Title

arima — ARIMA, ARMAX, and other dynamic regression models

Syntax

Basic syntax for a regression model with ARMA *disturbances*

`arima` *depvar* [*indepvars*] `,` `ar(`*numlist*`)` `ma(`*numlist*`)`

Basic syntax for an ARIMA(p,d,q) *model*

`arima` *depvar* `,` `arima(`$\#_p$`,`$\#_d$`,`$\#_q$`)`

Basic syntax for a multiplicative seasonal ARIMA$(p,d,q)\times(P,D,Q)_s$ *model*

`arima` *depvar* `,` `arima(`$\#_p$`,`$\#_d$`,`$\#_q$`)` `sarima(`$\#_P$`,`$\#_D$`,`$\#_Q$`,`$\#_s$`)`

Full syntax

`arima` *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [`,` *options*]

options	description
Model	
`noconstant`	suppress constant term
`arima(`$\#_p$`,`$\#_d$`,`$\#_q$`)`	specify ARIMA(p,d,q) model for dependent variable
`ar(`*numlist*`)`	autoregressive terms of the structural model disturbance
`ma(`*numlist*`)`	moving-average terms of the structural model disturbance
`constraints(`*constraints*`)`	apply specified linear constraints
`collinear`	keep collinear variables
Model 2	
`sarima(`$\#_P$`,`$\#_D$`,`$\#_Q$`,`$\#_s$`)`	specify period-$\#_s$ multiplicative seasonal ARIMA term
`mar(`*numlist*`,` $\#_s$`)`	multiplicative seasonal autoregressive term; may be repeated
`mma(`*numlist*`,` $\#_s$`)`	multiplicative seasonal moving-average term; may be repeated
Model 3	
`condition`	use conditional MLE instead of full MLE
`savespace`	conserve memory during estimation
`diffuse`	use diffuse prior for starting Kalman filter recursions
`p0(`# | *matname*`)`	use alternate prior for starting Kalman recursions; seldom used
`state0(`# | *matname*`)`	use alternate state vector for starting Kalman filter recursions
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `opg`, `robust`, or `oim`

Reporting	
`level(#)`	set confidence level; default is `level(95)`
`detail`	report list of gaps in time series
`nocnsreport`	do not display constraints
display_options	control spacing
Maximization	
maximize_options	control the maximization process; seldom used
†`coeflegend`	display coefficients' legend instead of coefficient table

† `coeflegend` does not appear in the dialog box.

You must `tsset` your data before using `arima`; see [TS] **tsset**.

depvar and *indepvars* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

`by`, `rolling`, `statsby`, and `xi` are allowed; see [U] **11.1.10 Prefix commands**.

`iweight`s are allowed; see [U] **11.1.6 weight**.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Menu

Statistics > Time series > ARIMA and ARMAX models

Description

`arima` fits univariate models with time-dependent disturbances. `arima` fits a model of *depvar* on *indepvars* where the disturbances are allowed to follow a linear autoregressive moving-average (ARMA) specification. The dependent and independent variables may be differenced or seasonally differenced to any degree. When independent variables are included in the specification, such models are often called ARMAX models; and when independent variables are not specified, they reduce to Box–Jenkins autoregressive integrated moving-average (ARIMA) models in the dependent variable. Multiplicative seasonal ARMAX and ARIMA models can also be fit. Missing data are allowed and are handled using the Kalman filter and methods suggested by Harvey (1989 and 1993); see *Methods and formulas*.

In the full syntax, *depvar* is the variable being modeled, and the structural or regression part of the model is specified in *indepvars*. `ar()` and `ma()` specify the lags of autoregressive and moving-average terms, respectively; and `mar()` and `mma()` specify the multiplicative seasonal autoregressive and moving-average terms, respectively.

`arima` allows time-series operators in the dependent variable and independent variable lists, and making extensive use of these operators is often convenient; see [U] **11.4.4 Time-series varlists** and [U] **13.9 Time-series operators** for an extended discussion of time-series operators.

`arima` typed without arguments redisplays the previous estimates.

Options

Model

`noconstant`; see [R] **estimation options**.

`arima(`$\#_p$`,`$\#_d$`,`$\#_q$`)` is an alternative, shorthand notation for specifying models with ARMA disturbances. The dependent variable and any independent variables are differenced $\#_d$ times, and 1 through $\#_p$ lags of autocorrelations and 1 through $\#_q$ lags of moving averages are included in the model. For example, the specification

```
. arima D.y, ar(1/2) ma(1/3)
```

is equivalent to

```
. arima y, arima(2,1,3)
```

The latter is easier to write for simple ARMAX and ARIMA models, but if gaps in the AR or MA lags are to be modeled, or if different operators are to be applied to independent variables, the first syntax is required.

`ar(`*numlist*`)` specifies the autoregressive terms of the structural model disturbance to be included in the model. For example, `ar(1/3)` specifies that lags of 1, 2, and 3 of the structural disturbance be included in the model; `ar(1 4)` specifies that lags 1 and 4 be included, perhaps to account for additive quarterly effects.

If the model does not contain regressors, these terms can also be considered autoregressive terms for the dependent variable.

`ma(`*numlist*`)` specifies the moving-average terms to be included in the model. These are the terms for the lagged innovations (white-noise disturbances).

`constraints(`*constraints*`)`, `collinear`; see [R] **estimation options**.

If constraints are placed between structural model parameters and ARMA terms, the first few iterations may attempt steps into nonstationary areas. This process can be ignored if the final solution is well within the bounds of stationary solutions.

Model 2

`sarima(`$\#_P$`,`$\#_D$`,`$\#_Q$`,`$\#_s$`)` is an alternative, shorthand notation for specifying the multiplicative seasonal components of models with ARMA disturbances. The dependent variable and any independent variables are lag-$\#_s$ seasonally differenced $\#_D$ times, and 1 through $\#_P$ seasonal lags of autoregressive terms and 1 through $\#_Q$ seasonal lags of moving-average terms are included in the model. For example, the specification

```
. arima DS12.y, ar(1/2) ma(1/3) mar(1/2,12) mma(1/2,12)
```

is equivalent to

```
. arima y, arima(2,1,3) sarima(2,1,2,12)
```

`mar(`*numlist*`,` $\#_s$`)` specifies the lag-$\#_s$ multiplicative seasonal autoregressive terms. For example, `mar(1/2,12)` requests that the first two lag-12 multiplicative seasonal autoregressive terms be included in the model.

`mma(`*numlist*`,` $\#_s$`)` specified the lag-$\#_s$ multiplicative seasonal moving-average terms. For example, `mma(1 3,12)` requests that the first and third (but not the second) lag-12 multiplicative seasonal moving-average terms be included in the model.

Model 3

`condition` specifies that conditional, rather than full, maximum likelihood estimates be produced. The presample values for ϵ_t and μ_t are taken to be their expected value of zero, and the estimate of the variance of ϵ_t is taken to be constant over the entire sample; see Hamilton (1994, 132). This estimation method is not appropriate for nonstationary series but may be preferable for long series or for models that have one or more long AR or MA lags. `diffuse`, `p0()`, and `state0()` have no meaning for models fit from the conditional likelihood and may not be specified with `condition`.

