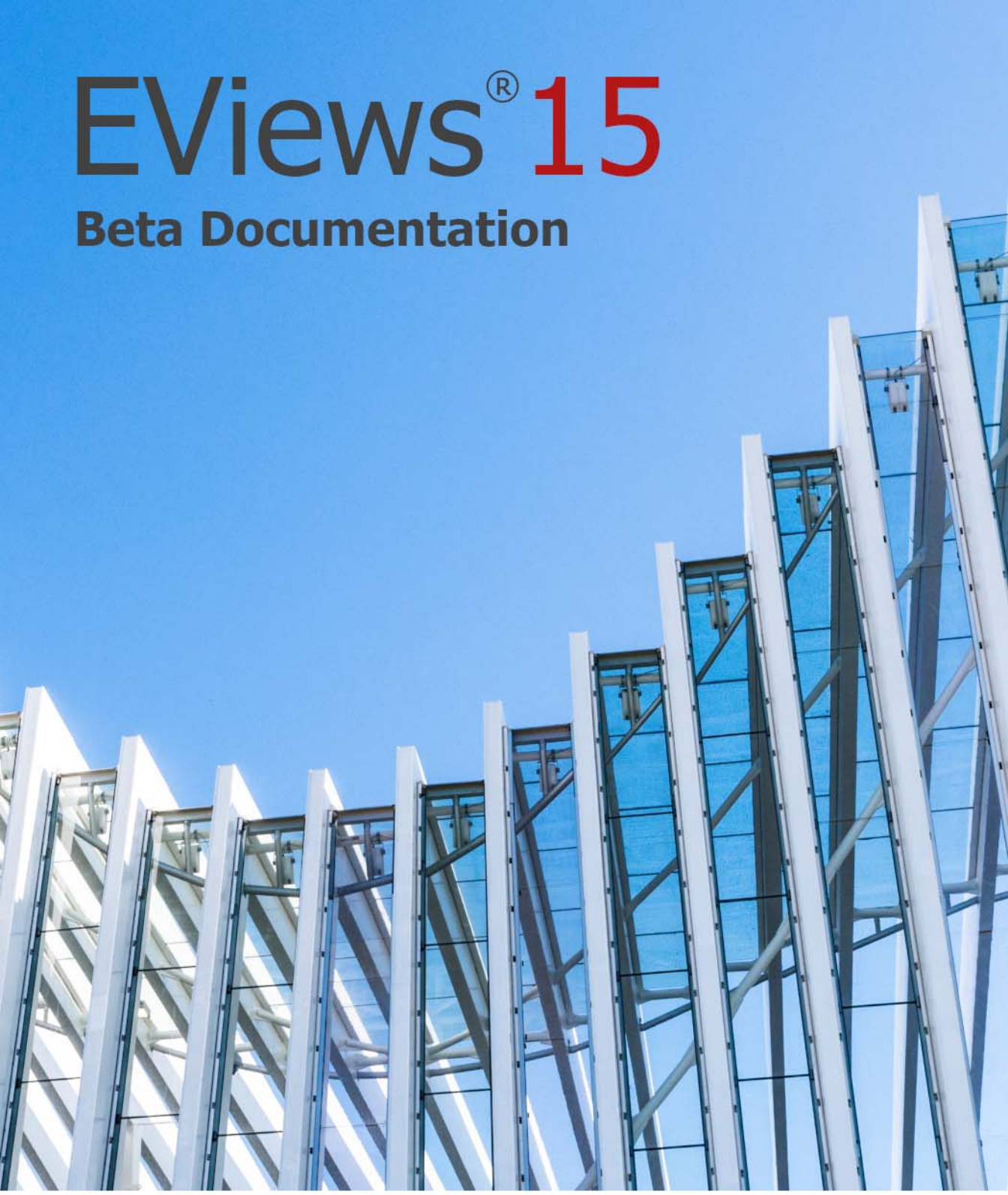
EViews® 15

Beta Documentation



S&P Global

EViews® 15 Beta Documentation

S&P Global

EViews® 15 Beta Documentation

Disclaimer

The authors and S&P Global Inc. assume no responsibility for any errors that may appear in this manual or the EViews program. The user assumes all responsibility for the selection of the program to achieve intended results, and for the installation, use, and results obtained from the program.

Trademarks

EViews® is a registered trademark of S&P Global Inc. Windows, Excel, PowerPoint, and Access are registered trademarks of Microsoft Corporation. PostScript is a trademark of Adobe Corporation. Bloomberg is a trademark of Bloomberg Finance L.P. All other product names mentioned in this manual may be trademarks or registered trademarks of their respective companies.

Third Party Licenses

This section contains third party notices or additional terms and conditions applicable to certain software technologies which may be used in one or more EViews products and/or services. Please be sure to consult the individual product files, about box, and/or install or manual documentation for specific copyright notices and author attributions. Notices on this page are current for EViews products released on or after October 1, 2017.

- diff template Library - Copyright © 2015 Tatsuhiko Kubo cubicdaiya@gmail.com. All rights reserved.
- FFmpeg Library - FFmpeg is a trademark of Fabrice Bellard, originator of the FFmpeg project.
- GZipHelper - Copyright © 1995-2002 Gao Dasheng dsgao@hotmail.com.
- Java SE Runtime Environment v1.8.0_401 licensed under Oracle Binary Code License Agreement.
- JDemetra + v3 licensed under European Union Public License v1.2.
- jsonCPP Library - Copyright © 2007-2010 Baptiste Lepilleur and The JsonCPP Authors.
- openssl Library - Copyright © 1998-2016 The OpenSSL Project. All rights reserved.
- libcurl Library - Copyright © 1996-2013, Daniel Stenberg daniel@haxx.se.
- libharu Library - Copyright © 2000-2006 Takeshi Kanno, Copyright © 2007-2009 Antony Dovgal et all.

S&P Global Inc.
Telephone: (949) 856-3368
e-mail: sales@eviews.com
web: www.eviews.com
August 3, 2026

Table of Contents

Chapter 1. EViews 15 Beta Documentation

The EViews development team is pleased to announce that EViews 15 is now available for beta testing.

The Beta Version will not run unless you already have EViews 14 installed and registered on your computer. In addition, this release of the Beta Version is designed for limited time use, and will stop running after the expiration date displayed when you run the program. We will continue to provide regular updates and bug fixes for the program and documentation up through the final release of EViews 15.

You may send us email with comments, suggestions, and bug reports at:

```
support@eviews.com
```

If you experience any problems, or have any suggestions, we would very much like to hear from you. To report problems, please describe as completely as possible the nature of the problem, including the sequence of operations you were carrying out, and any messages provided. Please include your EViews 15 Beta build-date in any correspondence. You can always tell the build date of your copy of EViews by selecting **Help/About EViews** from the main menu.

Beta Documentation Notes

This version of the documentation is dated August 3, 2026.

This document contains very rough, preliminary documentation for some (not all) of the new features in EViews 15.

Note that this document is, in part, an extract from the full document so that portions may not be formatted properly, the index is incomplete, and cross-reference links to pages and sections may fail.

Despite all of that, you should feel free to comment on the documentation; in particular, let us know if you find portions to be unclear or in error.

What's New in EViews 15?

EViews 15 features a wide range of exciting changes and improvements. What follows is a list of some of the new features. Note that items may appear in more than one category. ***This is not a complete list of the new features in EViews 15***.

Data Handling

Seasonal Adjustment

- Spline Decomposition Seasonal Adjustment ("Spline Decomposition Seasonal Adjustment" on page 3).

Econometrics and Statistics

Estimation and Analysis

- Spline Regression ("Spline Regression" on page 5).
- Regression Discontinuity in Time ("Regression Discontinuity in Time" on page 7).
- Model Averaging ("Model Averaging" on page 8).
- Elastic-net GLM ("Elastic-net GLM" on page 14).
- Factor-Augmented VAR ("Factor-Augmented VAR" on page 15).
- Enhancements to Bayesian VAR ("Enhancements to BVAR" on page 15).

Forecasting

- Enhanced Facebook Prophet ("Enhanced Prophet" on page 20).
- NeuralProphet ("NeuralProphet" on page 20).
- Long Short-Term Memory (LSTM) Models ("LSTM Models" on page 21).
- AutoGluon ("AutoGluon" on page 21).
- Lag-Llama ("Lag-Llama" on page 22).
- Random Forests and Decision Trees ("Random Forests and Decision Trees" on page 23).
- Chronos-2 ("Chronos-2" on page 24).
- Enhanced ETS Smoothing ("Enhanced ETS Smoothing" on page 24).

Preliminary Updates to Command Reference

- Preliminary list of new and updated commands (Chapter 4. "Preliminary Updates to Command Reference," on page 25).

Chapter 2. Data Handling

EViews 15 offers a new method for seasonal adjustment.

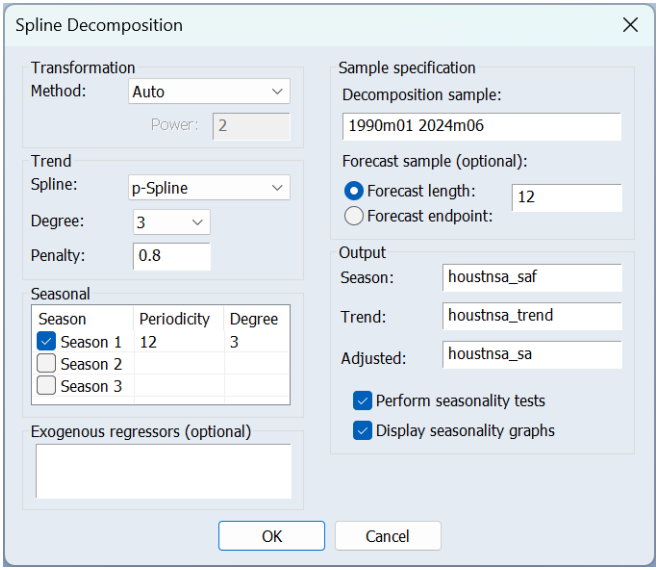
Seasonal Adjustment

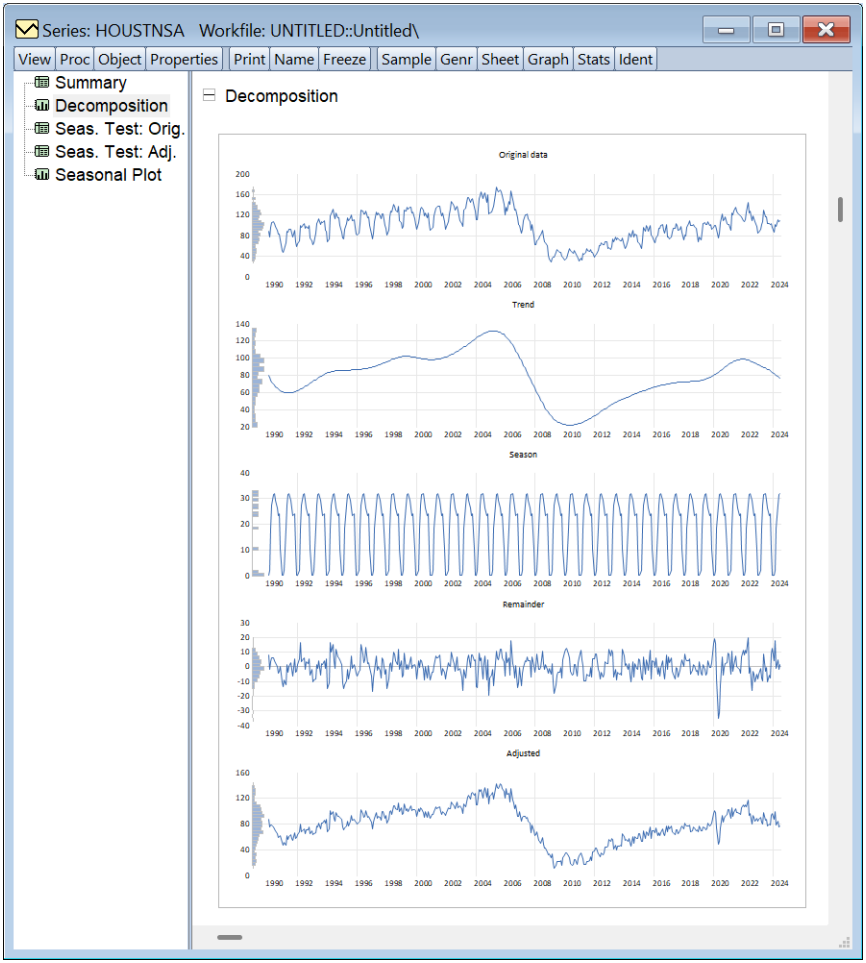
- Spline Decomposition Seasonal Adjustment ("Spline Decomposition Seasonal Adjustment" on page 3).

Spline Decomposition Seasonal Adjustment

EViews 15 adds a new seasonal adjustment method capable of modeling multiple seasonalities for any frequency of data using spline regressions. The advantage of a spline decomposition over traditional seasonal adjustment methods is that it requires few assumptions on the structure of the time series and does not require strict parameterization.

EViews allows specification of either a p-Spline, a bSpline, or a polynomial spline for the trend component of the decomposition, and then up to three cyclical splines to model the seasonal components. You may optionally add further linear regressors to the decomposition to model holiday or one-off effects.





New or Updated Commands

splinedecomp ..... perform seasonal adjustment of a series using a spline decomposition (p. 74). *(new)*

Chapter 3. Econometrics and Statistics

EViews 15 offers a variety of additions and improvements to our set of econometric and statistical features. The following is a brief outline of the most important new features, followed by discussion and links to preliminary documentation.

Estimation and Analysis

- Spline Regression ("Spline Regression" on page 5).
- Regression Discontinuity in Time ("Regression Discontinuity in Time" on page 7).
- Model Averaging ("Model Averaging" on page 8).
- Elastic-net GLM ("Elastic-net GLM" on page 14).
- Factor-Augmented VAR ("Factor-Augmented VAR" on page 15).
- Enhancements to Bayesian VAR ("Enhancements to BVAR" on page 15).

Forecasting

- Enhanced Facebook Prophet ("Enhanced Prophet" on page 20).
- NeuralProphet ("NeuralProphet" on page 20).
- Long Short-Term Memory (LSTM) Models ("LSTM Models" on page 21).
- AutoGluon ("AutoGluon" on page 21).
- Lag-Llama ("Lag-Llama" on page 22).
- Random Forests and Decision Trees ("Random Forests and Decision Trees" on page 23).
- Chronos-2 ("Chronos-2" on page 24).
- Enhanced ETS Smoothing ("Enhanced ETS Smoothing" on page 24).

Spline Regression

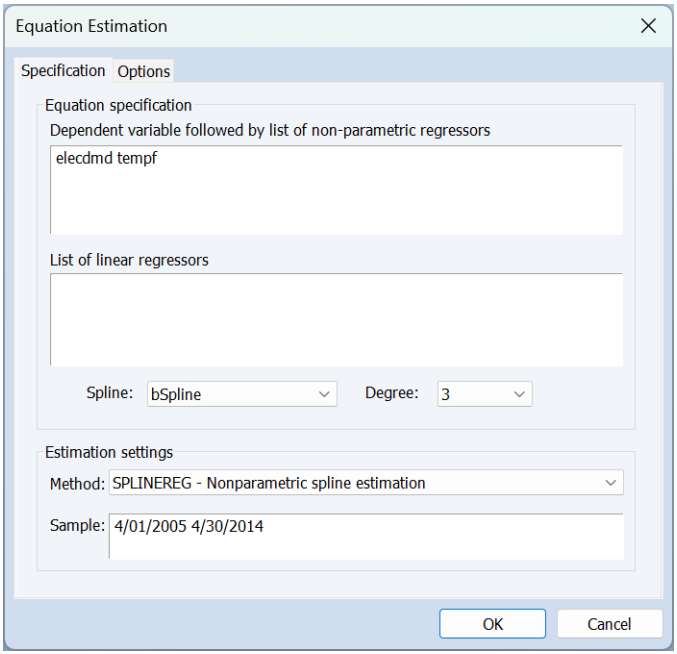
Spline regression is a powerful technique for modeling nonlinear relationships by fitting piecewise polynomial curves that join smoothly at predetermined points called knots.

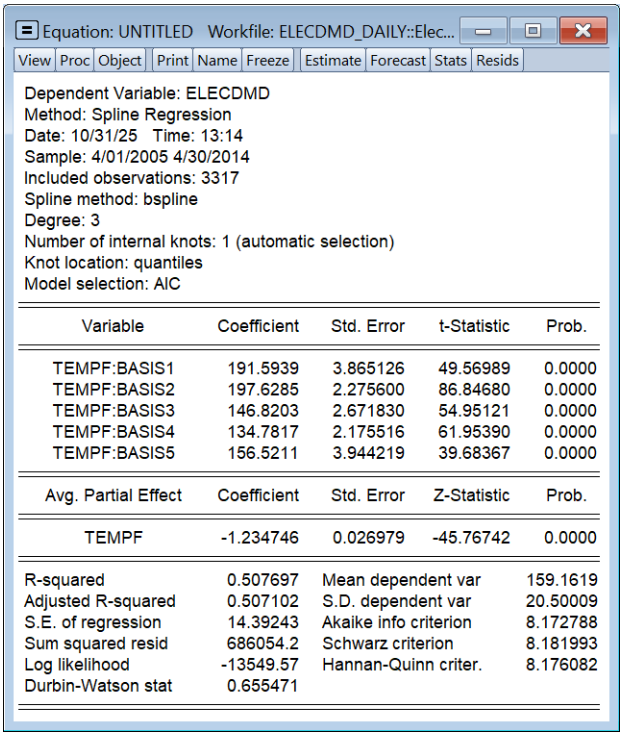
EViews 15 introduces three spline options:

- B-Splines: A set of basis functions that are nonzero over limited subintervals, ensuring numerical stability and local control.
- Polynomial Splines: Piecewise polynomials joined at knots with specified continuity constraints on derivatives.

