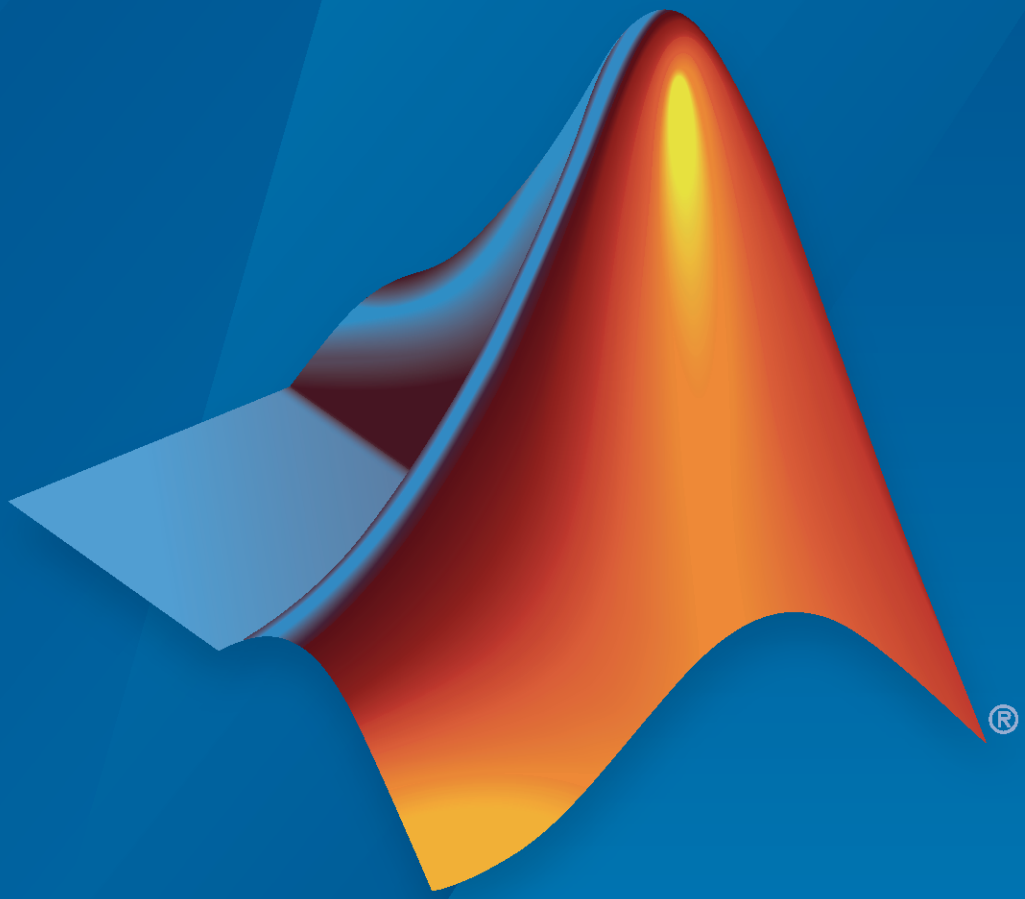
Econometrics Toolbox™ Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Econometrics Toolbox™ Release Notes

Contents

R2022b

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

No New Features or Changes

R2006b

R2006a

R14SP3

R2022b

Dynamic Stochastic Equilibrium Equations Example: Analyze DSGE model by using Bayesian state-space model methods

The “Analyze Linearized DSGE Models” example analyzes the dynamic stochastic general equilibrium (DSGE) model in [69] by using Bayesian state-space model tools available in Econometrics Toolbox.

The example shows how to perform the following tasks:

1. Simulate data from a known state-space model.
2. Estimate posterior moments of the model parameters by combining prior distributions and the simulated data.
3. Characterize the dynamic behavior of the model by using impulse response and forecast error variance analyses.
4. Forecast response variables by using posterior predictive distribution.

Bayesian State-Space Models: Assume non-Gaussian, fat-tailed distributions on the state disturbance and observation innovation processes

The Bayesian state-space model (`bssm` object) framework enables you to assume the Student's *t* distribution for the conditional distribution of the state disturbance or observation innovations process. These error settings are suited when the process or measurement errors have excess kurtosis (or is fat-tailed or leptokuric).

Specify the distribution of either error process by using the following name-value argument when you create a `bssm` object:

- `StateDistribution` — Distribution of the state disturbance process
- `ObservationDistribution` — Distribution of the observation innovation process

Specify the distribution using its name or a scalar structure array.

Distribution	Name	Structure Array
Gaussian	`"Gaussian"`	`struct('Name',"Gaussian")`
Student's *t*	`"t"`	`struct('Name',"t",'DoF',dof)`, where the degrees of freedom parameter `dof` is a positive scalar, `NaN` specifying it has a flat prior, or a function handle to a function that specifies its log prior distribution

When `dof` is `NaN` or a function handle, the degrees of freedom parameter is unknown and estimated with Θ, the parameters of the state-space model. In other words, the `estimate` and `simulate` functions sample from its posterior distribution when they sample Θ by using the Metropolis-Hastings method. You can specify the proposal standard deviation of the degrees of freedom parameter for the sampler by using the `StdDoF` name-value arguments.

Table and Timetable Support: Vector autoregression (VAR) and vector error-correction (VEC) model functions accept input data in tables and timetables, and return results in tables and timetables

In addition to accepting input data (insample and presample data) in numeric arrays, the functions of VAR and VEC models, represented by `varm` and `vecm` objects, listed in the table accept input data in tables or regular timetables. When you supply data in a table or timetable, the following conditions apply:

- The functions choose the default series on which to operate, but you can use the specified name-value argument to select variables.
- The functions return results and selected options in a table or timetable.

Function	Name-Value Arguments for Variable Specification
`estimate` (`varm`) and `estimate` (`vecm`)	• `ResponseVariables` specifies the response series names in the input data to which the model is fit. • `PredictorVariables` specifies the predictor series names in the input data for a model regression component. • `Presample` specifies the input table or timetable of presample response data. • `PresampleResponseVariables` specifies the response series names from `Presample`.
`fevd` (`varm`) and `fevd` (`vecm`)	• `Presample` specifies the input table or regular timetable of presample response data. • `PresampleResponseVariables` specifies the response series names from `Presample`. • `Insample` specifies the table or regular timetable of residual and predictor data to compute bootstrap estimates. • `ResidualVariables` specifies the residual series names in `InSample`. • `PredictorVariables` specifies the predictor series in `InSample` for a model regression component.
`filter` (`varm`) and `filter` (`vecm`)	• `DisturbanceVariables` specifies the disturbance series names in the input data to filter through the model. • `Presample` specifies the input table or regular timetable of presample response data. • `PresampleResponseVariables` specifies the response series names from `Presample`. • `PredictorVariables` specifies the predictor series names in the input data for a model regression component.
`forecast` (`varm`) and `forecast` (`vecm`)	• `PresampleResponseVariables` specifies the response series names in the input presample response data. • `Insample` specifies the table or regular timetable of future response and predictor data to compute conditional forecasts. • `ResponseVariables` specifies the response series names in `InSample`. • `PredictorVariables` specifies the predictor series in `InSample` for a model regression component.

Function	Name-Value Arguments for Variable Specification
`infer` (`varm`) and `infer` (`vecm`)	• `ResponseVariables` specifies the response series names in the input data from which residuals are inferred. • `PredictorVariables` specifies the predictor series names in the input data for a model regression component. • `Presample` specifies the input table or timetable of presample response data. • `PresampleResponseVariables` specifies the response series names from `Presample`.
`irf` (`varm`) and `irf` (`vecm`)	• `Presample` specifies the input table or regular timetable of presample response data. • `PresampleResponseVariables` specifies the response series names from `Presample`. • `Insample` specifies the table or regular timetable of residual and predictor data to compute bootstrap estimates. • `ResidualVariables` specifies the residual series names in `InSample`. • `PredictorVariables` specifies the predictor series in `InSample` for a model regression component.
`simulate` (`varm`) and `simulate` (`vecm`)	• `Presample` specifies the input table or regular timetable of presample response data. • `PresampleResponseVariables` specifies the response series names in `Presample`. • `Insample` specifies the table or regular timetable of future response and predictor data for conditional simulation. • `ResponseVariables` specifies the response series names in `InSample`. • `PredictorVariables` specifies the predictor series in `InSample` for a model regression component.

Data files include economic data in timetables

Most data files provided with Econometrics Toolbox and stored in `mlr/toolbox/econ/econdata` now include sample data sets in timetables (`mlr` is the value of `matlabroot`). For more details, see “Data Sets and Examples”.

Compatibility Considerations

For consistency, the data files `Data_USEconModel.mat`, `Data_ElectricityPrices.mat`, and `Data_DieboldLi.mat` store the following variables containing the same time series data:

- `Data` — A numeric matrix
- `DataTable` — A table
- `DataTimeTable` — A timetable

Before R2022b, the variables in the data files containing time series data were:

- `Data` — A numeric matrix

- `DataTable` — A timetable

Update your code that uses the timetable in the data files by replacing all instances of `DataTable` with `DataTimeTable`.

R2022a

Version: 6.0

New Features

Compatibility Considerations

Econometrics Modeler App: Conduct cointegration tests and fit multivariate time series models

The **Econometric Modeler** app enables you to test multiple time series for cointegration among the variables by performing the Engle-Granger cointegration test or the Johansen cointegration test.

After assessing the cointegrating relationships among the series, you can fit vector autoregression (VAR) models, including optional exogenous variables (VARX) or vector error-correction (VEC) models. As with univariate models, after working in the app, you can export variables to the MATLAB® workspace and generate code or a report summarizing your session. For more details, Econometric Modeler App Overview.

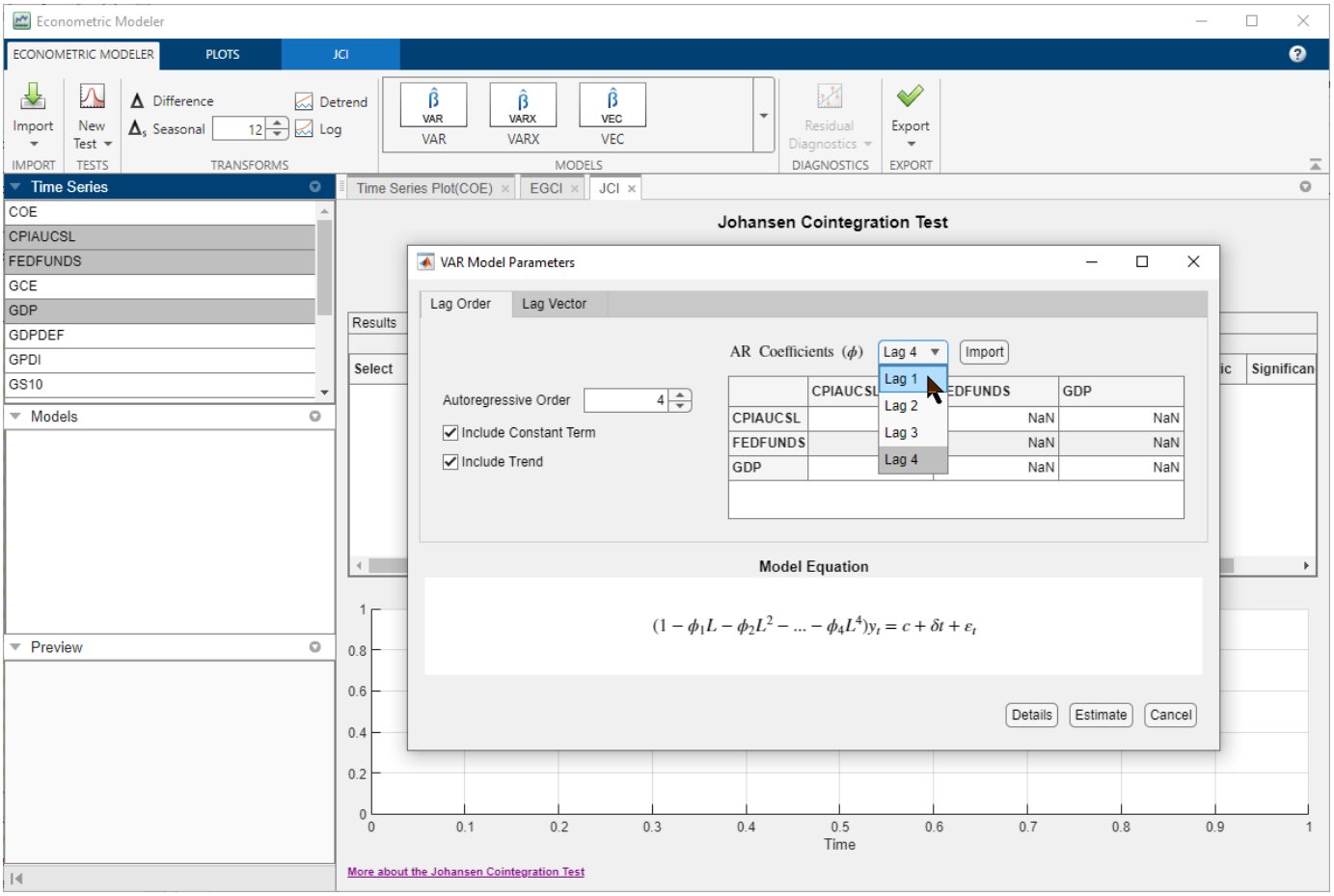


Table and Timetable Support: Functions accept input data in tables and timetables, and return results in tables

In addition to accepting input data in numeric arrays, the Econometrics Toolbox functions listed in the table accept input data in tables and timetables. When you supply data in a table or timetable, the following conditions apply:

- The functions choose the default series on which to operate, but you can use the specified name-value argument to select variables.
- The functions return results and selected options in a table or timetable.