If the series is long and stationary and the underlying data-generating process does not have a long memory, estimates will be similar, whether estimated by unconditional maximum likelihood (the default), conditional maximum likelihood (`condition`), or maximum likelihood from a diffuse prior (`diffuse`).

In small samples, however, results of conditional and unconditional maximum likelihood may differ substantially; see Ansley and Newbold (1980). Whereas the default unconditional maximum likelihood estimates make the most use of sample information when all the assumptions of the model are met, Harvey (1989) and Ansley and Kohn (1985) argue for diffuse priors often, particularly in ARIMA models corresponding to an underlying structural model.

The `condition` or `diffuse` options may also be preferred when the model contains one or more long AR or MA lags; this avoids inverting potentially large matrices (see `diffuse` below).

When `condition` is specified, estimation is performed by the `arch` command (see [TS] **arch**), and more control of the estimation process can be obtained using `arch` directly.

`condition` cannot be specified if the model contains any multiplicative seasonal terms.

`savespace` specifies that memory use be conserved by retaining only those variables required for estimation. The original dataset is restored after estimation. This option is rarely used and should be used only if there is not enough space to fit a model without the option. However, `arima` requires considerably more temporary storage during estimation than most estimation commands in Stata.

`diffuse` specifies that a diffuse prior (see Harvey 1989 or 1993) be used as a starting point for the Kalman filter recursions. Using `diffuse`, nonstationary models may be fit with `arima` (see the `p0()` option below; `diffuse` is equivalent to specifying `p0(1e9)`).

By default, `arima` uses the unconditional expected value of the state vector $\boldsymbol{\xi}_t$ (see *Methods and formulas*) and the mean squared error (MSE) of the state vector to initialize the filter. When the process is stationary, this corresponds to the expected value and expected variance of a random draw from the state vector and produces unconditional maximum likelihood estimates of the parameters. When the process is not stationary, however, this default is not appropriate, and the unconditional MSE cannot be computed. For a nonstationary process, another starting point must be used for the recursions.

In the absence of nonsample or presample information, `diffuse` may be specified to start the recursions from a state vector of zero and a state MSE matrix corresponding to an effectively infinite variance on this initial state. This method amounts to an uninformative and improper prior that is updated to a proper MSE as data from the sample become available; see Harvey (1989).

Nonstationary models may also correspond to models with infinite variance given a particular specification. This and other problems with nonstationary series make convergence difficult and sometimes impossible.

`diffuse` can also be useful if a model contains one or more long AR or MA lags. Computation of the unconditional MSE of the state vector (see *Methods and formulas*) requires construction and inversion of a square matrix that is of dimension $\{\max(p, q+1)\}^2$, where p and q are the maximum AR and MA lags, respectively. If $q = 27$, for example, we would require a 784-by-784 matrix. Estimation with `diffuse` does not require this matrix.

For large samples, there is little difference between using the default starting point and the `diffuse` starting point. Unless the series has a long memory, the initial conditions affect the likelihood of only the first few observations.

`p0(#|matname)` is a rarely specified option that can be used for nonstationary series or when an alternate prior for starting the Kalman recursions is desired (see `diffuse` above for a discussion of the default starting point and *Methods and formulas* for background).

matname specifies a matrix to be used as the MSE of the state vector for starting the Kalman filter recursions—$\mathbf{P}_{1|0}$. Instead, one number, *#*, may be supplied, and the MSE of the initial state vector $\mathbf{P}_{1|0}$ will have this number on its diagonal and all off-diagonal values set to zero.

This option may be used with nonstationary series to specify a larger or smaller diagonal for $\mathbf{P}_{1|0}$ than that supplied by `diffuse`. It may also be used with `state0()` when you believe that you have a better prior for the initial state vector and its MSE.

`state0(#|matname)` is a rarely used option that specifies an alternate initial state vector, $\boldsymbol{\xi}_{1|0}$ (see *Methods and formulas*), for starting the Kalman filter recursions. If # is specified, all elements of the vector are taken to be #. The default initial state vector is `state0(0)`.

SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification and that are derived from asymptotic theory; see [R] ***vce_option***.

For state-space models in general and ARMAX and ARIMA models in particular, the robust or quasi–maximum likelihood estimates (QMLEs) of variance are robust to symmetric nonnormality in the disturbances, including, as a special case, heteroskedasticity. The robust variance estimates are not generally robust to functional misspecification of the structural or ARMA components of the model; see Hamilton (1994, 389) for a brief discussion.

Reporting

`level(#)`; see [R] **estimation options**.

`detail` specifies that a detailed list of any gaps in the series be reported, including gaps due to missing observations or missing data for the dependent variable or independent variables.

`nocnsreport`; see [R] **estimation options**.

display_options: `vsquish`; see [R] **estimation options**.

Maximization

maximize_options: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `gtolerance(#)`, `nonrtolerance(#)`, `from(init_specs)`; see [R] **maximize** for all options except `gtolerance()`, and see below for information on `gtolerance()`.

These options are sometimes more important for ARIMA models than most maximum likelihood models because of potential convergence problems with ARIMA models, particularly if the specified model and the sample data imply a nonstationary model.

Several alternate optimization methods, such as Berndt–Hall–Hall–Hausman (BHHH) and Broyden–Fletcher–Goldfarb–Shanno (BFGS), are provided for ARIMA models. Although ARIMA models are not as difficult to optimize as ARCH models, their likelihoods are nevertheless generally not quadratic and often pose optimization difficulties; this is particularly true if a model is nonstationary or nearly nonstationary. Because each method approaches optimization differently, some problems can be successfully optimized by an alternate method when one method fails.

Setting `technique()` to something other than the default or BHHH changes the *vcetype* to `vce(oim)`.

The following options are all related to maximization and are either particularly important in fitting ARIMA models or not available for most other estimators.

`technique(`*algorithm_spec*`)` specifies the optimization technique to use to maximize the likelihood function.

`technique(bhhh)` specifies the Berndt–Hall–Hall–Hausman (BHHH) algorithm.

`technique(dfp)` specifies the Davidon–Fletcher–Powell (DFP) algorithm.

`technique(bfgs)` specifies the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

`technique(nr)` specifies Stata's modified Newton–Raphson (NR) algorithm.

You can specify multiple optimization methods. For example,

```
technique(bhhh 10 nr 20)
```

requests that the optimizer perform 10 BHHH iterations, switch to Newton–Raphson for 20 iterations, switch back to BHHH for 10 more iterations, and so on.

The default for `arima` is `technique(bhhh 5 bfgs 10)`.

`gtolerance(#)` specifies the tolerance for the gradient relative to the coefficients. When $|g_i\, b_i| \leq$ `gtolerance()` for all parameters b_i and the corresponding elements of the gradient g_i, the gradient tolerance criterion is met. The default gradient tolerance for `arima` is `gtolerance(.05)`.

`gtolerance(999)` may be specified to disable the gradient criterion. If the optimizer becomes stuck with repeated "(backed up)" messages, the gradient probably still contains substantial values, but an uphill direction cannot be found for the likelihood. With this option, results can often be obtained, but whether the global maximum likelihood has been found is unclear.

When the maximization is not going well, it is also possible to set the maximum number of iterations (see [R] **maximize**) to the point where the optimizer appears to be stuck and to inspect the estimation results at that point.