- P-Splines (Penalized Splines): Polynomial splines augmented with a roughness penalty on the differences of adjacent spline coefficients to control smoothness.

EViews optionally allows estimation of spline regression including variables treated linearly.





Equation: UNTITLED Workfile: ELECDMD_DAILY::Elec...

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: ELECDMD
Method: Spline Regression
Date: 10/31/25 Time: 13:14
Sample: 4/01/2005 4/30/2014
Included observations: 3317
Spline method: bspline
Degree: 3
Number of internal knots: 1 (automatic selection)
Knot location: quantiles
Model selection: AIC

Variable	Coefficient	Std. Error	t-Statistic	Prob.
TEMPF:BASIS1	191.5939	3.865126	49.56989	0.0000
TEMPF:BASIS2	197.6285	2.275600	86.84680	0.0000
TEMPF:BASIS3	146.8203	2.671830	54.95121	0.0000
TEMPF:BASIS4	134.7817	2.175516	61.95390	0.0000
TEMPF:BASIS5	156.5211	3.944219	39.68367	0.0000

Avg. Partial Effect	Coefficient	Std. Error	Z-Statistic	Prob.
TEMPF	-1.234746	0.026979	-45.76742	0.0000

R-squared	0.507697	Mean dependent var	159.1619
Adjusted R-squared	0.507102	S.D. dependent var	20.50009
S.E. of regression	14.39243	Akaike info criterion	8.172788
Sum squared resid	686054.2	Schwarz criterion	8.181993
Log likelihood	-13549.57	Hannan-Quinn criter.	8.176082
Durbin-Watson stat	0.655471		

The output of a spline regression includes the coefficients on the spline basis matrix columns, as well as the average partial effects of the underlying variables. A full table of partial effects for each independent variable is also available from the **View** menu.

New or Updated Commands

splinereg.....estimate an equation using nonparametric splines (p. 76). *(new)*

knots.....view a table of the knot values from an equation estimated with splines (p. 61). *(new)*

partialeffects.....view a graph of the partial effects from an equation estimated with splines (p. 66). *(new)*

Regression Discontinuity in Time

Regression Discontinuity in Time (RDiT) is a quasi-experimental method for estimating the causal effect of an intervention that occurs at a known point in time. It treats time relative to the intervention date as the running variable and looks for a discontinuous jump in the outcome at that cutoff, comparing observations just before and just after. The key identifying assumption is that, absent the intervention, the outcome would have evolved smoothly through the cutoff, so any sharp break can be attributed to the treatment.

A key advantage of RDiT is that it focuses only on observations local to the intervention cut-off, so it does not require the underlying relationship to be stable over the entire sample the way traditional structural break models do.

EViews 15 introduces RDiT following the local polynomial regression discontinuity framework of Calonico, Cattaneo, and Titiunik (2014a-c, 2015) and its extensions to time-based running variables.

New or Updated Commands

rdit..... estimate a regression discontinuity in time model. (p. 72). *(new)*

balance ..... view the covariate balance chart of a regression discontinuity in time model estimated with covariates (p. 32). *(new)*

bwsensitivity..... view the bandwidth sensitivity chart of a regression discontinuity in time model (p. 35). *(new)*

discontinuity ..... view the discontinuity chart of a regression discontinuity in time model (p. 38). *(new)*

donutsensitivity.... view the donut sensitivity chart of a regression discontinuity in time model (p. 38). *(new)*

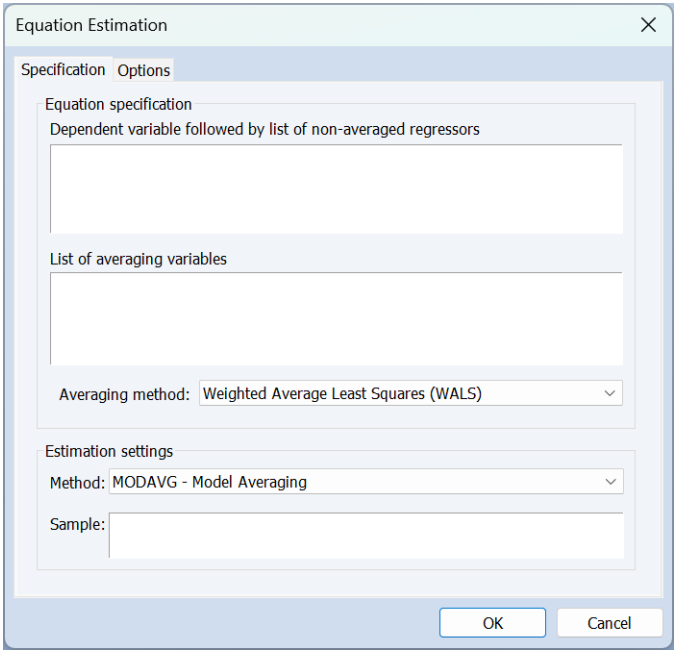
placebo ..... view the placebo chart of a regression discontinuity in time model (p. 66). *(new)*

Model Averaging

Model averaging is an estimation technique that aims to address model uncertainty by averaging inference over many models. Model weights are used in the averaging so that models supported by the data have more influence than those not supported by the data.

EViews 15 offers the model averaging methods of Magnus et al. (2010): Weighted Average Least Squares (WALS) and Bayesian Model Averaging (BMA). WALS is a classical estimation method that uses model weights that approximate posterior model probabilities. BMA is a fully Bayesian technique that uses posterior model probabilities for model weights. For an in-depth coverage of the topic, see Chapter 9. “Model Averaging,” beginning on page 289.

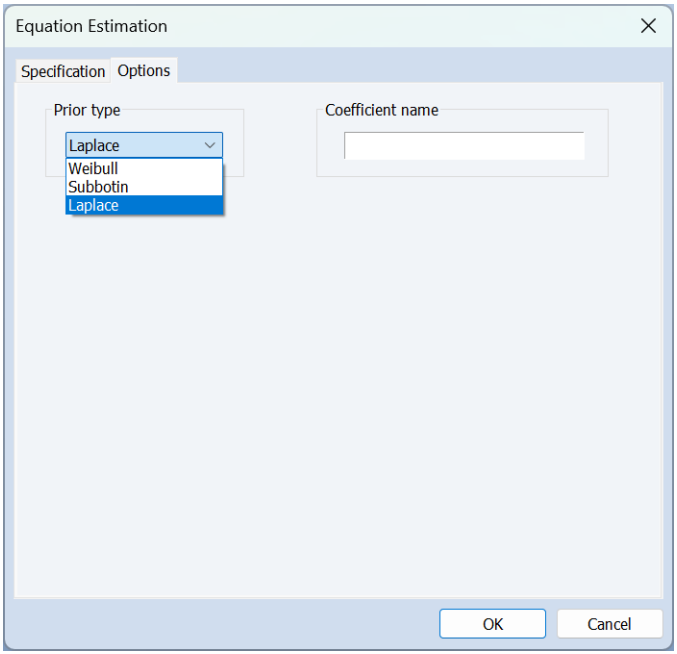
To estimate an equation using model averaging in EViews, go to **Quick/Estimate Equation…** or type `equation` in the command window and hit Enter. In the **Specification** page, select **MODAVG - Model Averaging** from the **Method:** drop-down menu. EViews will display the following dialog:



There are two edit fields within the **Equation specification** group box. Enter the name of the dependent variable, followed by the names of the required regressors, in the first edit field. Enter the names of the optional regressors in the second edit field.

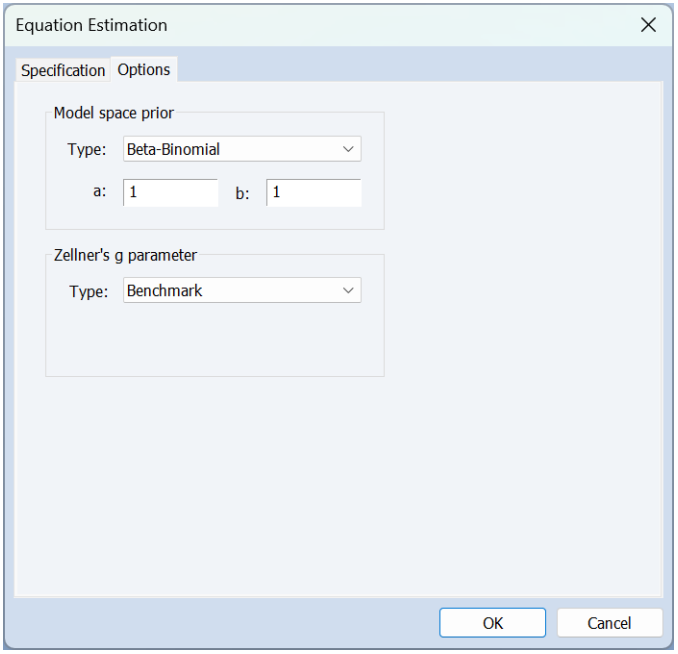
Use the **Averaging method:** dropdown menu to select between **Weighted Average Least Squares (WALS)** and **Bayesian Model Averaging (BMA)**. The averaging method will determine what options are available to you in the **Options** page.

If the averaging method is set to WALS, the **Options** page lets you specify the approximate distribution of the model weights:



See "Weighted Average Least Squares (WALS)," beginning on page 298 for details.

If the averaging method is set to BMA, the **Options** page lets you specify the prior distribution over the model space and model parameters:



See "Bayesian Model Averaging (BMA)," beginning on page 299 for details.

In what follows, $\{M_\gamma\}$ denotes a set of candidate models.

WALS

WALS forms an averaged estimator by weighting the OLS estimator from M_γ:

$$\hat{\beta}^{\text{WALS}} = \sum_{\gamma \in \{0,1\}^{k_2}} w_\gamma \hat{\beta}_\gamma, \qquad \sum_\gamma w_\gamma = 1, \qquad w_\gamma \geq 0.$$

Magnus (2010) shows that the optimal weights can be calculated through an orthogonalization of the auxiliary regressors to both each other and the focus regressors; see "Weighted Average Least Squares (WALS)," beginning on page 298 for details.

BMA

BMA is based upon the formula

$$\pi(\Delta \mid y) = \sum_\gamma \pi(\Delta \mid y, M_\gamma) \Pr(M_\gamma \mid y)$$

where $\pi(\Delta \mid y)$ is the posterior distribution over some quantity of interest, Δ, marginally of all candidate models, $\pi(\Delta \mid y, M_\gamma)$ is the posterior distribution over Δ conditionally on

model M_γ, and $\Pr(M_\gamma|y)$ is the posterior model probability for M_γ. Common choices for Δ include the set of common parameters θ, as well as predicted values y^*.

The BMA methodology of Magnus et al. (2010) applies the formula above to linear regression. For further detail, see "Bayesian Model Averaging (BMA)," beginning on page 299.

Example

Here we reproduce WALS and BMA results from Magnus et al. (2010). For the complete example, see "Examples," beginning on page 294.

WALS

Go to **Quick/Estimate Equation…** and select **MODAVG - Model Averaging** from the **Method:** dropdown menu. In the **Specification** page, enter

```
growth c gdp60 equipinv school60 life60 dpop
```

in the first edit field and

```
law tropics avelf confuc
```

in the second. Click **OK** to get

Equation: MYEQ Workfile: DATA::Data\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: GROWTH
Method: Model Averaging
Date: 12/09/25 Time: 11:24
Sample: 1 74
Included observations: 74
Averaging Method: Weighted Average Least Squares (WALS)
Number of required variables: 6
Number of optional variables: 4

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.061751	0.021791	2.833819	0.0061
GDP60	-0.015650	0.003144	-4.977897	0.0000
EQUIPINV	0.158213	0.054421	2.907201	0.0050
SCHOOL60	0.016676	0.009667	1.725019	0.0894
LIFE60	0.000851	0.000350	2.429667	0.0179
DPOP	0.271387	0.242528	1.118990	0.2673
LAW	0.013411	0.005804	2.310705	0.0241
TROPICS	-0.005997	0.003456	-1.735518	0.0875
AVELF	-0.007676	0.005066	-1.515223	0.1346
CONFUC	0.046455	0.014277	3.253943	0.0018

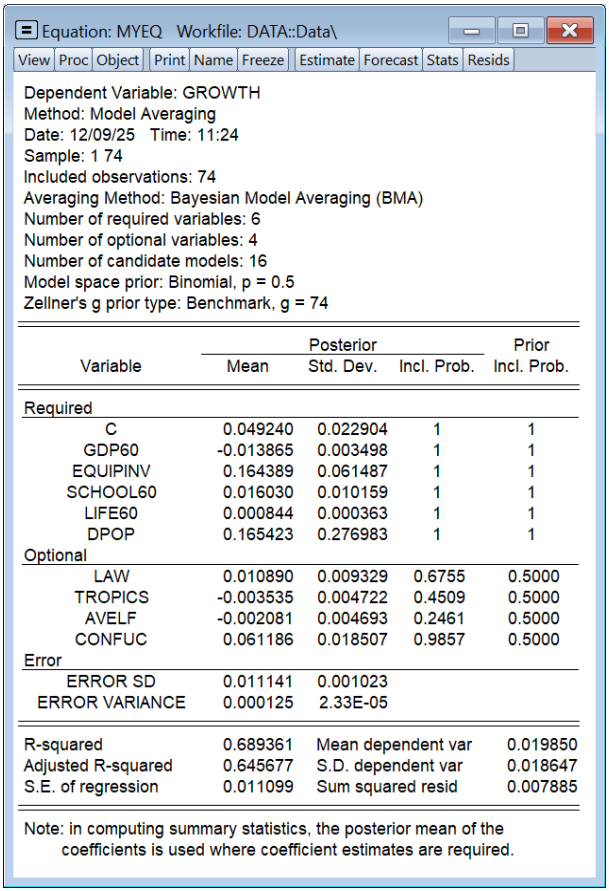
R-squared	0.698238	Mean dependent var	0.019850
Adjusted R-squared	0.655803	S.D. dependent var	0.018647
S.E. of regression	0.010940	Sum squared resid	0.007659
Durbin-Watson stat	1.668501		

These results match the output produced by code accompanying Magnus et al. (2010).

BMA

Continuing with the previous example, click on the **Estimate** button. From the **Averaging method:** dropdown menu, select **Bayesian Model Averaging (BMA)**.

Click on the **Options** tab. Set the model space prior type to **Binomial**. Click **OK**.



Equation: MYEQ Workfile: DATA::Data\

View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids

Dependent Variable: GROWTH
Method: Model Averaging
Date: 12/09/25 Time: 11:24
Sample: 1 74
Included observations: 74
Averaging Method: Bayesian Model Averaging (BMA)
Number of required variables: 6
Number of optional variables: 4
Number of candidate models: 16
Model space prior: Binomial, p = 0.5
Zellner's g prior type: Benchmark, g = 74

Variable	Posterior Mean	Posterior Std. Dev.	Posterior Incl. Prob.	Prior Incl. Prob.
Required				
C	0.049240	0.022904	1	1
GDP60	-0.013865	0.003498	1	1
EQUIPINV	0.164389	0.061487	1	1
SCHOOL60	0.016030	0.010159	1	1
LIFE60	0.000844	0.000363	1	1
DPOP	0.165423	0.276983	1	1
Optional				
LAW	0.010890	0.009329	0.6755	0.5000
TROPICS	-0.003535	0.004722	0.4509	0.5000
AVELF	-0.002081	0.004693	0.2461	0.5000
CONFUC	0.061186	0.018507	0.9857	0.5000
Error				
ERROR SD	0.011141	0.001023		
ERROR VARIANCE	0.000125	2.33E-05		

R-squared	0.689361	Mean dependent var	0.019850
Adjusted R-squared	0.645677	S.D. dependent var	0.018647
S.E. of regression	0.011099	Sum squared resid	0.007885

Note: in computing summary statistics, the posterior mean of the coefficients is used where coefficient estimates are required.

The first two columns under the "Posterior" header show posterior means and posterior standard deviations on the coefficients. These values match those reported in the last column of Table 2 in Magnus et al. (2010).

The third column under the "Posterior" header shows posterior inclusion probabilities (PIP). This column matches the second column in the body of Table 6 in Magnus et al. (2010).