Function	Name-Value Arguments for Variable Specification
`adftest`	`DataVariable` specifies the response series to test.
`archtest`	`DataVariable` specifies the residual series to test.
`autocorr`	`DataVariable` specifies the series for which the autocorrelation is computed.
`chowtest`	`ResponseVariable` specifies the series to use for the regression response and `PredictorVariables` specifies the series to use for the regression predictors.
`collintest`	`DataVariables` specifies the series among which collinearity is assessed.
`corrplot`	`DataVariables` specifies the series among which pairwise correlations are plotted.
`crosscorr`	`DataVariables` specifies the two series for which the cross-correlation is computed.
`cusumtest`	`ResponseVariable` specifies the series to use for the regression response and `PredictorVariables` specifies the series to use for the regression predictors.
`egcitest`	`ResponseVariable` specifies the series to use for the response in the cointegrating regression and `PredictorVariables` specifies the series to use for the predictors in the cointegrating regression.
`fgls`	`ResponseVariable` specifies the series to use for the regression response and `PredictorVariables` specifies the series to use for the regression predictors.
`hac`	`ResponseVariable` specifies the series to use for the regression response and `PredictorVariables` specifies the series to use for the regression predictors.
`hpfilter`	`DataVariables` specifies the series to filter.
`i10test`	`DataVariables` specifies the series to test.
`jcitest`	`DataVariables` specifies the series to test.
`jcontest`	`DataVariables` specifies the series to test.
`kpsstest`	`DataVariable` specifies the response series to test.
`lagmatrix`	`DataVariables` specifies the series to lag.
`lbqtest`	`DataVariable` specifies the residual series to test.
`lmctest`	`DataVariable` specifies the response series to test.
`parcorr`	`DataVariable` specifies the series for which the partial autocorrelation is computed.
`pptest`	`DataVariable` specifies the response series to test.
`price2ret`	`DataVariables` specifies the series to convert to returns.
`recreg`	`ResponseVariable` specifies the series to use for the regression response and `PredictorVariables` specifies the series to use for the regression predictors.
`ret2price`	`DataVariables` specifies the series to convert to prices.
`vratiotest`	`DataVariable` specifies the response series to test.

Compatibility Considerations

The functions in the table now return tables of results when you supply input data in a table or timetable. Before R2022a, the functions returned results in numeric arrays. The table compares the previous and new syntaxes for table and timetable data inputs `Tbl`.

Function	Before R2022a	R2022a and Later Releases
`chowtest`	`[h,pValue,stat,cValue] = chowtest(Tbl,...)`, all outputs are vectors	`StatTbl = chowtest(Tbl,...)`, `StatTbl` is a table containing variables `h`, `pValue`, `stat`, and `cValue`
`collintest`	`[sValue,condIdx,VarDecomp,h] = collintest(Tbl,...)`, all outputs are arrays	`[VarDecompTbl,h] = collintest(Tbl,...)`, `VarDecompTbl` is a table containing variables `sValue`, `condIdx`, and separate variables for each column of `VarDecomp`, with names equal to the names in `Tbl`
`corrplot`	`[R,PValue,H] = corrplot(Tbl,...)`, all outputs are arrays	`[R,PValue,H] = corrplot(Tbl,...)`, all outputs are tables with the same dimensions as the arrays and the same variable and row names as the names in `Tbl`.
`cusumtest`	`[h,H,Stat,W,B] = cusumtest(Tbl,...)`, all outputs are arrays	`[h,H,Stat,W,B] = cusumtest(Tbl,...)`, outputs `H`, `Stat`, and `W` are tables with the same dimensions as the corresponding arrays
`fgls`	`[coeff,se,EstCoeffCov,iterPlots] = fgls(Tbl,...)`, all outputs are arrays	`[CoeffTbl,CovTbl,iterPlots] = fgls(Tbl,...)`, `CoeffTbl` is a table containing variables for `coeff` (`Coeff`), `se` (`SE`), and `CovTbl` is a table containing `EstCoeffCov`, with row and variable names equal to the variable names in `Tbl`
`hac`	`[EstCoeffCov,coeff,se] = hac(Tbl,...)`, all outputs are arrays	`[CovTbl,CoeffTbl] = hac(Tbl,...)`, `CovTbl` is a table containing `EstCoeffCov`, with row and variable names equal to the variable names in `Tbl`, `CoeffTbl` is a table containing variables for `coeff` (`Coeff`), `se` (`SE`)
`i10test`	`[H,PValue] = i10test(Tbl,...)`, all outputs are arrays	`DecisionTbl = i10test(Tbl,...)`, `DecisionTbl` is a table containing variables for each column of `H` (labeled `I1` and `I0`) and `PValue` (labeled `P1` and `P0`)
`recreg`	`[Coeff,SE,coeffPlots] = recreg(Tbl,...)`, all outputs are arrays	`[CoeffTbl,SETbl,coeffPlots] = recreg(Tbl,...)`, `Coeff` and `SE` are tables containing the corresponding arrays, with row names equal to the variable names in `[Const Tbl.Properties.VariableNames]` and variable names equal to `Iter`*`j`*, where *`j`* is column *`j`*

Update your code by returning the appropriate outputs for release R2022a or later, and by using table indexing to access results. For more details, see Access Data in Tables.

Bayesian State-Space Models: Analyze posterior distributions of random parameters in multivariate linear state-space models

Econometrics Toolbox provides Bayesian state-space models for analyzing multivariate time series data from a Bayesian point of view. The `bssm` function creates a Bayesian state-space model (`bssm` object) that characterizes the joint prior distribution on the state and observation equation parameters, including coefficient, state-disturbance-loading, and observation-innovation matrices.

The framework enables you to create a Bayesian state-space model in two ways:

- Convert a standard state-space model (`ssm` object) to a Bayesian state-space model by using the `ssm2bssm` function.
- For maximum flexibility, pass the following inputs to `bssm`:
  - A parameter-to-matrix mapping function that specifies the structure of the state-space model and determines the likelihood computed by the Kalman filter
  - Prior distribution of the parameters, a function representing the log prior or matrix of random draws from the distribution

After creating a Bayesian state-space model, you can combine the prior model with data to form and analyze the posterior distribution. The data likelihood is formed by assuming that the state disturbances and observation innovations are multivariate Gaussian random variables with a mean of 0. The posterior distribution does not have a closed form; the framework uses Metropolis-Hastings to sample from the posterior. Posterior distribution formation and analysis functions include:

- `estimate` – Estimate moments of the posterior distribution given the state-space model structure, data, and initial values for optimization.
- `simulate` – Draw random state-space model parameters for custom Monte Carlo estimation.
- `tune` – Compute proposal distributions for the Metropolis-Hasting sampler to facilitate Monte Carlo simulation.

lagmatrix Function: Specify presample, postsample, or output shape, and return time base

In addition to supporting input data in a table or timetable, the `lagmatrix` function enables you to optionally specify the following values by using name-value argument syntax:

- `Y0` — Presample data, specified as a matrix, table, or timetable
- `YF` — Postsample data, specified as a matrix, table, or timetable
- `Shape` — Output shape specifying which part of the shifted series to return. Values include:
  - `"full"` — Outputs include presample, in-sample, and postsample values in the expanded time base.
  - `"same"` — Outputs include values on the time base of the input data.
  - `"valid"` — Outputs include values for times at which all series have specified (non-`NaN`) values.

Also, `lagmatrix` returns the time base defined by the value of `Shape`, either in the second output position when you specify numeric data or as a variable in the output table when you specify data in a table or timetable.

price2ret and ret2price Functions: Specify datetime observation dates and corresponding time units, and specify name-value arguments for optional positional inputs

In addition to supporting input data in a table or timetable, the `price2ret` and `ret2price` functions enable you to optionally specify the following values by using name-value argument syntax:

- `Ticks` — Observation times, specified as a datetime or numeric vector
- `Units` — Time units when `Ticks` is a datetime vector, specified as a time unit name.

Compatibility Considerations

Now, `price2ret` and `ret2price` accept all optional inputs using name-value arguments. There are no plans to remove the optional positional input syntaxes; the functions will continue to accept them as in previous releases. The following table lists the recommended replacement options.

Function	Syntaxes Before R2022a	Recommended Syntaxes for R2022a and Later Releases
`price2ret`	`price2ret(TickSeries,`*`TickTimes`*`,`*`Method`*`)` Required input `TickSeries` is a matrix.	`price2ret(TickSeries,Name=Value)` • Required input `TickSeries` is a matrix, table, or timetable. • `Ticks=`*`TickTimes`* replaces the second input. • `Method=`*`Method`* replaces the third input.
`ret2price`	`ret2price(RetSeries,`*`StartPrice`*`,`*`RetIntervals`*`,`*`StartTime`*`,`*`Method`*`)` Required input `RetSeries` is a matrix.	`ret2price(RetSeries,Name=Value)` • Required input `RetSeries` is a matrix, table, or timetable. • `StartPrices=`*`StartPrice`* replaces the second input. • `Intervals=`*`RetIntervals`* replaces the third input. • `StartTime=`*`StartTime`* replaces the fourth input. • `Method=`*`Method`* replaces the fifth input.

As with any set of name-value arguments, you can specify them in any order.

hpfilter Function: The function removes missing values from the data and supports name-value argument syntax for optional inputs

The `hpfilter` function removes missing observations, specified by `NaN` values, from the data by using listwise deletion. That is, if a row of the input data contains one `NaN` value, the function removes the entire row. The action reduces the sample size and can create an irregular time series.

Compatibility Considerations

Now, `hpfilter` accepts the optional, positional smoothing parameter value using name-value argument syntax. There are no plans to remove the optional positional input syntax; the function will continue to accept it as in previous releases.

The syntax before R2022a is

`hpfilter(Data,`*`smoothing`*`)`

The required input `Data` is a matrix.

The recommended syntax for R2022a and later releases is

`hpfilter(Data,Smoothing=`*`smoothing`*`)`

The required input `Data` is a matrix, table, or timetable. `Smoothing=`*`smoothing`* replaces the second input of earlier releases.

Time Series Preprocessing: Add business calendar awareness to time series data

The `addBusinessCalendar` function adds business calendar awareness to the time base of a timetable. The function optionally accepts custom holiday and weekend definitions.

recessionplot Function: Plot band for new recession interval

The National Bureau of Economic Research defined the period February, 2020, through April, 2020, in the US as a recession (https://www.nber.org/research/business-cycle-dating). The data set `Data_Recessions.mat` includes the new recession period. Consequently, the `recessionplot` function plots a recession band for that period in time series plots.

R2021b

Version: 5.7

New Features

Threshold-Switching Dynamic Regression Models: Analyze and model nonlinear multivariate time series

Econometrics Toolbox has a class of threshold-switching dynamic regression tools for modeling multivariate time series in the presence of structural breaks or regime changes. The framework supports threshold autoregressive (TAR), self-exciting threshold autoregressive (SETAR), and smooth transition autogressive (STAR) models.

The `threshold` function creates a threshold transitions model that describes the switching mechanism among latent states. The specified levels of an observable variable determine the thresholds for abrupt or variable-rate, smooth transitions between states. `threshold` model tools enable you to study the structure of the switching mechanism.

- Visualize the threshold transitions model with data on the observable variable by using `ttplot`.
- Evaluate the transition function at values of the observable variable by using `ttdata`.
- Determine the state of the system at values of the observable variable by using `ttstates`.