`from(`*init_specs*`)` allows you to set the starting values of the model coefficients; see [R] **maximize** for a general discussion and syntax options.

The standard syntax for `from()` accepts a matrix, a list of values, or coefficient name value pairs; see [R] **maximize**. `arima` also accepts `from(armab0)`, which sets the starting value for all ARMA parameters in the model to zero prior to optimization.

ARIMA models may be sensitive to initial conditions and may have coefficient values that correspond to local maximums. The default starting values for `arima` are generally good, particularly in large samples for stationary series.

The following option is available with `arima` but is not shown in the dialog box:

`coeflegend`; see [R] **estimation options**.

Remarks

Remarks are presented under the following headings:

Introduction
ARIMA models
Multiplicative seasonal ARIMA models
ARMAX models
Dynamic forecasting

Introduction

`arima` fits both standard ARIMA models that are autoregressive in the dependent variable and structural models with ARMA disturbances. Good introductions to the former models can be found in Box, Jenkins, and Reinsel (1994); Hamilton (1994); Harvey (1993); Newton (1988); Diggle (1990); and many others. The latter models are developed fully in Hamilton (1994) and Harvey (1989), both of which provide extensive treatment of the Kalman filter (Kalman 1960) and the state-space form used by `arima` to fit the models.

Consider a first-order autoregressive moving-average process. Then `arima` estimates all the parameters in the model

$$
\begin{aligned}
y_t &= \mathbf{x}_t\boldsymbol{\beta} + \mu_t && \textit{structural equation} \\
\mu_t &= \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t && \textit{disturbance, } \mathrm{ARMA}(1,1)
\end{aligned}
$$

where

ρ	is the first-order autocorrelation parameter
θ	is the first-order moving-average parameter
ϵ_t	$\sim$ *i.i.d.* $N(0,\sigma^2)$, meaning that ϵ_t is a white-noise disturbance

You can combine the two equations and write a general $\mathrm{ARMA}(p,q)$ in the disturbances process as

$$
\begin{aligned}
y_t = \mathbf{x_t}\boldsymbol{\beta} &+ \rho_1(y_{t-1} - \mathbf{x}_{t-1}\boldsymbol{\beta}) + \rho_2(y_{t-2} - \mathbf{x}_{t-2}\boldsymbol{\beta}) + \cdots + \rho_p(y_{t-p} - \mathbf{x}_{t-p}\boldsymbol{\beta}) \\
&+ \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} + \epsilon_t
\end{aligned}
$$

It is also common to write the general form of the ARMA model more succinctly using lag operator notation as

$$
\boldsymbol{\rho}(L^p)(y_t - \mathbf{x}_t\boldsymbol{\beta}) = \boldsymbol{\theta}(L^q)\epsilon_t \qquad \mathrm{ARMA}(p,q)
$$

where

$$
\begin{aligned}
\boldsymbol{\rho}(L^p) &= 1 - \rho_1 L - \rho_2 L^2 - \cdots - \rho_p L^p \\
\boldsymbol{\theta}(L^q) &= 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q
\end{aligned}
$$

and $L^j y_t = y_{t-j}$.

For stationary series, full or unconditional maximum likelihood estimates are obtained via the Kalman filter. For nonstationary series, if some prior information is available, you can specify initial values for the filter by using `state0()` and `p0()` as suggested by Hamilton (1994) or assume an uninformative prior by using the `diffuse` option as suggested by Harvey (1989).

ARIMA models

Pure ARIMA models without a structural component do not have regressors and are often written as autoregressions in the dependent variable, rather than autoregressions in the disturbances from a structural equation. For example, an $\mathrm{ARMA}(1,1)$ model can be written as

$$
y_t = \alpha + \rho y_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \qquad (1a)
$$

Other than a scale factor for the constant term α, these models are equivalent to the ARMA in the disturbances formulation estimated by `arima`, though the latter are more flexible and allow a wider class of models.

To see this effect, replace $\mathbf{x}_t\boldsymbol{\beta}$ in the structural equation above with a constant term β_0 so that

$$\begin{aligned} y_t &= \beta_0 + \mu_t \\ &= \beta_0 + \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \\ &= \beta_0 + \rho(y_{t-1} - \beta_0) + \theta\epsilon_{t-1} + \epsilon_t \\ &= (1-\rho)\beta_0 + \rho y_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \end{aligned} \tag{1b}$$

Equations (1a) and (1b) are equivalent, with $\alpha = (1-\rho)\beta_0$, so whether we consider an ARIMA model as autoregressive in the dependent variable or disturbances is immaterial. Our illustration can easily be extended from the ARMA$(1,1)$ case to the general ARIMA(p,d,q) case.

▷ Example 1: ARIMA model

Enders (2004, 87–93) considers an ARIMA model of the U.S. Wholesale Price Index (WPI) using quarterly data over the period 1960q1 through 1990q4. The simplest ARIMA model that includes differencing and both autoregressive and moving-average components is the ARIMA(1,1,1) specification. We can fit this model with `arima` by typing

```
. use http://www.stata-press.com/data/r11/wpi1
. arima wpi, arima(1,1,1)
(setting optimization to BHHH)
Iteration 0:   log likelihood = -139.80133
Iteration 1:   log likelihood =  -135.6278
Iteration 2:   log likelihood = -135.41838
Iteration 3:   log likelihood = -135.36691
Iteration 4:   log likelihood = -135.35892
(switching optimization to BFGS)
Iteration 5:   log likelihood = -135.35471
Iteration 6:   log likelihood = -135.35135
Iteration 7:   log likelihood = -135.35132
Iteration 8:   log likelihood = -135.35131
ARIMA regression
Sample:  1960q2 - 1990q4                        Number of obs      =       123
                                                Wald chi2(2)       =    310.64
Log likelihood = -135.3513                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
       D.wpi |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
wpi          |
       _cons |   .7498197   .3340968     2.24   0.025     .0950019    1.404637
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .8742288   .0545435    16.03   0.000     .7673256     .981132
          ma |
         L1. |  -.4120458   .1000284    -4.12   0.000    -.6080979   -.2159938
-------------+----------------------------------------------------------------
      /sigma |   .7250436   .0368065    19.70   0.000     .6529042    .7971829
------------------------------------------------------------------------------
```

Examining the estimation results, we see that the AR(1) coefficient is 0.874, the MA(1) coefficient is -0.412, and both are highly significant. The estimated standard deviation of the white-noise disturbance ϵ is 0.725.

This model also could have been fit by typing

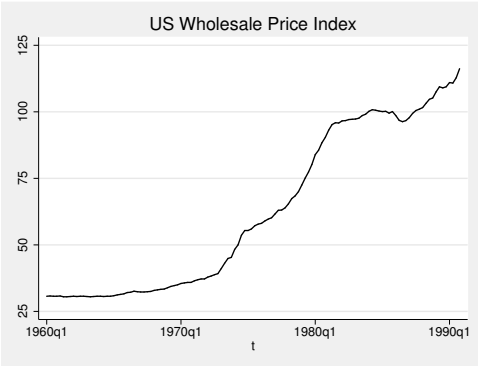
```
. arima D.wpi, ar(1) ma(1)
```

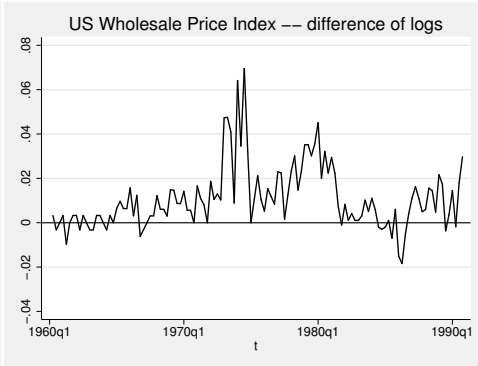
The `D.` placed in front of the dependent variable `wpi` is the Stata time-series operator for differencing. Thus we would be modeling the first difference in WPI from the second quarter of 1960 through the fourth quarter of 1990 because the first observation is lost because of differencing. This second syntax allows a richer choice of models.