New or Updated Commands

modavg ..... estimate equation using weighted-average least squares (WALS) or Bayesian model averaging (BMA) (p. 61). *(new)*

modelranking ..... display models ranked by posterior probability (p. 62). *(new)*

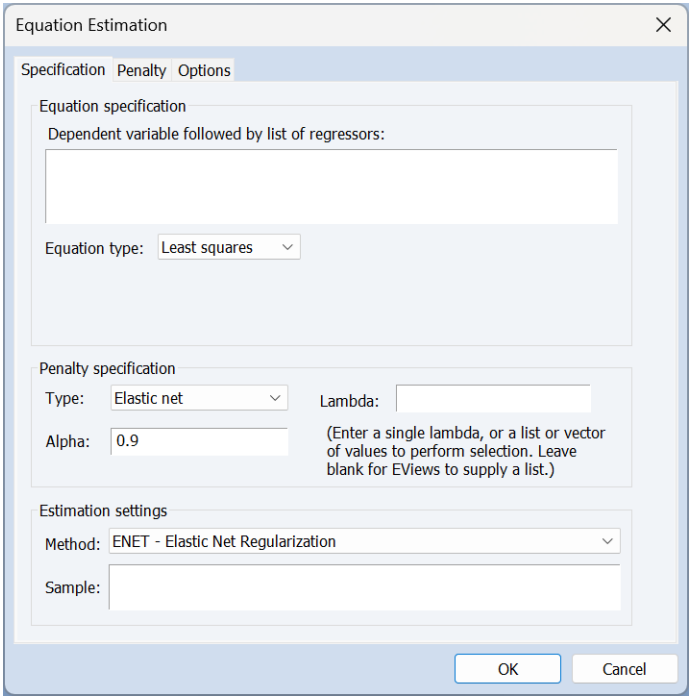
modelsize ..... display results pertaining to the size of the model (p. 63). *(new)*

postmarginals ..... display the univariate marginals of the posterior distribution over parameters (p. 68). *(new)*

Elastic-net GLM

EViews 15 expands upon existing elastic-net estimation by including generalized linear models (GLM) as part of the enet toolkit.

Elastic-net GLM extends regularized modeling beyond continuous outcomes, allowing the same penalized estimation framework to be applied to binary, count, proportion, and other non-Gaussian response variables through appropriate distribution and link-function choices.

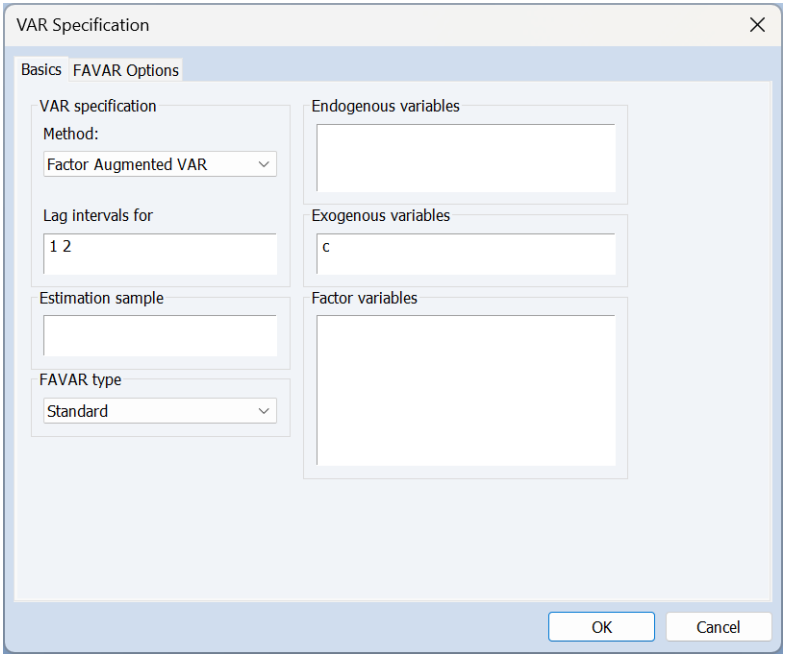


New or Updated Commands

enet ..... elastic net regression (including Lasso and ridge regression) (p. 39). *(updated)*

Factor-Augmented VAR

Factor-augmented vector autoregression (FAVAR) models extend the standard VAR framework by combining a small system of observed endogenous variables with information extracted from a larger set of related series. Rather than including the large information set directly in the VAR, EViews summarizes the comovement in the factor variables using a small number of estimated factors. The estimated factors are then appended to the observed endogenous variables and the augmented system is estimated as a VAR.



New or Updated Commands

favar .....estimate a Factor-Augmented VAR specification (p. 50). *(new)*

Enhancements to BVAR

The Bayesian VAR (BVAR) combines the likelihood function from an unrestricted VAR with a prior distribution over the VAR parameters. BVAR modeling is popular because the prior distribution gives the modeler the ability to shrink the fitted model towards a benchmark model. Shrinkage is desirable and sometimes even necessary in running VAR analysis.

EViews 15 makes the following updates to BVAR:

- The selection of available prior types have changed.
- The posterior sample is now cached for reuse in post-estimation procedures.

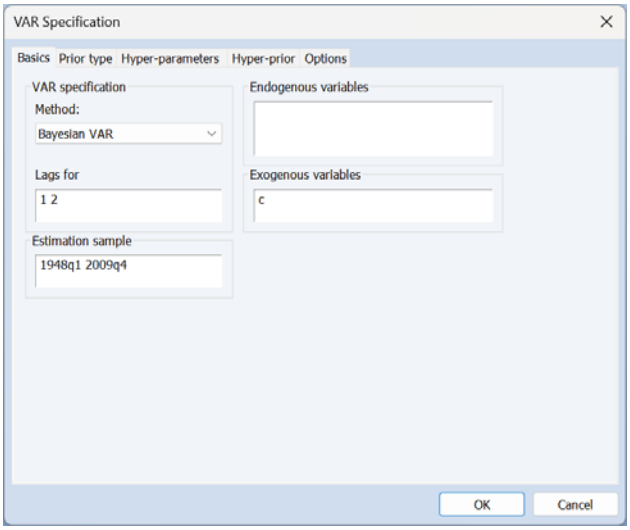
- GLP has been redesigned.

The User's Guide chapter on BVAR has also gone through significant changes. For more on EViews 15 BVAR updates, see "Version Compatibility Notes," beginning on page 394.

(Note that BVARs estimated in EViews 15 cannot be read by previous versions of EViews. Conversely, BVARs estimated in previous versions of EViews will need to be re-estimated in EViews 15).

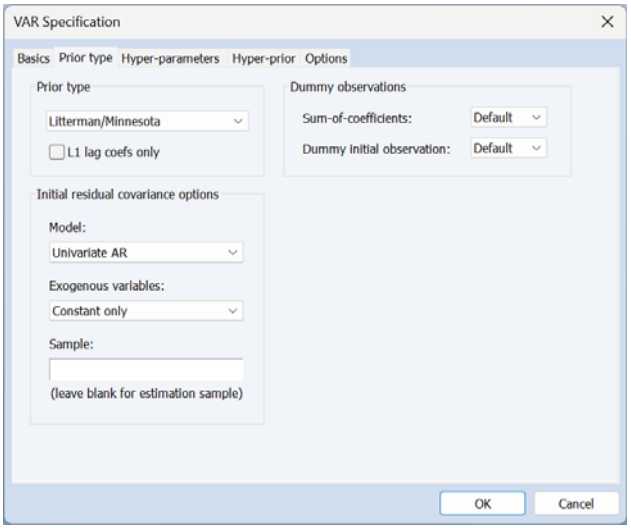
To estimate a BVAR in EViews, click on **Quick/Estimate VAR...** or type `var` in the command window and hit Enter. The VAR dialog will open. Select **Bayesian VAR** from the **Method:** dropdown menu. The VAR dialog will update to reflect your selection.

Endogenous variables, exogenous variables, included lags, and the estimation sample are set in the **Basics** page.

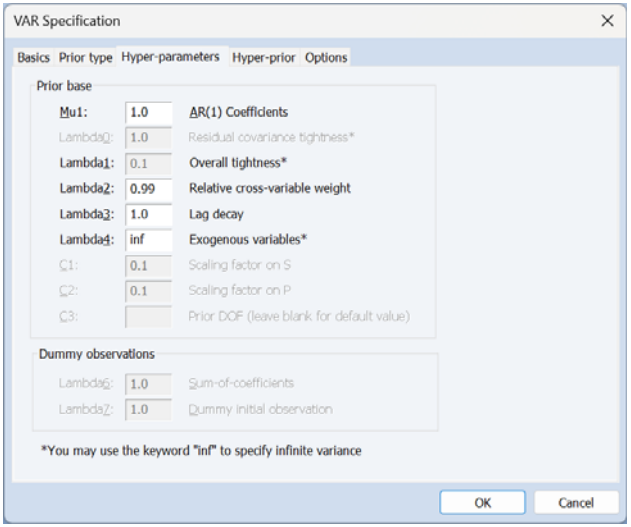


The remaining tabs let you customize your BVAR.

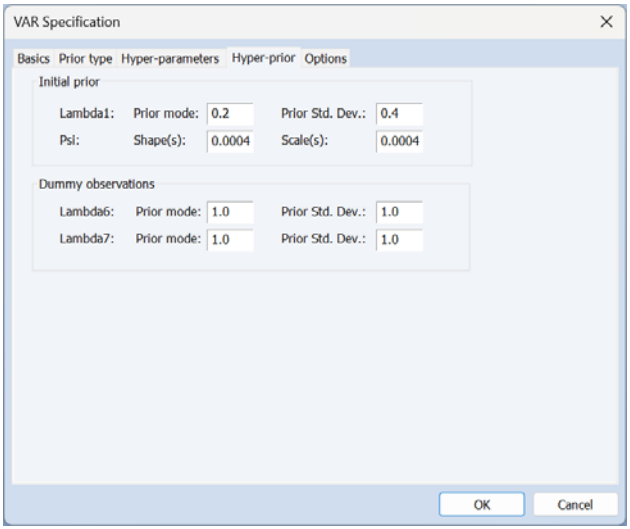
The **Prior type** page lets you specify the prior type, whether or not to include dummy observations, and the settings used to calculate the initial residual covariance matrix (when applicable). In addition, if the prior type is set to GLP, the user must make their selection of unknown prior hyperparameters (those being inferred) here.



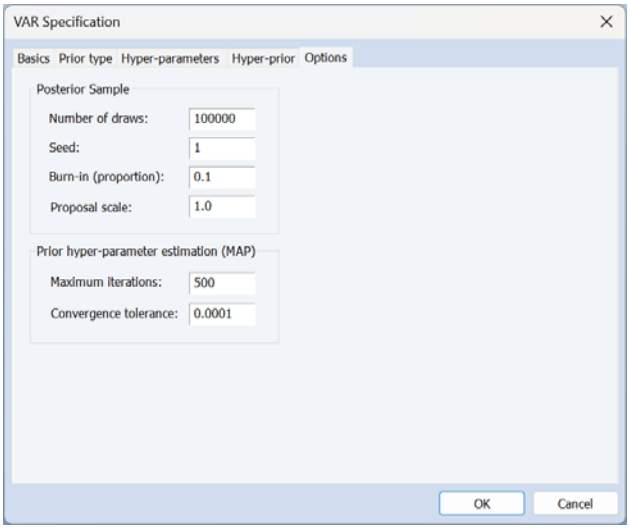
Prior hyperparameters may be set in the **Hyper-parameters** page. The available set of prior hyperparameters varies with other settings, e.g., the prior type.



The hyperprior (the distribution over unknown prior hyperparameters) may be set in the **Hyper-prior** page when GLP is being used.



All remaining options (e.g., those pertaining to simulation) are set in the **Options** page.



Updated GLP Method

The GLP method has been redesigned for EViews 15. Options have been added, modified, or removed to better align the EViews implementation of GLP with the literature.

EViews 15 updates to GLP include the following:

- The ability to specify the hyperprior (the distribution over prior hyperparameters).

- A different parameterization for prior hyperparameters.
- The timing of posterior simulation has been moved to during estimation. Estimates for VAR parameters are now posterior means rather than maximum-a-posteriori (MAP) estimates.

Example of GLP Method

In this example, we replicate some estimation results from Giannone et al. (2015) for the small-scale BVAR using the dataset in glpdata.wf1.

The small-scale BVAR is a Bayesian VAR(5) model on GDP (Y), GLP deflator (P), and the federal funds rate (FFR). To fit this, run the following in the command window:

```
var bvar.bvar(prior=glp, l1glp, nsumcoef, ninitobs,
    l4=@sqrt(10e6), l1lagcoefsonly, initcov=ar1, icsmpl=1960q3
    2008q4, propscale=50) 1 5 y p ffr
```

The options `prior=glp` and `l1glp` instructs EViews to use the GLP to infer λ_1. The keywords `nsumcoef` and `ninitobs` are needed to exclude dummy observations; dummy observations are included by default when GLP is used. The options `l4=@sqrt(10e6)`, `l1lagcoefsonly`, `initcov=ar1`, `icsmpl=1960q3 2008q4` are used to match our BVAR setup to that of the paper. Finally, the option `propscale=50` sets the proposal distribution scaling factor to 50, resulting in an MH acceptance rate of approximately the recommended 20%.

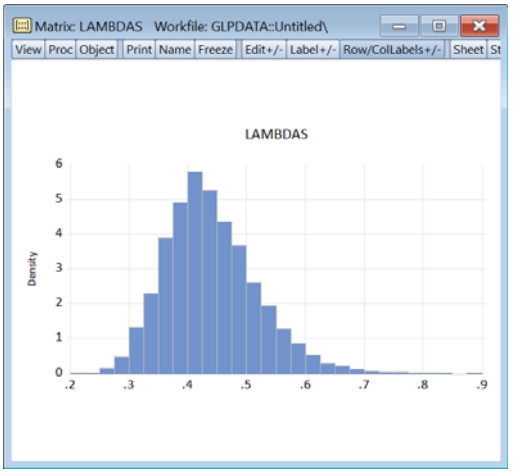
To display the posterior distribution of λ_1, run

```
bvar.postdraws coefs ecovs lambdas
```

first to save posterior draws to workfile, then run

```
lambdas.distplot(s) hist(anchor=0, scale=dens)
```

The resulting histogram, shown below, corresponds to the curve in Figure 1 of Giannone et al. (2015) for the small-scale BVAR.



New or Updated Commands

bvar ..... estimate a Bayesian VAR specification (p. 32). *(updated)*

postdraws..... put posterior draws for BVAR or BTVCVAR in current workfile (p. 67). *(updated)*

stablegr ..... carries out the stable Gelman-Rubin (GR) diagnostic (p. 77). *(new)*

Enhanced Prophet

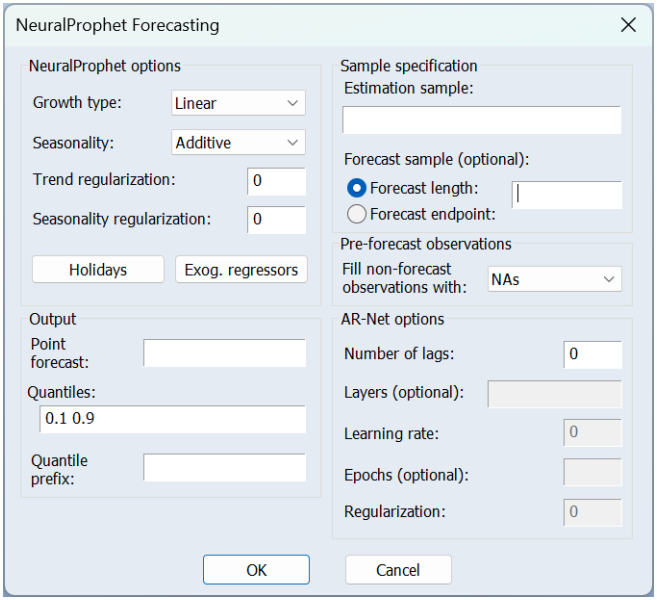
Prophet has been enhanced to allow specification of holiday effects.

New or Updated Commands

prophet ..... Performs Facebook's Prophet forecasting on the underlying series (p. 69). *(updated)*

NeuralProphet

NeuralProphet is an extension to the Prophet engine that combines the original Prophet algorithm with more advanced deep learning techniques. Specifically, NeuralProphet is a hybrid model that uses a neural network to add autoregressive components to Prophet.



New or Updated Commands

nprophet .....performs NeuralProphet forecasting on the underlying series (p. 63). *(new)*

LSTM Models

Long short-term memory (LSTM) models are a class of recurrent neural networks designed for ordered (sequential) data, see Hochreiter and Schmidhuber (1997); Gers, Schmidhuber and Cummins (2000). In forecasting applications, an LSTM learns a nonlinear mapping from the recent history of a target series (and, optionally, additional feature series) to future values of the target. EViews provides a built-in LSTM forecasting engine that automatically constructs training and validation sets, estimates the network by gradient-based optimization, and produces forecasts along with optional diagnostic output (loss plots, fitted values, and tuning summaries).

