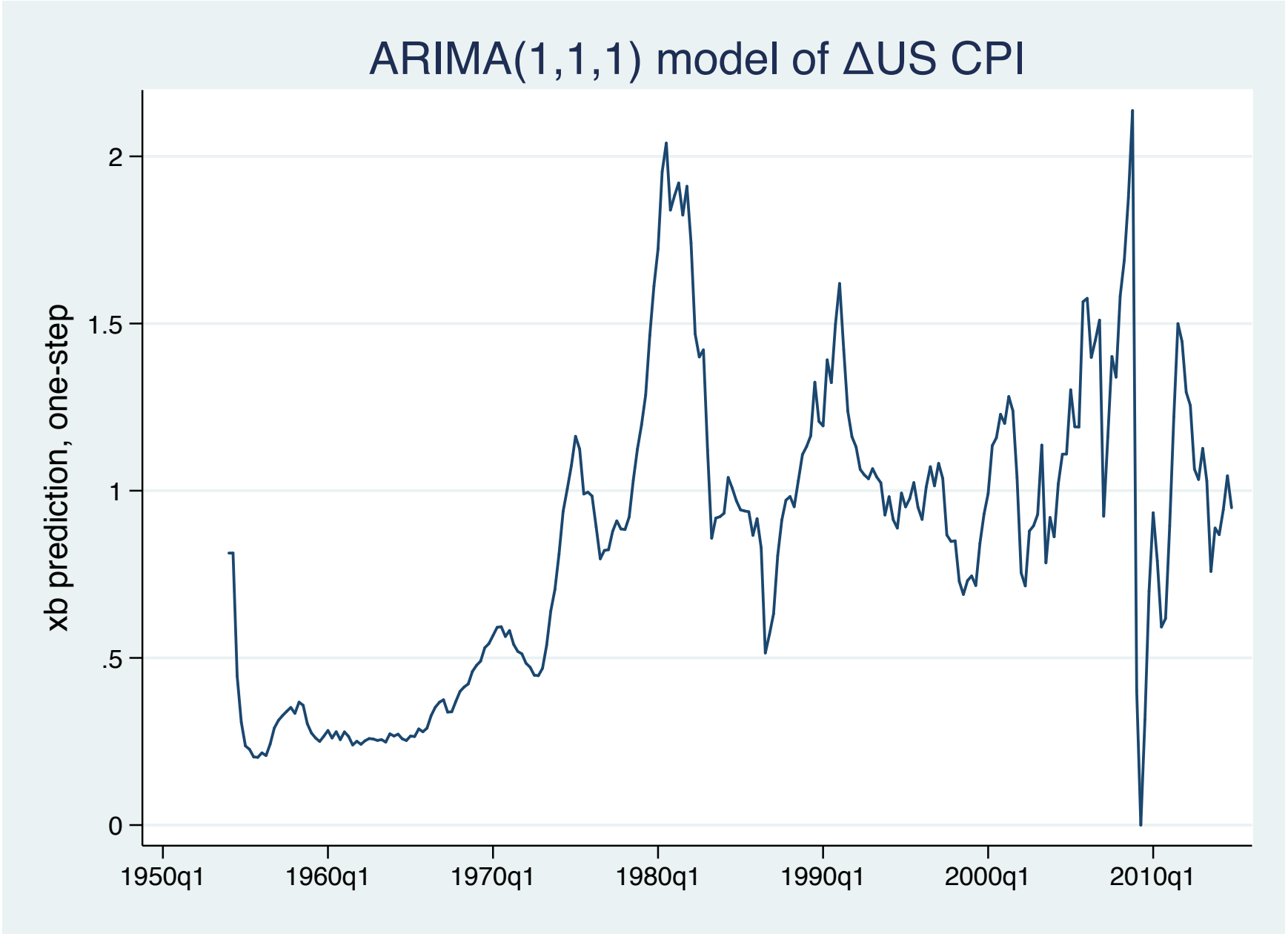
```
. bcuse macro14, nodesc
. arima cpiaucsl, arima(1,1,1) nolog
ARIMA regression
Sample:  1954q2 - 2014q4                        Number of obs     =        243
                                                Wald chi2(2)      =     330.75
Log likelihood = -259.7311                      Prob > chi2       =     0.0000
```

D.cpiaucsl	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
cpiaucsl						
_cons	.8134891	.3291271	2.47	0.013	.1684118	1.458566
ARMA						
ar L1.	.931381	.0532187	17.50	0.000	.8270742	1.035688
ma L1.	-.7014376	.0582227	-12.05	0.000	-.8155519	-.5873233
/sigma	.7037909	.0160152	43.95	0.000	.6724018	.7351801

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
. predict double dcpihat, xb
. tsline dcpihat, ylab(,angle(0) labs(small)) xlab(,labs(small)) legend(size(small)) ///
> ti("ARIMA(1,1,1) model of {&Delta}US CPI") xti("")
```

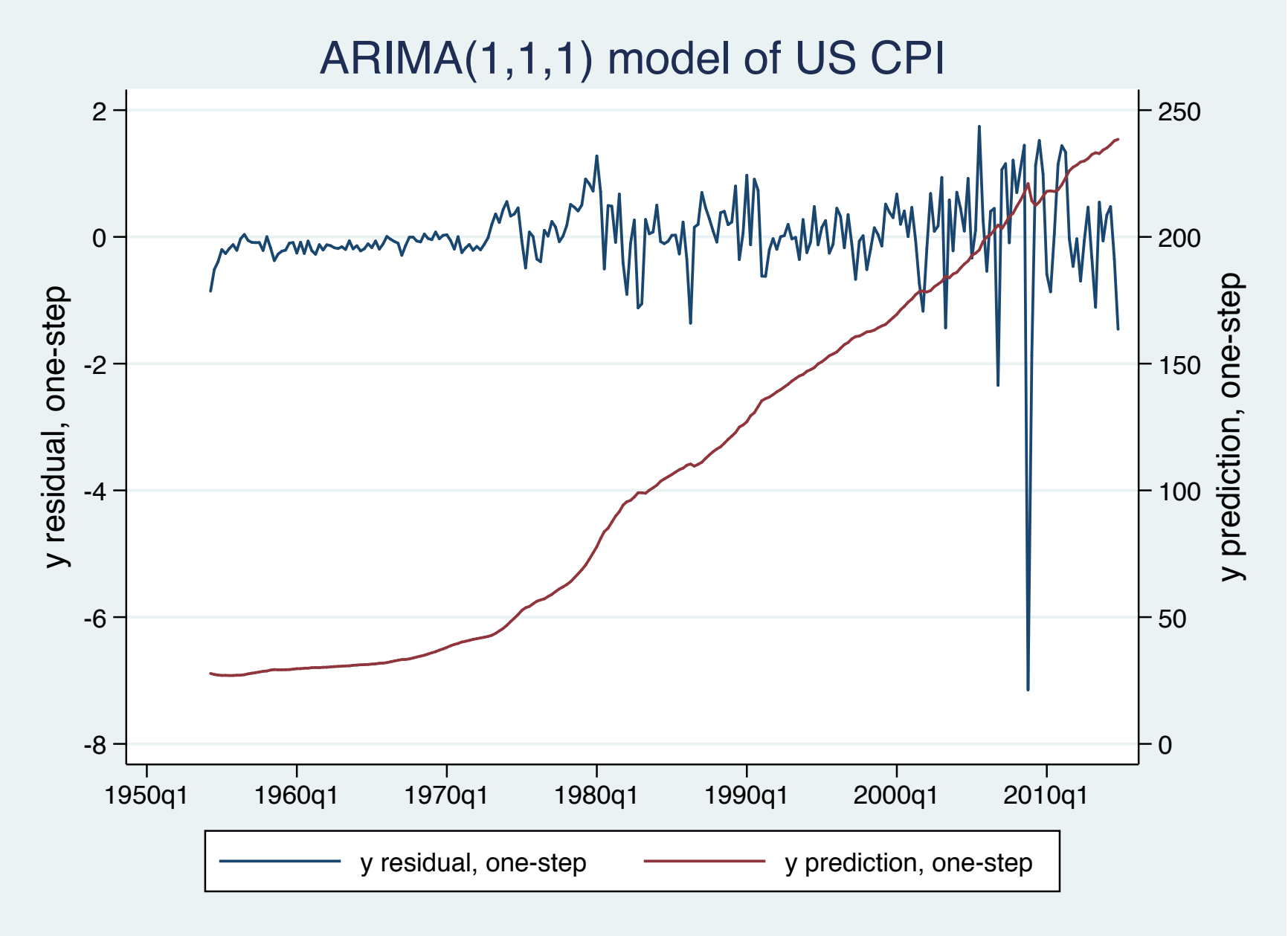
Several prediction options are available after estimating an `arima` model. The default option, `xb`, predicts the actual dependent variable: so if `D.cpi` is the dependent variable, predictions are made for that variable. In contrast, the `y` option generates predictions of the original variable, in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

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


ARIMA(1,1,1) model of ΔUS CPI
xb prediction, one-step
2
1.5
1
.5
0
1950q1
1960q1
1970q1
1980q1
1990q1
2000q1
2010q1

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

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

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

. tw (tsline cpieps, yaxis(2) ylab(,angle(0) axis(2) labs(small)) xlab(,labs(small))) ///
>  (tsline cpihat, ylab(,angle(0) labs(small)) ///
> ti("ARIMA(1,1,1) model of US CPI") xti("") legend(size(small)) )
```


ARIMA(1,1,1) model of US CPI
y residual, one-step
y prediction, one-step
2
0
-2
-4
-6
-8
250
200
150
100
50
0
1950q1
1960q1
1970q1
1980q1
1990q1
2000q1
2010q1
y residual, one-step
y prediction, one-step

In the prior example, we used 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:

```
. bcuse macro14, clear nodesc
. arima D.cpiaucsl, ar(1 4) nolog
ARIMA regression
Sample:  1954q2 - 2014q4                        Number of obs     =        243
                                                Wald chi2(2)      =      59.35
Log likelihood = -261.8755                      Prob > chi2       =     0.0000
```

D.cpiaucsl	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
cpiaucsl						
_cons	.8494434	.1501867	5.66	0.000	.555083	1.143804
ARMA						
ar						
L1.	.4695209	.0633586	7.41	0.000	.3453402	.5937015
L4.	.1132828	.0568215	1.99	0.046	.0019147	.224651
/sigma	.7103758	.0176754	40.19	0.000	.6757326	.745019

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

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

Try it out!