◁

▷ Example 2: ARIMA model with additive seasonal effects

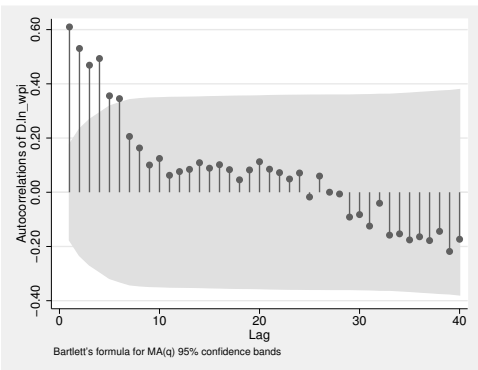
After examining first-differences of WPI, Enders chose a model of differences in the natural logarithms to stabilize the variance in the differenced series. The raw data and first-difference of the logarithms are graphed below.

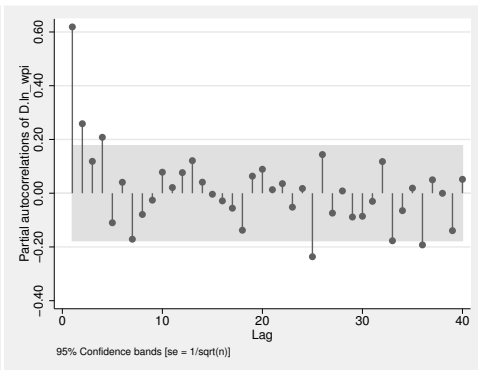




On the basis of the autocorrelations, partial autocorrelations (see graphs below), and the results of preliminary estimations, Enders identified an ARMA model in the log-differenced series.

```
. ac D.ln_wpi, ylabels(-.4(.2).6)
. pac D.ln_wpi, ylabels(-.4(.2).6)
```





In addition to an autoregressive term and an MA(1) term, an MA(4) term is included to account for a remaining quarterly effect. Thus the model to be fit is

$$\Delta \ln(\text{wpi}_t) = \beta_0 + \rho_1\{\Delta \ln(\text{wpi}_{t-1}) - \beta_0\} + \theta_1\epsilon_{t-1} + \theta_4\epsilon_{t-4} + \epsilon_t$$

We can fit this model with `arima` and Stata's standard difference operator:

```
. arima D.ln_wpi, ar(1) ma(1 4)
(setting optimization to BHHH)
Iteration 0:   log likelihood =  382.67447
Iteration 1:   log likelihood =  384.80754
Iteration 2:   log likelihood =  384.84749
Iteration 3:   log likelihood =  385.39213
Iteration 4:   log likelihood =  385.40983
(switching optimization to BFGS)
Iteration 5:   log likelihood =   385.9021
Iteration 6:   log likelihood =  385.95646
Iteration 7:   log likelihood =  386.02979
Iteration 8:   log likelihood =  386.03326
Iteration 9:   log likelihood =  386.03354
Iteration 10:  log likelihood =  386.03357

ARIMA regression
Sample:  1960q2 - 1990q4                        Number of obs      =       123
                                                Wald chi2(3)       =    333.60
Log likelihood =  386.0336                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
    D.ln_wpi |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
ln_wpi       |
       _cons |   .0110493   .0048349     2.29   0.022     .0015731    .0205255
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .7806991   .0944946     8.26   0.000     .5954931     .965905
             |
          ma |
         L1. |  -.3990039   .1258753    -3.17   0.002    -.6457149   -.1522928
         L4. |   .3090813   .1200945     2.57   0.010     .0737003    .5444622
-------------+----------------------------------------------------------------
      /sigma |   .0104394   .0004702    22.20   0.000     .0095178    .0113609
------------------------------------------------------------------------------
```

In this final specification, the log-differenced series is still highly autocorrelated at a level of 0.781, though innovations have a negative impact in the ensuing quarter (-0.399) and a positive seasonal impact of 0.309 in the following year.

◁

❑ Technical note

In one way, the results differ from most of Stata's estimation commands: the standard error of the coefficients is reported as `OPG Std. Err`. As noted in *Options*, the default standard errors and covariance matrix for `arima` estimates are derived from the outer product of gradients (OPG). This is one of three asymptotically equivalent methods of estimating the covariance matrix of the coefficients (only two of which are usually tractable to derive). Discussions and derivations of all three estimates can be found in Davidson and MacKinnon (1993), Greene (2008), and Hamilton (1994). Bollerslev, Engle, and Nelson (1994) suggest that the OPG estimates are more numerically stable in time-series regressions when the likelihood and its derivatives depend on recursive computations, which is certainly the case for the Kalman filter. To date, we have found no numerical instabilities in either estimate of the covariance matrix—subject to the stability and convergence of the overall model.

Most of Stata's estimation commands provide covariance estimates derived from the Hessian of the likelihood function. These alternate estimates can also be obtained from `arima` by specifying the `vce(oim)` option.

❑

Multiplicative seasonal ARIMA models

Many time series exhibit a periodic seasonal component, and a seasonal ARIMA model, often abbreviated SARIMA, can then be used. For example, monthly sales data for air conditioners have a strong seasonal component, with sales high in the summer months and low in the winter months.

In the previous example, we accounted for quarterly effects by fitting the model

$$(1-\rho_1 L)\{\Delta\ln(\text{wpi}_t)-\beta_0\} = (1+\theta_1 L+\theta_4 L^4)\epsilon_t$$

This is an *additive* seasonal ARIMA model, in the sense that the first- and fourth-order MA terms work additively: $(1+\theta_1 L+\theta_4 L^4)$.

Another way to handle the quarterly effect would be to fit a multiplicative seasonal ARIMA model. A multiplicative SARIMA model of order $(1,1,1)\times(0,0,1)_4$ for the $\ln(\text{wpi}_t)$ series is

$$(1-\rho_1 L)\{\Delta\ln(\text{wpi}_t)-\beta_0\} = (1+\theta_1 L)(1+\theta_{4,1}L^4)\epsilon_t$$

or, upon expanding terms,

$$\Delta\ln(\text{wpi}_t) = \beta_0+\rho_1\{\Delta\ln(\text{wpi}_t)-\beta_0\}+\theta_1\epsilon_{t-1}+\theta_{4,1}\epsilon_{t-4}+\theta_1\theta_{4,1}\epsilon_{t-5}+\epsilon_t \tag{2}$$

In the notation $(1,1,1)\times(0,0,1)_4$, the $(1,1,1)$ means that there is one nonseasonal autoregressive term $(1-\rho_1 L)$ and one nonseasonal moving-average term $(1+\theta_1 L)$ and that the time series is first-differenced one time. The $(0,0,1)_4$ indicates that there is no lag-4 seasonal autoregressive term, that there is one lag-4 seasonal moving-average term $(1+\theta_{4,1}L^4)$, and that the series is seasonally differenced zero times. This is known as a *multiplicative* SARIMA model because the nonseasonal and seasonal factors work multiplicatively: $(1+\theta_1 L)(1+\theta_{4,1}L^4)$. Multiplying the terms imposes nonlinear constraints on the parameters of the fifth-order lagged values; `arima` imposes these constraints automatically.