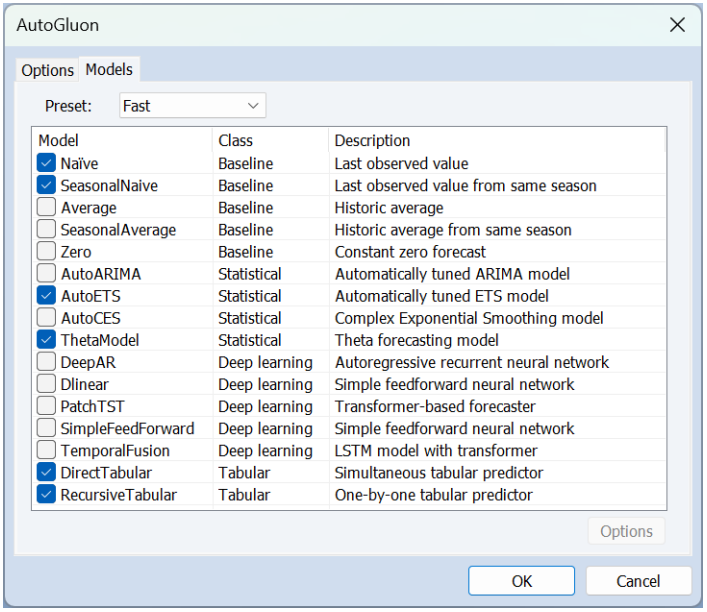
New or Updated Commands

lstm .....performs long short-term memory (LSTM) estimation and forecasting on the underlying series (p. 54). *(new)*

AutoGluon

AutoGluon is an open-source library designed by Amazon to simplify the process of building and optimizing machine learning models. Initially focused on tasks like tabular data, image

classification, and text processing, AutoGluon has extended its capabilities to support time series forecasting, offering a suite of time based forecasting models under one package. Access to the suite of machine learning techniques offered by AutoGluon is available within EViews 15, with an easy to use interface.



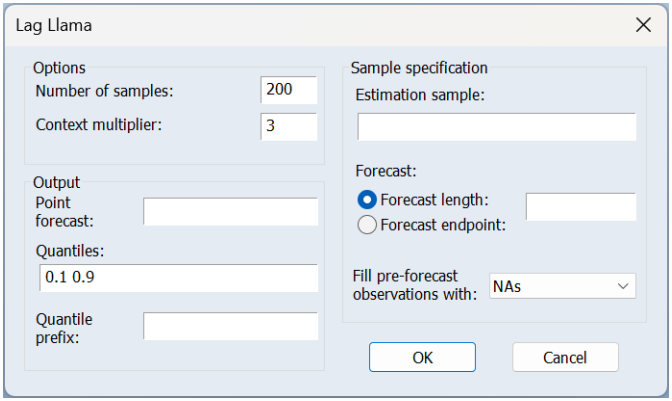
New or Updated Commands

autogluon..... performs AutoGluon forecasting on the underlying series (p. 26). *(new)*

Lag-Llama

Lag-Llama is an automatic forecasting package built on top of a large language model (LLM) architecture, designed to provide accurate forecasts across a wide range of frequencies and forecast horizons. Its strength lies in its ability to model complex dynamics that may be difficult for traditional models to detect.

EViews 15 introduces an easy to use interface to Lag-Llama that makes accessing its powerful forecasting methods straight forward.

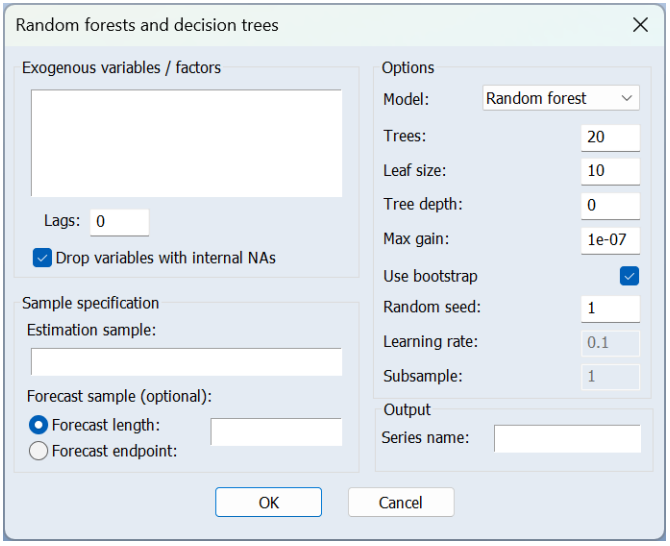


New or Updated Commands

lagllama .....performs Lag-Llama forecasting on the underlying series (p. 52). *(new)*

Random Forests and Decision Trees

Random Forest forecasting leverages an ensemble of decision trees to generate predictions by averaging the outputs of multiple tree-based models. Random Forests are robust to overfitting, straightforward to tune, and provide built-in measures of variable importance, making them an attractive choice for time series forecasting when traditional linear methods prove insufficient.

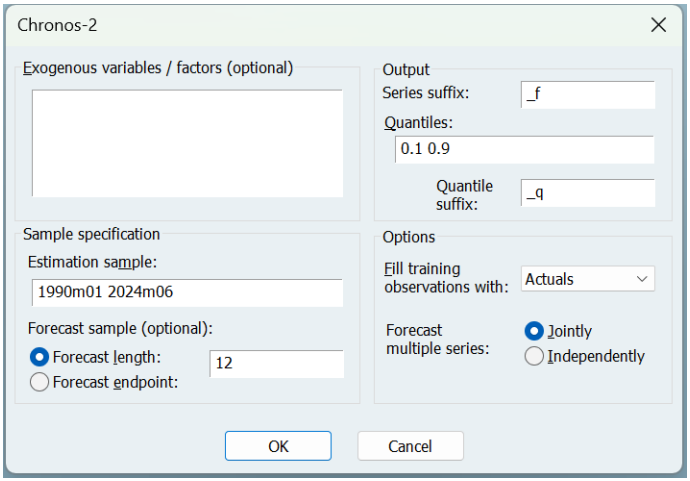


New or Updated Commands

randomforest ..... performs Random Forest style regression forecasting (p. 71). *(new)*

Chronos-2

Chronos-2 is an open-source transformer-based foundation model designed for both univariate and multivariate time series forecasting, developed by Amazon. Unlike other automatic forecasting models such as Prophet, NeuralProphet, or Lag-Llama, Chronos-2 leverages advanced natural language processing techniques by tokenizing time series data into discrete bins and predicting future values as a language modeling task. This approach enables zero-shot forecasting capabilities, meaning it can generate accurate predictions without requiring series-specific training.



New or Updated Commands

chronos2 ..... performs Chronos-2 forecasting on the underlying series (p. 37). *(new)*

chronos2 ..... performs Chronos-2 forecasting on the series in the group (p. 36). *(new)*

Enhanced ETS Smoothing

ETS smoothing now supports forecast intervals via bootstrap simulation.

New or Updated Commands

ets ..... perform Error-Trend-Season (ETS) estimation and exponential smoothing (p. 47). *(updated)*

Chapter 4. Preliminary Updates to Command Reference

This chapter contains preliminary documentation for commands that are new or have been updated in EViews 15.

Note, that this document is preliminary and is also extracted from a larger document so that portions may not be formatted properly and cross-reference links to pages and sections may not work properly.

Preliminary Listing of New and Updated EViews Commands

Equation

Equation Methods

enet .....elastic net regression (including Lasso and ridge regression) (p. 39). *(updated)*

modavg.....estimate equation using weighted-average least squares (WALS) or Bayesian model averaging (BMA) (p. 61). *(new)*

rdit .....estimate a regression discontinuity in time model. (p. 72). *(new)*

splinereg .....estimate an equation using nonparametric splines (p. 76). *(new)*

Equation Views

balance .....view the covariate balance chart of a regression discontinuity in time model estimated with covariates (p. 32). *(new)*

bwsensitivity.....view the bandwidth sensitivity chart of a regression discontinuity in time model (p. 35). *(new)*

discontinuity .....view the discontinuity chart of a regression discontinuity in time model (p. 38). *(new)*

donutsensitivity.....view the donut sensitivity chart of a regression discontinuity in time model (p. 38). *(new)*

knots .....view a table of the knot values from an equation estimated with splines (p. 61). *(new)*

modelranking.....display models ranked by posterior probability (p. 62). *(new)*

modelsize.....display results pertaining to the size of the model (p. 63). *(new)*

partialeffects.....view a graph of the partial effects from an equation estimated with splines (p. 66). *(new)*

placebo .....view the placebo chart of a regression discontinuity in time model (p. 66). *(new)*

postmarginals.....display the univariate marginals of the posterior distribution over parameters (p. 68). *(new)*

Group

Group Procs

chronos2 ..... performs Chronos-2 forecasting on the series in the group (p. 36). *(new)*

Series

Series Procs

autogluon ..... performs AutoGluon forecasting on the underlying series (p. 26). *(new)*

chronos2 ..... performs Chronos-2 forecasting on the underlying series (p. 37). *(new)*

ets ..... perform Error-Trend-Season (ETS) estimation and exponential smoothing (p. 47). *(updated)*

lagllama ..... performs Lag-Llama forecasting on the underlying series (p. 52). *(new)*

lstm ..... performs long short-term memory (LSTM) estimation and forecasting on the underlying series (p. 54). *(new)*

nprophet ..... performs NeuralProphet forecasting on the underlying series (p. 63). *(new)*

prophet ..... Performs Facebook's Prophet forecasting on the underlying series (p. 69). *(updated)*

randomforest ..... performs Random Forest style regression forecasting (p. 71). *(new)*

splinedecomp ..... perform seasonal adjustment of a series using a spline decomposition (p. 74). *(new)*

Var

Var Methods

bvar ..... estimate a Bayesian VAR specification (p. 32). *(updated)*

favar ..... estimate a Factor-Augmented VAR specification (p. 50). *(new)*

Var Procs

postdraws ..... put posterior draws for BVAR or BTVCVAR in current workfile (p. 67). *(updated)*

Var Views

stablegr ..... carries out the stable Gelman-Rubin (GR) diagnostic (p. 77). *(new)*

autogluon	Series Procs

Performs AutoGluon forecasting on the underlying series.

Prophet requires the AutoGluon Python library installed with EViews.

Syntax

series_name.**autogluon**(*options*) *forecast_name [quantile_prefix] [@modelspecs]*

Enter a name for the forecast series, and, optionally, a naming prefix for the quantile forecasts followed by specifications for each of the individual models used by AutoGluon.

A preselected set of models can be specified using the `preset=` option. If a `@modelspec` argument is included, that model will be included as well as any models specified in the preset. If the `preset` option is not used, only models specified by the `@modelspec` argument will be used.

Options

preset = *arg*	Select a pre-specified set of models. *arg* can be "naive", "fast" (default), "medium", "high".
eval = *arg*	Set the forecast evaluation metric. *arg* can be SQL (Scaled quantile loss), WQL (Weighted quantile loss, default), MAE (Mean absolute error), MAPE (Mean absolute percentage error), MASE (Mean absolute scaled error), MSE (Mean squared error), RMSE (Root mean squared error), RMSLE (Root mean squared logarithmic error), SMAPE (Symmetric mean absolute percentage error), WAPE (Weighted absolute percentage error).
quantiles = *"arg"*	Specify the forecast quantiles. By default, 0.1 and 0.9 quantiles are forecasted.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), or "na" (fill observations outside the forecast sample with missing values, default).

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Model Specifications

Individual models can be added to the list of models estimated by AutoGluon by including their keyword, with an @ symbol, after the `autogluon` command. If a model offers a set of

options, those options can be specified in parenthesis after the keyword. The model keywords are:

@naive

@seasonalnaive

@average

@seasaverage

@zero

@autoarima(options)

Options:

diff = *int*	Set the level of differencing. int may take values of 0, 1, or 2. If not specified, AutoGluon will automatically determine the most appropriate level of differencing.
sdiff = *int*	Set the level of seasonal differencing. int may take values of 0, or 1. If not specified, AutoGluon will automatically determine the most appropriate level of seasonal differencing.
maxar = *int*	Set the maximum number of AR terms. Default value is 5.
maxma = *int*	Set the maximum number of MA terms. Default value is 5.
maxsar = *int*	Set the maximum number of seasonal AR terms. Default value is 0.
maxsma = *int*	Set the maximum number of seasonal MA terms. Default value is 0.
nomean	Do not include an intercept term.
stationary	Only include stationary ARMA models in the search process.
drift	Include a drift term.

@autoets(options)

Options:

damped	Dampen the trend component of the ETS model.

@autoces

@theta(options)

Options:

decomp = *arg*	Set the decomposition to either multiplicative (`mult`, default) or additive (`add`).

@deepar(options)

Options:

context = *int*	Set the number of steps to unroll the recurrent neural network.
maxepoch = *int*	Set the number of epochs the model will be trained with.
batchperepoch = *int*	Set the number of batches of the training data processed per epoch.
lr = *num*	Set the learning rate / step size of the training process.
rnnlayers = *int*	Set the number of layers in the recurrent neural network.
rnncells = *int*	Set the number of cells in each layer.
dropout = *num*	Set the regularization parameter used during training in the recurrent layers.
noscale	Do not apply mean absolute scaling to each context window.

@dlinear(options)

Options:

context = *int*	Set the number of observations that condition the forecast.
maxepoch = *int*	Set the number of epochs the model will be trained with.
batchperepoch = *int*	Set the number of batches of the training data processed per epoch.
lr = *num*	Set the learning rate / step size of the training process.
hiddenlayers = *int*	Set the size of the hidden layers in the feedforward network.
wgtdecay = *num*	Set the weight decay regularization parameter.
scale = *arg*	Set the scaling applied to each context window during both training and forecasting. *arg* can be “mean” (default), “stdev”, or “none”.

@patchtst(options)

Options:

context = *int*	Set the number of observations that condition the forecast.
maxepoch = *int*	Set the number of epochs the model will be trained with.
batchperep-och = *int*	Set the number of batches of the training data processed per epoch.
lr = *num*	Set the learning rate / step size of the training process.
patchlen = *int*	Set the length of each patch.
stride = *int*	Set the stride of each patch.
hiddenlayers = *int*	Set the size of the hidden layers in the transformer encoder.
heads = *int*	Set the number of attention heads in the transformer encoder.
wgtdecay = *num*	Set the weight decay regularization parameter.

@simpfeedfwd(options)

Options:

context = *int*	Set the number of observations that condition the forecast.
maxepoch = *int*	Set the number of epochs the model will be trained with.
batchperep-och = *int*	Set the number of batches of the training data processed per epoch.
lr = *num*	Set the learning rate / step size of the training process.
hiddenlayers = *int*	Set the size of the hidden layers in the feed forward network.
batchnorm	Normalize/standardize the inputs to each batch during training.
noscale	Do not apply mean absolute scaling to each context window.

@tempfusion(options)

Options:

context = *int*	Set the number of observations that condition the forecast.
maxepoch = *int*	Set the number of epochs the model will be trained with.
batchperep-och = *int*	Set the number of batches of the training data processed per epoch.
lr = *num*	Set the learning rate / step size of the training process.
hiddenstates = *int*	Set the size of the LSTM and transformer hidden states.
embedsize = *int*	Set the size of the embeddings.

heads = *int*	Set the number of attention heads in the transformer encoder.
dropout = *num*	Set the regularization parameter used during training in the recurrent layers.
patience = *int*	Set the early stopping limit.

@directtab(options)

Options:

scale = *arg*	Set the scaling applied to the underlying series. *arg* can be "mean" (default), "standard", or "none".

@recurstab(options)

Options:

scale = *arg*	Set the scaling applied to the underlying series. *arg* can be "mean" (default), "standard", or "none".

Examples

```
elecdmd.autogluon elecdmd_f elec_f_q
```

This will execute the AutoGluon routine on the ELECDMD series with all default setting, putting the forecasted values into the series ELECDMD_F, and the 10% and 90% quantiles being stored in ELEC_F_Q_1 and ELEC_F_Q_9, respectively. Since no preset was specified, the default Fast preset is used, meaning the Naïve, Seasonal Naïve, Auto ETS, Theta, Direct Tabular and Recursive Tabular methods are used.

```
elecdmd.autogluon(preset=medium, f=actual) elecdmd_f elec_f_q
```

instructs AutoGluon to use the Medium preset, meaning the Naïve, Seasonal Naïve, Auto ETS, Theta, Temporal Fusion, Direct Tabular and Recursive Tabular methods are used. Actual values for out-of-forecast sample observations are inserted into the forecast series instead of NAs.

```
elecdmd.autogluon(preset=medium) elecdmd_f elec_f_q
   @autoarima(maxma=3)
```

instruct AutoGluon to use the Medium preset again, but to also add in the Auto ARIMA method, setting the maximum MA term to 3.

Cross-references

See "AutoGluon" on page 123 of *User's Guide I* for additional discussion.

balance	Equation Views

View the covariate balance chart of a regression discontinuity in time model estimated with covariates.

Syntax

eq_name.**balance**

Examples

```
equation eq1.rdit log(conceptions) adoptions @ 2007m7
eq1.balance
```

estimates a regression discontinuity in time model with the log of CONCEPTIONS as the outcome variable and July 2007 as the intervention date with ADOPTIONS used as an additional covariate, and then displays the covariate balance chart.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

bvar	Var Methods

Estimate a Bayesian VAR specification.

