Title

stata.com

arima postestimation — Postestimation tools for arima

Postestimation commands

The following postestimation commands are of special interest after `arima`:

Command	Description
`estat acplot`	estimate autocorrelations and autocovariances
`estat aroots`	check stability condition of estimates
`irf`	create and analyze IRFs
`psdensity`	estimate the spectral density

The following standard postestimation commands are also available:

Command	Description
`estat ic`	Akaike's and Schwarz's Bayesian information criteria (AIC and BIC)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`forecast`	dynamic forecasts and simulations
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of coefficients
`lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	predictions, residuals, influence statistics, and other diagnostic measures
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates a new variable containing predictions such as expected values and mean squared errors. All predictions are available as static one-step-ahead predictions or as dynamic multistep predictions, and you can control when dynamic predictions begin.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [`,` *statistic options*]

statistic	Description
Main	
`xb`	predicted values for mean equation—the differenced series; the default
`stdp`	standard error of the linear prediction
`y`	predicted values for the mean equation in y—the undifferenced series
`mse`	mean squared error of the predicted values
`residuals`	residuals or predicted innovations
`yresiduals`	residuals or predicted innovations in y, reversing any time-series operators

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Predictions are not available for conditional ARIMA models fit to panel data.

options	Description
Options	
`dynamic(`*time_constant*`)`	how to handle the lags of y_t
`t0(`*time_constant*`)`	set starting point for the recursions to *time_constant*
`structural`	calculate considering the structural component only

time_constant is a # or a time literal, such as `td(1jan1995)` or `tq(1995q1)`; see *Conveniently typing SIF values* in [D] **datetime**.

Options for predict

Six statistics can be computed using `predict` after `arima`: the predictions from the model (the default also given by `xb`), the standard error of the linear prediction (`stdp`), the predictions after reversing any time-series operators applied to the dependent variable (`y`), the MSE of `xb` (`mse`), the predictions of residuals or innovations (`residual`), and the predicted residuals or innovations in terms of y (`yresiduals`). Given the dynamic nature of the ARMA component and because the dependent variable might be differenced, there are other ways of computing each. We can use all the data on the dependent variable that is available right up to the time of each prediction (the default, which is

often called a one-step prediction), or we can use the data up to a particular time, after which the predicted value of the dependent variable is used recursively to make later predictions (`dynamic()`). Either way, we can consider or ignore the ARMA disturbance component (the component is considered by default and is ignored if you specify `structural`).

All calculations can be made in or out of sample.

Main

`xb`, the default, calculates the predictions from the model. If `D.`*depvar* is the dependent variable, these predictions are of `D.`*depvar* and not of *depvar* itself.

`stdp` calculates the standard error of the linear prediction `xb`. `stdp` does not include the variation arising from the disturbance equation; use `mse` to calculate standard errors and confidence bands around the predicted values.

`y` specifies that predictions of *depvar* be made, even if the model was specified in terms of, say, `D.`*depvar*.

`mse` calculates the MSE of the predictions.

`residuals` calculates the residuals. If no other options are specified, these are the predicted innovations ϵ_t; that is, they include the ARMA component. If `structural` is specified, these are the residuals μ_t from the structural equation; see `structural` below.

`yresiduals` calculates the residuals in terms of *depvar*, even if the model was specified in terms of, say, `D.`*depvar*. As with `residuals`, the `yresiduals` are computed from the model, including any ARMA component. If `structural` is specified, any ARMA component is ignored, and `yresiduals` are the residuals from the structural equation; see `structural` below.

Options

`dynamic(`*time_constant*`)` specifies how lags of y_t in the model are to be handled. If `dynamic()` is not specified, actual values are used everywhere that lagged values of y_t appear in the model to produce one-step-ahead forecasts.

`dynamic(`*time_constant*`)` produces dynamic (also known as recursive) forecasts. *time_constant* specifies when the forecast is to switch from one step ahead to dynamic. In dynamic forecasts, references to y_t evaluate to the prediction of y_t for all periods at or after *time_constant*; they evaluate to the actual value of y_t for all prior periods.

For example, `dynamic(10)` would calculate predictions in which any reference to y_t with $t < 10$ evaluates to the actual value of y_t and any reference to y_t with $t \geq 10$ evaluates to the prediction of y_t. This means that one-step-ahead predictions are calculated for $t < 10$ and dynamic predictions thereafter. Depending on the lag structure of the model, the dynamic predictions might still refer some actual values of y_t.