To further clarify the notation, consider a $(2,1,1)\times(1,1,2)_4$ multiplicative SARIMA model:

$$(1-\rho_1 L-\rho_2 L^2)(1-\rho_{4,1}L^4)\Delta\Delta_4 z_t = (1+\theta_1 L)(1+\theta_{4,1}L^4+\theta_{4,2}L^8)\epsilon_t \tag{3}$$

where Δ denotes the difference operator $\Delta y_t = y_t - y_{t-1}$ and Δ_s denotes the lag-s seasonal difference operator $\Delta_s y_t = y_t - y_{t-s}$. Expanding (3), we have

$$\begin{aligned}\widetilde{z}_t = {}&\rho_1\widetilde{z}_{t-1}+\rho_2\widetilde{z}_{t-2}+\rho_{4,1}\widetilde{z}_{t-4}-\rho_1\rho_{4,1}\widetilde{z}_{t-5}-\rho_2\rho_{4,1}\widetilde{z}_{t-6}\\ &+\theta_1\epsilon_{t-1}+\theta_{4,1}\epsilon_{t-4}+\theta_1\theta_{4,1}\epsilon_{t-5}+\theta_{4,2}\epsilon_{t-8}+\theta_1\theta_{4,2}\epsilon_{t-9}\end{aligned}$$

where

$$\widetilde{z}_t = \Delta\Delta_4 z_t = \Delta(z_t - z_{t-4}) = z_t - z_{t-1} - (z_{t-4} - z_{t-5})$$

and $z_t = y_t - \mathbf{x}_t\boldsymbol{\beta}$ if regressors are included in the model, $z_t = y_t - \beta_0$ if just a constant term is included, and $z_t = y_t$ otherwise.

More generally, a $(p, d, q) \times (P, D, Q)_s$ multiplicative SARIMA model is

$$\boldsymbol{\rho}(L^p)\boldsymbol{\rho}_s(L^P)\Delta^d\Delta_s^D z_t = \boldsymbol{\theta}(L^q)\boldsymbol{\theta}_s(L^Q)$$

where

$$\boldsymbol{\rho}_s(L^P) = (1 - \rho_{s,1}L^s - \rho_{s,2}L^{2s} - \cdots - \rho_{s,P}L^{Ps})$$
$$\boldsymbol{\theta}_s(L^Q) = (1 + \theta_{s,1}L^s + \theta_{s,2}L^{2s} + \cdots + \theta_{s,Q}L^{Qs})$$

$\boldsymbol{\rho}(L^p)$ and $\boldsymbol{\theta}(L^q)$ were defined previously, Δ^d means apply the Δ operator d times, and similarly for Δ_s^D. Typically, d and D will be 0 or 1; and p, q, P, and Q will seldom be more than 2 or 3. s will typically be 4 for quarterly data and 12 for monthly data. In fact, the model can be extended to include both monthly and quarterly seasonal factors, as we explain below.

If a plot of the data suggests that the seasonal effect is proportional to the mean of the series, then the seasonal effect is probably multiplicative and a multiplicative SARIMA model may be appropriate. Box, Jenkins, and Reinsel (1994, sec. 9.3.1) suggest starting with a multiplicative SARIMA model with any data that exhibit seasonal patterns and then exploring nonmultiplicative SARIMA models if the multiplicative models do not fit the data well. On the other hand, Chatfield (2004, 14) suggests that taking the logarithm of the series will make the seasonal effect additive, in which case an additive SARIMA model as fit in the previous example would be appropriate. In short, the analyst should probably try both additive and multiplicative SARIMA models to see which provides better fits and forecasts.

Unless the `diffuse` option is used, `arima` must create square matrices of dimension $\{\max(p, q+1)\}^2$, where p and q are the maximum AR and MA lags, respectively; and the inclusion of long seasonal terms can make this dimension rather large. For example, with monthly data, you might fit a $(0, 1, 1) \times (0, 1, 2)_{12}$ SARIMA model. The maximum MA lag is $2 \times 12 + 1 = 25$, requiring a matrix with $26^2 = 676$ rows and columns.

▷ Example 3: Multiplicative SARIMA model

One of the most common multiplicative SARIMA specifications is the $(0, 1, 1) \times (0, 1, 1)_{12}$ "airline" model of Box, Jenkins, and Reinsel (1994, sec. 9.2). The dataset `airline.dta` contains monthly international airline passenger data from January 1949 through December 1960. After first- and seasonally differencing the data, we do not suspect the presence of a trend component, so we use the `noconstant` option with `arima`:

(*Continued on next page*)

```
. use http://www.stata-press.com/data/r11/air2
(TIMESLAB: Airline passengers)
. generate lnair = ln(air)
. arima lnair, arima(0,1,1) sarima(0,1,1,12) noconstant
(setting optimization to BHHH)
Iteration 0:   log likelihood =   223.8437
Iteration 1:   log likelihood =  239.80405
  (output omitted)
Iteration 8:   log likelihood =  244.69651
ARIMA regression
Sample:  14 - 144                               Number of obs      =       131
                                                Wald chi2(2)       =     84.53
Log likelihood =  244.6965                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
  DS12.lnair |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
ARMA         |
          ma |
         L1. |  -.4018324   .0730307    -5.50   0.000    -.5449698   -.2586949
-------------+----------------------------------------------------------------
ARMA12       |
          ma |
         L1. |  -.5569342   .0963129    -5.78   0.000     -.745704   -.3681644
-------------+----------------------------------------------------------------
      /sigma |   .0367167   .0020132    18.24   0.000     .0327708    .0406625
------------------------------------------------------------------------------
```

Thus our model of the monthly number of international airline passengers is

$$\Delta\Delta_{12}\texttt{lnair}_t = -0.402\epsilon_{t-1} - 0.557\epsilon_{t-12} + 0.224\epsilon_{t-13} + \epsilon_t$$
$$\widehat{\sigma} = 0.037$$

In (2), for example, the coefficient on ϵ_{t-13} is the product of the coefficients on the ϵ_{t-1} and ϵ_{t-12} terms $(0.224 \approx -0.402 \times -0.557)$. `arima` labeled the dependent variable `DS12.lnair` to indicate that it has applied the difference operator Δ and the lag-12 seasonal difference operator Δ_{12} to `lnair`; see [U] **11.4.4 Time-series varlists** for more information.

We could have fit this model by typing

```
. arima DS12.lnair, ma(1) mma(1, 12) noconstant
```

For simple multiplicative models, using the `sarima()` option is easier, though this second syntax allows us to incorporate more complicated seasonal terms.