The `tsVAR` function creates a threshold-switching dynamic regression model from:

- A threshold transitions model (`threshold`) that describes the switching mechanism. The corresponding observable variable can be endogenous, for a self-exciting model, or exogenous.
- A collection of state-specific autoregressive (`arima`) or vector autoregression (`varm`) models, which can contain exogenous regression components.

`tsVAR` model tools enable you to:

- Fit model parameters to observed data by using `estimate`. Estimable parameters include transition mid-levels, rates of smooth transition functions, submodel coefficients, and innovations covariances. The `summarize` function provides switching model estimates and state-specific coefficient estimates, inferences, and information criteria.
- Simulate response or state paths by using `simulate`.
- Predict future responses by using `forecast`.

State-Space Models: Apply Kalman filter in real time to estimate state distribution moments

The `update` function estimates state distribution moments at time *T* for a specified state-space model (`ssm` object) and *T* observations from a measurement variable, by applying one forward recursion of the Kalman filter.

Alternatively, you can estimate state distribution moments in real time by calling the `filter` function and setting the `'RealTimeUpdate'` name-value argument to `true`.

Markov-Switching Dynamic Regression Models: Assign names to time series and summarize estimation results

- The `'SeriesNames'` name-value argument of the `msVAR` function assigns names to the time series of the model.

- The `summarize` function displays or returns Markov-switching dynamic regression model estimation results. Results include parameter estimates and corresponding standard errors, information criteria, and the estimated transition probability matrix.

Time Series Preprocessing: Aggregate timetable data to different periodicities ranging from daily to annually

The following functions aggregate timetable data to the specified periodicity:

- `convert2daily`
- `convert2weekly`
- `convert2monthly`
- `convert2quarterly`
- `convert2semiannual`
- `convert2annual`

Diebold-Li Model Example: Analyze yield-curve model augmented with macroeconomic variables

The Apply State-Space Methodology to Analyze Diebold-Li Yield Curve Model example contains two additional analyses of the Diebold-Li yield-curve model:

- **Yields-macro model** — In addition to the yields for each maturity, the state equation includes the endogenous macroeconomic variables: manufacturing capacity utilization, effective federal funds rate, and inflation measured by the change in personal consumption expenditures price index. The analysis characterizes the dynamic relationship between the yield and macroeconomic variables.
- **Yields model with exogenous macroeconomic variables** — This model is similar to the yields-macro model, but the macroeconomic variables are exogenous.

To support the new analyses, the `Data_DieboldLi.mat` data set includes a timetable `DataTable` containing the yield data and the macroeconomic series `CU`, `FEDFUNDS`, and `PI`. Supporting variables of the data set are similarly expanded for the macroeconomic variables.

R2021a

Version: 5.6

New Features

State-Space Models: Characterize dynamics using forecast error variance decomposition (FEVD)

The `fevd` function computes the FEVD of a standard state-space model (`ssm` object) or diffuse state-space model (`dssm` object). The FEVD provides information about the relative importance of each state disturbance in affecting the forecast error variance of all measurement variables in the system.

State-Space Models: Characterize dynamics using model-implied temporal correlation among state and observation variables

The `corr` function clarifies the dynamics of a standard state-space model (`ssm` object) by computing the model-implied temporal correlations or covariances of the state and observation variables.

R2020b

Version: 5.5

New Features

Impulse response function (IRF) of standard and diffuse state-space models

The `irf` function filters state-disturbance shocks through each state equation of a standard state-space model (`ssm` object) or diffuse state-space model (`dssm` object), and returns the resulting state and measurement variable IRF as a numeric array. `irf` optionally computes corresponding pointwise confidence intervals on the true IRF.

To plot the IRF and confidence intervals, use the `irfplot` function.

aicbic returns additional information criteria and can normalize results

In addition to Akaike and Bayesian information criteria (AIC and BIC, respectively), `aicbic` returns the corrected AIC (AICc), consistent AIC (CAIC), and Hanna-Quinn criterion (HQC). Also, the `'Normalize'` name-value pair argument scales all criteria by the specified sample sizes used in estimation.

R2020a

Version: 5.4

New Features

Bayesian vector autoregression models

Econometrics Toolbox provides Bayesian vector autoregression (VAR) models for analyzing multivariate time series data from a Bayesian point of view. The `bayesvarm` function creates a Bayesian VAR model object that characterizes the joint prior distribution on the coefficients and innovations covariance matrix. The framework enables you to regularize coefficients by using the Minnesota prior [124]. This table lists supported prior models, ordered by increasing flexibility and complexity.

Model Object	Description
`normalbvarm`	• Normal model • Normal coefficients; fixed innovations covariance matrix
`conjugatebvarm`	• Conjugate matrix-normal-inverse-Wishart model • Dependent coefficients and innovations covariance matrix
`semiconjugatebvarm`	• Semiconjugate normal-inverse-Wishart model • Independent coefficients and innovations covariance matrix
`diffusebvarm`	• Diffuse model • Joint prior distribution is inversely proportional to the determinant of the innovations covariance matrix

After creating a Bayesian VAR model object, you can combine the prior model with data to form and analyze the posterior distribution. The data likelihood is always formed by assuming that the innovations are iid multivariate normal random variables with a mean of 0 and a constant covariance matrix. Posterior distribution formation and analysis functions include:

- `estimate` – Estimate posterior distribution characteristics given data.
- `simulate` – Draw random coefficients or innovations covariances from posterior distributions for Monte Carlo estimation.
- `forecast` – Forecast responses into a forecast horizon with respect to the posterior predictive distribution.

corrplot accepts data in a timetable

The `corrplot` function additionally accepts input time series data in a MATLAB timetable.

R2019b

Version: 5.3

New Features

Compatibility Considerations

Markov-switching autoregression models

Econometrics Toolbox has a class of tools for modeling multivariate time series in the presence of structural breaks and unobserved latent states (`msVAR`).

The `msVAR` function creates a switching model from:

- A discrete-time Markov chain object (`dtmc`), which describes a switching mechanism among states
- A collection of state-specific autoregressive (`arima`) or vector autoregression (`varm`) models, which can contain exogenous regression components

Using `msVAR` tools, you can:

- Fit model parameters to observed data by using `estimate`, which implements the expectation-maximization procedure of Hamilton, 1994. Estimable parameters include state transition probabilities, model coefficients, and innovations covariances.
- Infer time-dependent state probabilities by using `filter` and `smooth`.
- Simulate response or state paths by using `simulate`, and predict future responses by using `forecast`.

Study the structure of the switching mechanism by using `dtmc` tools.

Finite-step analysis for discrete-time Markov chains

The `hitprob` and `hittime` functions compute hitting probabilities and expected hitting times in a discrete-time Markov chain (`dtmc`). Options allow for visualization of mixing times.

Equality constraints on innovations covariance matrix for VAR and VEC models

If you specify a positive definite innovations covariance matrix for the `Covariance` property of a `varm` or `vecm` object, the `estimate` function treats your specification as an equality constraint in model estimation.

Functionality being removed or changed

estimate includes the final lag in all estimated univariate time series model polynomials

The `estimate` functions of the `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` univariate time series model objects now include the final polynomial lag as specified in the input model template for estimation. In other words, the specified polynomial degrees of an input model template returned by an object creation function and the corresponding polynomial degrees of the estimated model returned by `estimate` are equal.

To illustrate the behavior difference between R2019b and earlier releases, consider creating a GARCH(4,2) model template, generating 50 random observations from the standard Gaussian distribution, and fitting the model to the data.

```
Mdl = garch(4,2)
rng(1) % For reproducibility
```

```
y = randn(50,1);
EstMdl = estimate(Mdl,y,'Display','off')
```

The model is a poor representation of the data generating process. The likely reason is that several of the estimated lag coefficients are zero, particularly the larger lags.

In releases before R2019b, the display is:

```
EstMdl =

  garch with properties:

     Description: "GARCH(2,1) Conditional Variance Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 2
               Q: 1
        Constant: 0.30454
           GARCH: {0.532338} at lag [2]
            ARCH: {0.172308} at lag [1]
          Offset: 0
```

`EstMdl` represents an estimated GARCH(2,1) model because the GARCH polynomial coefficient estimates at lags 3 and 4, and the ARCH polynomial coefficient estimate at lag 2, are below the tolerance of `1e–12`. Consequently, `estimate` removes these coefficients from the respective polynomials.

In R2019b and future releases, `estimate` keeps the final coefficient in each polynomial to ensure that the specified degrees in the model template and the degrees of the estimated polynomials are equal. The display is:

```
EstMdl =

  garch with properties:

     Description: "GARCH(4,2) Conditional Variance Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 4
               Q: 2
        Constant: 0.30454
           GARCH: {0.532338 2e-12} at lags [2 4]
            ARCH: {0.172308 2e-12} at lags [1 2]
          Offset: 0
```

`EstMdl` represents an estimated GARCH(4,2) model. Estimates above the tolerance `1e–12` are equivalent among all releases.

Update Code

Polynomial degrees require minimum presample observations for operations downstream of estimation, such as model forecasting and simulation. If a model template in your code does not describe the data generating process well, then the polynomials in the estimated model can have higher degrees than in previous releases. Consequently, you must supply additional presample responses for operations on the estimated model; otherwise, the function issues an error. For more details, see the `'Y0'` name-value pair argument of the function.

estimate returns only an estimated model object and estimation summary
Errors

For a simpler interface, the `estimate` function of Bayesian linear regression models returns only an estimated model and an estimation summary table. `estimate` now supports these syntaxes:

```
PosteriorMdl = estimate(...);
[PosteriorMdl,Summary] = estimate(...);
```

You can obtain estimated posterior means and covariances, based on the marginal or conditional distributions, from the estimation summary table.

In past releases, `estimate` returned these output arguments:

```
[PosteriorMdl,estBeta,EstBetaCov,estSigma2,estSigma2Var,Summary] = estimate(...);
```

`estBeta`, `EstBetaCov`, `estSigma2`, and `estSigma2Var` are posterior means and covariances of β and σ^2.

Starting in R2019b, if you request any output argument in a position greater than the second position, `estimate` issues this error:

```
Too many output arguments.
```

For details on how to update your code, see Replacing Removed Syntaxes of estimate.

R2019a

Version: 5.2

New Features

Compatibility Considerations

Granger causality test

The `gctest` function conducts the Granger causality test to assess whether past values of a set of time series variables affect the present values of another set of time series variables. `gctest` accepts either observed paths of the time series variables or a `varm` model object, which represents a vector autoregression (VAR) model fitted to the observed paths.

Impulse response and forecast error decomposition functions for VAR and VEC models

The `irf` function filters innovation shocks through each variable in a fully specified VAR model object (`varm`) or vector error-correction (VEC) model object (`vecm`), and returns the response as a numeric array. The `fevd` function estimates the contributions to the forecast error variance of each variable in a fully specified VAR or VEC model object. The forecast error variance results from shocks to all variables that constitute the specified model.