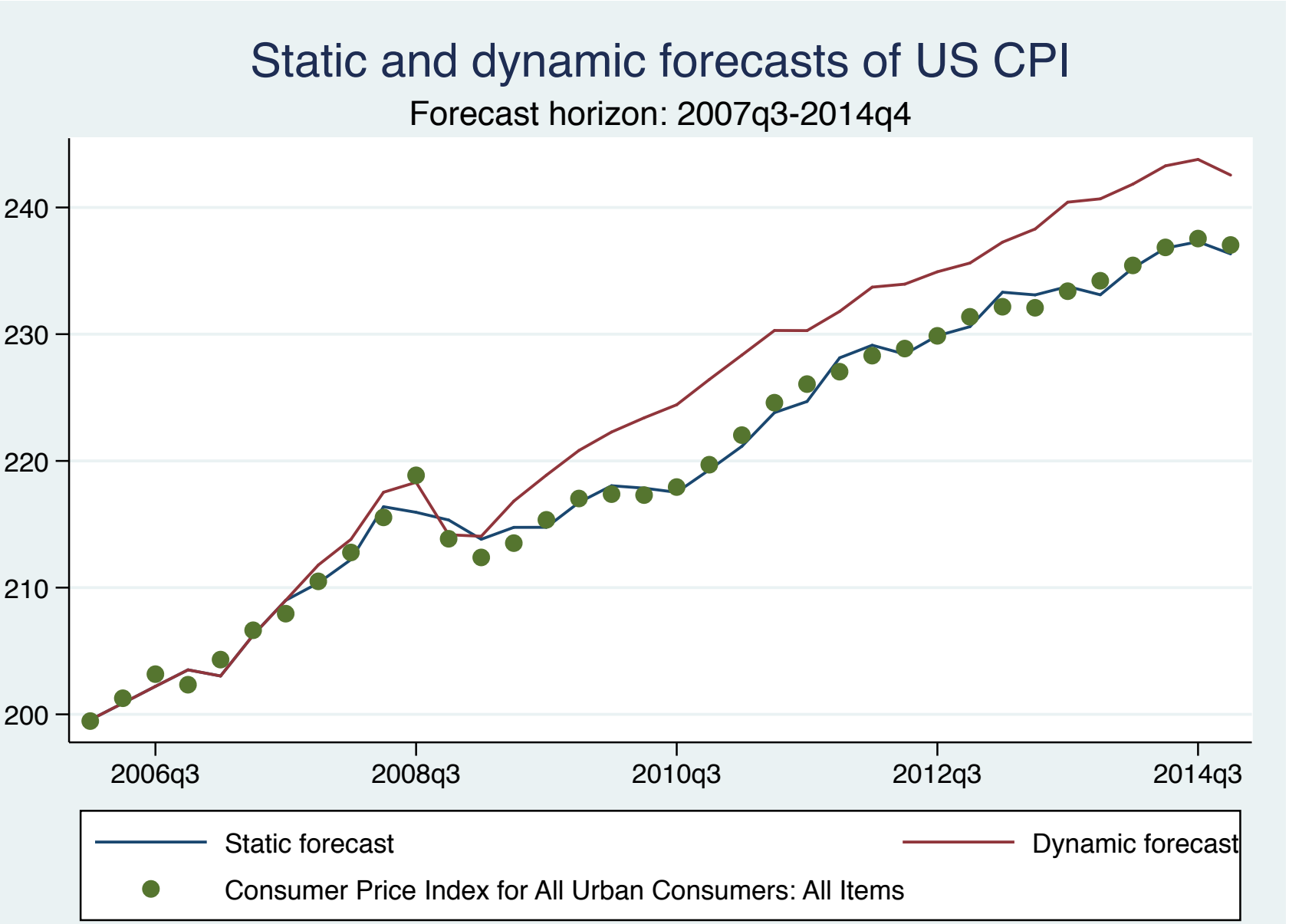
```
. arima d.cpiaucsl d.mcoilwtico if tin(,2007q3), ar(1) ma(1) nolog vsquish
ARIMA regression
Sample:  1959q2 - 2007q3                        Number of obs     =        194
                                                Wald chi2(3)      =    1695.47
Log likelihood = -91.14007                      Prob > chi2       =     0.0000
```

D.cpiaucsl	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
cpiaucsl						
mcoilwtico						
D1.	.0911746	.0055546	16.41	0.000	.0802878	.1020615
_cons	.8333632	.2807785	2.97	0.003	.2830475	1.383679
ARMA						
ar						
L1.	.9594617	.0252121	38.06	0.000	.9100469	1.008877
ma						
L1.	-.6247632	.0528421	-11.82	0.000	-.7283317	-.5211947
/sigma	.3858822	.0138622	27.84	0.000	.3587128	.4130516

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

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.

```
. qui predict double cpihat_s if tin(2006q1,), y
. label var cpihat_s "Static forecast"
. qui predict double cpihat_d if tin(2006q1,), dynamic(tq(2007q3)) y
. label var cpihat_d "Dynamic forecast"
. tw (tsline cpihat_s cpihat_d if !mi(cpihat_s)) ///
> (scatter cpiaucsl yq if !mi(cpihat_s), c(i) ///
> ti("Static and dynamic forecasts of US CPI") ///
> t2("Forecast horizon: 2007q3-2014q4") legend(size(small)) ///
> ylab(,angle(0) labs(small)) xti("") xlab(,labs(small)))
```


Static and dynamic forecasts of US CPI
Forecast horizon: 2007q3-2014q4
240
230
220
210
200
2006q3
2008q3
2010q3
2012q3
2014q3
Static forecast
Dynamic forecast
Consumer Price Index for All Urban Consumers: All Items

Forecast comparison procedures

A common task in forecast evaluation is the comparison of two forecasts to judge which is more accurate. Diebold and Mariano (*JBES*, 1995) proposed a measure of predictive accuracy that compares two competing forecasts with the actual series. The null hypothesis is that of equal accuracy.

This routine can be installed from the SSC Archive via `ssc install dmariano`. The computation depends on an estimate of the long-run variance of the series. Options `maxlag` and `kernel` can be used to influence that estimation. The option `crit` specifies the forecast accuracy criterion to be employed: MSE, MAE or MAPE, with a default of MSE.

Consider the one- and two-month futures rates on the Japanese yen to be forecasts of the expected future spot rate:

```
. bcuse barclaymonthly, nodesc
        time variable:  month, 1984m12 to 1998m12
                delta:  1 month

. // consider one-month and two-month futures as predictors of spot rate
. dmariano bbjpysp L.bbjpy1f L2.bbjpy2f, max(6)

Diebold-Mariano forecast comparison test for actual : bbjpysp
Competing forecasts:  L.bbjpy1f versus L2.bbjpy2f
Criterion: MSE over 167 observations
Maxlag = 6   Kernel : uniform

Series                    MSE
_____________________________
L.bbjpy1f                27.56
L2.bbjpy2f               56.87
Difference              -29.31

By this criterion, L.bbjpy1f is the better forecast
H0: Forecast accuracy is equal.
S(1) =    -4.081  p-value = 0.0000
```

The long-run variance is a ‘Newey–West’ estimate based on 6 lags.

Consider the same test based on mean absolute percentage error, or MAPE:

```
. dmariano bbjpysp L.bbjpy1f L2.bbjpy2f, max(6) kernel(bartlett) crit(mape)

Diebold-Mariano forecast comparison test for actual : bbjpysp
Competing forecasts:  L.bbjpy1f versus L2.bbjpy2f
Criterion: MAPE over 167 observations
Maxlag = 6   Kernel : bartlett

Series                    MAPE
_____________________________________
L.bbjpy1f                .02879
L2.bbjpy2f               .04399
Difference               -.0152

By this criterion, L.bbjpy1f is the better forecast
H0: Forecast accuracy is equal.
S(1) =    -7.842  p-value = 0.0000
```

The qualitative result is unchanged: the one-month futures contract is the superior predictor.

Consider an ARIMA model of the Japanese yen spot rate, with the specification ARIMA(2,1,1):

```
. arima bbjpysp, arima(2,1,1) nolog

ARIMA regression

Sample:  1985m1 - 1998m12                       Number of obs     =        168
                                                Wald chi2(3)      =      13.23
Log likelihood = -511.8448                      Prob > chi2       =     0.0042
```

D.bbjpysp	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
bbjpysp						
_cons	-.8287808	.5895949	-1.41	0.160	-1.984366	.3268039
ARMA						
ar						
L1.	.309928	.3237227	0.96	0.338	-.3245568	.9444128
L2.	.186342	.0715153	2.61	0.009	.0461745	.3265094
ma						
L1.	-.3077743	.3255738	-0.95	0.344	-.9458872	.3303386
/sigma	5.091067	.2412995	21.10	0.000	4.618128	5.564005

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.

. predict jpyarima, y
(1 missing value generated)
```

As a competing forecast, consider a pure AR(3) model of the Japanese yen spot rate:

```
. reg bbjpysp L(1/3).bbjpysp
```

Source	SS	df	MS		
Model	165408.288	3	55136.0959	Number of obs	= 166
Residual	3848.82392	162	23.7581724	F(3, 162)	= 2320.72
				Prob > F	= 0.0000
				R-squared	= 0.9773
				Adj R-squared	= 0.9768
Total	169257.112	165	1025.80068	Root MSE	= 4.8742

bbjpysp	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
bbjpysp						
L1.	.8987353	.077176	11.65	0.000	.7463346	1.051136
L2.	.2005905	.1055989	1.90	0.059	-.0079373	.4091183
L3.	-.1510395	.0743511	-2.03	0.044	-.2978617	-.0042173
_cons	6.133964	1.589804	3.86	0.000	2.994553	9.273375

```
. predict jpyols, xb
(3 missing values generated)
```

The Diebold–Mariano test compares the in-sample accuracy of these competing approaches to modeling the Japanese yen spot rate:

```
. dmariano bbjpysp jpyarima jpyols, max(6) kernel(bartlett)

Diebold-Mariano forecast comparison test for actual : bbjpysp
Competing forecasts:  jpyarima versus jpyols
Criterion: MSE over 166 observations
Maxlag = 6   Kernel : bartlett

Series                    MSE
_____________________________
jpyarima                25.97
jpyols                  23.19
Difference              2.783