◁

The `mar()` and `mma()` options can be repeated, allowing us to control for multiple seasonal patterns. For example, we may have monthly sales data that exhibit a quarterly pattern as businesses purchase our product at the beginning of calendar quarters when new funds are budgeted, and our product is purchased more frequently in a few months of the year than in most others, even after we control for quarterly fluctuations. Thus we might choose to fit the model

$$(1-\rho L)(1-\rho_{4,1}L^4)(1-\rho_{12,1}L^{12})(\Delta\Delta_4\Delta_{12}\texttt{sales}_t-\beta_0) = (1+\theta L)(1+\theta_{4,1}L^4)(1+\theta_{12,1}L^{12})\epsilon_t$$

Although this model looks rather complicated, estimating it using `arima` is straightforward:

```
. arima DS4S12.sales, ar(1) mar(1, 4) mar(1, 12) ma(1) mma(1, 4) mma(1, 12)
```

If we instead wanted to include two lags in the lag-4 seasonal AR term and the first and third (but not the second) term in the lag-12 seasonal MA term, we would type

```
. arima DS4S12.sales, ar(1) mar(1 2, 4) mar(1, 12) ma(1) mma(1, 4) mma(1 3, 12)
```

However, models with multiple seasonal terms can be difficult to fit. Usually, one seasonal factor with just one or two AR or MA terms is adequate.

ARMAX models

Thus far all our examples have been pure ARIMA models in which the dependent variable was modeled solely as a function of its past values and disturbances. Also, `arima` can fit ARMAX models, which model the dependent variable in terms of a linear combination of independent variables, as well as an ARMA disturbance process. The `prais` command, for example, allows you to control for only AR(1) disturbances, whereas `arima` allows you to control for a much richer dynamic error structure. `arima` allows for both nonseasonal and seasonal ARMA components in the disturbances.

▷ Example 4: ARMAX model

For a simple example of a model including covariates, we can estimate an update of Friedman and Meiselman's (1963) equation representing the quantity theory of money. They postulate a straightforward relationship between personal-consumption expenditures (`consump`) and the money supply as measured by M2 (`m2`).

$$\texttt{consump}_t = \beta_0 + \beta_1 \texttt{m2}_t + \mu_t$$

Friedman and Meiselman fit the model over a period ending in 1956; we will refit the model over the period 1959q1 through 1981q4. We restrict our attention to the period prior to 1982 because the Federal Reserve manipulated the money supply extensively in the later 1980s to control inflation, and the relationship between consumption and the money supply becomes much more complex during the later part of the decade.

To demonstrate `arima`, we will include both an autoregressive term and a moving-average term for the disturbances in the model; the original estimates included neither. Thus we model the disturbance of the structural equation as

$$\mu_t = \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t$$

As per the original authors, the relationship is estimated on seasonally adjusted data, so there is no need to include seasonal effects explicitly. Obtaining seasonally unadjusted data and simultaneously modeling the structural and seasonal effects might be preferable.

We will restrict the estimation to the desired sample by using the `tin()` function in an `if` expression; see [D] **functions**. By leaving the first argument of `tin()` blank, we are including all available data through the second date (`1981q4`). We fit the model by typing

```
. use http://www.stata-press.com/data/r11/friedman2, clear
. arima consump m2 if tin(, 1981q4), ar(1) ma(1)
(setting optimization to BHHH)
Iteration 0:   log likelihood = -344.67575
Iteration 1:   log likelihood = -341.57248
  (output omitted)
Iteration 10:  log likelihood = -340.50774
ARIMA regression
Sample:  1959q1 - 1981q4                        Number of obs      =        92
                                                Wald chi2(3)       =   4394.80
Log likelihood = -340.5077                      Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |                 OPG
     consump |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
consump      |
          m2 |   1.122029   .0363563    30.86   0.000     1.050772    1.193286
       _cons |  -36.09872   56.56703    -0.64   0.523    -146.9681    74.77062
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .9348486   .0411323    22.73   0.000     .8542308    1.015467
             |
          ma |
         L1. |   .3090592   .0885883     3.49   0.000     .1354293    .4826891
-------------+----------------------------------------------------------------
      /sigma |   9.655308   .5635157    17.13   0.000     8.550837    10.75978
------------------------------------------------------------------------------
```

We find a relatively small money velocity with respect to consumption (1.122) over this period, although consumption is only one facet of the income velocity. We also note a very large first-order autocorrelation in the disturbances, as well as a statistically significant first-order moving average.

We might be concerned that our specification has led to disturbances that are heteroskedastic or non-Gaussian. We refit the model by using the `vce(robust)` option.

```
. arima consump m2 if tin(, 1981q4), ar(1) ma(1) vce(robust)
(setting optimization to BHHH)
Iteration 0:   log pseudolikelihood = -344.67575
Iteration 1:   log pseudolikelihood = -341.57248
  (output omitted)
Iteration 10:  log pseudolikelihood = -340.50774
ARIMA regression
Sample:  1959q1 - 1981q4                        Number of obs      =        92
                                                Wald chi2(3)       =   1176.26
Log pseudolikelihood = -340.5077                Prob > chi2        =    0.0000

------------------------------------------------------------------------------
             |               Semirobust
     consump |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
consump      |
          m2 |   1.122029   .0433303    25.89   0.000     1.037103    1.206954
       _cons |  -36.09872   28.10479    -1.28   0.199     -91.1831    18.98566
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   .9348486   .0493428    18.95   0.000     .8381385    1.031559
             |
          ma |
         L1. |   .3090592   .1605359     1.93   0.054    -.0055854    .6237038
-------------+----------------------------------------------------------------
      /sigma |   9.655308   1.082639     8.92   0.000     7.533375    11.77724
------------------------------------------------------------------------------
```

We note a substantial increase in the estimated standard errors, and our once clearly significant moving-average term is now only marginally significant.

◁

Dynamic forecasting

Another feature of the `arima` command is the ability to use `predict` afterward to make dynamic forecasts. Suppose that we wish to fit the regression model

$$y_t = \beta_0 + \beta_1 x_t + \rho y_{t-1} + \epsilon_t$$

by using a sample of data from $t = 1 \ldots T$ and make forecasts beginning at time f.

If we use `regress` or `prais` to fit the model, then we can use `predict` to make one-step-ahead forecasts. That is, `predict` will compute

$$\widehat{y}_f = \widehat{\beta_0} + \widehat{\beta_1} x_f + \widehat{\rho} y_{f-1}$$

Most importantly, here `predict` will use the actual value of y at period $f-1$ in computing the forecast for time f. Thus, if we use `regress` or `prais`, we cannot make forecasts for any periods beyond $f = T + 1$ unless we have observed values for y for those periods.

If we instead fit our model with `arima`, then `predict` can produce dynamic forecasts by using the Kalman filter. If we use the `dynamic(`f`)` option, then for period f `predict` will compute

$$\widehat{y}_f = \widehat{\beta_0} + \widehat{\beta_1} x_f + \widehat{\rho} y_{f-1}$$

by using the observed value of y_{f-1} just as `predict` after `regress` or `prais`. However, for period $f+1$ `predict` *newvar*, `dynamic(`f`)` will compute

$$\widehat{y}_{f+1} = \widehat{\beta_0} + \widehat{\beta_1} x_{f+1} + \widehat{\rho}\widehat{y}_f$$

using the *predicted* value of y_f instead of the observed value. Similarly, the period $f+2$ forecast will be

$$\widehat{y}_{f+2} = \widehat{\beta_0} + \widehat{\beta_1} x_{f+2} + \widehat{\rho}\widehat{y}_{f+1}$$

Of course, because our model includes the regressor x_t, we can make forecasts only through periods for which we have observations on x_t. However, for pure ARIMA models, we can compute dynamic forecasts as far beyond the final period of our dataset as desired.