You may also specify `dynamic(.)` to have `predict` automatically switch from one-step-ahead to dynamic predictions at $p + q$, where p is the maximum AR lag and q is the maximum MA lag.

`t0(`*time_constant*`)` specifies the starting point for the recursions to compute the predicted statistics; disturbances are assumed to be 0 for $t <$ `t0()`. The default is to set `t0()` to the minimum t observed in the estimation sample, meaning that observations before that are assumed to have disturbances of 0.

`t0()` is irrelevant if `structural` is specified because then all observations are assumed to have disturbances of 0.

`t0(5)` would begin recursions at $t = 5$. If the data were quarterly, you might instead type `t0(tq(1961q2))` to obtain the same result.

The ARMA component of ARIMA models is recursive and depends on the starting point of the predictions. This includes one-step-ahead predictions.

`structural` specifies that the calculation be made considering the structural component only, ignoring the ARMA terms, producing the steady-state equilibrium predictions.

margins

Description for margins

`margins` estimates margins of response for expected values.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`xb`	predicted values for mean equation—the differenced series; the default
`y`	predicted values for the mean equation in y—the undifferenced series
`stdp`	not allowed with `margins`
`mse`	not allowed with `margins`
`residuals`	not allowed with `margins`
`yresiduals`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Forecasting after ARIMA
IRF results for ARIMA

Forecasting after ARIMA

We assume that you have already read [TS] **arima**. In this section, we illustrate some of the features of `predict` after fitting ARIMA, ARMAX, and other dynamic models by using `arima`. In example 2 of [TS] **arima**, we fit the model

$$\Delta \ln(wpi_t) = \beta_0 + \rho_1\{\Delta \ln(wpi_{t-1}) - \beta_0\} + \theta_1\epsilon_{t-1} + \theta_4\epsilon_{t-4} + \epsilon_t$$

by typing

```
. use http://www.stata-press.com/data/r15/wpi1
. arima D.ln_wpi, ar(1) ma(1 4)
  (output omitted)
```

If we use the command

```
. predict xb, xb
```

then Stata computes $\texttt{xb}_t$ as

$$\texttt{xb}_t = \widehat{\beta}_0 + \widehat{\rho}_1\{\Delta \ln(wpi_{t-1}) - \widehat{\beta}_0\} + \widehat{\theta}_1\widehat{\epsilon}_{t-1} + \widehat{\theta}_4\widehat{\epsilon}_{t-4}$$

where

$$\widehat{\epsilon}_{t-j} = \begin{cases} \Delta \ln(wpi_{t-j}) - \texttt{xb}_{t-j} & t-j > 0 \\ 0 & \text{otherwise} \end{cases}$$

meaning that `predict` *newvar*`, xb` calculates predictions by using the metric of the dependent variable. In this example, the dependent variable represented *changes* in $\ln(wpi_t)$, and so the predictions are likewise for *changes* in that variable.

If we instead use

```
. predict y, y
```

Stata computes $\texttt{y}_t$ as $\texttt{y}_t = \texttt{xb}_t + \ln(wpi_{t-1})$ so that $\texttt{y}_t$ represents the predicted *levels* of $\ln(wpi_t)$. In general, `predict` *newvar*`, y` will reverse any time-series operators applied to the dependent variable during estimation.

If we want to ignore the ARMA error components when making predictions, we use the `structural` option,

```
. predict xbs, xb structural
```

which generates $\texttt{xbs}_t = \widehat{\beta}_0$ because there are no regressors in this model, and

```
. predict ys, y structural
```

generates $\texttt{ys}_t = \widehat{\beta}_0 + \ln(wpi_{t-1})$

▷ Example 1: Dynamic forecasts

An attractive feature of the `arima` command is the ability to make dynamic forecasts. In example 4 of [TS] **arima**, we fit the model

$$\texttt{consump}_t = \beta_0 + \beta_1\texttt{m2}_t + \mu_t$$
$$\mu_t = \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t$$

First, we refit the model by using data up through the first quarter of 1978, and then we will evaluate the one-step-ahead and dynamic forecasts.

```
. use http://www.stata-press.com/data/r15/friedman2
. keep if time<=tq(1981q4)
(67 observations deleted)
. arima consump m2 if tin(, 1978q1), ar(1) ma(1)
  (output omitted)
```