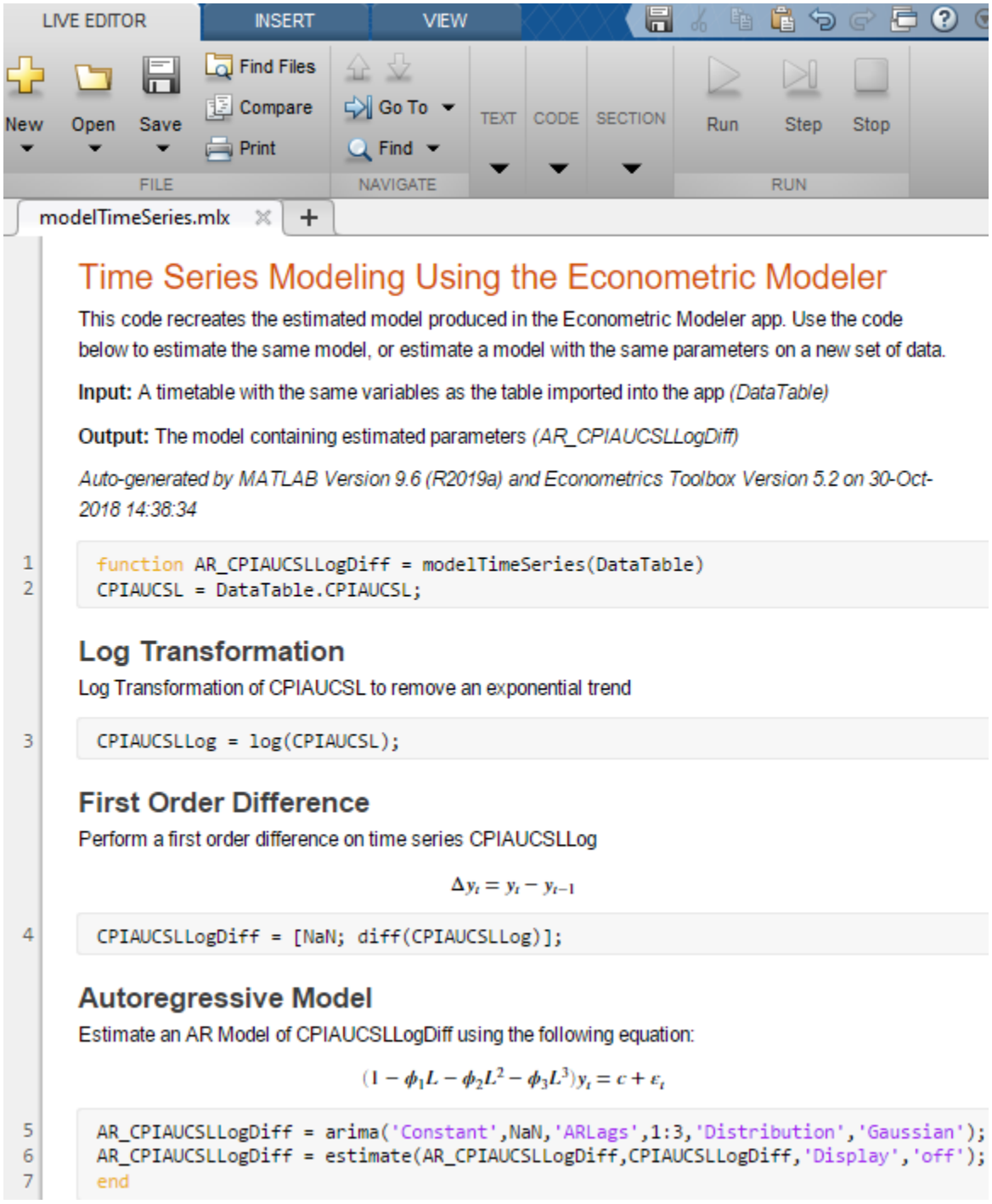
For more details on estimating the impulse response function (IRF) or forecast error variance decomposition (FEVD) of VAR models, see `irf` and `fevd`. For more details on estimating the IRF or FEVD of a VEC model, see `irf` and `fevd`.

Econometric Modeler generates live functions

The **Econometric Modeler** app enables you to generate a live function from a session during which you estimate a time series model. The live function accepts the loaded data set as input, and then outputs the estimated model.

After you estimate a model, you can generate a live function by following this procedure:

1. In the **Data Browser**, select the estimated model that you want the live function to return.
2. On the **Econometric Modeler** tab, in the **Export** section, click **Export** > **Generate Live Function**. The Live Editor opens and displays the live function.

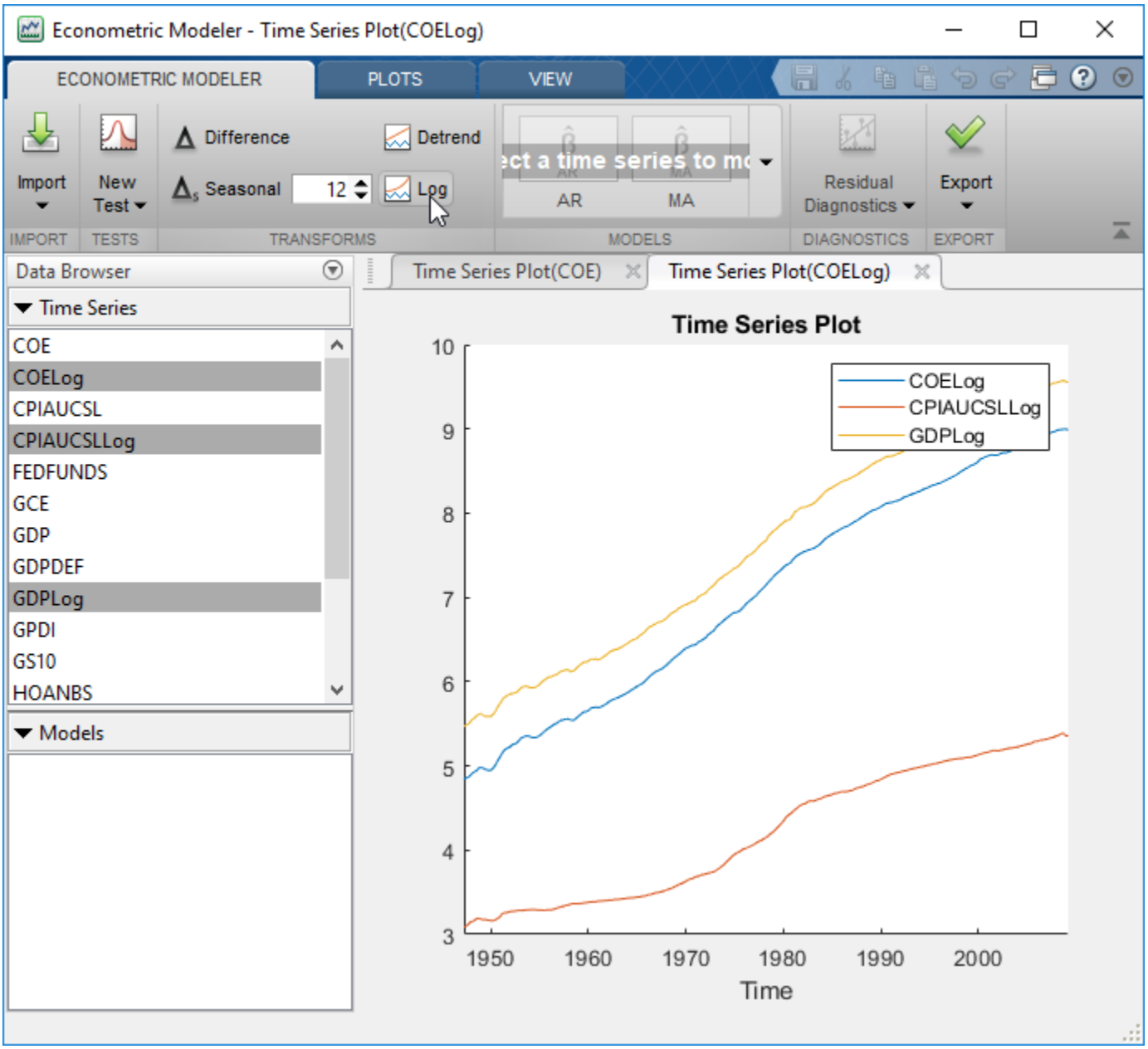


For more details on live functions, see Create Live Functions (MATLAB).

Econometric Modeler transforms multiple series simultaneously

The **Econometric Modeler** app allows you to apply a transformation to multiple time series simultaneously by following this procedure:

1. In the **Data Browser**, select all variables to which to apply the transformation.
2. On the **Econometric Modeler** tab, in the **Transforms** section, click the transformation.



Econometric Modeler fits intercept-only linear models

The **Econometric Modeler** app allows you to fit an intercept-only linear model by following this procedure:

1. In the **Data Browser**, select the response variable to model.
2. On the **Econometric Modeler** tab, in the **Models** section, click the arrow to display the models gallery.

3. In the models gallery, in the **Regression Models** section, click **MLR**.
4. In the **MLR Model Parameters** dialog box, click the **Estimate** button (do not select any check box in the **Include?** column).

Spot price simulation example with skew-normal modeling

The example Model and Simulate Electricity Spot Prices Using the Skew-Normal Distribution simulates the future behavior of electricity spot prices from a time series model fitted to historical data. Because electricity spot prices can exhibit large deviations, the example models the innovations using a skew-normal distribution.

New data set

Econometrics Toolbox includes a new data set containing simulated, daily electricity spot prices from January 1, 2010 through November 11, 2013.

To load the data set into the workspace, enter:

```
load Data_ElectricityPrices
```

Forecast ARIMAX responses without specifying presample predictor data

You can forecast responses from an ARIMAX model (represented by an `arima` model object containing a regression component) by specifying only the forecasted predictor data `XF` when you call the `forecast` function. That is, `forecast` does not require the presample predictor data `X0`.

Functionality being removed or changed

Univariate time series models require specification of presample data to forecast responses

The `forecast` functions of the `arima`, `garch`, `egarch`, and `gjr` univariate time series model objects now have a third input argument for you to supply presample response data:

```
forecast(Mdl,numperiods,Y0)
forecast(Mdl,numperiods,Y0,Name,Value)
```

Before R2019a, these syntaxes were:

```
forecast(Mdl,numperiods)
forecast(Mdl,numperiods,Name,Value)
```

With these syntaxes, you could optionally supply presample responses by using the `'Y0'` name-value pair argument.

There are no plans to remove the previous syntaxes or the `'Y0'` name-value pair argument at this time, but you are encouraged to use the new syntaxes.

To forecast responses from an econometric model, MATLAB must initialize the model by using enough response data occurring before the forecast period. Without specified presample responses,

`forecast` initializes models by using reasonable default values. However, the default initialization might not support all workflows. This table describes the default presample values for each model object.

Model Object	Presample Default
`arima`	For stationary models without a regression component, all presample responses are the unconditional mean of the process. For nonstationary models or models containing a regression component, all presample responses are `0`.
`garch`	All presample responses are the unconditional standard deviation of the process.
`egarch`	All presample responses are `0`.
`gjr`	All presample responses are the unconditional standard deviation of the process.

Update Code

Update your code by supplying presample responses when you forecast observations from univariate time series models using `forecast`. Specify the presample responses by using the third input argument.

If you do not supply presample responses, then `forecast` provides default presample values that might not support all workflows.

R2018b

armafevd Function: Estimate the forecast error variance decomposition (FEVD) for VARMA models

The `armafevd` function estimates the contributions to the forecast error variance of each variable in a multivariate vector autoregressive, moving average (VARMA) model. The forecast error variance results from shocks to all variables in the system. `armafevd` returns the FEVD as a numeric array or in separate figures for each variable.

Bayesian Linear Regression: Perform Bayesian variable selection for dimensionality reduction

The `bayeslm` function supports the creation of additional Bayesian linear regression model objects suitable for variable (predictor or feature) selection. The additional objects impose an inverse gamma prior distribution on the disturbance variance (σ^2), but three alternative priors on the regression coefficients (β) are available, as described in this table.

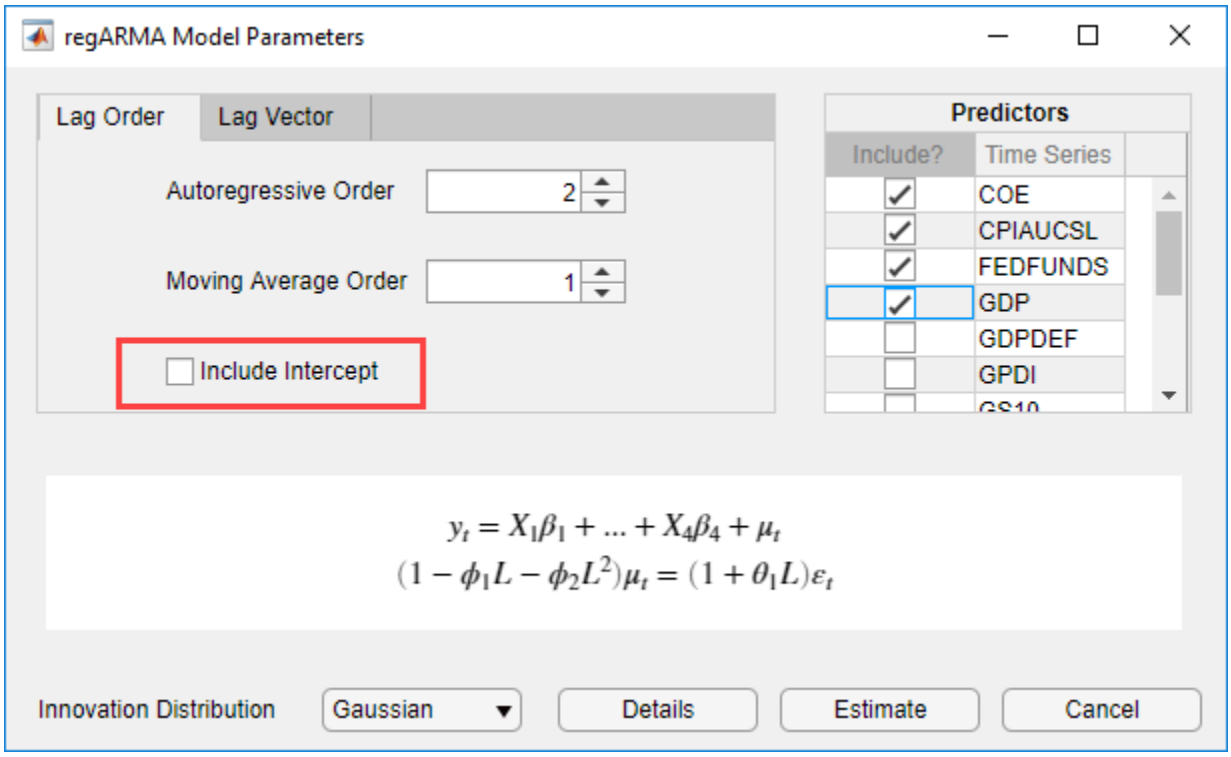
Model Object	Description
`mixconjugateblm`	• The variable selection algorithm is stochastic search variable selection (SSVS). • The data likelihood, prior distribution, and posterior distributions compose a conjugate Gaussian mixture model. • β and σ^2 are dependent random variables.
`mixsemiconjugateblm`	• The variable selection algorithm is SSVS. • The data likelihood, prior distribution, and the prior and posterior distributions compose a semiconjugate Gaussian mixture model. • β and σ^2 are independent random variables.
`lassoblm`	• The variable selection method is shrinkage estimation by Bayesian lasso regression. • Conditioned on σ^2, the prior distribution of each regression coefficient is double exponential with a mean of 0 and scale σ/λ, where λ is the lasso shrinkage parameter. As λ increases, the coefficients tend towards 0. • β and σ^2 are dependent random variables. Regression coefficients are independent, a priori.