For more information on `predict` after `arima`, see [TS] **arima postestimation**.

Saved results

`arima` saves the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_gaps)`	number of gaps
`e(k)`	number of parameters
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in model Wald test
`e(k_dv)`	number of dependent variables
`e(k1)`	number of variables in first equation
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(sigma)`	sigma
`e(chi2)`	χ^2 statistic
`e(p)`	significance
`e(tmin)`	minimum time
`e(tmax)`	maximum time
`e(ar_max)`	maximum AR lag
`e(ma_max)`	maximum MA lag
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

e(cmd)	arima
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(eqnames)	names of equations
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(tmins)	formatted minimum time
e(tmaxs)	formatted maximum time
e(chi2type)	Wald; type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. Err.
e(ma)	lags for moving-average terms
e(ar)	lags for autoregressive terms
e(mar*i*)	multiplicative AR terms and lag i=1… (# seasonal AR terms)
e(mma*i*)	multiplicative MA terms and lag i=1… (# seasonal MA terms)
e(seasons)	seasonal lags in model
e(unsta)	unstationary or blank
e(opt)	type of optimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(tech_steps)	number of iterations performed before switching techniques
e(crittype)	optimization criterion
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

Methods and formulas

arima is implemented as an ado-file.

Estimation is by maximum likelihood using the Kalman filter via the prediction error decomposition; see Hamilton (1994), Gourieroux and Monfort (1997), or, in particular, Harvey (1989). Any of these sources will serve as excellent background for the fitting of these models with the state-space form; each source also provides considerable detail on the method outlined below.

Methods and formulas are presented under the following headings:

ARIMA model
Kalman filter equations
Kalman filter or state-space representation of the ARIMA model
Kalman filter recursions
Kalman filter initial conditions
Likelihood from prediction error decomposition
Missing data

ARIMA model

The model to be fit is

$$
\begin{aligned}
y_t &= \mathbf{x}_t\boldsymbol{\beta} + \mu_t \\
\mu_t &= \sum_{i=1}^{p} \rho_i \mu_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t
\end{aligned}
$$

which can be written as the single equation

$$
y_t = \mathbf{x}_t\boldsymbol{\beta} + \sum_{i=1}^{p} \rho_i (y_{t-i} - x_{t-i}\boldsymbol{\beta}) + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t
$$

Some of the ρs and θs may be constrained to zero or, for multiplicative seasonal models, the products of other parameters.

Kalman filter equations

We will roughly follow Hamilton's (1994) notation and write the Kalman filter

$$
\begin{aligned}
\boldsymbol{\xi}_t &= \mathbf{F}\boldsymbol{\xi}_{t-1} + \mathbf{v}_t && (\textit{state equation}) \\
\mathbf{y}_t &= \mathbf{A}'\mathbf{x}_t + \mathbf{H}'\boldsymbol{\xi}_t + \mathbf{w}_t && (\textit{observation equation})
\end{aligned}
$$

and

$$
\begin{pmatrix} \mathbf{v}_t \\ \mathbf{w}_t \end{pmatrix} \sim N\left\{\mathbf{0}, \begin{pmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}\right\}
$$

We maintain the standard Kalman filter matrix and vector notation, although for univariate models $\mathbf{y}_t$, $\mathbf{w}_t$, and $\mathbf{R}$ are scalars.

Kalman filter or state-space representation of the ARIMA model

A univariate ARIMA model can be cast in state-space form by defining the Kalman filter matrices as follows (see Hamilton [1994], or Gourieroux and Monfort [1997], for details):

$$F = \begin{bmatrix} \rho_1 & \rho_2 & \dots & \rho_{p-1} & \rho_p \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

$$\mathbf{v}_t = \begin{bmatrix} \epsilon_{t-1} \\ 0 \\ \dots \\ \dots \\ \dots \\ 0 \end{bmatrix}$$

$$\mathbf{A}' = \boldsymbol{\beta}$$

$$\mathbf{H}' = [\,1 \quad \theta_1 \quad \theta_2 \quad \dots \quad \theta_q\,]$$

$$\mathbf{w}_t = 0$$

The Kalman filter representation does not require the moving-average terms to be invertible.

Kalman filter recursions

To demonstrate how missing data are handled, the updating recursions for the Kalman filter will be left in two steps. Writing the updating equations as one step using the gain matrix $\mathbf{K}$ is common. We will provide the updating equations with little justification; see the sources listed above for details.

As a linear combination of a vector of random variables, the state $\boldsymbol{\xi}_t$ can be updated to its expected value on the basis of the prior state as

$$\boldsymbol{\xi}_{t|t-1} = \mathbf{F}\boldsymbol{\xi}_{t-1} + \mathbf{v}_{t-1} \tag{1}$$

This state is a quadratic form that has the covariance matrix

$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}' + \mathbf{Q} \tag{2}$$

The estimator of $\mathbf{y}_t$ is

$$\widehat{\mathbf{y}}_{t|t-1} = \mathbf{x}_t\boldsymbol{\beta} + \mathbf{H}'\boldsymbol{\xi}_{t|t-1}$$

which implies an innovation or prediction error

$$\widehat{\boldsymbol{\iota}}_t = \mathbf{y}_t - \widehat{\mathbf{y}}_{t|t-1}$$

This value or vector has mean squared error (MSE)

$$\mathbf{M}_t = \mathbf{H}'\mathbf{P}_{t|t-1}\mathbf{H} + \mathbf{R}$$

Now the expected value of $\boldsymbol{\xi}_t$ conditional on a realization of $\mathbf{y}_t$ is

$$\boldsymbol{\xi}_t = \boldsymbol{\xi}_{t|t-1} + \mathbf{P}_{t|t-1}\mathbf{H}\mathbf{M}_t^{-1}\widehat{\boldsymbol{\iota}}_t \tag{3}$$

with MSE

$$\mathbf{P}_t = \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{H}\mathbf{M}_t^{-1}\mathbf{H}'\mathbf{P}_{t|t-1} \tag{4}$$

This expression gives the full set of Kalman filter recursions.

Kalman filter initial conditions

When the series is stationary, conditional on $\mathbf{x}_t\boldsymbol{\beta}$, the initial conditions for the filter can be considered a random draw from the stationary distribution of the state equation. The initial values of the state and the state MSE are the expected values from this stationary distribution. For an ARIMA model, these can be written as

$$\xi_{1|0} = \mathbf{0}$$

and

$$\text{vec}(\mathbf{P}_{1|0}) = (\mathbf{I}_{r^2} - \mathbf{F} \otimes \mathbf{F})^{-1}\text{vec}(\mathbf{Q})$$

where vec() is an operator representing the column matrix resulting from stacking each successive column of the target matrix.