By this criterion, jpyols is the better forecast
H0: Forecast accuracy is equal.
S(1) =     2.008  p-value = 0.0447
```

The OLS regression model is significantly more accurate using the MSE criterion, just surpassing the 95% level of significance.

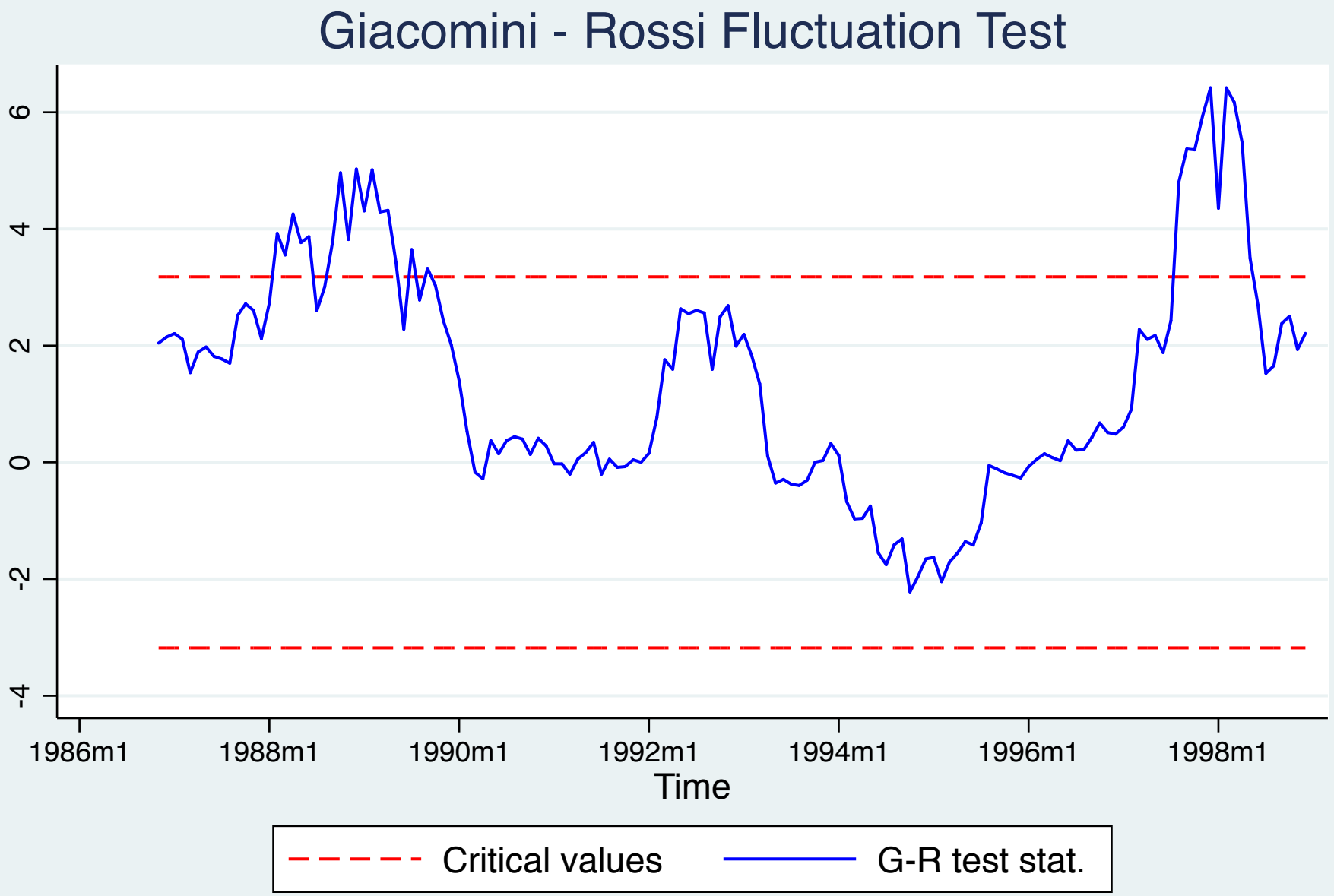
An extension of the Diebold–Mariano test have been proposed by Giacomini and Rossi (*J. Applied Econometrics*, 2010). This extended test deals with instabilities in the underlying process by using rolling regression procedures (`help rolling`).

A complementary routine by Rossi and Sekhposyan (*J. Applied Econometrics*, 2016) deals with the rationality of forecasts in the presence of instabilities. Both tests can be accessed from the SSC Archive by `ssc install forec_instab`.

We run the Giacomini–Rossi test with a 24-month rolling window on the Japanese yen spot rate series:

```
. giacross bbjpysp jpyarima jpyols, window(24) alpha(0.05)

Running the Giacomini - Rossi (2010) test for forecast comparison...

REMINDER
First forecast: jpyarima
Second forecast: jpyols
Actual series: bbjpysp

Newey - West HAC estimator bandwidth chosen automatically with the Schwert criterion.


Giacomini and Rossi´s (2010) test rejects the null hypothesis of equal predictive ability
when the test statistic is outside the band lines (2 sided alternative) or above the
band line (1 sided alternative).
When using the two sided test: when the test statistic is below the lowest band line,
the first model forecasts significantly better.
When using the one sided test: when the test statistic is above the critical value line,
the first model forecasts significantly worse.
```

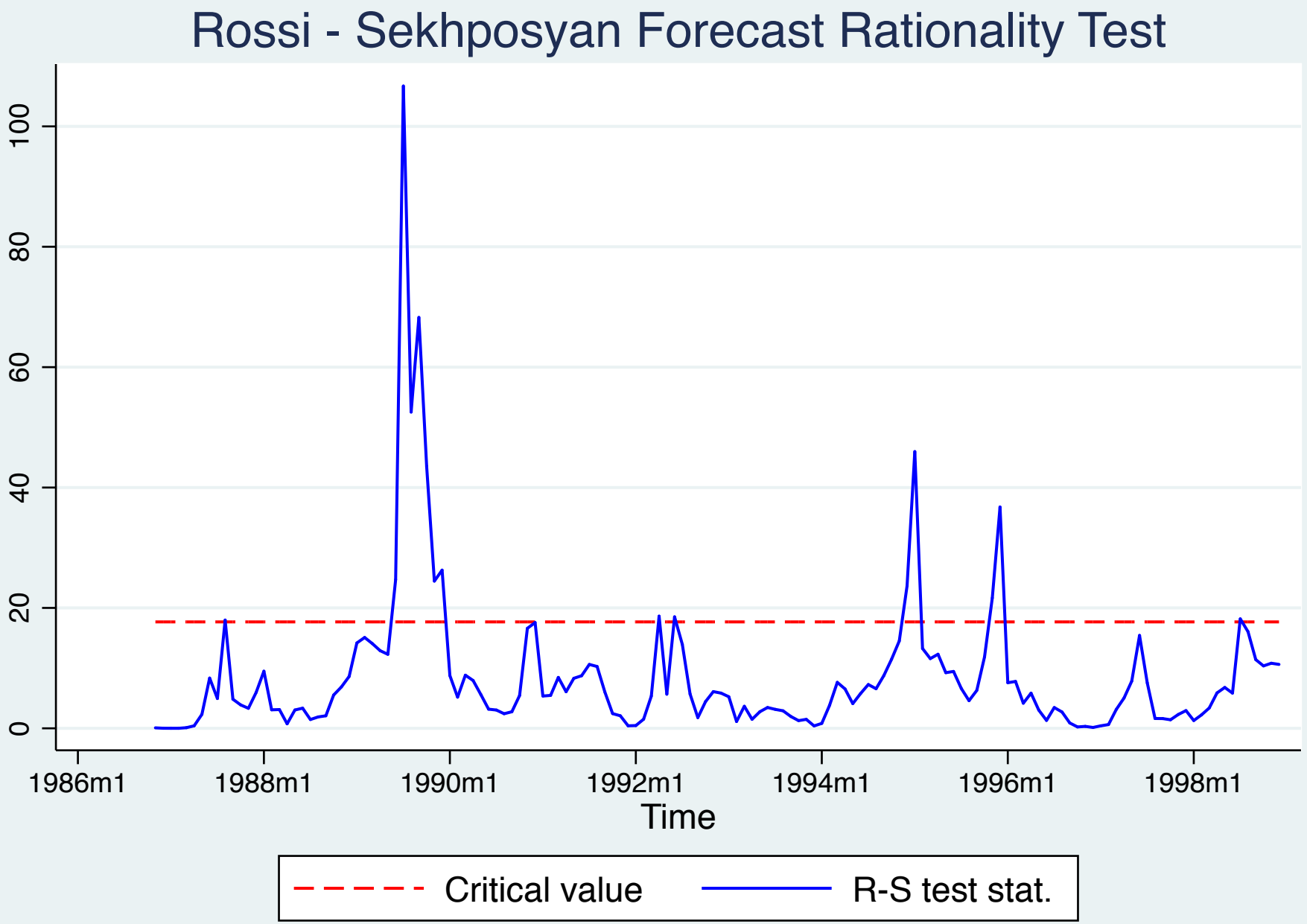
The graphical representation shows that in the late 1980s and the late 1990s the OLS model was forecasting significantly better than the ARIMA model.


Giacomini - Rossi Fluctuation Test
6
4
2
0
-2
-4
1986m1
1988m1
1990m1
1992m1
1994m1
1996m1
1998m1
Time
Critical values
G-R test stat.
Giacomini and Rossi's (2010) Fluctuation test

We run the Rossi–Sekhposyan test with a 24-month rolling window on the Japanese yen spot rate series:

```
. rosssekh bbjpysp jpyols, window(24) alpha(0.05) nw(6)


Running the Rossi - Sekhposyan (2016) forecast rationality test...