After creating a Bayesian linear regression model object using `bayeslm`, you can combine the prior model with data to form the posterior distribution and perform variable selection. The disturbances are iid Gaussian random variables with a mean of 0 and a constant variance. Variable selection, posterior distribution estimation, and analysis functions include:

- `estimate` — Perform variable selection and estimate posterior distribution characteristics given predictor and response data.
- `simulate` — Simulate posterior distribution values for Monte Carlo estimation.
- `forecast` — Forecast responses given new predictor data with respect to the posterior predictive distribution.

Econometric Modeler App: Exclude intercept from linear models or regression models with ARMA errors

In the **Econometric Modeler** app, when you specify a multiple linear regression model or regression model with ARMA errors for estimation, you can exclude an intercept from the model by clearing the **Include Intercept** check box.



Bayesian Linear Regression: Access covariance estimates from estimation summary tables

The estimation summary table, returned by `estimate` and `summarize` as a MATLAB table, stores the estimated covariance of the regression coefficient and disturbance variance estimates in the `Covariances` variable.

armairf Function: Supply or return graphics object handles

The `armairf` function accepts a vector of axes handles in which to plot and returns handles to plotted graphics objects.

For details on graphics objects, see Graphics Objects.

Functionality being removed or changed

armairf returns separate figures for each variable in the system
Behavior change

The `armairf` function now plots, in separate figures, the impulse response function (IRF) of the `numVars` variables in a system. Each figure contains `numVars` line plots representing the responses of a variable to a shock to all variables in the system at time 0.

In previous releases, `armairf` returned one figure containing separate subplots for each variable.

armairf permutes the second and third dimensions of the impulse response array output
Behavior change

When you estimate and return a multivariate impulse response function (IRF) by using `armairf`:

- The `numVars` columns of the 3-D array output now correspond to the `numVars` variables in the system receiving a shock at time 0.
- The `numVars` pages of the 3-D array output now correspond to the IRF of the `numVars` variables in the system.

In other words, element t,j,k of the returned 3-D array is the IRF of variable k to a shock to variable j at time 0, for t = 1,...,`numObs`. `numObs` is the number of observations in the IRF representing times 0, ...,`numObs` – 1, respectively, and j,k = 1,...,`numVars`.

In previous releases, the `numVars` columns of the 3-D array output of `armairf` corresponded to the IRF of the `numVars` variables in the system, and pages corresponded to the `numVar` variables in the system receiving a shock at time 0.

If you index the columns or pages of the 3-D array output, and you want to update your code so that you obtain results in the same format, permute the column and page indices. For example, suppose `Y` is the IRF returned by `armairf` in R2018b or a later release. To reproduce the formatted results in previous releases, rearrange the columns and pages of `Y` by using `permute`:

```
permute(Y, [1 3 2])
```

estimate will return only an estimated model object and estimation summary
Warns

For a simpler interface, the `estimate` function of Bayesian linear regression models will return only an estimated model and an estimation summary table in a future release. That is, `estimate` will return only these output arguments:

```
[PosteriorMdl,Summary] = estimate(...);
```

You will be able to obtain estimated posterior means and covariances, based on the marginal or conditional distributions, from the estimation summary table.

In the current and past releases, `estimate` returns these output arguments:

```
[PosteriorMdl,estBeta,EstBetaCov,estSigma2,estSigma2Var,Summary] = estimate(...);
```

Starting in this release, if you request any output argument other than `PosteriorMdl`, then `estimate` issues this warning:

```
Warning: Current syntax supports 6 output arguments, and will be removed in a future release. Fo
arguments, see estimate.
```

For details on how to update your code to prepare for this change, see Replacing Discouraged Syntaxes of `estimate`.

R2018a

Version: 5.0

New Features

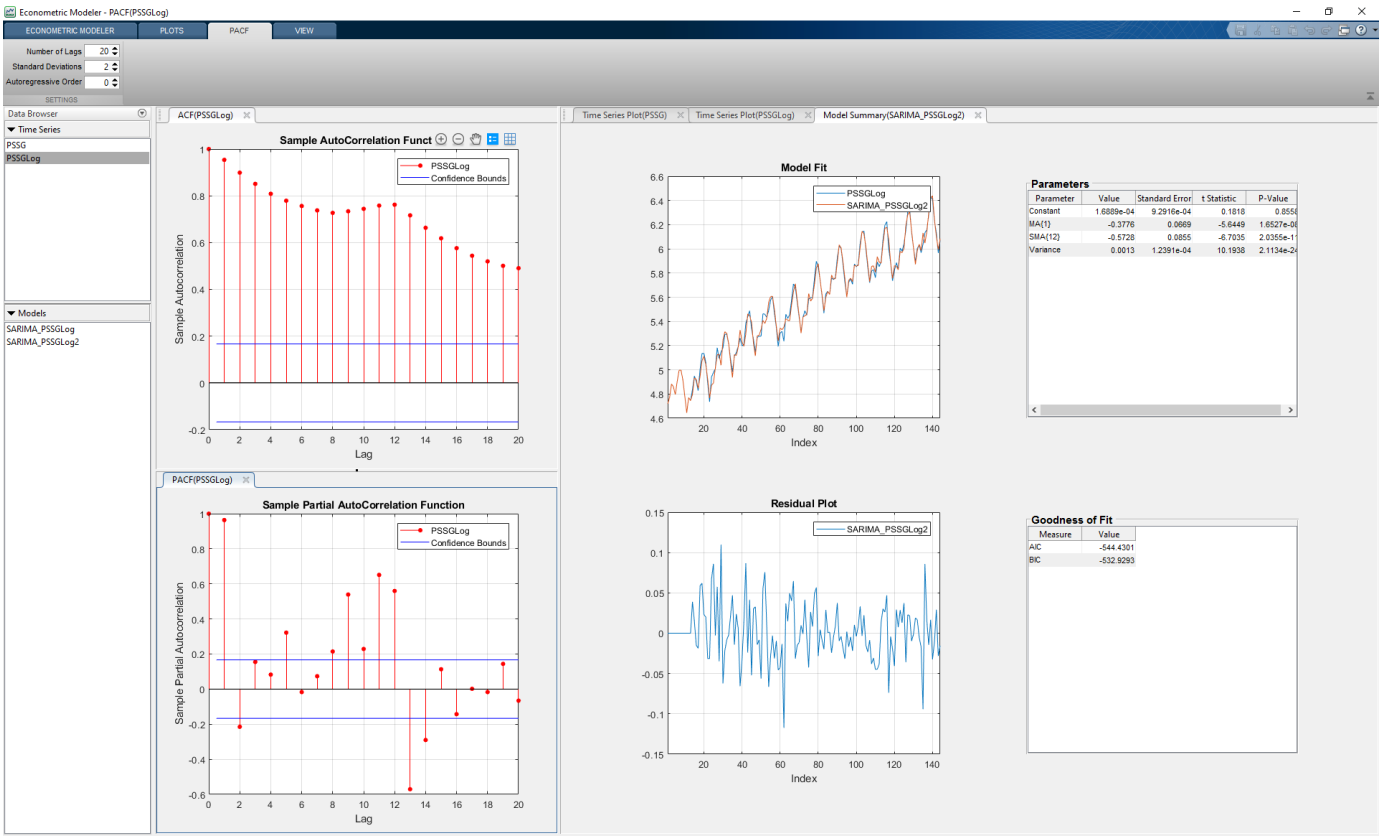
Compatibility Considerations

Econometric Modeler App: Perform time series analysis, specification testing, modeling, and diagnostics

The **Econometric Modeler** app lets you visualize and transform time series data, perform statistical specification and model identification tests, estimate candidate models, and perform post-fit assessments. After working in the app, you can export variables to the MATLAB Workspace and generate code or a report summarizing your session.

To launch the Econometric Modeler app, click the **Apps** tab, then click the app in the **Computational Finance** category of the apps gallery. Or, enter `econometricModeler` at the command line.

For more details, see Econometric Modeler Overview.



Missing Data Support: Autocorrelation, partial autocorrelation, and Ljung-Box Q-test calculations account for missing observations

The `autocorr`, `parcorr`, and `lbqtest` functions treat `NaN` values in the data as missing completely at random.

Toolbox Plotting Functions: Supply or return graphics object handles

These functions accept a graphics handle in which to plot and return handles to plotted graphics objects.

- `autocorr`
- `distplot`
- `collintest`
- `corrplot`
- `crosscorr`
- `cusumtest`
- `eigplot`
- `fgls`
- `graphplot`
- `parcorr`
- `recreg`
- `simplot`

For details on graphics objects, see Graphics Objects.

Model Summary: Display estimation results from fitting models

The `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` model objects allow you to display estimation results from estimated models by using the `summarize` function. Estimation results include parameter estimates, standard errors, *t* statistics, *p*-values, and model fit statistics. For more details, see `summarize` for `arima` models, `summarize` for `regARIMA` models, and `summarize` for conditional variance models.

Model Description: Assign descriptions to econometric model objects

The `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` model objects allow you to assign a description to the model by setting the `Description` property.

Model Lag Parameters: Use indices that are consistent with MATLAB cell array indexing

The indices of cell arrays of lag operator polynomial coefficients for `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` model objects follow MATLAB cell array indexing rules.

Affected model properties are:

- `AR`, `MA`, `SAR`, and `SMA` properties of `arima` and `regARIMA` models
- `GARCH` and `ARCH` properties of `garch`, `egarch`, and `gjr` models
- `Leverage` property of `egarch` and `gjr` models

Compatibility Considerations

- You cannot access any lag-zero coefficients by using an index of `0`. For example, `Mdl.AR{0}` issues an error, where `Mdl` is an `arima` model.

Remove any instances of such indices of zero from your code. The value of all lag-zero coefficients is `1` except for lag operator polynomials corresponding to the `ARCH` and `Leverage` properties, which have the value `0`.

- You cannot index beyond the maximal lag in the polynomial. For example, if `Mdl.P` is 4, then `Mdl.AR{p}` issues an error when `p` is greater than `4`. For details on the maximal lags of the lag operator polynomials, see the corresponding property descriptions in the reference pages.

  Remove any instances of such indices beyond the maximal lag from your code. All coefficients beyond the maximal lag are `0`.

Model Distribution: Store innovation distribution name as string scalar

In `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` model objects, the `Name` field of the `Distribution` property stores the innovation distribution name as a string scalar, for example, `"Gaussian"`.

Compatibility Considerations

Previously, MATLAB stored the innovation distribution name as a character vector, for example `'Gaussian'`. Although most text-data operations accept character vectors and string scalars for text-data input, there are some differences between the two data types. For more details, see Characters and Strings.

Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations

`autocorr(y,numLags)`
 	
`autocorr(y,numLags,numMA)`
 	
`autocorr(y,numLags,numMA,numSTD)`
	Still runs	Name-value pair argument syntax, for example, `autocorr(y,'NumLags',numLags)`	Replace all instances of `numLags`, `numMA`, and `numSTD` with corresponding name-value pair argument syntax.

`crosscorr(y1,y2,numLags)`
 	
`crosscorr(y1,y2,numLags,numSTD)`
	Still runs	Name-value pair argument syntax, for example, `crosscorr(y1,y2,'NumLags',numLags)`	Replace all instances of `numLags` and `numSTD` with corresponding name-value pair argument syntax.

`parcorr(y,numLags)`
 	
`parcorr(y,numLags,numAR)`
 	
`parcorr(y,numLags,numAR,numSTD)`
	Still runs	Name-value pair argument syntax, for example, `parcorr(y,'NumLags',numLags)`	Replace all instances of `numLags`, `numAR`, and `numSTD` with corresponding name-value pair argument syntax.

Functionality	Result	Use Instead	Compatibility Considerations
`print` function for `arima`, `regARIMA`, `garch`, `egarch`, and `gjr` model objects	Warns	`summarize` for `arima` models `summarize` for `regARIMA` models `summarize` for conditional variance models	Replace all instances of `print` with `summarize`.

`archtest(residuals,lags)`
 	
`archtest(residuals,lags,alpha)`
	Issues error	Name-value pair argument syntax, for example, `archtest(residuals,'Lags',lags)`	Replace all positional argument usages of `lags` and `alpha` with corresponding name-value pair argument syntax.

`lbqtest(series,lags)`
 	
`lbqtest(series,lags,alpha)`
 	
`lbqtest(series,lags,alpha,dof)`
	Issues error	Name-value pair argument syntax, for example, `lbqtest(series,'Lags',lags)`	Replace all positional argument usages of `lags`, `alpha`, and `dof` with corresponding name-value pair argument syntax.

`vgxar`	Issues error	`arma2ar`	Replace all instances of `vgxar` with `arma2ar`.
`vgxcount`	Issues error	`summarize` function of the `varm` object	Replace all instances of `vgxcount` with `summarize`.
`vgxdisp`	Issues error	`summarize` function of the `varm` object	Replace all instances of `vgxdisp` with `summarize`.
`vgxget`	Issues error	Dot notation on the `varm` model object	Replace all instances of `vgxget` with dot notation on the `varm` model object.
`vgxinfer`	Issues error	`infer` function of the `varm` object	Replace all instances of `vgxinfer` with `infer`.
`vgxloglik`	Issues error	`infer` function of the `varm` object	Replace all instances of `vgxloglik` with `infer`.
`vgxma`	Issues error	`arma2ma`	Replace all instances of `vgxma` with `arma2ma`.
`vgxplot`	Issues error	`subplot` and `plot`	Replace all instances of `vgxplot` with `subplot` and `plot`.

Functionality	Result	Use Instead	Compatibility Considerations
`vgxpred`	Issues error	`forecast` function of the `varm` object	Replace all instances of `vgxpred` with `forecast`.
`vgxproc`	Issues error	`filter` function of the `varm` object	Replace all instances of `vgxproc` with `filter`.
`vgxqual`	Issues error	`isStable` function of the `LagOp` object	Replace all instances of `vgxqual` with `isStable`.
`vgxset`	Issues error	`varm`	Replace all instances of `vgxset` with `varm` or use dot notation.
`vgxsim`	Issues error	`simulate` function of the `varm` object	Replace all instances of `vgxsim` with `simulate`.
`vgxvarx`	Issues error	`estimate` function of the `varm` object	Replace all instances of `vgxvarx` with `estimate`.
`'print'` name-value pair argument of the `estimate` function for `arima`, `garch`, `egarch`, and `gjr` model objects	Issues error	`'Display'` name-value pair argument	Replace all instances of `'print',true` with `'Display','on'`, and `'print',false` with `'Display','off'`.
`Data_VARMA22` data set	Issues error	N/A	Remove all instances from your code.

R2017b

Version: 4.1

New Features

Discrete-Time Markov Chains: Analyze the structure and evolution of Markov models

Econometrics Toolbox has a class of tools for analyzing the structure and evolution of discrete-time Markov chains. The `dtmc` function creates model objects from empirical transition counts or theoretical transition probabilities. The `mcmix` function creates random models with a specified structure for testing purposes. The tools support state classification and the analysis of asymptotic behavior. A variety of visualization methods complement the analytic tools.

Vector Error-Correction Model: Analyze multivariate time series with cointegrating relationships

Econometrics Toolbox has a vector error-correction (VEC) model for multivariate time series with cointegrating relationships. The `vecm` function creates model objects characterizing VEC models with cointegration. Additional functions support modeling workflows for estimation, simulation, and forecasting.

In addition to the VEC model analysis tools, you can convert a VAR model (a `varm` model object) to its equivalent VEC model representation using `vecm`.

Bayesian Linear Regression: Draw samples from posterior distributions using the Hamiltonian Monte Carlo sampler

Bayesian linear regression models with custom prior distributions (that is, `customblm` model objects) can efficiently draw from corresponding parameter posterior distributions using the Hamiltonian Monte Carlo sampler. For more details, see Using Hamiltonian Monte Carlo (Statistics and Machine Learning Toolbox™).

New Data Set

Econometrics Toolbox includes a new data set containing quarterly U.S. macroeconomic measurements from 1957:Q1–2016:Q4 and quarterly econometric projections for the following 10 years from the Congressional Budget Office.

To load the data set, enter the following code at the command line.

```
load Data_USEconVECModel
```

R2017a

Version: 4.0

New Features

Compatibility Considerations

Bayesian Linear Regression: Analyze posterior distributions of random parameters in multiple regression models

Econometrics Toolbox has a Bayesian linear regression model for analyzing the linear relationship between a response and a set of predictor variables. The `bayeslm` function creates a Bayesian linear regression model object that characterizes the joint prior distribution on the regression coefficients and disturbance variance. This table lists supported prior models, ordered by increasing flexibility and complexity.

Model object	Description
`conjugateblm`	• Conjugate normal-inverse-gamma model • The regression coefficients and the disturbance variance are dependent
`semiconjugateblm`	• Semiconjugate normal-inverse-gamma model • The regression coefficients and disturbance variance are independent
`diffuseblm`	• Diffuse model • The joint prior distribution is inversely proportional to the disturbance variance
`empiricalblm`	• Empirical model • You supply draws from the joint distribution, which characterizes the joint prior
`customblm`	• Custom model • The log of the joint prior distribution is described in a declared function.

After creating a Bayesian linear regression model object using `bayeslm`, you can combine the prior model with data to form and analyze the posterior distribution. The data likelihood is always formed by assuming that the disturbances are I.I.D. Gaussian random variables with a mean of 0 and a constant variance. Posterior distribution formation and analysis functions include:

- Estimation of posterior distribution characteristics given predictor and response data using `estimate`
- Simulation of posterior distribution values for Monte Carlo estimation using `simulate`
- Forecasting responses for new predictor data with respect to the posterior predictive distribution using `forecast`

Vector Autoregressive Model: Analyze multivariate time series data including exogenous predictors

Econometrics Toolbox has a vector autoregressive model including exogenous predictor data (VARX) for analyzing multivariate time series data. The `varm` function creates a `varm` model object, which characterizes a VARX(p) model. After creating a `varm` model object, you can:

- Fit the model to predictor and response data using `estimate`.
- Simulate observations given predictor data using `simulate`.
- Forecast observations given new predictor data using `forecast`.

Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
`vgxar`	Warns	`arma2ar`	Replace all instances of `vgxar` with `arma2ar`.
`vgxcount`	Warns	`summarize` function of the `varm` object	Replace all instances of `vgxcount` with `summarize`.
`vgxdisp`	Warns	`summarize` function of the `varm` object	Replace all instances of `vgxdisp` with `summarize`.
`vgxget`	Warns	Dot notation on `varm` model object	Replace all instances of `vgxget` with dot notation on `varm` model object.
`vgxinfer`	Warns	`infer` function of the `varm` object	Replace all instances of `vgxinfer` with `infer`.
`vgxloglik`	Warns	`infer` function of the `varm` object	Replace all instances of `vgxloglik` with `infer`.
`vgxma`	Warns	`arma2ma`	Replace all instances of `vgxma` with `arma2ma`.
`vgxplot`	Warns	`subplot` and `plot`	Replace all instances of `vgxplot` with `subplot` and `plot`.
`vgxpred`	Warns	`forecast` function of the `varm` object	Replace all instances of `vgxpred` with `forecast`.
`vgxproc`	Warns	`filter` function of the `varm` object	Replace all instances of `vgxproc` with `filter`.

Functionality	Result	Use Instead	Compatibility Considerations
`vgxqual`	Warns	`isStable` function of the `LagOp` object.	Replace all instances of `vgxqual` with `isStable`.
`vgxset`	Warns	`varm`	Replace all instances of `vgxset` with `varm` or use dot notation.
`vgxsim`	Warns	`simulate` function of the `varm` object	Replace all instances of `vgxsim` with `simulate`.
`vgxvarx`	Warns	`estimate` function of the `varm` object	Replace all instances of `vgxvarx` with `estimate`.

R2016b

Version: 3.5

Bug Fixes

Compatibility Considerations

Econometrics data folder renamed to econdata

The `econdemos` folder in *`matlabroot`*`/toolbox/econ` is now named `econdata`.

Compatibility Considerations

To avoid errors, replace all instances of `econdemos` in your code with `econdata`.

R2016a

Version: 3.4

New Features

Compatibility Considerations

Cusum Structural Change Tests: Assess stability of multiple linear regression models

The `cusumtest` function implements the Brown-Durbin-Evans cusum test to assess the stability of coefficients over time in multiple linear regression models. `cusumtest` supports:

- Both cusum and cusum of squares tests
- Estimating recursive residuals using forward and backward regressions
- Plotting recursive residuals with critical lines

Recursive Linear Regression: Recursively estimate coefficients of multiple linear regression models

The `recreg` function plots estimated, multiple linear regression model coefficients from recursive linear regressions to examine the stability of the coefficients over time. `recreg` supports:

- Nested or rolling-window subsamples of the data
- Estimating coefficients and standard errors using ordinary-least-squares (OLS) or robust methods (HAC and FGLS)

Functionality being removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`vartovec`	Errors	`var2vec`	Replace all existing instances of `vartovec` with the correct `var2vec` syntax.
`vectovar`	Errors	`vec2var`	Replace all existing instances of `vectovar` with the correct `vec2var` syntax.

R2015b

Model Conversion Functions: Convert between VEC models and VAR models

Given known coefficient matrices of the first difference terms in a VEC model, the `vec2var` function returns the autoregressive coefficient matrices of the equivalent VAR model representation.

Given known, autoregressive coefficient matrices of a VAR model, the `var2vec` function returns the coefficient matrices of the first difference and error correction terms of the equivalent VEC model representation.

Diffuse Kalman Filter: Model state-space systems having diffuse initial state distributions

The `dssm` model object applies the diffuse Kalman filter to obtain exact filtered and smoothed states in the presence of states having infinite initial distribution variance.

Specify a diffuse state-space model using `dssm`, identify the diffuse states, and then:

- Estimate its parameters using `estimate`.
- Implement forward recursion of the state-space model using `filter`.
- Implement backward recursion of the state-space model using `smooth`.
- Forecast states and observations using `forecast`.

Chow Structural Change Test: Assess stability of multiple linear regression models

The `chowtest` function implements a variety of Chow tests to assess the stability of coefficients over time in multiple linear regression models. The framework supports both “breakpoint” and “forecast” tests, and the testing of specific coefficient subsets.

ARMAIRF Function: Calculate impulse responses for ARMA models

The `armairf` function filters innovation shocks through each variable in a multivariate autoregressive, moving average model, and returns the response as a numeric array or a impulse response plot.

New Data Set

Econometrics Toolbox includes a new data set containing U.S. food consumption statistics, 1927–1962. Load the data set using `load Data_Consumption`. The reference that analyzes the data is Maddala, G. S. *Introduction to Econometrics.* 2nd Ed., New York, NY: Macmillan, 1992.

Functionality being removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`garchar`	Errors	`arma2ar`	Replace all existing instances of `garchar` with the correct `arma2ar` syntax.
`garchma`	Errors	`arma2ma`	Replace all existing instances of `garchma` with the correct `arma2ma` syntax.
`vartovec`	Warns	`var2vec`	Replace all existing instances of `vartovec` with the correct `var2vec` syntax.
`vectovar`	Warns	`vec2var`	Replace all existing instances of `vectovar` with the correct `vec2var` syntax.