If the series is not stationary, the initial state conditions do not constitute a random draw from a stationary distribution, and some other values must be chosen. Hamilton (1994) suggests that they be chosen based on prior expectations, whereas Harvey suggests a diffuse and improper prior having a state vector of $\mathbf{0}$ and an infinite variance. This method corresponds to $\mathbf{P}_{1|0}$ with diagonal elements of ∞. Stata allows either approach to be taken for nonstationary series—initial priors may be specified with `state0()` and `p0()`, and a diffuse prior may be specified with `diffuse`.

Likelihood from prediction error decomposition

Given the outputs from the Kalman filter recursions and assuming that the state and observation vectors are Gaussian, the likelihood for the state-space model follows directly from the resulting multivariate normal in the predicted innovations. The log likelihood for observation t is

$$\ln L_t = -\frac{1}{2}\left\{ \ln(2\pi) + \ln(|\mathbf{M}_t|) - \widehat{\boldsymbol{\iota}}_t'\mathbf{M}_t^{-1}\widehat{\boldsymbol{\iota}}_t \right\}$$

This command supports the Huber/White/sandwich estimator of the variance using `vce(robust)`. See [P] **_robust**, in particular, in *Maximum likelihood estimators* and *Methods and formulas.*

Missing data

Missing data, whether a missing dependent variable y_t, one or more missing covariates $\mathbf{x}_t$, or completely missing observations, are handled by continuing the state-updating equations without any contribution from the data; see Harvey (1989 and 1993). That is, (1) and (2) are iterated for every missing observation, whereas (3) and (4) are ignored. Thus, for observations with missing data, $\boldsymbol{\xi}_t = \boldsymbol{\xi}_{t|t-1}$ and $\mathbf{P}_t = \mathbf{P}_{t|t-1}$. Without any information from the sample, this effectively assumes that the prediction error for the missing observations is 0. Other methods of handling missing data on the basis of the EM algorithm have been suggested, e.g., Shumway (1984, 1988).

George Edward Pelham Box (1919–) was born in Kent, England, and earned degrees in statistics at the University of London. After work in the chemical industry, he taught and researched at Princeton and the University of Wisconsin. His many major contributions to statistics include papers and books in Bayesian inference, robustness (a term he introduced to statistics), modeling strategy, experimental design and response surfaces, time-series analysis, distribution theory, transformations, and nonlinear estimation.

Gwilym Meirion Jenkins (1933–1982) was a British mathematician and statistician who spent his career in industry and academia, working for extended periods at Imperial College London and the University of Lancaster before running his own company. His interests were centered on time series and he collaborated with G. E. P. Box on what are often called Box–Jenkins models. The last years of Jenkins' life were marked by a slowly losing battle against Hodgkin's disease.

References

Ansley, C. F., and R. Kohn. 1985. Estimation, filtering, and smoothing in state space models with incompletely specified initial conditions. *Annals of Statistics* 13: 1286–1316.

Ansley, C. F., and P. Newbold. 1980. Finite sample properties of estimators for autoregressive moving average models. *Journal of Econometrics* 13: 159–183.

Baum, C. F. 2000. sts15: Tests for stationarity of a time series. *Stata Technical Bulletin* 57: 36–39. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 356–360. College Station, TX: Stata Press.

Baum, C. F., and T. Room. 2001. sts18: A test for long-range dependence in a time series. *Stata Technical Bulletin* 60: 37–39. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 370–373. College Station, TX: Stata Press.

Baum, C. F., and R. Sperling. 2000. sts15.1: Tests for stationarity of a time series: Update. *Stata Technical Bulletin* 58: 35–36. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 360–362. College Station, TX: Stata Press.

Baum, C. F., and V. L. Wiggins. 2000. sts16: Tests for long memory in a time series. *Stata Technical Bulletin* 57: 39–44. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 362–368. College Station, TX: Stata Press.

Berndt, E. K., B. H. Hall, R. E. Hall, and J. A. Hausman. 1974. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement* 3/4: 653–665.

Bollerslev, T., R. F. Engle, and D. B. Nelson. 1994. ARCH models. In *Handbook of Econometrics, Volume IV*, ed. R. F. Engle and D. L. McFadden. New York: Elsevier.

Box, G. E. P. 1983. Obituary: G. M. Jenkins, 1933–1982. *Journal of the Royal Statistical Society, Series A* 146: 205–206.

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel. 1994. *Time Series Analysis: Forecasting and Control*. 3rd ed. Englewood Cliffs, NJ: Prentice–Hall.

Chatfield, C. 2004. *The Analysis of Time Series: An Introduction*. 6th ed. Boca Raton, FL: Chapman & Hall/CRC.

David, J. S. 1999. sts14: Bivariate Granger causality test. *Stata Technical Bulletin* 51: 40–41. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 350–351. College Station, TX: Stata Press.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

DeGroot, M. H. 1987. A conversation with George Box. *Statistical Science* 2: 239–258.

Diggle, P. J. 1990. *Time Series: A Biostatistical Introduction*. Oxford: Oxford University Press.

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

Friedman, M., and D. Meiselman. 1963. The relative stability of monetary velocity and the investment multiplier in the United States, 1897–1958. In *Stabilization Policies*, Commission on Money and Credit, 123–126. Englewood Cliffs, NJ: Prentice–Hall.

Gourieroux, C., and A. Monfort. 1997. *Time Series and Dynamic Models*. Trans. ed. G. M. Gallo. Cambridge: Cambridge University Press.

Greene, W. H. 2008. *Econometric Analysis*. 6th ed. Upper Saddle River, NJ: Prentice–Hall.

Hamilton, J. D. 1994. *Time Series Analysis*. Princeton: Princeton University Press.

Harvey, A. C. 1989. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge: Cambridge University Press.

——. 1993. *Time Series Models*. 2nd ed. Cambridge, MA: MIT Press.

Hipel, K. W., and A. I. McLeod. 1994. *Time Series Modelling of Water Resources and Environmental Systems*. Amsterdam: Elsevier.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering, Series D* 82: 35–45.

McDowell, A. W. 2002. From the help desk: Transfer functions. *Stata Journal* 2: 71–85.

——. 2004. From the help desk: Polynomial distributed lag models. *Stata Journal* 4: 180–189.

Newton, H. J. 1988. *TIMESLAB: A Time Series Analysis Laboratory*. Belmont, CA: Wadsworth.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes in C: The Art of Scientific Computing*. 3rd ed. Cambridge: Cambridge University Press.

Shumway, R. H. 1984. Some applications of the EM algorithm to analyzing incomplete time series data. In *Time Series Analysis of Irregularly Observed Data*, ed. E. Parzen, 290–324. New York: Springer.

——. 1988. *Applied Statistical Time Series Analysis*. Upper Saddle River, NJ: Prentice–Hall.

Also see

[TS] **arima postestimation** — Postestimation tools for arima

[TS] **tsset** — Declare data to be time-series data

[TS] **arch** — Autoregressive conditional heteroskedasticity (ARCH) family of estimators

[TS] **dfactor** — Dynamic-factor models

[TS] **dvech** — Diagonal vech multivariate GARCH models

[TS] **prais** — Prais–Winsten and Cochrane–Orcutt regression

[TS] **sspace** — State-space models

[R] **regress** — Linear regression

[U] **20 Estimation and postestimation commands**