Critical value for the test: 14.9966


Rossi and Sekhposyan´s (2016) test rejects the null hypothesis of forecast rationality
when the test statistic is above the critical value line.
```

The graphical representation shows that in the late 1980s and the mid-1990s the hypothesis of forecast rationality was rejected for the OLS forecast of the Japanese yen spot rate.


Rossi - Sekhposyan Forecast Rationality Test
0
20
40
60
80
100
1986m1
1988m1
1990m1
1992m1
1994m1
1996m1
1998m1
Time
Critical value
R-S test stat.

ARCH models

Heteroskedasticity can occur in time series models, just as it may in a cross-sectional context. It has the same consequences: the OLS point estimates are unbiased and consistent, but their standard errors will be inconsistent, as will hypothesis test statistics and confidence intervals.

We may prevent that loss of consistency by using heteroskedasticity-robust standard errors. The "Newey–West" or HAC standard errors available from `newey` in the OLS context or `ivreg2` in the instrumental variables context will be robust to arbitrary heteroskedasticity in the error process as well as serial correlation.

The most common model of heteroskedasticity employed in the time series context is that of *autoregressive conditional heteroskedasticity*, or ARCH. As proposed by Nobel laureate Robert Engle in 1982, an ARCH model starts from the premise that we have a static regression model

$$y_t = \beta_0 + \beta_1 z_t + u_t$$

and all of the Gauss–Markov assumptions hold, so that the OLS estimators are BLUE. This implies that $Var(u_t|Z)$ is constant. But even when this unconditional variance of u_t is constant, we may have time variation in the *conditional variance* of u_t:

$$E(u_t^2|u_{t-1}, u_{t-2}, \dots) = E(u_t^2|u_{t-1}) = \alpha_0 + \alpha_1 u_{t-1}^2$$

so that the conditional variance of u_t is a linear function of the squared value of its predecessor.

If the original u_t process is serially uncorrelated, the variance conditioned on a single lag is identical to that conditioned on the entire history of the series. We can rewrite this as

$$h_t = \alpha_0 + \alpha_1 u_{t-1}^2$$

where $u_t = \sqrt{h_t}\, v_t$, $v_t \sim (0, 1)$. This formulation represents the ARCH(1) model, in which a single lagged u^2 enters the ARCH equation. A higher-order ARCH equation would include additional lags of u^2. To ensure a positive variance, $\alpha_0 > 0$ and $\alpha_1 > 0$. When $\alpha_1 > 0$, the squared errors are positively serially correlated even though the u_t themselves are not.

Since we could estimate this equation and derive OLS *b* which are BLUE, why should we be concerned about ARCH? First, we could derive consistent estimates of *b* which are asymptotically more efficient than the OLS estimates, since the ARCH structure is no longer a linear model.

Second, the dynamics of the conditional variance are important in many contexts: particularly financial models, in which movements in volatility are themselves important. Many researchers have found “ARCH effects" in higher-frequency financial data, and to the extent to which they are present, we may want to take advantage of them. We may test for the existence of ARCH effects in the residuals of a time series regression by using the command `estat archlm`. The null hypothesis is that of no ARCH effects; a rejection of the null implies the existence of significant ARCH effects, or persistence in the squared errors.

The ARCH model is inherently nonlinear. If we assume that the u_t are distributed Normally, we may use a maximum likelihood procedure such as that implemented in Stata's `arch` command to jointly estimate its mean and conditional variance equation.

The ARCH model has been extended to a generalized form which has proven to be much more appropriate in many contexts. In the simplest example, we may write

$$h_t = \alpha_0 + \alpha_1 u_{t-1}^2 + \gamma_1 h_{t-1}$$

which is known as the GARCH(1,1) model since it involves a single lag of both the ARCH term and the conditional variance term. We must impose the additional constraint that $\gamma_1 > 0$ to ensure a positive variance.

We may also have a so-called ARCH-in-mean model, in which the h_t term itself enters the regression equation. This sort of model would be relevant if we had a theory that suggests that the level of a variable might depend on its variance, which may be very plausible in financial markets contexts or in terms of, say, inflation, where we often presume that the level of inflation may be linked to inflation volatility. In such instances we may want to specify a ARCH- or GARCH-in-mean model and consider interactions of this sort in the conditional mean (level) equation.

Alternative GARCH specifications

A huge literature on alternative GARCH specifications exists; many of these models are preprogrammed in Stata's `arch` command, and references for their analytical derivation are given in the Stata manual.

One of particular interest is Nelson's (1991) exponential GARCH, or EGARCH. He proposed:

$$\log h_t = \eta + \sum_{j=1}^{\infty} \lambda_j \left(\left| \epsilon_{t-j} \right| - E \left| \epsilon_{t-j} \right| + \theta \epsilon_{t-j} \right)$$

which is then parameterized as a rational lag of two finite–order polynomials, just as in Bollerslev's GARCH.

Advantages of the EGARCH specification include the positive nature of h_t irregardless of the estimated parameters, and the asymmetric nature of the impact of innovations: with $\theta \neq 0$, a positive shock will have a different effect on volatility than will a negative shock, mirroring findings in equity market research about the impact of "bad news" and "good news" on market volatility. For instance, a simple EGARCH(1,1) model will provide a variance equation such as

$$\log h_t = -\delta_0 + \delta_1 z_{t-1} + \delta_2 \left| z_{t-1} - \sqrt{2/\pi} \right| + \delta_3 \log h_{t-1}$$

where $z_t = \epsilon_t / \sigma_t$, which is distributed as $N(0, 1)$.

Nelson's model is only one of several extensions of GARCH that allow for asymmetry, or consider nonlinearities in the process generating the conditional variance: for instance, the threshold ARCH model of Zakoian (1990) and the Glosten et al. model (1993).

Stata provides a suite of commands to estimate time series models in the ARCH (Autoregressive Conditional Heteroskedasticity) family. The command `arch` is used to estimate single-equation models. Its options allow the specification of over a dozen models from the literature, including ARCH, GARCH, ARCH-in-mean, GARCH with ARMA errors, EGARCH (exponential GARCH), TARCH (threshold ARCH), GJR (Glosten et al., 1993), SAARCH (simple asymmetric ARCH), PARCH (power ARCH), NARCH (nonlinear ARCH), APARCH (asymmetric power ARCH) and NPARCH (nonlinear power ARCH).

Errors may be specified as Gaussian, *t*, or GED (generalized error distribution).

To estimate an ARCH model, you give the `arch` *varname* command, followed by (optionally) the independent variables in the mean equation and the options indicating the type of model. For instance, to fit a GARCH(1,1) to the mean regression of `cpi` on `wage`,

```
arch cpi wage, arch(1) garch(1)
```

It is important to note that a GARCH(2,1) model would be specified with the option `arch(1/2)`. If the option was given as `arch(2)`, only the second-order term would be included in the conditional variance equation.

A test for ARCH effects in a linear regression can be conducted with the `estat archlm` command. We can use Stata's `urates` dataset of monthly unemployment rates for several US states.

Try it out!

```
. webuse urates, clear
. qui reg D.tenn LD.tenn
. estat archlm, lags(3)
LM test for autoregressive conditional heteroskedasticity (ARCH)
```

lags(*p*)	chi2	df	Prob > chi2
3	11.195	3	0.0107

H0: no ARCH effects *vs.* H1: ARCH(*p*) disturbance

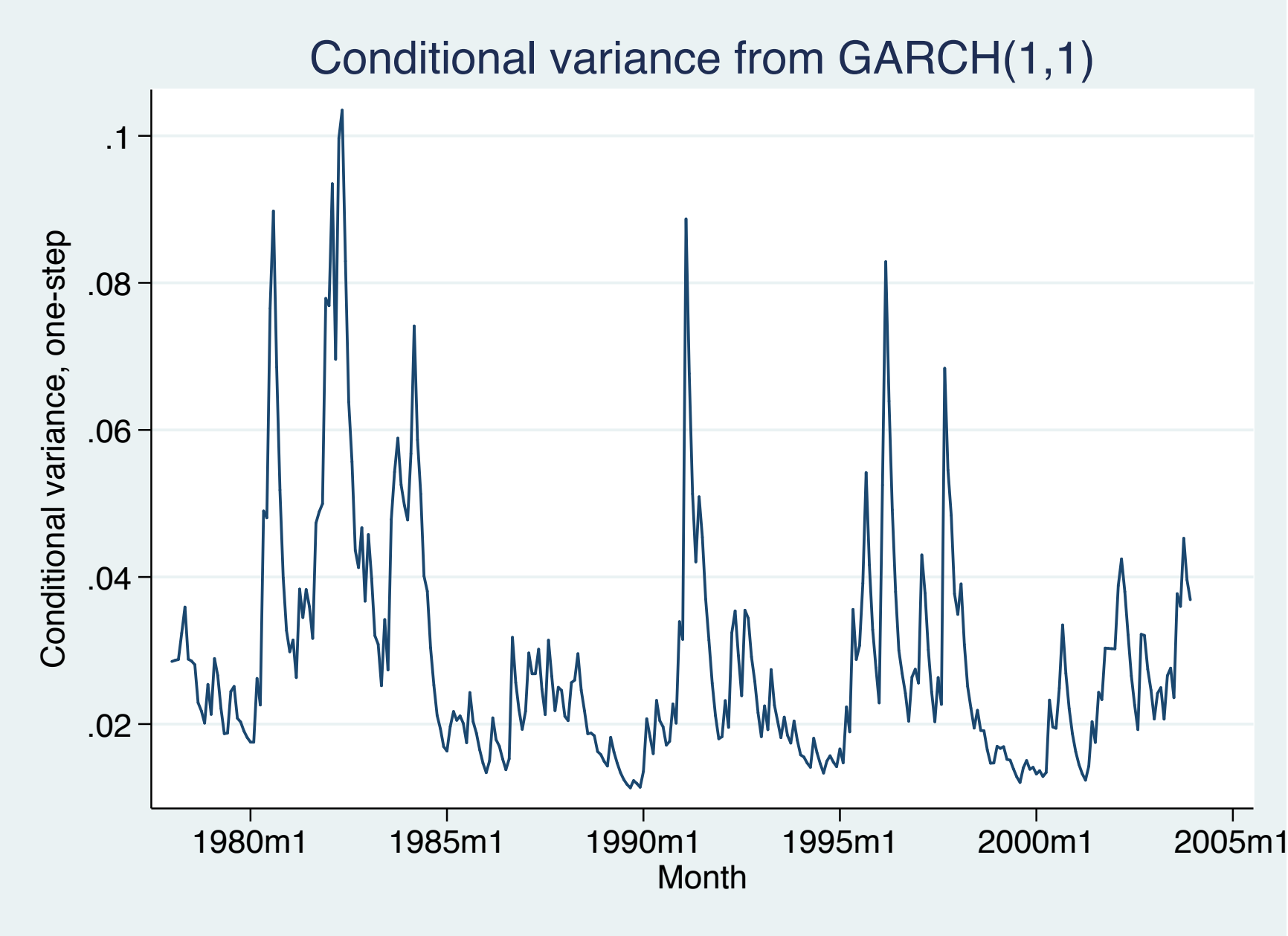
The LM test indicates the presence of significant ARCH effects.

We may also estimate a GARCH(1,1) model. **Try it out!**