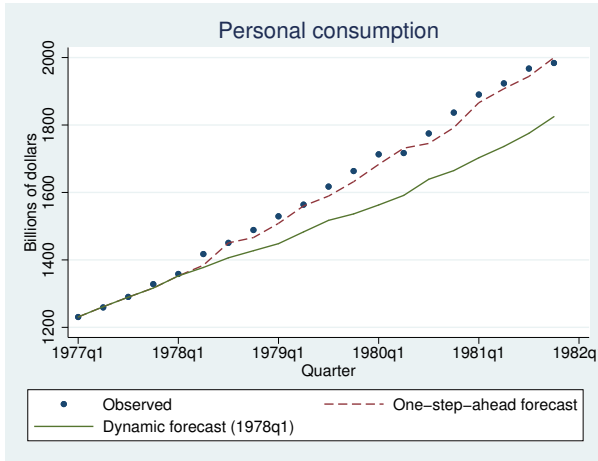
To make one-step-ahead forecasts, we type

```
. predict chat, y
(52 missing values generated)
```

(Because our dependent variable contained no time-series operators, we could have instead used `predict chat, xb` and accomplished the same thing.) We will also make dynamic forecasts, switching from observed values of `consump` to forecasted values at the first quarter of 1978:

```
. predict chatdy, dynamic(tq(1978q1)) y
(52 missing values generated)
```

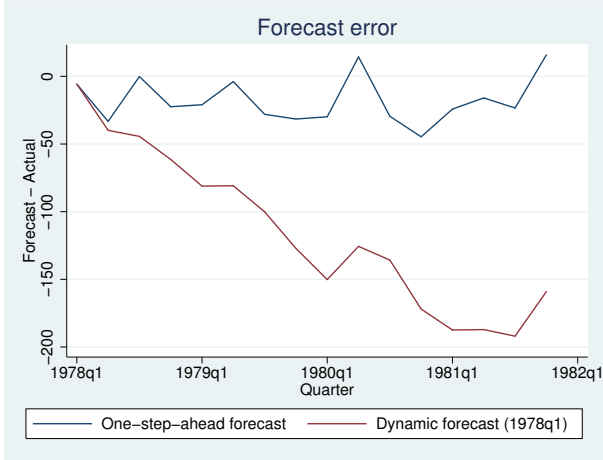
The following graph compares the forecasted values to the observed values for the first few years following the estimation sample:



The one-step-ahead forecasts never deviate far from the observed values, though over time the dynamic forecasts have larger errors. To understand why that is the case, rewrite the model as

$$\begin{aligned}\texttt{consump}_t &= \beta_0 + \beta_1 \texttt{m2}_t + \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \\ &= \beta_0 + \beta_1 \texttt{m2}_t + \rho\left(\texttt{consump}_{t-1} - \beta_0 - \beta_1 \texttt{m2}_{t-1}\right) + \theta\epsilon_{t-1} + \epsilon_t\end{aligned}$$

This form shows that the forecasted value of consumption at time t depends on the value of consumption at time $t-1$. When making the one-step-ahead forecast for period t, we know the actual value of consumption at time $t-1$. On the other hand, with the `dynamic(tq(1978q1))` option, the forecasted value of consumption for period 1978q1 is based on the observed value of consumption in period 1977q4, but the forecast for 1978q2 is based on the forecast value for 1978q1, the forecast for 1978q3 is based on the forecast value for 1978q2, and so on. Thus, with dynamic forecasts, prior forecast errors accumulate over time. The following graph illustrates this effect.



IRF results for ARIMA

We assume that you have already read [TS] **irf** and [TS] **irf create**. In this section, we illustrate how to calculate the impulse–response function (IRF) of an ARIMA model.

Example 2

Consider a model of the quarterly U.S. money supply, as measured by M1, from Enders (2004). Enders (2004, 93–97) discusses why seasonal shopping patterns cause seasonal effects in M1. The variable `lnm1` contains data on the natural log of the money supply. We fit seasonal and nonseasonal ARIMA models and compare the IRFs calculated from both models.