Syntax:

var_name.**bvar***(options) lag_pairs endog_list [@ exog_list]*

`bvar` estimates a Bayesian VAR. You must specify the order of the VAR (using one or more pairs of lag intervals), and then provide a list of series or groups to be used as endogenous variables. You may include exogenous variables such as trends and seasonal dummies in the VAR by including an "@-sign" followed by a list of series or groups. A constant is automatically added to the list of exogenous variables; to estimate a specification without a constant, you should use the option "noconst".

Options

General options

noconst	Do not include a constant in the exogenous regressors list.
prior = *keyword* (*default* = "lit")	Set the prior type: "lit" (Litterman/Minnesota), "nw" (Normal-Wishart), "inw" (independent normal-Wishart), "sz" (Sims-Zha), and "glp" (Giannone, Lenze, and Primiceri). The Litterman/Minnesota prior is used if no prior type is specified.
generic	Use a generic version of the prior (applies to the normal-Wishart and the independent normal-Wishart priors only). If the keyword "generic" is absent from the options list, the Litterman-type version of the prior is used.
initcov = *keyword* (*default* = "uni")	The model used in computing the initial residual covariance matrix: "ar1" (univariate AR(1)s), "uni" (univariate ARs), "diag" (diagonal of full VAR), and "full" (full VAR). If unspecified, EViews uses the "initcov = uni" option.
initexog = *keyword* (*default* = "const")	Set the exogenous regressors list used in computing the initial residual covariance matrix: "none" (no exogenous regressors), "const" (const only), and "all" (all exogenous regressors used in the BVAR). If unspecified, EViews uses the "initexog = const" option.
icsmpl = *arg*	Set the sample used in computing the initial residual covariance matrix. The estimation sample is used if no sample is specified.
sumcoef, nsumcoef	Use "sumcoef" to include, "nsumcoef" to exclude, sum-of-coefficients dummy observations. If neither keyword is included in the options list, the prior-specific default is used.
initobs, ninitobs	Use "initobs" to include, "ninitobs" to exclude, the dummy initial observation. If neither keyword is included in the options list, the prior-specific default is used.
l0 = *num*	Set the error covariance tightness (only applies to the Sims-Zha prior).
l1 = *num*	Set the overall tightness.
l1lagcoefsonly	Apply the overall tightness hyper-parameter (l1) to lag coefficients only.
l2 = *num*	Set the relative cross-variable weight.

l3 = *num*	Set the lag decay rate.
l4 = *num*	Set the exogenous variables tightness.
l6 = *num*	Set the scale for sum-of-coefficient dummies.
l7 = *num*	Set the scale for the dummy initial observation.
m1 = *num*	Set the prior mean on coefficients. Depending on the prior type, this can be a scalar, an n-vector, or a $k \times n$ matrix. The off-diagonal elements are set to zero if a scalar or vector is used.
c1 = *num*	Set the prior scale for the error covariance matrix. Depending on the prior type, this can be a scalar, an n-vector, or an $n \times n$ symmetric matrix. The off-diagonal elements are set to zero if a scalar or vector is used.
c2 = *num*	Set the prior scale on coefficients. Depending on the prior type, this can be a scalar, a kn-vector, or a $kn \times kn$ symmetric matrix. The off-diagonal elements are set to zero if a scalar or vector is used.
c3 = *num*	Set the prior degrees-of-freedom for the error covariance matrix. The default value is $n + 2$.
probstable	Compute the posterior probability that the VAR is stable. Note that this involves the use of a posterior sample.
l1glp	Estimate the l1 prior hyper-parameter.
l1mo = *num*	Set the hyper-prior mode for l1.
l1sd = *num*	Set the hyper-prior standard deviation for l1.
l6glp	Estimate the l6 prior hyper-parameter.
l6mo = *num*	Set the hyper-prior mode for l6.
l6sd = *num*	Set the hyper-prior standard deviation for l6.
l7glp	Estimate the l7 prior hyper-parameter.
l7mo = *num*	Set the hyper-prior mode for l7.
l7sd = *num*	Set the hyper-prior standard deviation for l7.
psiglp	Estimate the prior mean of the error variances.
psishapes = *num*	Set the hyper-prior shape parameters for the prior mean of the error variances.
psiscales = *num*	Set the hyper-prior scale parameters for the prior mean of the error variances.
draws = *num*	Set the number of draws to sample.

seed = *num*	Set the random seed for the posterior simulator.
burn = *num*	Set the percentage of draws to discard for burn-in.
propscale = *num*	Set the scale for the Metropolis-Hastings (MH) proposal-generating density. Only applicable to GLP.
m = *integer*	Set the optimization maximum number of iterations. Only applicable to GLP.
c = *scalar*	Set the optimization convergence criterion. Only applicable to GLP.
prompt	Force the dialog to appear from within a program.
p	Print basic estimation results.

Examples

```
var var01.bvar 1 3 m1 gdp
```

declares and estimates a VAR object named VAR01, which is a Bayesian VAR that uses the Litterman/Minnesota prior and has two endogenous variables (M1 and GDP), a constant, and 3 lags (1 through 3).

```
var01.bvar(noconst) 1 3 m1 gdp
```

estimates the same BVAR but omits the constant.

```
var var02.bvar(prior=nw, m1=0.5) 1 3 m1 gdp
```

declares and estimates a BVAR that uses the (conjugate) normal-Wishart prior, with the prior hyperparameter m1 set to 0.5.

```
var var03.bvar(prior=inw, m1=0.5) 1 3 m1 gdp
```

does the same under the independent normal-Wishart prior.

```
var var04.bvar(prior=glp, l1glp) 1 3 m1 gdp
```

uses the GLP prior to estimate not only the VAR parameters but also the overall tightness hyper-parameter l1.

Cross-references

See Chapter 12. “Bayesian VAR Models,” on page 369 of *User’s Guide II* for details.

bwsensitivity	Equation Views

View the bandwidth sensitivity chart of a regression discontinuity in time model.

Syntax

eq_name.**bwsensitivity**

Examples

```
equation eq1.rdit log(conceptions) @ 2007m7
eq1.bwsensitivity
```

estimates a regression discontinuity in time model with the log of CONCEPTIONS as the outcome variable and July 2007 as the intervention date, and then displays the bandwidth sensitivity chart.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

chronos2	Group Procs

Performs Chronos-2 forecasting on the series in the group.

This command requires the Chronos-2 Python library installed with EViews.

Syntax

group_name.**chronos2**(*options*) *forecast_suffix [quantile_suffix] @ exog*

Enter a naming pattern suffix for the forecast series, and, optionally, a naming pattern suffix for the quantile forecasts. If you wish to include exogenous variables, use an "@" followed by the list of series names.

Options

independent	Perform independent univariate forecasts of the series. If omitted, Chronos-2 will perform a joint forecast.
quantiles = *"arg"*	Specify the forecast quantiles. By default, 0.1 and 0.9 quantiles are forecasted.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable) or "na" (fill observations outside the forecast sample with missing values, default).

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
macrovars.chronos2 _f _f_q @ @trend
```

will perform Chronos-2 forecasting on the series inside the group MACROVARS, putting the forecasted values into series with the same name as the original series, but with a _F appended, and the 10% and 90% quantiles being stored in *_F_Q_1 and *_F_Q_9, respectively. A time trend is included as an exogenous variable.

Cross-references

See "Chronos-2" on page 170 of *User's Guide I* for additional discussion.

chronos2	Series Procs

Performs Chronos-2 forecasting on the underlying series.

This command requires the Chronos-2 Python library installed with EViews.

Syntax

series_name.**chronos2**(*options*) *forecast_name [quantile_prefix] @ exog*

Enter a name for the forecast series, and, optionally, a naming prefix for the quantile forecasts. If you wish to include exogenous variables, use an "@" followed by the list of series names.

Options

quantiles = *"arg"*	Specify the forecast quantiles. By default, 0.1 and 0.9 quantiles are forecasted.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), or "na" (fill observations outside the forecast sample with missing values, default).

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
elecdmd.chronos2 elecdmd_f elec_f_q @ temp @expand(@month)
```

This will execute the Chronos-2 routine on the ELECDMD series putting the forecasted values into the series ELECDMD_F, and the 10% and 90% quantiles being stored in ELEC_F_Q_1 and ELEC_F_Q_9, respectively. The series TEMP, along with monthly dummy variables, are used as exogenous variables.

Cross-references

See "Chronos-2" on page 170 of *User's Guide I* for additional discussion.

discontinuity	Equation Views

View the discontinuity chart of a regression discontinuity in time model.

Syntax

eq_name.**discontinuity**

Examples

```
equation eq1.rdit log(conceptions) @ 2007m7
eq1.discontinuity
```

estimates a regression discontinuity in time model with the log of CONCEPTIONS as the outcome variable and July 2007 as the intervention date, and then displays the discontinuity chart.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

donutsensitivity	Equation Views

View the donut sensitivity chart of a regression discontinuity in time model.

Syntax

eq_name.**donutsensitivity**

Examples

```
equation eq1.rdit log(conceptions) @ 2007m7
eq1.donutsensitivity
```

estimates a regression discontinuity in time model with the log of CONCEPTIONS as the outcome variable and July 2007 as the intervention date, and then displays the donut sensitivity chart.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

enet	Equation Methods

Estimation of an elastic net model, including options for Lasso and ridge regression.

Syntax

equation_name.**enet***(options) y x1 [x2 x3 ...] [@vw(...)]*

List the dependent variable first, followed by a list of the independent variables. Use a "C" if you wish to include an unpenalized intercept term.

Note that PDL and ARMA terms are not permitted in elastic net specifications.

If you wish to specify regressors with an individual penalty weight ω_j, or to place inequality restrictions on the coefficient values, you may do so using special expressions of the form:

*@***vw**(*series_name, weight_value)*

or

*@***vw**(*series_name[,* **wgt** = *weight_value,* **cmin** = *coef_min,* **cmax** = *coef_max]*)

where *weight_value* is a non-negative value, *coef_min* is a non-positive minimum coefficient value, and *coef_max* is a non-negative maximum coefficient value.

There are two forms of the special expression.

In the abbreviated form, you specify the variable name followed by the penalty weight value.

In the more general form, you specify the variable name followed by one or more keyword expressions in arbitrary order, with the "`wgt=`" argument specifying the penalty weight, "`cmin=`" with a non-positive minimum coefficient value, and "`cmax=`" with a non-negative maximum coefficient value.

When specifying individual regressor behavior using `@vw,` keep in mind that:

- The special intercept variable "C" is always non-penalized and has an implicit weight $\omega = 0.0$.
- Individual penalty weights may be also specified using a vector in the **Individual lambda wgts. vector** edit field on the **Penalty** dialog page (or using the command

estimation option "**lambdawgt** = *vector_name*"). If the vector weights are specified and individual weights are specified using the `@vw` keyword, the vector weights will be applied first, followed by the individual variable weights.

- Individual coefficient limit values may also be specified using vectors in the **Min values vector** and **Max values vector** edit fields on the **Options** dialog page (or the command estimation options "**coefmin** = *vector_name*" and "**coefmax** = *vector_name*"). If vector coefficient limits are specified and individual regressor limits are specified using the `@vw` keyword, the vector limits will be applied first, followed by the individual limits weights.

EViews will normalize the individual penalty weights so that they sum to the number of coefficients.

Options

Specification Options

eqtype = glm	Estimate a generalized linear model (GLM); omit for ordinary linear model.
penalty = *arg* *(default* = "el")	Type of threshold estimation: "enet" (elastic net), "ridge" (ridge), "lasso" (Lasso).
alpha = *arg* *(default* = ".5")	Value of the mixing parameter. Must be a value from zero to one.
lambda = *arg*	Value(s) of the penalty parameter. Can be one or more numbers or vector objects.Values must be zero or greater. If left blank (default) EViews will generate a list.

GLM Options

The following specification options are only applicable when the equation type is set to generalized linear model (GLM).

family = *arg* *(default* = "normal"*)*	Distribution family: Normal ("normal"), Poisson ("poisson"), Binomial Count ("binomial"), Binomial Proportion ("binprop"), Negative Binomial ("negbin"), Gamma ("gamma"), Inverse Gaussian ("igauss"), Exponential Mean ("emean)", Power Mean ("pmean"), Binomial Squared ("binsq"). The Binomial Count, Binomial Proportion, Negative Binomial, and Power Mean families all require specification of a distribution parameter:
n = *arg (default* = 1*)*	Number of trials for Binomial Count ("family = binomial") or Binomial Proportions ("family = binprop") families.

fparam = *arg*	Family parameter value for Negative Binomial ("family = negbin") and Power Mean ("family = pmean") families.
link = *arg* *(default* = "identity"*)*	Link function: Identity ("identity"), Log ("log"), Log Compliment ("logc"), Logit ("logit"), Probit ("probit"), Log-log ("loglog"), Complementary Log-log ("cloglog"), Reciprocal ("recip"), Power ("power"), Box-Cox ("boxcox"), Power Odds Ratio ("opow"), Box-Cox Odds Ratio ("obox"). The Power, Box-Cox, Power Odds Ratio, and Box-Cox Odds Ratio links all require specification of a link parameter specified using "lparam = ".
lparam = *arg*	Link parameter for Power ("link = power"), Box-Cox ("link = boxcox"), Power Odds Ratio ("link = opow") and Box-Cox Odds Ratio ("link = obox") link functions.
offset = *arg*	Offset terms.
disp = *arg*	Dispersion estimator: Pearson χ^2 statistic ("pearson"), deviance statistic ("deviance"), unit ("unit"), user-specified ("user"). The default is family specific: "unit" for Binomial Count, Binomial Proportion, Negative Binomial, and Poison, and "pearson" for all others. The "deviance" option is only offered for families in the exponential family of distributions (Normal, Poisson, Binomial Count, Binomial Proportion, Negative Binomial, Gamma, Inverse Gaussian).
dispval = *arg*	User-dispersion value (if "disp = user").
fwgts = *arg*	Frequency weights.
w = *arg*	Weight series or expression.
wtype = *arg* *(default* = "istdev"*)*	Weight specification type: inverse standard deviation ("istdev"), inverse variance ("ivar"), standard deviation ("stdev"), variance ("var").
wscale = *arg*	Weight scaling: EViews default ("eviews"), average ("avg"), none ("none"). The default setting depends upon the weight type: "eviews" if "wtype = istdev", "avg" for all others.

Penalty Options

ytrans = *arg* *(default* = "none")	Scaling of the dependent variable: "none" (none), "L1" (L1), "L2" (L2), "stdsmpl" (sample standard deviation), "stdpop" (population standard deviation), "minmax" (min-max).
xtrans = *arg* *(default* = "stdpop")	Scaling of the regressor variables: "none" (none), "L1" (L1 norm), "L2" (L2 norm), "stdsmpl" (sample standard deviation), "stdpop" (population standard deviation), "minmax" (min-max).
nlambdas = *integer* *(default = 100)*	Number of penalty values for EViews-supplied list.
lambdaratio = *arg*	Ratio of minimum to maximum lambda for EViews-supplied list. You may specify a value for the ratio parameter, or you may leave the edit field blank to let EViews specify a default value based on the number of observations T and the number of potential regressors p. By default, EViews will set the ratio to 0.001.
lambdawgt = *vector_name*	Vector of individual penalty weights, containing non-negative values sized to and matching the order of the variables in the specification. If a vector is provided and individual weights are specified using one or more `@vw` regressors, the vector weights will be applied first, then overwritten by the individual variable weights. For comparability purposes, we normalize the final weights so that they sum to $p*$ where $p*$ the number of non-zero ω_j.
nlambdamin = *integer* *(default = 5)*	Minimum number of lambda values in the path before applying stopping rules.
minddev = *arg* *(default* = 1e-05*)*	Minimum change in deviance fraction to continue estimation. Truncate path estimation if relative change in deviance is smaller than this value.

maxedev = *arg* *(default* = 0.99*)*	Maximum of deviance explained fraction attained to terminate estimation. Truncate path estimation if fraction of null deviance explained is larger than this value.
maxvars = *arg*	Maximum number of regressors in the model. Truncate path estimation if the number of coefficients (including those for non-penalized variables like the intercept) reaches this value.
maxvarsratio = *arg*	Maximum number of regressors in the model as a fraction of the number of observations. Truncate path estimation if the number of coefficients (including those for non-penalized variables like the intercept) divided by the number of observations reaches this value.