```
. arch D.tenn LD.tenn, arch(1) garch(1) nolog vsquish
ARCH family regression
Sample: 1978m3 - 2003m12                        Number of obs   =       310
Distribution: Gaussian                          Wald chi2(1)    =      9.39
Log likelihood =  127.4172                      Prob > chi2     =    0.0022
```

D.tenn	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
tenn						
tenn						
LD.	.2129528	.0694996	3.06	0.002	.076736	.3491695
_cons	-.0155809	.0085746	-1.82	0.069	-.0323868	.0012251
ARCH						
arch						
L1.	.1929262	.0675544	2.86	0.004	.0605219	.3253305
garch						
L1.	.7138542	.0923551	7.73	0.000	.5328415	.894867
_cons	.0028566	.0016481	1.73	0.083	-.0003736	.0060868

Following estimation, we may use `predict` with the `variance` option to produce the conditional variance series.


Conditional variance from GARCH(1,1)
Conditional variance, one-step
.1
.08
.06
.04
.02
1980m1
1985m1
1990m1
1995m1
2000m1
2005m1
Month

We may also fit a model with additional variables in the mean equation:

```
. arch D.tenn LD.tenn LD.indiana LD.arkansas, arch(1) garch(1) nolog vsquish
ARCH family regression
Sample: 1978m3 - 2003m12                        Number of obs   =        310
Distribution: Gaussian                          Wald chi2(3)    =      41.31
Log likelihood =  135.1611                      Prob > chi2     =     0.0000
```

D.tenn	OPG Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
tenn						
tenn						
LD.	.1459972	.0723994	2.02	0.044	.004097	.2878974
indiana						
LD.	.1751591	.047494	3.69	0.000	.0820727	.2682455
arkansas						
LD.	.1170958	.0757688	1.55	0.122	-.0314083	.2655999
_cons	-.0078106	.0087075	-0.90	0.370	-.0248769	.0092558
ARCH						
arch						
L1.	.1627143	.0712808	2.28	0.022	.0230064	.3024221
garch						
L1.	.6793291	.1388493	4.89	0.000	.4071896	.9514687
_cons	.0042064	.0026923	1.56	0.118	-.0010704	.0094832

Following estimation, we may test hypotheses on the coefficients of the conditional variance equation: for instance, that they sum to unity, indicating integrated GARCH:

```
. test [ARCH]L.arch + [ARCH]L.garch == 1
 ( 1)  [ARCH]L.arch + [ARCH]L.garch = 1
           chi2(  1) =    2.30
         Prob > chi2 =    0.1297
```

In this case, that hypothesis cannot be rejected at 90%.

Multiple-equation GARCH models

Stata's `mgarch` command estimates multivariate GARCH models, allowing both the conditional mean and conditional covariance matrix to be dynamic. Four commonly used parameterizations are supported: the diagonal vech model, the constant conditional correlation model, the dynamic conditional correlation model, and the time-varying conditional correlation model.

A general MGARCH(1,1)) model may be written as:

$$\text{vech}(H_t) = s + A\,\text{vech}(\epsilon_{t-1}\epsilon_{t-1}') + B\,\text{vech}(H_{t-1})$$

where the `vech` function returns a vector containing the unique elements of its matrix argument. The various parameterizations of MGARCH provide alternative restrictions on H, the conditional covariance matrix, which must be positive definite.

Alternative parameterizations differ in terms of flexibility, allowing for more complex *H* processes, and parsimony, allowing the model to be specified with fewer parameters.

The oldest and simplest parameterization is the *diagonal vech* (DVECH) of Bollerslev, Engle, Wooldridge (1988), which restricts the *A* and *B* matrices to be diagonal. The number of parameters grows rapidly with the size of the model.

Conditional correlation (CC) models use nonlinear combinations of univariate GARCH models to represent the conditional covariances in *H*. They often have less difficulty with satisfying the restrictions on the estimated *H*, and their number of parameters grows more slowly than in the DVECH specification.

The constant CC model of Bollerslev (1990) specifies the correlation matrix as time invariant. Engle's (2002) extension, the dynamic CC model, allows the conditional correlations (technically, quasicorrelations) to follow a GARCH(1,1)-like process. Tse and Tsui's (2002) variant, the varying CC model, expresses the conditional correlations using a time-invariant component, a measure of recent correlations among the residuals, and last period's values.

The four MGARCH specifications are invoked with the `mgarch` command, with a first argument being the model specification: `dvech`, `ccc`, `dcc` or `vcc`.

To illustrate, we use Stata's `stocks` dataset, and model daily Toyota and Honda equity returns as AR(1) processes with the `ccc` and `dcc` specifications. In the CCC specification, the sizable correlation indicates the interaction between the two equations' error processes.

```
. webuse stocks, clear
(Data from Yahoo! Finance)
. mgarch ccc (toyota honda = L.toyota L.honda), arch(1) garch(1) nolog vsquish
Constant conditional correlation MGARCH model
Sample: 1 - 2015                                  Number of obs   =      2014
Distribution: Gaussian                            Wald chi2(4)    =      4.34
Log likelihood =  11602.61                        Prob > chi2     =    0.3620
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
toyota						
toyota						
L1.	-.03374	.032697	-1.03	0.302	-.097825	.030345
honda						
L1.	-.005188	.0288975	-0.18	0.858	-.0618261	.0514502
_cons	.0004523	.0003094	1.46	0.144	-.0001542	.0010587
ARCH_toyota						
arch						
L1.	.0661046	.0095018	6.96	0.000	.0474814	.0847279
garch						
L1.	.916793	.0117942	77.73	0.000	.8936769	.9399092
_cons	4.50e-06	1.19e-06	3.78	0.000	2.17e-06	6.83e-06
honda						
toyota						
L1.	-.0066352	.0343028	-0.19	0.847	-.0738675	.0605971
honda						
L1.	-.0332976	.0316213	-1.05	0.292	-.0952743	.028679
_cons	.0006128	.0003394	1.81	0.071	-.0000524	.0012781
ARCH_honda						
arch						
L1.	.0498417	.0080311	6.21	0.000	.0341009	.0655824
garch						
L1.	.9321435	.0111601	83.52	0.000	.9102701	.9540168
_cons	5.26e-06	1.41e-06	3.73	0.000	2.50e-06	8.02e-06
Correlation						
toyota						
honda	.7176095	.0108477	66.15	0.000	.6963483	.7388707

In the DCC model, the diagonal elements of H_t are modeled as univariate GARCH models. The off-diagonal elements are modeled as nonlinear functions of the diagonal terms:

$$h_{ij,t} = \rho_{ij,t}\sqrt{h_{ii,t}h_{jj,t}}$$

where $\rho_{ij,t}$ follows a dynamic process, rather than being constrained to be constant as in the CCC specification.

Two additional parameters, λ_1 and λ_2, are adjustment parameters that govern the evolution of the conditional quasicorrelations. They must be positive and sum to less than one. A test for the sum of these parameters equalling zero tests the DCC model against the special case of the CCC model.