R2015a

Version: 3.2

New Features

Compatibility Considerations

State-space example for Diebold-Li model

The example analyzes yield curves derived from government bond data using the popular Diebold-Li yields-only model in the state-space model framework: Using the Kalman Filter to Estimate and Forecast the Diebold-Li Model.

Autoregressive moving average (ARMA) to AR and MA conversions

Given coefficient values for lagged terms of an ARMA model, the `arma2ar` function returns the coefficients of the truncated AR model, and `arma2ma` function returns the coefficients of the truncated MA model.

Functionality being removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`garchar`	Warns	`arma2ar`	Replace all existing instances of `garchar` with the correct `arma2ar` syntax.
`garchma`	Warns	`arma2ma`	Replace all existing instances of `garchma` with the correct `arma2ma` syntax.

R2014b

Simulation smoothing for state-space models

The `ssm` model object has the method `simsmooth` for sampling from the posterior distribution of the states using forward filtering, backward sampling.

Feasible generalized least squares (FGLS) estimators

The `fgls` function uses generalized least squares (GLS) to estimate coefficients and standard errors in multiple linear regression models with nonspherical errors by first estimating the covariance of the innovations process.

Time-series regression example

The example, following a series of time series regression examples, illustrates how to estimate multiple linear regression models of time series data in the presence of heteroscedastic or autocorrelated (nonspherical) innovations: Time Series Regression X: Generalized Least Squares and HAC Estimators.

Shipped data sets now support tabular arrays

Econometrics Toolbox data sets organize data in tabular arrays rather than `dataset` arrays.

Compatibility Considerations

To access or modify a tabular array, you must use `table` indexing and functions. For details, see Tables.

Functions now support tabular arrays

`hac`, `i10test`, `corrplot`, and `collintest` accept tabular arrays as input arguments. `jcitest` returns tabular arrays.

Compatibility Considerations

To access or modify the output tables of `jcitest`, you must use `table` indexing and functions. For details, see Tables.

R2014a

Version: 3.0

New Features

Compatibility Considerations

Time-invariant and time-varying, linear, Gaussian state-space models

Econometrics Toolbox has a model for performing univariate and multivariate time-series data analysis.

- The `ssm` model supports time-invariant and time-varying, linear state-space models.
- Specify a state-space model using `ssm`, and then:
  - Estimate its parameters using `estimate`.
  - Implement forward recursion of the state-space model using `filter`.
  - Implement backward recursion of the state-space model using `smooth`.
  - Simulate states and observations using `simulate`.
  - Forecast states and observations using `forecast`.

Kalman filter with missing data

The methods of the state-space model, `ssm`, use the Kalman filter to estimate the states, and also use this framework to manage missing data.

Performance enhancements for ARIMA and GARCH models

The `estimate` methods of the `arima`, `egarch`, `garch`, `gjr`, and `regARIMA` models have been enhanced to converge more quickly, and, therefore, you might experience faster estimation durations.

SDE functions moved from Econometrics Toolbox to Financial Toolbox

The following stochastic differential equation (SDE) functions have moved from Econometrics Toolbox to Financial Toolbox™:

- `bm`
- `cev`
- `cir`
- `diffusion`
- `drift`
- `gbm`
- `heston`
- `hwv`
- `interpolate`
- `sde`
- `sdeddo`
- `sdemrd`
- `simByEuler`
- `simBySolution`
- `simulate`

- `cev`
- `ts2func`

Data set and example functions moved from Econometrics Toolbox to Financial Toolbox

The following data set and example functions from the `matlab/toolbox/econ/econdemos` folder have moved to `matlab/toolbox/finance/findemos`:

- `Demo_AmericanBasket`
- `Example_BarrierOption`
- `Example_BlackScholes`
- `Example_CEVModel`
- `Example_CIRModel`
- `Example_CopulaRNG`
- `Example_LongstaffSchwartz`
- `Example_StratifiedRNG`
- `Data_GlobalIdx2.mat`

Functionality being removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`garchcount`	Errors	`sum(any(EstParamCov))`, where `EstParamCov` is an estimated parameter covariance matrix of a fitted `arima`, `garch`, `egarch`, or `gjr` model	N/A
`garchdisp`	Errors	`print` method of the classes	Replace all existing instances of `garchdisp` with the correct `print` syntax.
`garchfit`	Errors	`estimate` method of the classes `arima`, `garch`, `egarch`, or `gjr`	Replace all existing instances of `garchfit` with the correct `estimate` syntax.

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`garchget`	Errors	`arima`, `garch`, `egarch`, or `gjr`	Specify a model using the appropriate model creator `arima`, `garch`, `egarch`, or `gjr`. Use Dot Notation to retrieve parameter values from the model.
`garchinfer`	Errors	`infer` method of the classes `arima`, `garch`, `egarch`, or `gjr`	Replace all existing instances of `garchinfer` with the correct `infer` syntax.
`garchplot`	Errors	N/A	N/A
`garchpred`	Errors	`forecast` method of the classes `arima`, `garch`, `egarch`, or `gjr`	Replace all existing instances of `garchpred` with the correct `forecast` syntax.
`garchset`	Errors	`arima`, `garch`, `egarch`, or `gjr`	Specify a model using the appropriate model creator `arima`, `garch`, `egarch`, or `gjr`. Use Dot Notation to set parameter values for the model.
`garchsim`	Errors	`simulate` method of the classes `arima`, `garch`, `egarch`, or `gjr`	Replace all existing instances of `garchsim` with the correct `simulate` syntax.

R2013b

Version: 2.4

New Features

Compatibility Considerations

Regression models with ARIMA errors

Econometrics Toolbox has a new model for performing time series regression analysis.

- The `regARIMA` model supports linear regression models with ARIMA error processes, including AR, MA, ARMA, and seasonal error models.
- Specify a regression model with ARIMA errors using `regARIMA`, then
  - Estimate its parameters using the data and `estimate`.
  - Simulate responses using `simulate`.
  - Forecast responses using `forecast`.
  - Infer residuals using `infer`.
  - Filter innovations using `filter`.
  - Plot an impulse response using `impulse`.
  - Convert it to an ARIMA model using `arima`.

Time series regression example for lag order selection

The example, following a series of time series regression examples, illustrates predictor history selection for multiple linear regression models: Time Series Regression IX: Lag Order Selection.

optimoptions support

`optimoptions` support when using solver optimization options to:

- Estimate `arima` models using `estimate`.
- Estimate `garch` models using `estimate`.
- Estimate `egarch` models using `estimate`.
- Estimate `gjr` models using `estimate`.

Compatibility Considerations

When estimating `arima`, `garch`, `egarch`, or `gjr` models using `estimate`, the default solver options now reference an `optimoptions` object, instead of an `optimset` structure. If you now use default solver options and operate on them assuming this is an `optimset` structure, some operations might not work.

`optimoptions` is the default and recommended method to set solver options, though `optimset` is also supported.

Estimation display options

The options for the Command Window display of `arima/estimate`, `garch/estimate`, `egarch/estimate`, and `gjr/estimate` is simplified and enhanced. You can easily:

- Display only the maximum likelihood parameter estimates, standard errors, and *t* statistics. This is the new default.

- Display only iterative optimization information.
- Display only optimization diagnostics.
- Display all of the above.
- Turn off all output.

Compatibility Considerations

The new, recommended name-value pair argument that controls the display is `Display`. However, the software still supports the previous name-value pair argument, `print`.

Functionality being removed

R2013a

Heteroscedasticity and autocorrelation consistent (HAC) covariance estimators

The new `hac` function estimates robust covariances for ordinary least squares coefficients of multiple linear regression models under general forms of heteroscedasticity and autocorrelation.

Regression component added to ARIMA models

You can include a regression component to an `arima` model to measure the linear effects that exogenous covariate series have on a response series. This new functionality also enhances `estimate`, `filter`, `forecast`, `infer`, and `simulate`.

Compatibility Considerations

This new `arima` functionality replaces `garchfit`, `garchdisp`, `garchinfer`, `garchget`, `garchset`, `garchpred`, and `garchsim`. Change all instances of those functions using the new `arima` syntax.

Changes to lmctest

`lmctest` uses `estimate` rather than `garchfit` to calculate the MLEs under the alternative hypothesis.

Compatibility Considerations

You might receive slightly different estimates and, in some cases, p-values for the same data under the previous functionality of `lmctest`.

Functionality being removed

R2012b

Version: 2.2

New Features

Impulse response (dynamic multipliers) for ARIMA models

The `arima` model object has a new `impulse` method for generating and plotting impulse response functions for ARIMA models.

Filter user-specified disturbances through ARIMA and conditional variance models

There are new methods to filter user-specified disturbances through ARIMA and conditional variance models:

- `filter` for `arima` model objects to filter disturbances through an ARIMA process.
- `filter` for `garch` model objects to filter disturbances through a GARCH process.
- `filter` for `egarch` model objects to filter disturbances through an EGARCH process.
- `filter` for `gjr` model objects to filter disturbances through a GJR process.

A series of new examples on time-series regression techniques

Eight new examples that illustrate common principles and tasks in time-series regression modeling:

- Time Series Regression I: Linear Models
- Time Series Regression II: Collinearity and Estimator Variance
- Time Series Regression III: Influential Observations
- Time Series Regression IV: Spurious Regression
- Time Series Regression V: Predictor Selection
- Time Series Regression VI: Residual Diagnostics
- Time Series Regression VII: Forecasting
- Time Series Regression VIII: Lagged Variables and OLS Estimator Bias

R2012a

Version: 2.1

New Features

New Model Objects and Their Functions

Econometrics Toolbox has four new model objects for modeling univariate time series data.

- The `arima` model object supports ARIMA processes, including AR, MA, ARMA, and seasonal models.
- For modeling conditionally heteroscedastic series, there are new `garch`, `egarch`, and `gjr` model objects, supporting GARCH models and the EGARCH and GJR variants.

Five new functions for each model object simplify the modeling workflow: `estimate`, `infer`, `forecast`, `print`, and `simulate`.

New Utility Functions

Four new utility functions assist in time series analysis:

- `corrplot` plots predictor correlations.
- `collintest` performs Belsley collinearity diagnostics.
- `i10test` conducts paired integration and stationarity tests.
- `recessionplot` adds recession bands to time series plots.

Demo for Static Time Series Model Specification

A new demo, “Specifying Static Time Series Models,” steps through the model specification workflow for static multiple linear regression models.

Steps include:

- Detecting multicollinearity
- Identifying influential observations
- Testing for spurious regression due to integrated data
- Selecting predictor subsets using stepwise regression and `lasso`
- Conducting residual diagnostics
- Forecasting

The demo uses many tools from Econometrics Toolbox, and introduces new utility functions useful for model specification.

To run the demo in the Command Window, use the command `showdemo Demo_StaticModels`.

New Data Sets

Econometrics Toolbox includes two new data sets:

- **Data_CreditDefaults**. Historical data on investment-grade corporate bond defaults and four predictors, 1984–2004. Data are those used in: Loeffler, G., and P. N. Posch. *Credit Risk Modeling Using Excel and VBA*. West Sussex, England: Wiley Finance, 2007.

- **Data_Recessions**. U.S. recession start and end dates from 1857 to 2011. Source: National Bureau of Economic Research. “U.S. Business Cycle Expansions and Contractions.” `http://www.nber.org/cycles.html`.

R2011b

Version: 2.0.1

New Features

Compatibility Considerations

Warning and Error ID Changes

Many warning and error IDs have changed from their previous versions. These warnings or errors typically appear during a function call.

Compatibility Considerations

If you use warning or error IDs, you might need to change the strings you use. For example, if you turned off a warning for a certain ID, the warning might now appear under a different ID. If you use a `try/catch` statement in your code, replace the old identifier with the new identifier. There is no definitive list of the differences, or of the IDs that changed.

R2011a

Version: 2.0

New Features

New Cointegration Functionality

Econometrics Toolbox now offers functions for cointegration testing and modeling. The `egcitest` function uses Engle-Granger methods to test for individual cointegrating relationships, and estimates their parameters. The `jcitest` function uses Johansen methods to test for multiple cointegrating relationships, and estimates parameters in corresponding vector error-correction models. The `jcontest` function tests linear restrictions on both error-correction speeds and the space of cointegrating vectors, and estimates restricted model parameters.