Cross Validation Options

cvmethod = *arg* *(default* = "kfold_cv")	Cross-validation method: "kfold" (k-fold), "simple" (simple split), "mcarlo" (Monte Carlo), "leavepout" (leave-P-out), "leave1out" (leave-1-out), "rolling" (rolling window), "expanding" (expanding window).
cvmeasure = *arg* *(default* = "mse")	Cross-validation fit measure: "mse" (mean-squared error), "r2" (R-squared), "mae" (mean absolute error), "mape" (mean absolute percentage error), "smape" (symmetric mean absolute percentage error).
cvnfolds = *arg* *(default* = 5)	Number of folds for K-fold cross-validation. For "cvmethod = kfold".
cvftrain = *arg* *(default* = 0.8)	Proportion of data for split and Monte Carlo methods. For "cvmethod = simple" and "cvmethod = mcarlo".
cvnreps = *arg* *(default* = 1)	Number of Monte Carlo method repetitions. For "cvmethod = mcarlo".
cvleaveout = *arg* *(default* = 2)	Number of data points left out for leave-p-out method. For "cvmethod = leavepout".
cvnwindows = *arg* *(default* = 4)	Number of windows for rolling window cross-validation method. For "cvmethod = rolling".
cvinitial = *arg* *(default* = 12)	Number of initial data points in the training set for expanding cross-validation. For "cvmethod = expanding".

cvpregap = *arg* (*default* = 0)	Number of observations between end of training set and beginning of test set. For "cvmethod = simple", "cvmethod = rolling" and "cvmethod = expanding".
cvhorizon = *arg* (*default* = 1)	Number of observation in the test set. For "cvmethod = rolling" and "cvmethod = expanding".
cvpostgap = *arg* (*default* = 0)	Number of observations between end of test set and beginning of next training set for rolling window or between end of test set and end of next training set for expanding window. For "cvmethod = rolling" and "cvmethod = expanding"

Random Number Options

seed = *positive_integer* from 0 to 2,147,483,647	Seed the random number generator. If not specified, EViews will seed random number generator with a single integer draw from the default global random number generator.
rnd = *arg* (*default* = "kn" or method previously set using `rndseed` (p. 576) in the *Command and Programming Reference)*.	Type of random number generator: improved Knuth generator ("kn"), improved Mersenne Twister ("mt"), Knuth's (1997) lagged Fibonacci generator used in EViews 4 ("kn4") L'Ecuyer's (1999) combined multiple recursive generator ("le"), Matsumoto and Nishimura's (1998) Mersenne Twister used in EViews 4 ("mt4").

Other Options

coefmin = *vector_name, number*	Vector of individual coefficient minimum values, containing negative or missing values sized to and matching the order of the variables in the specification, or a negative value for the minimum for all coefficients. Missing values in the vector should be used to indicate that the coefficient is unrestricted. If a vector of values is provided and individual minimums are specified using one or more `@vw` regressors, the vector values will be applied first, then overwritten by the individual values.

coefmax = *vector_name, number*	Vector of individual coefficient maximum values, containing positive or missing values sized to and matching the order of the variables in the specification, or a positive value for the maximum for all coefficients. Missing values in the vector should be used to indicate that the coefficient is unrestricted. If a vector of values is provided and individual maximums are specified using one or more `@vw` regressors, the vector values will be applied first, then overwritten by the individual values.
maxit = *integer*	Maximum number of iterations.
conv = *scalar*	Set convergence criterion. The criterion is based upon the maximum of the percentage changes in the scaled estimates. The criterion will be set to the nearest value between 1e-24 and 0.2.
w = *arg*	Weight series or expression.
wtype = *arg* *(default* = "istdev"*)*	Weight specification type: inverse standard deviation ("istdev"), inverse variance ("ivar"), standard deviation ("stdev"), variance ("var").
wscale = *arg*	Weight scaling: EViews default ("eviews"), average ("avg"), none ("none"). The default setting depends upon the weight type: "eviews" if "wtype = istdev", "avg" for all others.
estmethod = *arg*	Estimation method: "cov" (use covariance algorithm) or "naive" (use naive algorithm). Default is EViews automatic.
showopts / -showopts	[Do / do not] display estimation options in the output.
prompt	Force the dialog to appear from within a program.
p	Print basic results view after estimation.

Examples

The command

```
equation enet1.enet(xtrans=none, lambdaratio=.0001,
   cvseed=513255899) lpsa c lcavol lweight age lbph svi lcp
   gleason pgg45
```

estimates an elastic net model with α equal to the default of 0.9, no regressor or dependent variable scaling, automatically determined 100-element lambda path with minimum lambda of 0.0001 times the maximum value, using the default K-fold cross-validation with 5 folds

with an MSE objective and a random generator seed of 513255899 to determine the optimal value.

Similarly,

```
equation lasso1.enet(penalty=lasso, lambdaratio=.0001,
   cvseed=513255899) lpsa c lcavol lweight age lbph svi lcp
   gleason pgg45
```

estimates a Lasso model with regressor population standard deviation scaling, with the remaining settings as before, while

```
equation ridge1.enet(penalty=ridge, lambdaratio=.0001,
   cvseed=513255899) lpsa c lcavol lweight age lbph svi lcp
   gleason pgg45
```

estimates the equivalent ridge regression specification.

The command

```
equation enet2.enet(alpha=0.75, lambdaratio=.0001,
   cvmethod=rolling, cvmeasure=smape) lpsa c lcavol lweight age_s
   lbph svi lcp gleason pgg45
```

estimates an elastic net model with α equal to the 0.75 using rolling window cross-validation and SMAPE cross-validation.

We may use the `@vw` specifications to assign individual penalties and coefficient restrictions

```
equation enet3.enet(alpha=0.75, lambdaratio=.0001,
   cvmethod=rolling, cvseed=513255899) lpsa c @vw(lcavol, cmax=.4)
   lweight age lbph svi lcp gleason @vw(pgg45, cmax=0.075, w=1.2)
```

estimates an elastic net model with the coefficient of LCAVOL restricted to be less than or equal to 0.4, and the coefficient of PGG45 having a relative penalty weight of 1.2, and a maximum value of 0.075.

Identical specifications may be estimated using vectors of penalty weights and coefficient restrictions,

```
vector(9) cmax = na
cmax(2) = 0.4
cmax(9) = 1.2
vector(9) lwgt = 1
lwgt(9) = 1.2
equation enet4.enet(alpha=0.75, coefmax=cmax, lambdawgt=lwgt,
   lambdaratio=.0001, cvmethod=rolling, cvseed=513255899) lpsa c
   lcavol lweight age lbph svi lcp gleason pgg45
```

and

```
vector(9) cmax = na
```

```
cmax(2) = 0.4
vector(9) lwgt = 1
lwgt(9) = 50
equation enet5.enet(alpha=0.75, coefmax=cmax, lambdawgt=lwgt,
    lambdaratio=.0001, cvmethod=rolling, cvseed=513255899) lpsa c
    lcavol lweight age lbph svi lcp gleason @vw(pgg45, cmax=0.075,
    w=1.2)
```

since the penalty weight for PGG45 in the vector is overwritten by the individual weight specified using the `@vw`.

Note that in neither case is the intercept penalized, even though the corresponding element of LWGT is equal to 1 since the specification of "C" is always implicitly treated as "`@VW(C, 0)`".

Cross-references

See Chapter 10. "Elastic Net and Lasso," on page 303 of *User's Guide II* for a discussion of elastic net, ridge regression, and Lasso estimation for the linear and generalized linear models.

ets	Series Procs

Perform Error-Trend-Season (ETS) exponential smoothing.

The `ets` procedure forecasts a series using the ETS model framework with state-space based likelihood calculations, support for model selection, and calculation of forecast standard errors.

The ETS framework defines an extended class of exponential smoothing models, including the standard exponential smoothing models (e.g., Holt and Holt-Winters additive and multiplicative models).

Syntax

series_name.**ets***(options) smooth_name*

You should enter the `ets` keyword followed by options and then the a name for the smoothed output series. You can specify the smoothing method (the default setting is additive error, no trend, no seasonality) and the smoothing options in the parenthesis.

Options

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

One of these options is required.

General

prompt	Force the dialog to appear from within a program.
p	Print the view.

Model specification

e = *arg* (default = "a")	Set error type: "a" (additive), "m" (multiplicative), "e" (auto).
t = *arg* (default = "n")	Set trend type. *key* can be: "n" (none), "a" (additive), "m" (multiplicative), "ad" (additive dampened), "md" (multiplicative dampened), "e" (auto).
s = *arg* (default = "n")	Set season type. *key* can be: "n" (none), "a"(additive), "m" (multiplicative), "e" (auto).
modsel = *arg* (default = "aic")	Model selection method: "aic" (Akaike information criterion), "bic" (Bayesian information criterion/Schwartz criterion), "hq" (Hannan-Quinn information criterion), "amse" (average mean squared errors).
alpha = *arg*	Specify fixed value for level parameter α.
beta = *arg*	Specify fixed value for trend parameter β in models with trend.
gamma = *arg*	Specify fixed value for seasonal parameter γ in models with a seasonal component.
phi = *arg*	Specify fixed value for dampening parameter ϕ in models with dampened trends.
nomult	Do not allow multiplicative trend or seasonal terms. Only applies if the t = e or s = e options are set.

Optimization options

amse	Set Average Mean Square Error (AMSE) as the objective function (The default is log-likelihood as the objective function).
namse = *integer*	Specify the AMSE length—the number of observations over which to calculate AMSE if "amse" is selected.
c = *number*	Set the convergence criteria.

m = *integer*	Set the maximum number of iterations.
ustart	Employ user-supplied starting values (taken from the C vector in the workfile).
noi	Do not optimize the initial state values (fix at their starting values).

Output options

dgraph = *arg*	Include a decomposition graph for each specified element. *arg* may be composed of any of the following elements: "f" (forecast), "l" (level), "t" (trend), "s" (season).
dgopt = *arg* (default = "m")	Format for display of decomposition graph: "m" (multiple graph), "s" (single graph)
graph = *arg*	Include a comparison graph in the output for each specified element (if model selection is employed). *arg* may be composed of any of the following elements: "c" (forecast comparison) and "l" (likelihood comparison).
table = *arg*	Include a comparison table in the output (if model selection is employed). *arg* may be composed of any of the following elements: "c" (forecast comparison) and "l" (likelihood comparison).
level = *name*	Save the level component as a separate series in the workfile.
trend = *name*	Save the trend component as a separate series in the workfile (if applicable).
season = *name*	Save the seasonal component as a separate series in the workfile (if applicable).

Bootstrap options

boot	Perform bootstrap replications for forecast intervals.
nsamples = *int* (default = 0)	Number of bootstrap replications.
bootseed = *int* (default = 1)	Seed for the random number generator used in bootstrap resampling.
quantiles = *arg* (default = "0.1 0.9")	Space-delimited list of quantiles of the simulated forecast distribution to compute. Each value must lie strictly between 0 and 1.
quantprefix = *name*	Prefix used to name the saved quantile series. If not specified, "Q" is used as a prefix.

Examples

```
sales.ets(e=a, t=n, s=a)sales_f
```

smooths the series SALES using the an ANN (additive error, no trend, no seasonal) model and creates the smoothed series named “SALES_F”.

```
tb3.ets(e=e, t=e, s=n) tb3_smooth
```

will smooth TB3, automatically selecting the best smoothing model amongst the different Error and Trend specifications (the Seasonal specification is set at none).

```
sales.ets(e=a, t=a, s=a, dgopt=m, dgraph=flts)
```

will smooth the series SALES using the an AAA (additive error, additive trend, additive seasonal) model and display the output in a spool object which contains the actual and decomposition series (i.e., forecast, trend, level, and seasonal series) in multiple graphs.

```
sales.ets(e=a, t=a, s=a, level=level1, trend=trend1,
   season=season1, dgopt=s, dgraph=flts)
```

will smooth the series SALES using the an AAA (additive error, additive trend, additive seasonal) model, create the decomposition series named level, trend, and season series as level1, trend1, and season1, respectively, and display a spool object which contains the actual and decomposition graphs in a single graph.

```
tb3.ets(e=e, t=e, s=e, graph=cl)
```

will find out the best model amongst the different Error, Trend, and Seasonal specifications and present the estimation results in a spool object which contains the graphs with forecast and likelihood comparison graphs between all available models.

```
tb3.ets(e=a, t=e, s=e, amse, table=cl)
```

will search for the best model using average mean square errors calculations and display the estimation results in a spool object with forecast and likelihood comparison tables.

Cross-references

See “ETS Exponential Smoothing” on page 89 of *User’s Guide I* for a discussion of exponential smoothing methods.

favar	Var Methods

Estimate a Factor-Augmented VAR specification.

Syntax

The command form for a standard FAVAR is

var_name.**favar***(options) lag_spec y_list @ exog_list @FAVARFACTORS factor_list*

where `lag_spec` is the VAR lag specification, `y_list` is the list of observed endogenous variables, `exog_list` is the list of exogenous variables, and `factor_list` is the list of series used for factor extraction.

The command form for a BBE FAVAR is

var_name.favar*(options) lag_spec y_list @ exog_list @FAVARSLOW slow_list @FAVARFAST fast_list*

where `slow_list` and `fast_list` are the slow and fast factor-variable specifications.

If no exogenous regressors are desired, enter an empty exogenous field in the same manner as for standard VAR estimation.

Options

type = *arg* (*default* = `bbe`)	FAVAR type: standard (estimate a standard FAVAR using the factor variables supplied in `@FAVARFACTORS`) or BBE (estimate a Bernanke-Boivin-Eliasz FAVAR using `@FAVARSLOW` and `@FAVARFAST`, default).
demeantime	Demean each factor variable over the estimation sample before extracting factors.
sdizetime	Standardize each factor variable over the estimation sample before extracting factors.
demeancross	Demean observations across the factor-variable cross-section before extracting factors.
sdizecross	Standardize observations across the factor-variable cross-section before extracting factors.
mfmethod = *arg*	Maximum-factor method. Supported values are `schwert`, `ah`, `rootsize`, `size`, and `user`.
n = *int*	User-specified maximum number of factors when mfmethod = `user`.
fsmethod = *arg*	Factor-selection method. Supported values are `bn`, `ah`, `simple`, and `user`.
fsic_bn = *arg*	Bai-Ng criterion. Supported values are `ICP1`, `ICP2`, `ICP3`, `PCP1`, `PCP2`, `PCP3`, and `AVG`.
fsic_ah = *arg*	Ahn-Horenstein criterion. Supported values are `ER`, `GR`, and `AVG`.
fsic_simple = *arg*	Simple factor-selection criterion. Supported values are `MIN`, `MAX`, and `AVG`.

mineigen = *num*	Minimum eigenvalue threshold used by the simple factor-selection method.
cproport = *num*	Cumulative eigenvalue proportion threshold used by the simple factor-selection method.
r = *int*	User-specified number of retained factors when fsmethod = `user`.

By default, FAVAR estimation uses `type=bbe`, no demeaning or standardization, `mfmethod=user` with `n=1`, and `fsmethod=simple` with `fsic_simple=min`. The default simple-method thresholds are `mineigen=0` and `cproport=1`. If `fsmethod=user` is selected, the retained number of factors is supplied using `r=int`, with default value `r=1`. The default Bai-Ng criterion is `fsic_bn=icp1`, and the default Ahn-Horenstein criterion is `fsic_ah=er`.

Examples

The following command estimates a standard FAVAR with four lags, two observed endogenous variables, a constant, and factors extracted from the variables in XGROUP:

```
var1.favar(type=standard, fsmethod=user, r=3) 1 4 y1 y2 @ c
   @FAVARFACTORS xgroup
```

The next command estimates a BBE FAVAR with four lags, a constant, slow variables XSLOW1 XSLOW2 XSLOW3, and fast variables XFAST:

```
var2.favar(type=bbe, fsmethod=user, r=3) 1 4 policy @ c @FAVARSLOW
   xslow1 xslow2 xslow3 @FAVARFAST xfast
```

The following command estimates a standard FAVAR and selects the number of factors using Bai-Ng criteria:

```
var3.favar(type=standard, mfmethod=schwert, fsmethod=bn,
   fsic_bn=avg, demeantime, sdizetime) 1 6 y1 y2 @ c @FAVARFACTORS
   x
```

Cross-references

See "Factor-Augmented VAR (FAVAR)" on page 355 of *User's Guide II* for additional discussion.

lagllama

Series Procs

Performs Lag-Llama forecasting on the underlying series.

This command requires the EViews Python library installed with EViews.

The `lagllama` command in EViews performs Lag-Llama automatic forecasting. This command is a wrapper around the Python library that uses a large language model (LLM) type infrastructure to perform time series forecasting.

Syntax

series_name.**lagllama**(*options*) *forecast_name [quantile_prefix]*

Options

nsamples = *int*	Set the number of forecast samples performed (default 200).
ctxtmult = *int*	Set the context multiplier (default 3).
quantiles = *"arg"*	Specify the forecast quantiles. By default, 0.1 and 0.9 quantiles are forecasted.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), or "na" (fill observations outside the forecast sample with missing values, default).
seed = *int*	Set the Python seed value for the random draws of the forecast.
save = *arg*	Save all forecast draws into a matrix in the workfile, with a name of *arg*.