With the DCC specification:

```
. mgarch dcc (toyota honda = L.toyota L.honda), arch(1) garch(1) nolog vsquish
Dynamic conditional correlation MGARCH model
Sample: 1 - 2015                                  Number of obs   =      2014
Distribution: Gaussian                            Wald chi2(4)    =      4.81
Log likelihood =  11624.54                        Prob > chi2     =    0.3074
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
toyota						
toyota						
L1.	-.0346653	.0319267	-1.09	0.278	-.0972404	.0279098
honda						
L1.	-.0069742	.0284872	-0.24	0.807	-.0628081	.0488597
_cons	.000373	.0003108	1.20	0.230	-.0002362	.0009821
ARCH_toyota						
arch						
L1.	.0629146	.0093309	6.74	0.000	.0446263	.0812029
garch						
L1.	.9208039	.0116908	78.76	0.000	.8978904	.9437175
_cons	4.32e-06	1.16e-06	3.72	0.000	2.04e-06	6.60e-06

...

honda						
toyota						
L1.	.0030378	.0339118	0.09	0.929	-.0634281	.0695036
honda						
L1.	-.0367691	.0316091	-1.16	0.245	-.0987219	.0251836
_cons	.0005624	.000341	1.65	0.099	-.0001059	.0012307
ARCH_honda						
arch						
L1.	.0536899	.008511	6.31	0.000	.0370087	.0703711
garch						
L1.	.928433	.0115932	80.08	0.000	.9057107	.9511554
_cons	5.43e-06	1.44e-06	3.77	0.000	2.61e-06	8.26e-06
Correlation						
toyota						
honda	.7264858	.0132659	54.76	0.000	.7004852	.7524864
Adjustment						
lambda1	.0528653	.014217	3.72	0.000	.0250005	.0807301
lambda2	.746622	.0746374	10.00	0.000	.6003354	.8929085

In both the CCC and DCC specifications, the mean equations indicate that lagged daily returns of both stocks are not significant determinants of current returns, as would be expected from efficient markets theory.

There are very significant GARCH effects in both specifications. A sizable correlation parameter appears, as it did in the CCC specification. The magnitudes of the `lambda` parameters indicate that the evolution of the conditional covariances depends more on their past values than on lagged residuals' innovations.

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 lag-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:

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

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.

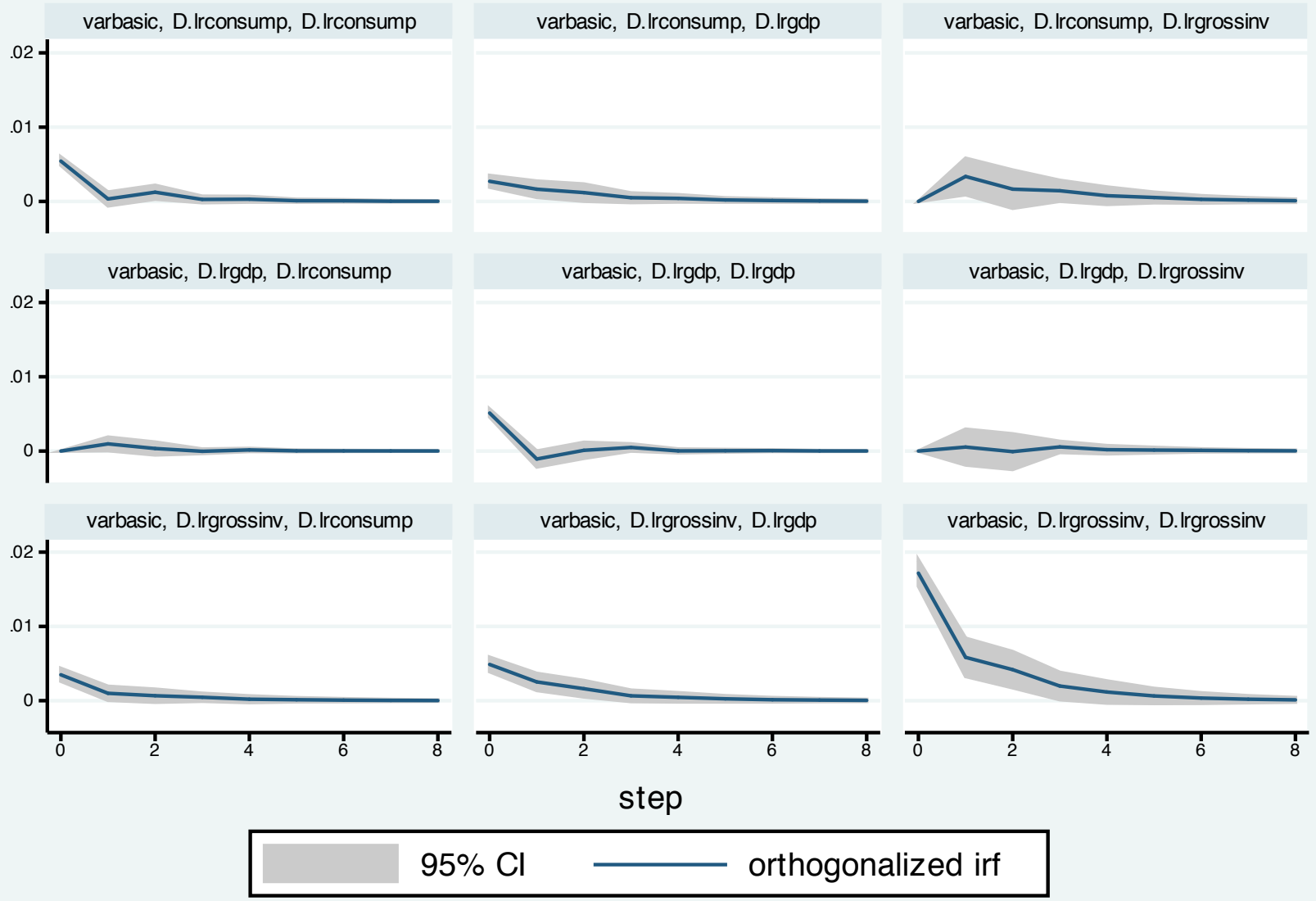
If there are additional (strictly) exogenous variables in the VAR, the dynamic multiplier functions or transfer functions can be computed. These measure the impact of a unit change in the exogenous variable on the endogenous variables over time. They are generated by `fcast compute` and graphed with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

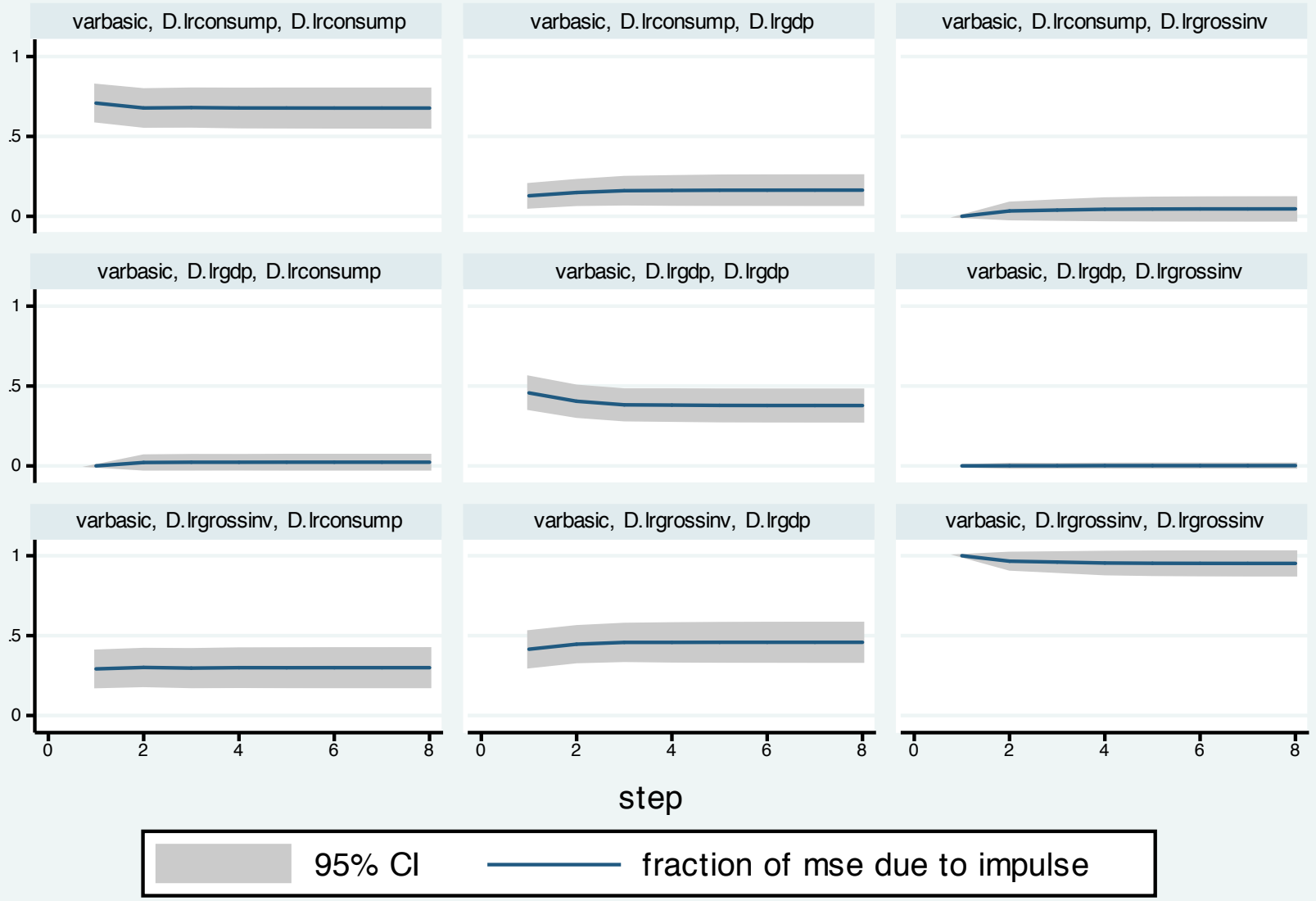
```
. bcuse usmacro1
. varbasic D.lrgrossinv D.lrconsump D.lrgdp if tin(,2005q4)
Vector autoregression

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
             |
       _cons |  -.0009508   .0027881    -0.34   0.733    -.0064153    .0045138
-------------+----------------------------------------------------------------
D_lrconsump  |
```



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)
```



Graphs by irfname, impulse variable, and response variable

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. **Try it out!**

```
. 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
lrgdp						
LD.	.1354875	.2455182	0.55	0.581	-.3457193	.6166942
L2D.	.0414686	.254353	0.16	0.870	-.4570541	.5399914
L3D.	.1304675	.2523745	0.52	0.605	-.3641774	.6251124
L4D.	-.135457	.2366945	-0.57	0.567	-.5993698	.3284558
lrmbase						
D1.	.0396035	.1209596	0.33	0.743	-.1974729	.2766799
_cons	-.0030005	.003383	-0.89	0.375	-.0096311	.0036302
D_lrconsump						
LD.	.0217782	.0363098	0.60	0.549	-.0493876	.092944

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 + .1840599*i*	.607851
-.5793137 - .1840599*i*	.607851
-.3792302 + .4714717*i*	.605063
-.3792302 - .4714717*i*	.605063
.1193592 + .5921967*i*	.604106
.1193592 - .5921967*i*	.604106
.5317127 + .2672997*i*	.59512
.5317127 - .2672997*i*	.59512
-.4579249	.457925
.1692559 + .3870966*i*	.422482
.1692559 - .3870966*i*	.422482

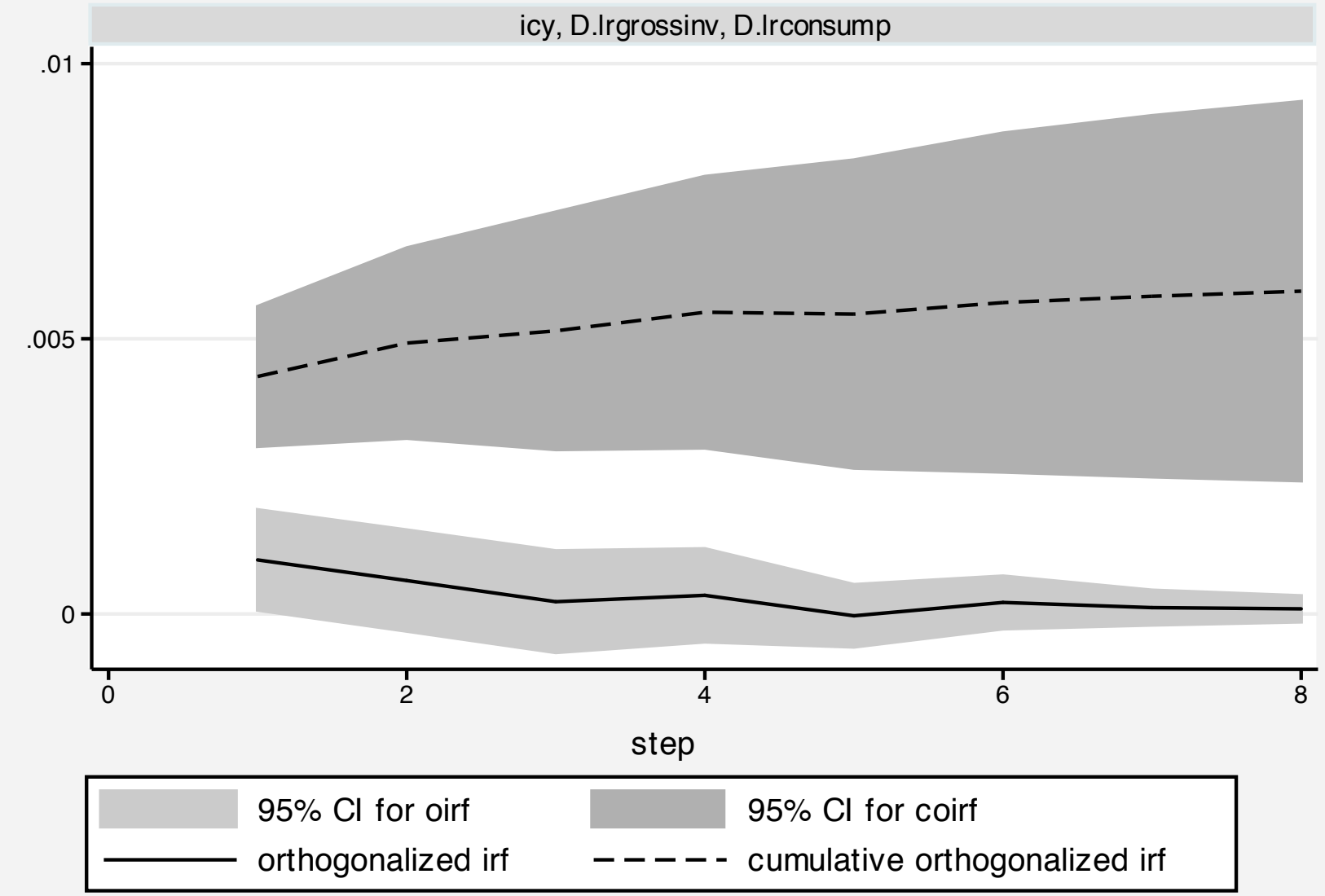
All the eigenvalues lie inside the unit circle.
VAR satisfies stability condition.

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
```