Convert Vector Autoregressive Models to and from Vector Error-Correction Models

The functions `vectovar` and `vartovec` allow you to convert between vector autoregressive (VAR) models and vector error-correction (VEC) models.

Data Sets for Calibrating Economic Models

Econometrics Toolbox includes three new data sets:

- **Data_Canada**. Mackinnon's data on inflation and interest rates in Canada, 1954–1994. Data are those used in: MacKinnon, J. G. "Numerical Distribution Functions for Unit Root and Cointegration Tests." *Journal of Applied Econometrics*. v. 11, 1996, pp. 601–618.
- **Data_JDanish**, **Data_JAustralian**. Johansen's data on money and income in Denmark, 1974–1987, and Australia/U.S. purchasing power and interest parity, 1972–1991. Data are those used in: Johansen, *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. Oxford: Oxford University Press, 1995.

R2010b

Version: 1.4

New Features

Compatibility Considerations

Functions Being Removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`dfARDTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfARDTest` with the correct `adftest` syntax.
`dfARTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfARTest` with the correct `adftest` syntax.
`dfTSTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfTSTest` with the correct `adftest` syntax.
`ppARDTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppARDTest` with the correct `pptest` syntax.
`ppARTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppARTest` with the correct `pptest` syntax.
`ppTSTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppTSTest` with the correct `pptest` syntax.

Additional Syntax Options for archtest and lbqtest

The functions `archtest` and `lbqtest` now take name-value pair arguments as inputs. The old syntax of individual arguments will continue to work but will not be documented.

New Data Set for Calibrating Economic Models

The economic data from the paper by Nielsen and Risager, “Stock Returns and Bond Yields in Denmark, 1922–99,” (Department of Economics, Copenhagen Business School; Working paper 3-2001, 2001) is now included with Econometrics Toolbox in the file `Data_Danish`.

R2010a

Version: 1.3

New Features

Compatibility Considerations

Functions Being Removed

Function Name	What Happens When You Use This Function?	Use This Function Instead	Compatibility Considerations
`dfARDTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfARDTest` with the correct `adftest` syntax.
`dfARTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfARTest` with the correct `adftest` syntax.
`dfTSTest`	Error	`adftest`	The new function syntax differs. Replace all existing instances of `dfTSTest` with the correct `adftest` syntax.
`ppARDTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppARDTest` with the correct `pptest` syntax.
`ppARTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppARTest` with the correct `pptest` syntax.
`ppTSTest`	Error	`pptest`	The new function syntax differs. Replace all existing instances of `ppTSTest` with the correct `pptest` syntax.

Demo Showing Multivariate Modeling of the U.S. Economy

A new demo, “Modeling the United States Economy,” develops a small macroeconomic model. This model is used to examine the impact of various shocks on the United States economy, particularly around the period of the 2008 fiscal crisis. It uses the multiple time series tools from the Econometrics Toolbox.

To run the demo in the command window, use the command `echodemo Demo_USEconModel`.

Lag Operator Polynomial Objects

The new `LagOp` polynomial class provides methods to create and manipulate lag operator polynomials and filter time series data, as well as methods to perform polynomial algebra including addition, subtraction, multiplication, and division.

Leybourne-McCabe Test for Stationarity

The new Leybourne-McCabe test function `lmctest` assesses the null hypothesis that a univariate time series *y* is a trend-stationary AR(*p*) process against the alternative that *y* is a nonstationary ARIMA(*p*,1,1) process.

Historical Data Sets for Calibrating Economic Models

The new data set `Data_SchwertMacro` contains original data from G. William Schwert's article "Effects of Model Specification on Tests for Unit Roots in Macroeconomic Data," (*Journal of Monetary Economics*, Vol. 20, 1987, pp. 73–103.). These data are a benchmark for unit root tests. The new data set `Data_SchwertStock` contains indices of U.S. stock prices as published in G. William Schwert's article "Indexes of U.S. Stock Prices from 1802 to 1987," (*The Journal of Business*,Vol. 63, 1990, pp. 399–42.). The new data set `Data_USEconModel`contains the macroeconomic series for the new demo `Demo_USEconModel`.

New Organization and Naming Standard for Data Sets

Econometrics Toolbox has a new set of naming conventions for data sets. Data set names are prefixed by `Data_`.

For full information on the available data sets, demos, and examples, see Data Sets, Demos, and Example Functions or type `help econ/econdemos` at the command line. For more information on Dataset Array objects, see `dataset` in the Statistics Toolbox™ documentation.

Compatibility Considerations

Replace any instances of `load Old_Data` with `load` and the new file name.

New Naming Convention for Demos and Example Functions

All demos and examples in the Econometrics Toolbox have been moved to the folder `econ/econdemos` and renamed according to the following convention:

- Demos are named `Demo_DemoName`
- Examples are named `Example_ExampleName`

Compatibility Considerations

Replace any instances of example functions with their new names. For full information on the available, demos, and examples, see Data Sets, Demos, and Example Functions or type `help econ/econdemos` at the command line.

R2009b

Version: 1.2

New Features

Compatibility Considerations

Unit Root Tests

There are now four classes of unit root tests. More information on the tests is available in the Unit Root Nonstationarity section of the User's Guide.

Dickey-Fuller and Phillips-Perron Tests

Dickey-Fuller and Phillips-Perron tests now have single interfaces, with new capabilities for multiple testing. Both `adftest` and `pptest` test a unit root null hypothesis against autoregressive, autoregressive with drift, or trend-stationary alternatives.

KPSS Test

The new `kpsstest` function tests a null hypothesis of (trend) stationarity against nonstationary unit root alternatives.

Variance Ratio Test

The new `vratiotest` function tests a null hypothesis of a random walk against alternatives with innovations that are not independent and identically distributed.

Compatibility Considerations

The `ardtest` function replaces the `dfARDTest`, `dfARTest`, and `dfTSTest` functions. The `pptest` function replaces the `ppARDTest`, `ppARTest`, and `ppTSTest` functions. The new function syntax differs from the functions they replace.

Financial Toolbox Required

Econometrics Toolbox requires Financial Toolbox as of this version.

Nelson-Plosser Data

The Nelson and Plosser [50] data set is now available. To access the data, enter `load Data_NelsonPlosser` at the MATLAB command line.

R2009a

Version: 1.1

New Features

Compatibility Considerations

Hypothesis Tests

There are two new hypothesis tests for model misspecification:

- Lagrange Multiplier tests, `lmtest`
- Wald tests, `waldtest`

Furthermore, the likelihood ratio test, `lratiotest`, has been enhanced to be able to “test up” as well as “test down” when performing multiple model comparisons. It now accepts vectors of model parameters for restricted log likelihoods, for unrestricted log likelihoods, or for both.

There is a new demo about these tests; see “New Demo” on page 28-2.

Compatibility Considerations

`lratiotest` error messages and message IDs differ from previous versions.

Structural VAR, VARX, and VARMAX models

Econometrics Toolbox multiple time series functions now include structural multiple time series. Structural models have the general form

$$A_0 Y_t = a + X_t b + \sum_{i=1}^{p} A_i Y_{t-i} + \sum_{j=1}^{q} B_j W_{t-j} + B_0 W_t .$$

Previously, Econometrics Toolbox multiple time series functions addressed models of the form

$$Y_t = a + X_t b + \sum_{i=1}^{p} A_i Y_{t-i} + \sum_{j=1}^{q} B_i W_{t-j} + W_t .$$

The mathematical difference is the inclusion of A_0 and B_0 matrices. These matrices allow practitioners to specify structural dependencies between variables. For more information, see the Multivariate Time Series Models chapter of the Econometrics Toolbox User's Guide.

Compatibility Considerations

Objects created with the Econometrics Toolbox V1.0 `vgxset` function, and saved in MAT files, do not work with Econometrics Toolbox V1.1 functions. Recreate the objects with the Econometrics Toolbox V1.1 `vgxset` function.

New Demo

There is a new demo on hypothesis tests. Run the demo at the MATLAB command line by entering `showdemo classicalTestsDemo`.

R2008b

Version: 1.0

New Features

Multivariate VAR, VARX, and VARMA Models

A new suite of functions, listed in the following table, adds support for multivariate VAR, VARX, and VARMA models.

Function	Description
`vgxar`	Convert VARMA specification into a pure vector autoregressive (VAR) model
`vgxcount`	Count restricted and unrestricted parameters in VAR or VARX models
`vgxdisp`	Display VGX model parameters and standard errors in different formats
`vgxget`	Get multivariate time-series specification parameters
`vgxinfer`	Infer innovations of a VGX process
`vgxloglik`	Compute conditional log-likelihoods of VGX process
`vgxma`	Convert VARMA specification into a pure vector moving average (VMA) model
`vgxplot`	Plot multivariate time series process
`vgxpred`	Generate transient response of VGX process during a specified forecast period
`vgxproc`	Generate a VGX process from an innovations process
`vgxqual`	Determine if a VGX process is stable and invertible
`vgxset`	Set or modify multivariate time-series specification parameters
`vgxsim`	Simulate VGX processes
`vgxvarx`	Solve VAR or VARX model using maximum likelihood estimation

Heston Stochastic Volatility Models

The new heston function adds support for Heston stochastic volatility models to the SDE engine.

R2008a

Version: 2.4

New Features

Monte Carlo Simulation of Stochastic Differential Equations

The GARCH Toolbox™ software now allows you to model dependent financial and economic variables, such as interest rates and equity prices, via Monte Carlo simulation of multivariate diffusion processes. For more information, see Stochastic Differential Equations in the GARCH Toolbox documentation.

R2007b

Version: 2.3.2

New Features

Changes to garchsim

The `garchsim` function previously allowed you to specify the `State` argument as either a scalar or a time series matrix of standardized, independent, identically distributed disturbances to drive the output `Innovations` in a time series process. The `State` argument must now be a time series matrix. See the `State` input argument on the `garchsim` reference page for more information.

R2007a

Version: 2.3.1

No New Features or Changes

R2006b

Version: 2.3

New Features

Data Preprocessing

A new Hodrick-Prescott filter, `hpfilter`, separates time series into trend and cyclical components

Demos

A new demo uses the `hpfilter` function to reproduce the results in Hodrick and Prescott's original paper on U.S. business cycles

R2006a

Version: 2.2

New Features

User's Guide

A new chapter in the *GARCH Toolbox User's Guide* explains how to conduct Dickey-Fuller and Phillips-Perron unit root tests with the new statistical functions in the toolbox.

Statistical Functions

Version 2.2 of the GARCH Toolbox software has six new functions. All of them support the ability to conduct univariate unit root tests on time series data. Three functions support augmented Dickey-Fuller unit root tests. The remaining three support Phillips-Perron unit root tests.

Dickey-Fuller Unit Root Tests

Function	Purpose
`dfARDTest`	Augmented Dickey-Fuller unit root test based on AR model with drift.
`dfARTest`	Augmented Dickey-Fuller unit root test based on zero drift AR model.
`dfTSTest`	Augmented Dickey-Fuller unit root test based on trend stationary AR model.

Phillips-Perron Unit Root Tests

Function	Purpose
`ppARDTest`	Phillips-Perron unit root test based on AR(1) model with drift.
`ppARTest`	Phillips-Perron unit root test based on zero drift AR(1) model.
`ppTSTest`	Phillips-Perron unit root test based on trend stationary AR(1) model.

R14SP3

Version: 2.1

New Features

Compatibility Considerations

Changes to garchsim

A change introduced in V2.1 of the GARCH Toolbox software concerns user-specified noise processes. The `garchsim` function now allows you to provide a time series matrix of standardized, i.i.d. disturbances to drive the output `Innovations` in a time series process. In previous versions, you could only provide a state that was used to generate a random noise process. See the `State` input argument on the `garchsim` reference page for more information.

Compatibility Considerations

garchsim argument Is renamed. In V2.1, the `garchsim` argument `Seed` is renamed to `State` for consistency with the MATLAB `rand` and `randn` functions. The name change, in itself, introduces no backward incompatibilities. The following topic explains a related change.

garchsim defaults to current random number generator state. In V2.0.1 of the GARCH Toolbox software, the `garchsim` function used the initial random number generator state, `0`, if you did not specify a value for the `Seed` argument. The `Seed` argument corresponded to the `rand` and `randn` state value.

In V2.1, if you do not specify a value for the `State` (formerly `Seed`) argument, `garchsim` uses the current state of `rand` and `randn`, rather than the initial state. Use the commands `s = rand('state')` and `s = randn('state')` to determine the current state of these random number generators. For more information, see the `rand` and `randn` reference pages.