We fit the following nonseasonal ARIMA model

$$\Delta\Delta_4 \texttt{lnm1}_t = \rho_1(\Delta\Delta_4 \texttt{lnm1}_{t-1}) + \rho_4(\Delta\Delta_4 \texttt{lnm1}_{t-4}) + \epsilon_t$$

The code below fits the above model and saves a set of IRF results to a file called `myirf.irf`.

```
. use http://www.stata-press.com/data/r15/m1nsa, clear
(U.S. money supply (M1) from Enders (2004), 95-99.)
. arima DS4.lnm1, ar(1 4) noconstant nolog
ARIMA regression
Sample:  1961q2 - 2008q2                        Number of obs     =        189
                                                Wald chi2(2)      =      78.34
Log likelihood =  579.3036                      Prob > chi2       =     0.0000
```

DS4.lnm1	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
ARMA						
ar						
L1.	.3551862	.0503011	7.06	0.000	.2565979	.4537745
L4.	-.3275808	.0594953	-5.51	0.000	-.4441895	-.210972
/sigma	.0112678	.0004882	23.08	0.000	.0103109	.0122246

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
. irf create nonseasonal, set(myirf) step(30)
(file myirf.irf created)
(file myirf.irf now active)
(file myirf.irf updated)
```

We fit the following seasonal ARIMA model

$$(1-\rho_1 L)(1-\rho_{4,1}L^4)\Delta\Delta_4 \texttt{lnm1}_t = \epsilon_t$$

The code below fits this nonseasonal ARIMA model and saves a set of IRF results to the active IRF file, which is `myirf.irf`.

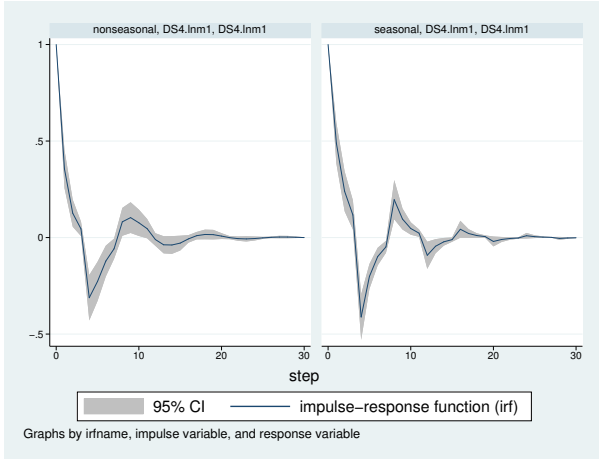
```
. arima DS4.lnm1, ar(1) mar(1,4) noconstant nolog
ARIMA regression
Sample:  1961q2 - 2008q2                        Number of obs     =        189
                                                Wald chi2(2)      =     119.78
Log likelihood =  588.6689                      Prob > chi2       =     0.0000
```

DS4.lnm1	Coef.	OPG Std. Err.	z	P>|z|	[95% Conf.	Interval]
ARMA						
ar						
L1.	.489277	.0538033	9.09	0.000	.3838245	.5947296
ARMA4						
ar						
L1.	-.4688653	.0601248	-7.80	0.000	-.5867076	-.3510229
/sigma	.0107075	.0004747	22.56	0.000	.0097771	.0116379

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
. irf create seasonal, step(30)
(file myirf.irf updated)
```

We now have two sets of IRF results in the file `myirf.irf`. We can graph both IRF functions side by side by calling `irf graph`.

```
. irf graph irf
```



The trajectories of the IRF functions are similar: each figure shows that a shock to `lnm1` causes a temporary oscillation in `lnm1` that dies out after about 15 time periods. This behavior is characteristic of short-memory processes.

◁

See [TS] **psdensity** for an introduction to estimating spectral densities using the parameters estimated by `arima`.

Reference

Enders, W. 2004. *Applied Econometric Time Series*. 2nd ed. New York: Wiley.

Also see

[TS] **arima** — ARIMA, ARMAX, and other dynamic regression models

[TS] **estat acplot** — Plot parametric autocorrelation and autocovariance functions

[TS] **estat aroots** — Check the stability condition of ARIMA estimates

[TS] **irf** — Create and analyze IRFs, dynamic-multiplier functions, and FEVDs

[TS] **psdensity** — Parametric spectral density estimation after arima, arfima, and ucm

[U] 20 Estimation and postestimation commands