step	(1) oirf	(1) S.E.	(1) coirf	(1) 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)
```



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
/b_2_1	(omitted)					
/b_3_1	(omitted)					
/b_1_2	(omitted)					
/b_2_2	.0054308	.0002823	19.24	0.000	.0048774	.0059841
/b_3_2	(omitted)					
/b_1_3	(omitted)					
/b_2_3	(omitted)					
/b_3_3	.0051196	.0002662	19.24	0.000	.0045979	.0056412

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 long-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:

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

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

Unlike VAR models, VECMs do not support additional exogenous variables other than constant, trend and seasonal indicator variables.

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 (9.0) 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.

```
. bcuse pwt90, clear nodesc
. keep if inlist(countrycode,"GBR")
(11,765 observations deleted)
. g lp = log(pl_da)
. g lxr = log(xr)
. varsoc lp lxr if tin(1959,2007)
```

Selection-order criteria
Sample: 1959 - 2007 Number of obs = 49

lag	LL	LR	df	p	FPE	AIC	HQIC	SBIC
0	-27.8269				.011582	1.21742	1.24672	1.29464
1	143.22	342.09	4	0.000	.000013	-5.60083	-5.51294	-5.36918
2	185.487	84.533*	4	0.000	2.7e-06*	-7.16273*	-7.01625*	-6.77665*
3	187.221	3.4685	4	0.483	2.9e-06	-7.07025	-6.86518	-6.52973
4	189.28	4.1183	4	0.390	3.2e-06	-6.99104	-6.72737	-6.29608

Endogenous: lp lxr
Exogenous: _cons

Two lags are selected by all of the criteria.

```
. vecrank lp lxr if tin(1959,2007)
                    Johansen tests for cointegration
Trend: constant                                   Number of obs =       49
Sample:  1959 - 2007                                    Lags =        2
------------------------------------------------------------------------------
                                                           5%
maximum                                          trace     critical
  rank     parms       LL         eigenvalue  statistic     value
    0        6      177.10544          .        16.7631     15.41
    1        9      184.14177       0.24964      2.6904*     3.76
    2       10      185.48697       0.05343
------------------------------------------------------------------------------
```

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 lxr if tin(1959,2007), lags(2)
Vector error-correction model
Sample:  1959 - 2007                              Number of obs     =         49
                                                  AIC               =  -7.148644
Log likelihood =  184.1418                        HQIC              =  -7.016811
Det(Sigma_ml)  =  1.87e-06                        SBIC              =  -6.801167

Equation           Parms      RMSE     R-sq      chi2     P>chi2
------------------------------------------------------------------
D_lp                   4     .067877   0.5190   48.55634   0.0000
D_lxr                  4     .062825   0.3940   29.25541   0.0000
------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
D_lp         |
        _ce1 |
         L1. |  -.0848576   .0230353    -3.68   0.000    -.1300059   -.0397092
             |
          lp |
         LD. |   .4920727   .2179366     2.26   0.024     .0649248    .9192206
             |
         lxr |
         LD. |  -.1062776   .2372436    -0.45   0.654    -.5712665    .3587114
             |
       _cons |   .0032079   .0161835     0.20   0.843    -.0285112     .034927
------------------------------------------------------------------------------
```

(continued)

D_lxr						
_ce1						
L1.	.0824095	.0213207	3.87	0.000	.0406216	.1241973
lp						
LD.	.3872178	.201715	1.92	0.055	-.0081365	.782572
lxr						
LD.	.9161414	.219585	4.17	0.000	.4857627	1.34652
_cons	.0033032	.0149789	0.22	0.825	-.026055	.0326614

Cointegrating equations

Equation	Parms	chi2	P>chi2
_ce1	1	58.30822	0.0000

Identification: beta is exactly identified

Johansen normalization restriction imposed

beta	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_ce1						
lp	1	.	.	.	.	.
lxr	-3.507915	.4593927	-7.64	0.000	-4.408308	-2.607521
_cons	-1.680548	.	.	.	.	.

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 exchange rate coefficient in this equation is not significantly different from zero.

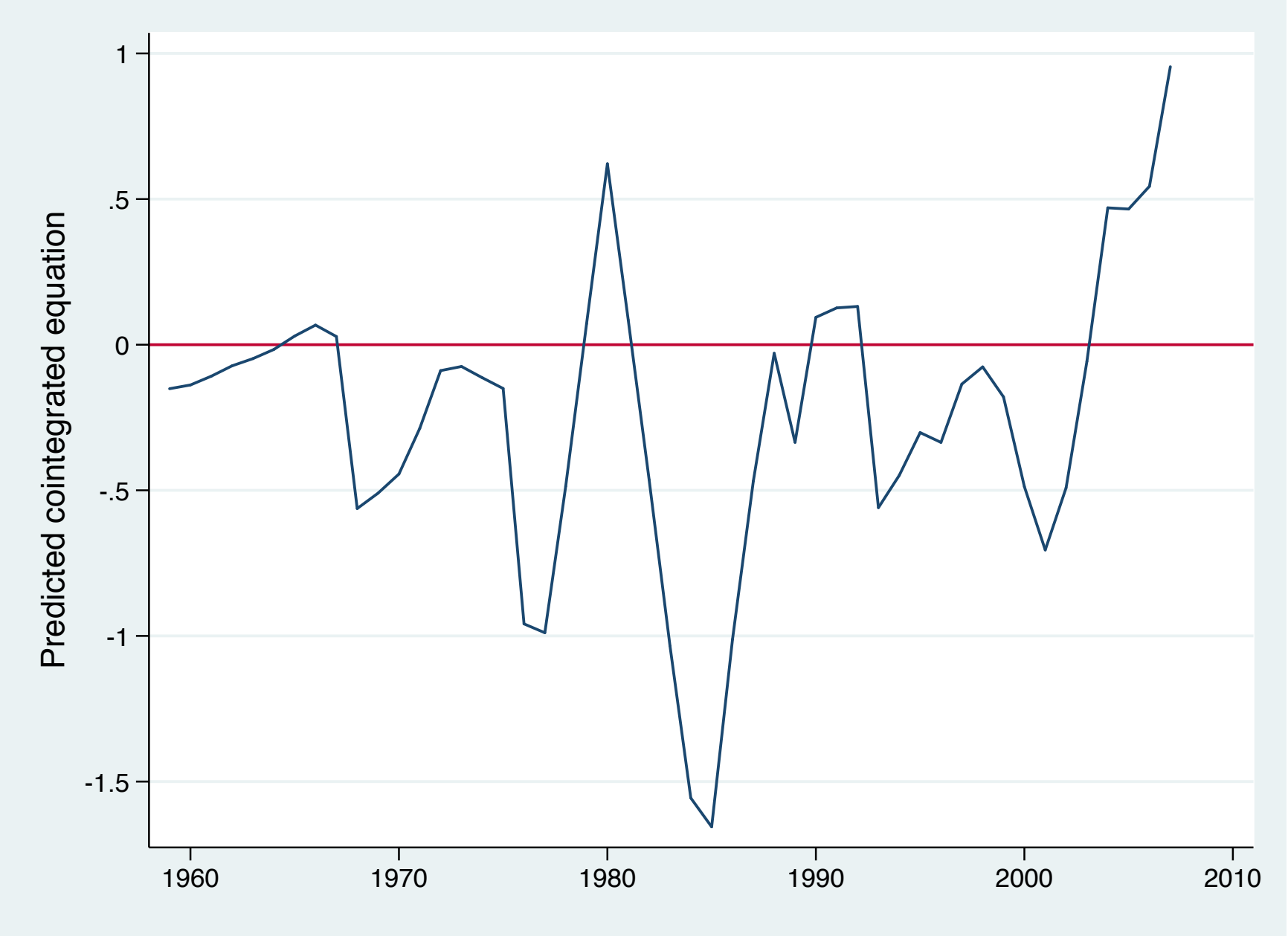
In the `lxr` equation, the lagged error correction term is positive, as it must be for the other variable in the relationship: that is, if $(\log p - \log e)$ is above long-run equilibrium, either p must fall or e must rise. Both short-run coefficients are positive and significant, with very high persistence in the exchange rate.