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
elecdmd.lagllama(nsamples=100, ctxtmult=2, quantiles="0.2 0.8")
   elecdmd_f
```

Uses Lag-Llama to produce a forecast of the ELECDMD series, with 100 forecast draws, a context multiplier of 2, and storing the 20% and 80% quantiles. The point forecasts will be stored in the ELECDMD_F series.

Cross-references

See "Lag-Llama" on page 159 of *User's Guide I* for additional discussion.

lstm	Series Procs

Performs long short-term memory (LSTM) estimation and forecasting on the underlying series.

The procedure constructs lagged target and optional feature inputs, divides the learning sample into chronological training and validation portions, trains the network, and creates a forecast series with optional diagnostic output.

Syntax

series_name.**lstm**(*options*) [*forecast_name] @target target_name [@features feature_list] [@hyperparams hp_options]*

Enter a name for the output forecast series. If *forecast_name* is omitted, EViews creates an unused name based on the target series name and the suffix `_LSTM_F`.

The grouped arguments are:

@target target_name	Identifies the target series for the LSTM specification. In normal use, *target_name* is the same series as *series_name*.
@features feature_list	Optional space-delimited list of feature series or valid series expressions. The feature lag and normalization options are applied to each listed feature.
@hyperparams hp_options	Optional comma-delimited hyperparameter assignments. Fixed values and tuning controls are described under “Hyperparameter options” below.

Option names and keyword values are not case-sensitive. Quote option values that contain spaces, such as `dims_hidden="64 32"`.

For Boolean controls, the tables show only the form that changes the default: an unprefixed name enables a default-off feature, while no disables a default-on feature.

Options

Data and network specification

normalize_series = *arg*	Target-series normalization. *arg* may be `none`, `minmax` (scale to [• 1, 1]), `minmax01` (scale to [0,1]), `stdize` (standardize, default), `robust`, `log`, or `tanh`.
lag = *int*	Target-series lag setting (*default* = 0).
nolagrange	Use only the selected target lag; the full range *0, …, lag* is used by default.
nolagsasfeatures	Process the target as one feature sequenced across the included lags; lags are treated as separate features by default.
normalize_features = *arg*	Feature normalization. *arg* uses the same keywords as `normalize_series`; the default is `stdize`. This option is used only when *@features* is supplied.
lag_features = *int*	Lag setting applied to every additional feature (*default* = 0).
nolagrange_features	Use only the selected feature lag; the full range *0, …, lag_features* is used by default.
dims_hidden = *arg*	Space-delimited list of positive hidden-unit counts, one for each LSTM layer (*default* = "1"). For example, `dims_hidden` = "32" specifies one layer, and `dims_hidden` = "64 32" specifies two stacked layers.
horizon_fcast = *int*	Step-ahead target used for training (*default* = 1). A value of 1 trains a one-step-ahead model; a larger value trains the model for that selected horizon. The procedure creates one forecast series for the selected horizon.
epochs = *int*	Maximum number of training epochs (*default* = 500). Early stopping may end training before this limit.
ratio_train = *num*	Fraction of the learning sample assigned chronologically to training (*default* = 0.80).
ratio_validate = *num*	Fraction of the learning sample assigned chronologically to validation (*default* = 0.20). The training and validation ratios must sum to no more than one. If they sum to less than one, the remaining learning observations are unused.

Normalization parameters are estimated from the training portion of the learning sample and then applied unchanged to the validation and forecast observations. Larger lag settings reduce the effective number of usable observations.

Forecast sample and in-sample fill

The learning sample is the current workfile sample. The forecast sample begins with the observation immediately following the learning sample. Specify the forecast endpoint using one of the following options:

forclen = *int*	Number of forecast observations. The value must produce at least one out-of-sample forecast observation.
forc = "*date*"	Forecast endpoint date or observation label. If neither `forc` nor `forclen` is supplied, the endpoint is the end of the workfile range.
f = *arg*	In-sample fill for the output forecast series. *arg* may be `na`, `actual`, `static` (default), or `dynamic`. `static` uses actual lagged target values where available; `dynamic` recursively uses prior forecasts.

Learning controls

loss = *arg*	Loss function. *arg* may be `mse` (default), `mae`, `huber`, or `ce`.
weight_init = *arg*	Weight initialization. *arg* may be `zeros`, `ones`, `random`, `xavier` (default), `he`, `orthogonal`, or `user`.
regularization = *arg*	Regularization type. *arg* may be `none` (default), `l1`, or `l2`. Set the penalty strength with `regularization_decay` in the `@hyperparams` group.
gradclip = *arg*	Gradient-clipping method. *arg* may be `none`, `value`, or `norm` (default).
gradclip_norm = *num*	L2 norm limit when `gradclip=norm` (*default* = 1.0).
gradclip_lower = *num*	Lower elementwise clipping limit when `gradclip=value` (*default* = -5).
gradclip_upper = *num*	Upper elementwise clipping limit when `gradclip=value` (*default* = 5).
randomize	Use randomized contiguous learning subsamples instead of one fixed split (*default* = off).
randomize_count = *int*	Number of randomized repetitions for a non-tuned model (*default* = 10). Used only when randomize is selected.

noearly_stop	Disable validation-based early stopping, which is enabled by default and restores the best-performing weights.
early_stop_warmup = *int*	Number of validation-loss observations required before early-stopping decisions begin (*default* = 50).
early_stop_patience = *int*	Number of consecutive epochs without a qualifying validation-loss improvement before stopping (*default* = 10).
early_stop_mindelta_absolute = *num*	Minimum absolute validation-loss improvement that resets patience (*default* = 0.001).
early_stop_mindelta_relative = *num*	Minimum relative validation-loss improvement that resets patience (*default* = 0.001). A change satisfying either positive threshold is treated as an improvement.

Optimizer and learning-rate scheduler

gd = *arg*	Gradient-descent optimizer. *arg* may be `sgd`, `momentum`, `adagrad`, `rmsprop`, `adadelta`, `adam`, `adamw` (default), `nadam`, `radam`, or `ranger`.
rs = *arg*	Learning-rate scheduler. *arg* may be `none`, `step`, `exponential`, `cosannealing`, `plateau` (default), `cycle`, or `restarts`.
rs_patience = *int*	Number of epochs without validation improvement before a `plateau` scheduler reduction (*default* = 5).

Set the learning rate, optimizer coefficients, scheduler coefficients, and their tuning controls in the `@hyperparams` group.

Trained weights and output

weights_page = *name*	Workfile page used to export trained weights or identify weights to load. Use a new page name when exporting to avoid a conflict with an existing page.
noweights_export	Suppress weight export; export is enabled by default when `weights_page` is supplied.
weights_load	Load saved weights and associated model settings from `weights_page` (*default* = off). Restored settings include lags, normalization, features, and layer dimensions.
noplotloss	Suppress the training and validation loss graph, which is produced by default.

plotlrevo	Produce the learning-rate evolution graph. Disabled by default.
tableloss	Produce a table of loss values by epoch. Disabled by default.
tabletuner	Produce the hyperparameter tuner table. Disabled by default and relevant when at least one tuner is active.

General options

prompt	Force the LSTM dialog to appear from within a program.
p	Print the output view.

Hyperparameter Options

Supply hyperparameter assignments after `@hyperparams`; separate assignments with commas. Except for the hidden-unit special case noted below, the general form is:

```
@hyperparams root=value, root_tuner=key, root_min=value,
    root_max=value, root_gridlen=int
```

root = *value*	Set the fixed current value. For hidden units, current layer sizes are set by `dims_hidden`; use the remaining `dim_hidden_*` controls to tune those layer sizes.
root_tuner = *key*	Tuner type. *key* may be `none` (default), `sequence` (evenly spaced candidate grid), or `random` (random candidate grid).
root_min = *value*	Lower search bound used when the tuner is `sequence` or `random`.
root_max = *value*	Upper search bound used when the tuner is `sequence` or `random`.
root_gridlen = *int*	Number of candidate values generated for the root (*default* = 10).

The supported roots and their defaults are:

dim_hidden	Hidden units. Current sizes come from `dims_hidden`; search minimum 1, maximum 100, grid length 10. Hidden-unit tuning is applied separately to each layer.
gd_rate_learn	Optimizer learning rate. Default 0.001; search minimum 0, maximum 1, grid length 10. `gd_rate_learn_min` also provides the absolute lower bound used by learning-rate schedulers.
gd_momentum	Momentum or velocity coefficient, where used by the selected optimizer. Default 0.9; search minimum 0, maximum 1, grid length 10.
gd_decay1	First optimizer decay or smoothing coefficient. Default 0.9; search minimum 0, maximum 1, grid length 10.
gd_decay2	Second optimizer decay or smoothing coefficient. Default 0.999; search minimum 0, maximum 1, grid length 10.
gd_decay3	Additional optimizer-specific decay coefficient. Default 0.00025; search minimum 0, maximum 1, grid length 10.
rs_decay	Scheduler reduction factor for step, exponential, and plateau schedules. Default 0.5; search minimum 0, maximum 1, grid length 10.
rs_cycle_len	Cycle length for cosine annealing, cyclical scheduling, and warm restarts. Default 10; search minimum 10, maximum 50, grid length 10.
rs_cycle_step	Decay-step interval used by step and exponential schedules. Default 10; search minimum 10, maximum 50, grid length 10.
rs_cycle_min	Lower learning-rate bound for cyclic and cosine-based schedules. Default 1×10^{-7}; search minimum 0, maximum 1, grid length 10.
rs_cycle_max	Upper learning-rate bound for cyclic and cosine-based schedules. Default 0.1; search minimum 0, maximum 1, grid length 10.
regularization_decay	L1/L2 penalty strength. Default 0.1; search minimum 0, maximum 1, grid length 10. Used only when `regularization=l1` or `regularization=l2`.

Other tuned optimizer, scheduler, and regularization parameters are global to the model. When randomization and tuning are both active, each candidate configuration is evaluated using the randomized setup generated for that candidate.

Examples

The command

```
elecdmd.lstm(forclen=30, lag=14, dims_hidden="32") elecdmd_lstm
    @target elecdmd
```

estimates a one-layer LSTM with 32 hidden units using target lags 0 through 14, and creates 30 out-of-sample forecasts in ELECDMD_LSTM.

The command

```
elecdmd.lstm(forc="12/31/2015", lag=14, lag_features=0,
    dims_hidden="64 32", plotlrevo, tableloss) elecdmd_lstm @target
    elecdmd @features @month @quarter
```

uses two stacked LSTM layers, adds contemporaneous calendar features, sets the forecast endpoint, and requests the learning-rate graph and loss table.

The command

```
elecdmd.lstm(forclen=30, lag=14, dims_hidden="32", tabletuner)
    elecdmd_tuned @target elecdmd @features @month @quarter
    @hyperparams dim_hidden_tuner=sequence, dim_hidden_min=16,
    dim_hidden_max=64, dim_hidden_gridlen=4,
    gd_rate_learn_tuner=random, gd_rate_learn_min=0.0001,
    gd_rate_learn_max=0.01, gd_rate_learn_gridlen=8
```

searches a four-value sequential grid for hidden units and an eight-value random grid for the learning rate, selecting values using validation performance.

The commands

```
elecdmd.lstm(forclen=30, lag=14, dims_hidden="32",
    weights_page=lstm_w) elecdmd_f @target elecdmd
elecdmd.lstm(forclen=30, weights_page=lstm_w, weights_load,
    noweights_export, weight_init=user) elecdmd_f_reuse @target
    elecdmd
```

first estimate a model and export its trained weights to the workfile page LSTM_W, then load the saved weights and associated model settings to produce a second forecast series.

Cross-references

See "Long Short-Term Memory (LSTM)" on page 137 of *User's Guide I* for discussion of model construction, learning options, diagnostics, and practical specification guidance.

knots	Equation Views

View a table of the knot values from an equation estimated with splines.

Syntax

eq_name.**knots**

Examples

```
equation eq1.splinereg elecdmd temp
show eq1.knots
```

estimates an equation with the series ELECDMD as the dependent variable, and using a B-spline to model the TEMP series as a regressor, and then displays a table of the computed knot values.

Cross-references

See "Spline Regression" on page 261 of *User's Guide II* for additional discussion.

modavg	Equation Methods

Estimate using model averaging methods.

Estimate equation using weighted-average least squares (WALS) or Bayesian model averaging (BMA).

Syntax

eq_name.**modavg**(*options*) *y x1 [x2 x3...] @ z1 z2...*

Specify the dependent variable followed by a list of variables that appear in every model, followed by the *"@"*-sign and a list of variables that are not required to be present in every model.

Options

method = *arg*	Model averaging method: "wals" for weighted-average least squares (default), "bma" for Bayesian model averaging.

WALS Options

walsprior = *arg*	arg can be "laplace" (default), "subbotin", or "weibull".

BMA Options

msp = *arg*	Model space prior: "beta-binomial" (default) or "binomial".
a = *number* (*default* = 1)	First shape parameter in $p \sim \text{Beta}(a, b)$, where p is the probability of inclusion for any optional regressor. Only applicable when the beta-binomial prior is used as the model space prior.
b = *number* (*default* = 1)	Second shape parameter in $p \sim \text{Beta}(a, b)$, where p is the probability of inclusion for any optional regressor. Only applicable when the beta-binomial prior is used as the model space prior.
p = *number* (*default* = 0.5)	The prior probability of inclusion for any optional regressor. If prior inclusion probabilities are different between optional regressors, set p to a vector or list of values in the open unit interval.
gtype = *arg*	The method type used to set *g* in Zellner's *g*-prior. The options are "benchmark" (default), "uip" for the unit information prior, "ric" for the risk inflation criterion, "leb" for local empirical Bayes, and "user" for a user-specified value. If "user" is selected, the user must also set *gvalue*.
gvalue = *number*	The value for *g* in Zellner's *g*-prior, set by the user. To use this option, set *gtype* to "user".

Examples

The command

```
equation eq1.modavg(method=bma,msp=binomial,p="0.25 0.25 0.5 0.5")
   y c @ x1 x2 x3 x4
```

performs Bayesian model averaging with Y as the dependent variable, the intercept C as the sole focus regressor, and four optional regressors X1, X2, X3, and X4. The model space prior is set to the binomial distribution, and X1, X2, X3, and X4 are included with prior probabilities 1/4, 1/4, 1/2, and 1/2, respectively.

Cross-references

See "Model Averaging" on page 289 of *User's Guide II* for additional discussion.

modelranking	Equation Views

Display models ranked by posterior probability.

This view is only available for equations estimated using Bayesian model averaging (BMA).

Syntax

obj_name.**modelranking**(*options*)

Options

showtop = *int*	The number of top models to show. Default value is the number of models in the model space or 100, whichever is smaller.

Examples

```
eq1.modelranking(showtop=20)
```

ranks models by posterior probability and displays information about the top 20.

Cross-references

See "Model Averaging" on page 289 of *User's Guide II* for additional discussion.

modelsize	Equation Views

Display results pertaining to the size of the model.

This view is only available for equations estimated using Bayesian model averaging (BMA).

Syntax

obj_name.**modelsize**

Cross-references

See "Model Averaging" on page 289 of *User's Guide II* for additional discussion.

nprophet	Series Procs

Performs NeuralProphet forecasting on the underlying series.

Prophet requires the NeuralProphet Python library installed with EViews.

Syntax

series_name.**nprophet**(*options*) *forecast_name [quantile_prefix]*

Enter a name for the forecast series, and, optionally, a naming prefix for the quantile forecasts.

Options

growth = *arg*	Set the growth type. *arg* can be "linear" (default) or "log".
season = *arg*	Set the seasonality type. *arg* can be "add" (additive, default) or "mult" (multiplicative).
trendreg = *arg*	Set the trend regularization parameter.
seasreg = *arg*	Set the seasonal regularization parameter.
quantiles = *"arg"*	Specify the forecast quantiles. By default, 0.1 and 0.9 quantiles are forecasted.
arlags = *int*	Set the number of autoregressive lags to estimate in the AR-Net component.
arreg = *arg*	Set the AR component regularization parameter.
arlayers = *"list"*	Specify the number and size of the hidden layers in the AR-Net component. *list* should be a comma delimited list of layer sizes. For example, to include two layers with sizes 64 and 32, enter arlayers = "64, 32".
epochs = *int*	Set the number of epochs in the AR-Net component.
learnrate = *arg*	Set the learning rate in the AR-Net component.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), "na" (fill observations outside the forecast sample with missing values, default), or "forc" (fill training-data observations with in-sample forecasts, and other data with NA).
userregs = *"list"*	Include user-defined exogenous variables. "list" should be a space delimited list of series or EViews series expressions, surrounded by quotes.
userregmodes = *"list"*	Set the user defined exogenous variables modes. "list" should be a space delimited list of either "mult" or "add", where the number of terms in the list matches the numbers of terms in the userregs option.
userregpriors = *"list"*	Set the user defined exogenous variables prior scales. "list" should be a space delimited list scales, where the number of terms in the list matches the numbers of terms in the userregs option.
countryhol = *arg*	Include one of Prophet's built-in set of country holidays. *arg* should be the country code for the country.
countryreg = *arg*	Set the regularization parameter for the country holidays.

countrylow = *int*	Set the lower window for the country holiday. To specify days before the holiday, the integer should be a negative number.
countryhigh = *int*	Set the upper window for the country holiday.
countrymode = *arg*	Set the country holiday mode. *arg* should be either "mult" or "add" (default).
userhols = *"list"*	Include user defined events or holidays. "list" should be a space delimited set of dates or EViews holiday keywords
userhollows = *"list"*	Specify the set of lower windows for user defined events or holidays. "list" should be a space delimited set of integers specifying the number of days to offset the user holiday to specify the start of the holiday period. Note to specify days before the holiday, the integer should be a negative number.
userhol-highs = *"list"*	Specify the set of upper windows for user defined events or holidays. "list" should be a space delimited set of integers specifying the number of days to offset the user holiday to specify the end of the user holiday.
userholm-odes = *"list"*	Set the user holiday modes. "list" should be a space delimited list of either "mult" or "add", where the number of terms in the list matches the numbers of terms in the userhols option.
userholpri-ors = *"list"*	Set the user holiday prior scales. "list" should be a space delimited list scales, where the number of terms in the list matches the numbers of terms in the userhols option

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
elecdmd.nprophet(f=actual, countryhol=UK, userregs="gdp
    @log(weather") elec_f elec_f_q
```

This will execute the NeuralProphet routine on the ELECDMD series, inserting the actual values for out-of-forecast sample observations, including NeuralProphet's built-in holiday set for the United Kingdom, and including both GDP and the log of WEATHER as exogenous regressors. The forecasted values will be stored in the new series ELEC_F, with the 10% and 90% quantiles being stored in ELEC_F_Q_1 and ELEC_F_Q_9, respectively.

Cross-references

See "NeuralProphet" on page 116 of *User's Guide I* for additional discussion.

partialeffects	Equation Views

View a graph of the partial effects from an equation estimated with splines.

Syntax

eq_name.**partialeffects**

Examples

```
equation eq1.splinereg elecdmd temp
show eq1.partialeffects
```

estimates an equation with the series ELECDMD as the dependent variable, and using a B-spline to model the TEMP series as a regressor, and then displays a graph of the partial effects.

Cross-references

See "Spline Regression" on page 261 of *User's Guide II* for additional discussion.

placebo	Equation Views

View the placebo chart of a Regression Discontinuity in Time model.

Syntax

eq_name.**placebo** *[dates]*

If no dates are provided, EViews will automatically determine an appropriate set of placebo dates.

Examples

```
equation eq1.rdit log(conceptions) @ 2007m7
eq1.placebo 2006m3 2006m6 2008m3 2008m6
```

estimates a regression discontinuity in time model with the log of CONCEPTIONS as the outcome variable and July 2007 as the intervention date, and then displays the placebo chart with placebo dates of March 2006, June 2006, March 2008 and June 2008.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

postdraws	Var Procs

Put posterior draws for BVAR or BTVCVAR in current workfile.

BVAR Description

In the case of BVAR, posterior draws are copied, vectorized, then stored as rows in one or more newly created matrix objects. Named matrix objects are added to the workfile.

Each matrix object represents a different parameter block. The parameter blocks are: coefficient matrix B, error covariance matrix Σ, and unknown prior hyperparameters γ. The prior type determines whether or not a parameter block is relevant:

- B is relevant under all prior types.
- Σ is relevant under all prior types except for the Litterman/Minnesota prior.
- γ is only relevant under GLP.

You may be prompted to re-estimate the BVAR if posterior draws are unavailable.

Compatibility notes: this proc was introduced in EViews 15 and replaces `var::drawcoefs` and `var::drawrescov` from EViews 11 to 14.

BTVCVAR Description

In the case of BTVCVAR, posterior draws are copied and stored as series objects in a new page. Each series holds draws for a particular parameter.

EViews may prompt you to run the posterior sampler if draws are not available.

Syntax

BVAR Syntax

var_name.**postdraws**(*options*) *[matrix_name_coef] [matrix_name_ecov] [matrix_name_hype]*

where *matrix_name_coef*, *matrix_name_ecov*, and *matrix_name_hype* are used to name the matrix objects containing draws for B, Σ, and γ, respectively.

BTVCVAR Syntax

var_name.**postdraws** *new_page_name*

Options

BVAR Options

bylag	If included, lag coefficients are grouped by lag instead of by variable.
dropunstable	If included, draws from the unstable region of the parameter space are dropped.
burn = *number*	Burn-in proportion. Note that *number* is a value in the open unit interval, (0, 1). This option is only relevant if draws were obtained using Markov chain Monte Carlo (MCMC) methods. The default value is the burn-in value used for estimation.

Examples

If MYBVAR is a Bayesian VAR that uses a prior other than the Litterman/Minnesota prior, then

```
mybvar.postdraws(dropunstable) coefs ecovs
```

creates two named matrices, COEFS and ECOVS. Coefficient draws are stored as rows in COEFS, and error covariance matrix draws are stored as rows in ECOVS. Draws taken from the unstable region of the parameter space are excluded from both COEFS and ECOVS.

Cross-references

See "Bayesian VAR Models" on page 369 of *User's Guide II* for additional discussion.

postmarginals

Equation Views

Display the univariate marginals of the posterior distribution over parameters.

This view is only available for equations estimated using Bayesian model averaging (BMA).

This view is a spool with three sections. The first section shows posterior densities and posterior means for coefficients on required regressors. The second section shows posterior densities, posterior means, and posterior exclusion probabilities for coefficients on optional regressors. The third section shows posterior densities and posterior means for the error standard deviation and error variance.

Syntax

eq_name.**postmarginals**

Cross-references

See "Model Averaging" on page 289 of *User's Guide II* for additional discussion.

prophet	Series Procs

Performs Facebook's Prophet forecasting on the underlying series.

Prophet requires the Prophet Python library installed with EViews.

Syntax

series_name.**prophet***(options) forecast_name [lower_name upper_name]*

Enter a name for the forecast series, and optionally, a name for the lower and upper bounds of the forecast.

Options

cpscale = *number*	Set the change point prior scale.
growth = *arg*	Set the growth type. *arg* can be "linear" (default) or "log".
season = *arg*	Set the seasonality type. *arg* can be "add" (additive, default) or "mult" (multiplicative).
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), "na" (fill observations outside the forecast sample with missing values, default), or "forc" (fill training-data observations with in-sample forecasts, and other data with NA).
bounds = *arg*	Set the interval width of the lower and upper bounds.
userregs = *"list"*	Include user-defined exogenous variables. "list" should be a space delimited list of series or EViews series expressions, surrounded by quotes.
userregmodes = *"list"*	Set the user defined exogenous variables modes. "list" should be a space delimited list of either "mult" or "add", where the number of terms in the list matches the numbers of terms in the userregs option.
userregpriors = *"list"*	Set the user defined exogenous variables prior scales. "list" should be a space delimited list scales, where the number of terms in the list matches the numbers of terms in the userregs option.

countryhol = *arg*	Include one of Prophet's built-in set of country holidays. *arg* should be the country code for the country.
countryprior = *arg*	Set the prior scale for the country holidays.
userhols = *"list"*	Include user defined events or holidays. "list" should be a space delimited set of dates or EViews holiday keywords.
userhollows = *"list"*	Specify the set of lower windows for user defined events or holidays. "list" should be a space delimited set of integers specifying the number of days to offset the user holiday to specify the start of the holiday period. Note to specify days before the holiday, the integer should be a negative number.
userhighs = *"list"*	Specify the set of upper windows for user defined events or holidays. "list" should be a space delimited set of integers specifying the number of days to offset the user holiday to specify the end of the user holiday.
userholmodes = *"list"*	Set the user holiday modes. "list" should be a space delimited list of either "mult" or "add", where the number of terms in the list matches the numbers of terms in the userhols option.
userholpriors = *"list"*	Set the user holiday prior scales. "list" should be a space delimited list scales, where the number of terms in the list matches the numbers of terms in the userhols option.
prompt	Force the dialog to appear from within a program.

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
elecdmd.prophet(f=actual) elec_f elec_l elec_u
```

This will execute the Facebook Prophet routine on the ELECDMD series, inserting the actual values for out-of-forecast sample observations. The forecasted values will be stored in the

new series ELEC_F, with the lower and upper bounds to the forecast being stored in ELEC_L and ELEC_U respectively.

Cross-references

See "Facebook Prophet Forecasting" on page 111 of *User's Guide I* for additional discussion.

randomforest — Series Procs

Performs Random Forest style regression forecasting

Randomforest requires the EViews Python library installed with EViews.

The randomforest command in EViews allows users to perform Random Forest regression forecasting. This command is a wrapper around the Random Forest implementation found in the SciKit Learn library.

Syntax

series_name.**randomforest**(*options*) *forecast_name [exog_variables]*

Options

model = *arg*	Specify the type of Random Forest model to be used. *arg* can be "rf" (random forest, default), "dt" (decision tree), "et" (extra trees), or "gb" (gradient boosting).
lags = *int*	Set the number of lags to include in the model.
nodropseries	Do not drop series with internal missing values from the exogenous variables / factors list.
f = *arg*	Out-of-forecast-sample fill behavior: "actual" (fill observations outside the forecast sample with actual values for the fitted variable), "na" (fill observations outside the forecast sample with missing values, default), or "forc" (fill training-data observations with in-sample forecasts, and other data with NA).
trees = *int*	Set the number of trees to be used in the forest (not applicable if model = dt).
depth = *int*	Set the maximum depth of the trees.
leafsize = *int*	Set the minimum size of leaf nodes.
gain = *num*	Set the minimum gain required for a split to occur.
nobootstrap	Disable bootstrapping (not applicable if model = dt).
seed = *int*	Set the seed for random number generation.
learnrate = *num*	Set the learning rate for the gradient boosting model.

subsample = *num*	Set the subsample for the gradient boosting model.
sharp = *arg*	Specify the name of the workfile matrix object containing the output Sharp values
import = *arg*	Specify the name of the workfile matrix object containing the output Importance values

Forecast sample options

The forecast sample will start at the observation immediately after the estimation sample (the current workfile sample). The forecast endpoint is given by either:

forclen = *int*	Number of periods to forecast.
forc = "*date*"	Specify the date of the forecast end point.

If omitted, the end point will be the end of the workfile range.

Examples

```
elecdmd.randomforest(trees=100, depth=10, lags=2) elecdmd_f
   features_group
```

Performs a random forest forecast of the series ELECDMD based upon the exogenous variables in the group FEATURES_GROUP, using 100 trees, a maximum depth of 10, and including 2 lags in the model. The results will be stored in the ELECDMD_F series.

Cross-references

See "Random Forest" on page 163 of *User's Guide I* for additional discussion.

rdit	Equation Methods

Estimate of a regression discontinuity in time model.

Syntax

eq_name.**rdit**(*options*) *y [z1 z2...] @ idate*

List the outcome variable first, followed by any optional covariates, then an @ symbol followed by the date of the intervention/cutoff.

Options

polyest = *int*	Specify the estimation polynomial order, *p*. *int* can be 0, 1(default) or 2.
polybias = *int*	Specify the bias-correction polynomial order, *q*. *int* can be 1, 2 (default) or 3, and must be greater than the estimation polynomial order.
kern = *arg*	Specify the kernel shape used during estimation. *arg* can be "Triangular" (default), "Epanechnikov", or "Uniform".
bwselect = *arg*	Specify the bandwidth selection method. *arg* can be "mserd" (MSE-Optimal (RD), default), "msetwo" (MSE-Optimal (Two-sided)), "msesum" (MSE-Optimal (Sum)), "msecomb1" (MSE-Optimal (Combined 1)), "msecomb2" (MSE-Optimal (Combined 2)), "cerrd" (CER-Optimal (RD)), "certwo" (CER-Optimal (Two-sided)), "cersum" (CER-Optimal (Sum)), "cercomb1" (CER-Optimal (Combined 1)), "cercomb2" (CER-Optimal (Combined 2)), "user" (User-specified).
userbw = *arg*	If the *bwselect* option is set to "user", this option specifies the user bandwidths. *arg* should be a comma separated list of 4 values (hl, hr, bl, br) surrounded in quotes.
donutl = *int*	Set the left side donut size (default is 0).
donutr = *int*	Set the right side donut size (default is 0).
cov = *arg*	Specify the covariance method used. *cov* may be "ordinary", "white", "hc" (default), "hac".
hctype = *arg*	Specify the HC covariance method. Only applicable if cov = *hc* option is used. arg may take the values of "hc1", "hc2" (default), "hc3".
covlag = *arg* *(default* = 1*)*	Whitening lag specification: *integer* (user-specified lag value), "a" (automatic selection).
covinfosel = *arg* *(default* = "aic"*)*	Information criterion for automatic selection: "aic" (Akaike), "sic" (Schwarz), "hqc" (Hannan-Quinn) (if "lag = a").
covmaxlag = *integer*	Maximum lag-length for automatic selection *(optional)* (if "lag = a"). The default is an observation-based maximum of $T^{1/3}$.
covkern = *arg* *(default* = "bart"*)*	Kernel shape: "none" (no kernel), "bart" (Bartlett, *default*), "bohman" (Bohman), "daniell" (Daniel), "parzen" (Parzen), "parzriesz" (Parzen-Riesz), "parzgeo" (Parzen-Geometric), "parzcauchy" (Parzen-Cauchy), "quadspec" (Quadratic Spectral), "trunc" (Truncated), "thamm" (Tukey-Hamming), "thann" (Tukey-Hanning), "tparz" (Tukey-Parzen).

covbw = *arg* (*default* = "fixednw")	Kernel Bandwidth: "fixednw" (Newey-West fixed), "andrews" (Andrews automatic), "neweywest" (Newey-West automatic), *number* (User-specified bandwidth).
covnwlag = *integer*	Newey-West lag-selection parameter for use in nonparametric kernel bandwidth selection (if "covbw = neweywest").
covbwint	Use integer portion of bandwidth.

Examples

```
equation eq1.rdit log(conceptions) @ 2007m07
```

estimates a regression discontinuity in time model, with the log of CONCEPTIONS as the outcome variable, and July 2007 as the intervention date.

```
equation eq2.rdit(polybias=3, bwselect=user, userbw="6.5, 6.5,
   8.0, 8.5") log(conceptions) @ 2007m7
```

estimates the same model, but increasing the polynomial used for bias correction to an order of 3, and fixing the bandwidths to hl = 6.5, hr = 6.5, bl = 8.0, br = 8.5.

Cross-references

See "Regression Discontinuity in Time (RDiT)" on page 273 of *User's Guide II* for additional discussion.

splinedecomp	Series Procs

Perform seasonal adjustment of a series using a spline decomposition.

Syntax

series_name.**command**(*options*) *[seasonal_spec] seasadj_name [factor_name trend_name]*

You should follow the `splinedecomp` keyword with an optional seasonality specification, followed by the name for the seasonally adjusted series. Optionally, you may also provide a name for the output factor series and trend series.

Options

tform = *arg*	Transform the underlying data before performing the spline decomposition. *arg* may be "none" (default), "box-cox", "yeo-johnson", "log", "square-root", "arcsin", "auto".
tform-lambda = *num*	Set the power level of the Box-Cox and Yeo-Johnson transformations.
trmethod = *arg*	Specify the spline method used for the trend component of the decomposition. *arg* may be "bspline", "polyspline", "pspline" (default).
trdegree = *int*	Set the degree of the trend spline component. Default value is 3.
trpenalty = *num*	Set the p-spline penalty for the trend component (only applicable if trmethod set to "pspline"). Default value is 0.8.
noseas	Do not include a seasonal component. By default EViews will determine an appropriate seasonal component to include in the decomposition.
noseastest	Do not perform the seasonality test.
noseasplot	Do not display the seasonal plots.

Seasonal Specification

Use the `@seas` keyword, followed by options, to specify a seasonal cyclical spline component. Multiple seasonal components can be included through the use of multiple `@seas` specifications.

Seasonal Options

degree = *int*	Set the degree of the cyclical spline.
cycle = *int*	Set the periodicity of the cyclical spline.

Examples

```
houstnsa.splinedecomp houst houst_f houst_tr
```

performs a spline decomposition seasonal adjustment on the series HOUSTNSA, producing a seasonally adjusted series named HOUST, seasonal factors stored into the series HOUST_F, and the trend component stored into HOUST_TR.

```
houstnsa.splinedecomp(tform=yeo-johnson, tformlambda=3) houst
    houst_f houst_tr
```

performs the same decomposition, but transforms the HOUSTNSA series using the Yeo-Johnson transformation with lambda = 3.

```
sp500.splinedecomp(trdegree=2) @seas(degree=2, cycle=366)
   @seas(degree=3, cycle=7) spsa
```

seasonally adjusts the series SP500, setting the trend component's spline degree to 2, and including two seasonal components, one with a degree of 2 and a cycle of 366, and one with a degree of 3 and a cycle of 7.

Cross-references

See "Spline Decomposition Seasonal Adjustment" on page 250 of *User's Guide I* for additional discussion.

splinereg — Equation Methods

Estimate an equation using nonparametric splines.

Syntax

eq_name.**splinereg**(*options*) *y x1 [x2 x3...] [@ z1 z2...]*

List the dependent variable, followed by a list of nonparametric exogenous regressors to be modeled with splines. Optionally you may use the "@" symbol followed by a list of linear exogenous regressors. You should not include a constant in the specification.

Options

method = *arg*	Select the spline type. arg can be "bspline" (default), "polyspline" or "pspline".
degree = *int*	Specify the degree of the splines.
knotsloc = *arg*	Specify the location selection method for knot location. *arg* can be "auto" (automatic choice between "quantile" or "uniform"), "quantile" (quantiles of the underlying variable, default), "uniform" (uniform spread between underlying variable's min and max values), or a space delimited list of values, or the name of a workfile vector object containing knot values.
knotsnum = *arg*	Specify the quantity of internal knots. arg can be "auto" (automatic selection, default), or a fixed integer.
maxknots = *arg*	Maximum number of internal knots for automatic selection.
boundaryadjust = *num*	Specify the scale factor for extending boundary knots beyond the min/max of the underlying variable. Default is 0.