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 -3.51 on `lxrat`, significantly different from zero. The value clearly rejects -1.0, the value implied by long-run PPP. 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 eq(#1)
. tsline ce1 if e(sample), ylab(,angle(0) labs(small)) yline(0) ///
> xlab(,labs(small)) xti("")
```


Predicted cointegrated equation
1
.5
0
-.5
-1
-1.5
1960
1970
1980
1990
2000
2010

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.

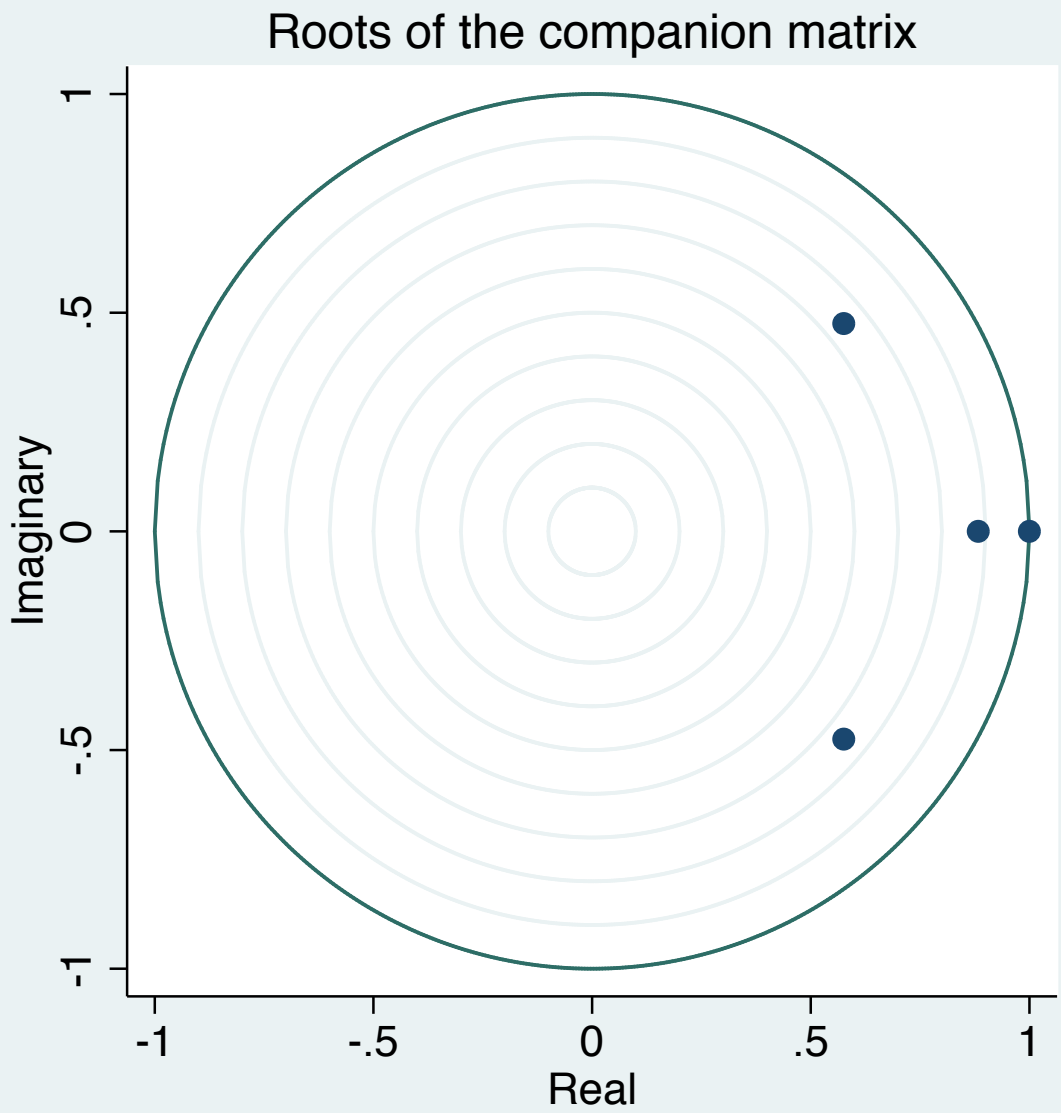
```
. vecstable, graph
```

Eigenvalue stability condition

Eigenvalue	Modulus
1	1
.8831188	.883119
.5755762 + .4751674*i*	.746373
.5755762 - .4751674*i*	.746373

The VECM specification imposes a unit modulus.

The eigenvalues meet the stability condition.



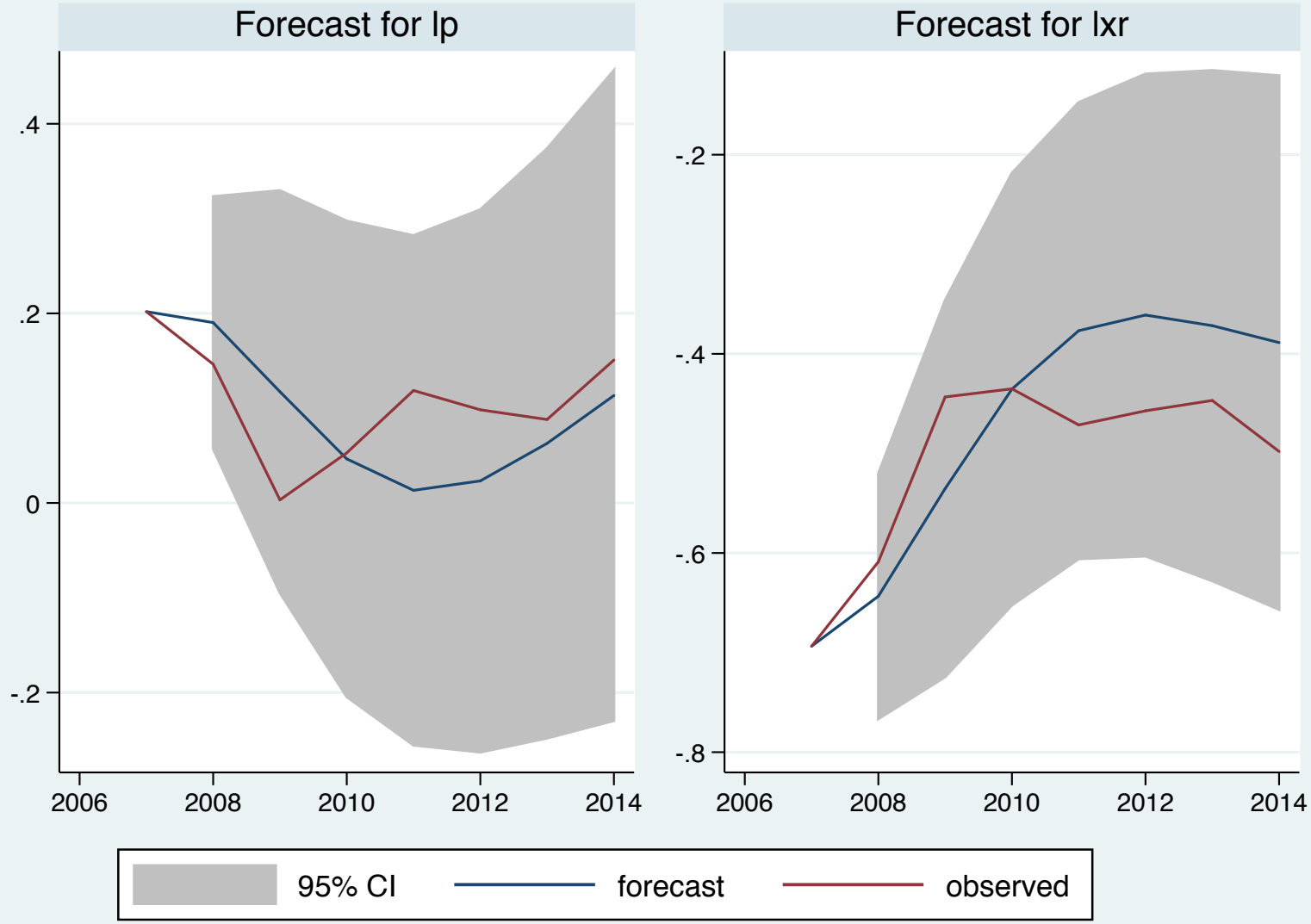
The VECM specification imposes 1 unit modulus

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 2007, and now forecast through the end of available data in 2014:

```
. tsset year, yearly
        time variable:  year, 1950 to 2014
                delta:  1 year
. fcast compute ppp_, step(7)
. fcast graph ppp_lp ppp_lxr, observed legend(rows(1) size(small)) ///
> ylab(,angle(0) labs(small)) xlab(,labs(small)) ///
> byopts(ti("Ex ante VECM forecasts, UK vs US RER components"))
```

Ex ante VECM forecasts, UK vs US RER components



Although the confidence intervals are quite wide, the model's dynamic forecasts do a reasonable job of tracking the components of the real exchange rate through this turbulent 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.

Panel vector autoregression

Abrigo and Love have developed a suite of commands to implement panel vector autoregression in Stata, as described in their 2016 *Stata Journal* article (16: 778–804). Their routines are designed to implement "model selection, estimation, and inference of homogeneous panel VAR models in a generalized method of moments framework." Earlier versions of their routines have been widely used in published work.

These routines include the estimation command `pvar` as well as `pvarsoc` (selection order criteria), `pvargranger` (Granger causality tests), `pvarstable` (stability tests), `pvarirf` (impulse response functions) and `pvarfevd` (forecast error vector decompositions), mirroring Stata's `var` suite of commands in their functionality.

Panel cointegration tests

Just as a VAR may be estimated in a panel setting, Stata provides implementations of three panel-data cointegration tests:

- Kao (1999), with the null that variables are cointegrated in all panels. Under the alternative, there is a single cointegrating vector.
- Pedroni (1999, 2004), with the null that variables are cointegrated in all panels. Under the alternative, there can be panel-specific cointegrating vectors.
- Westerlund (2005), with the null that variables are cointegrated in some (or all) panels. Under the alternative, there can be panel-specific cointegrating vectors.

Each of these tests allow for panel-specific means (fixed effects) and panel-specific time trends in the cointegrating regression. They are most fruitfully applied to “large N, large T” datasets.

Model specification, solution and dynamic forecasting

The `forecast` suite, introduced in Stata version 13, contains commands that support the definition of a model, containing a number of estimated equations and identities, and the ability to produce static and dynamic forecasts from the possibly nonlinear structure. For instance, a model might predict the percentage growth rate of GDP, but contain the national income identity in the level of GDP.

The `forecast` commands produce dynamic forecasts via Gauss–Seidel, Broyden–Powell or Newton–Raphson methods. Simulation methods may be used to obtain prediction intervals, and scenario analysis may easily be performed, allowing the comparison of a baseline forecast and a 'what-if' scenario involving alternate exogenous factors. For many Stata users, this suite of commands will greatly enhance Stata's usefulness in their work with time series.

Additional time series capabilities

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.

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

- Linearized dynamic stochastic general equilibrium models (`dsge`)
- linear state-space models via the Kalman filter (`sspace`)
- dynamic-factor multivariate time series models (`dfactor`)
- models of fractionally integrated (ARFIMA) time series (`arfima`)