## EViews ${ }^{\circledR} 13$ Beta Documentation

# EViews ${ }^{\circledR} 13$ Beta Documentation 

Copyright © 1994-2022 S\&P Global Inc.
All Rights Reserved
Printed in the United States of America

This software product, including program code and manual, is copyrighted, and all rights are reserved by S\&P Global Inc. The distribution and sale of this product are intended for the use of the original purchaser only. Except as permitted under the United States Copyright Act of 1976, no part of this product may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of IHS Global Inc.

## 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 ${ }^{\circledR}$ 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.

GZipHelper - Copyright © 1995-2002 Gao Dasheng dsgao@hotmail.com.
jsonCPP Library - Copyright © 2007-2010 Baptiste Lepilleur and The JsonCPP Authors.
openssl Library - Copyright ${ }^{\oplus}$ 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.
libssh2 Library - Copyright ${ }^{\oplus}$ 2004-2007 Sara Golemon sarag@libssh2.org, Copyright ${ }^{\oplus}$ 2005,2006 Mikhail Gusarov dottedmag@dottedmag.net, Copyright © 2006-2007 The Written Word, Inc.,

Copyright © 2007 Eli Fant elifantu@mail.ru, Copyright ${ }^{\circledR} 2009$ Daniel Stenberg, Copyright ${ }^{\odot}$ 2008, 2009 Simon Josefsson. All rights reserved.
ssleay License - Copyright © 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.
Tableau Data Extract API - Copyright © 2003-2017 Tableau and its licensors. All rights reserved.
Tramo/Seats - Copyright (c) 1996 Agustin Maravall and Victor Gomez. Windows version developed by G. Caporello and A. Maravall (Bank of Spain)
X11.2 and X12-ARIMA version 0.2.7 and X-13ARIMA-SEATS - Copyright (c) U.S. Census Bureau. zlib Data Compression Library - Copyright © 1995-2017 Jean-loup Gailly and Mark Adler.
Notices, terms and conditions pertaining to third party software are located at http:// www.eviews.com/thirdparty and incorporated by reference herein.

[^0]
## Table of Contents

Chapter 1. EViews 13 Beta Documentation Overview ..... 1
Beta Documentation Notes ..... 1
What's New in EViews 13? ..... 1
Chapter 2. General EViews Interface ..... 5
New Pane and Tab User Interface ..... 5
Program Debugging ..... 7
Programming Dependency Logging ..... 9
Jupyter Notebook Support ..... 10
Chapter 3. Graphs and Tables ..... 13
Graph Line and Shade Transparency ..... 13
Custom Graph Data Labels ..... 17
Customizable Geomap Labels ..... 20
High-Low-Median Colormap Preset ..... 22
Fixing Rows and Columns in Tables ..... 25
Chapter 4. Econometrics and Statistics ..... 29
ARDL Estimation ..... 29
Pool Mean Group (PMG) Estimation ..... 32
Difference-in-Difference Estimation ..... 33
Enhanced VEC Estimation ..... 34
Bayesian Time-varying Coefficient Vector Autoregression ..... 37
Cointegration Testing ..... 40
Diagnostics in ARDL ..... 41
Diagnostics in Panel ARDL/PMG ..... 49
Enhanced Impulse Response Display ..... 53
Chapter 5. Data Handling ..... 63
Data Handling ..... 63
Data Sources and File Formats ..... 69
Matrix Language Support ..... 81
Preliminary Updates to Function Reference ..... 91
Preliminary Updates to Command Reference ..... 105
Daily Seasonal Adjustment ..... 301
Background ..... 301
Performing Daily Seasonal Adjustment in EViews ..... 304
Example ..... 309
References ..... 314
Linear and Nonlinear ARDL ..... 315
Background ..... 315
Estimating ARDL and NARDL in EViews ..... 320
Views and Procs of ARDL ..... 324
Examples ..... 335
References ..... 359
Vector Error Correction Models (VECMs) ..... 361
Cointegration ..... 361
The VECM Specification ..... 362
Estimating VEC Models in EViews ..... 373
Examples ..... 379
References ..... 398
Difference-In-Difference Estimation ..... 399
Background ..... 399
Estimating DiD in EViews ..... 402
Examples ..... 411
References ..... 422
Bayesian Time-varying Coefficients VAR Models ..... 423
Background ..... 423
Estimating BTVCVAR in EViews ..... 425
Working with a BTCVAR ..... 428
Implementation Details ..... 430
Example ..... 433
References ..... 435
INDEX ..... 437

## Chapter 1. EViews 13 Beta Documentation Overview

The EViews development team is pleased to announce that EViews 13 is now available for beta testing.

The Beta Version will not run unless you already have EViews 12 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 13.

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 13 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 June 3, 2022.
This document contains preliminary documentation for some (not all) of the new features in EViews 13.

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 13?

EViews 13 features a wide range of exciting changes and improvements. What follows is a list of a few of the most important new features. (Note that items may appear in more than one category.) This is not a complete list of the new features in EViews 13.

## General EViews Interface

- Alternative graphical user interface ("New Pane and Tab User Interface," on page 5).
- Debugging tools for EViews programs ("Program Debugging" on page 7).
- Program dependency tracking ("Programming Dependency Logging" on page 9).
- Jupyter Notebook Support ("Jupyter Notebook Support" on page 10).


## Graphs and Tables

- Line and shade transparency in graphs ("Graph Line and Shade Transparency," on page 13).
- Custom graph data labels ("Custom Graph Data Labels," on page 17).
- Customizable Geomap labels ("Customizable Geomap Labels," on page 20).
- High-low-median colormap preset ("High-Low-Median Colormap Preset," on page 22).
- Fixed row and column display in tables ("Fixing Rows and Columns in Tables," on page 25).


## Estimation and Analysis

- Enhanced Autoregressive Distributed Lag (ARDL) support featuring estimation of Nonlinear ARDL models and new diagnostics ("ARDL Estimation" on page 29).
- Improved Pool Mean Group (PMG) estimation featuring expanded deterministic trend support, estimation with Nonlinear ARDL terms, and new diagnostics ("Pool Mean Group (PMG) Estimation" on page 32).
- Difference-in-Difference (DID) estimation and diagnostics ("Difference-in-Difference Estimation" on page 33).
- Enhanced Vector Error Correction (VEC) estimation, featuring improved support for deterministic regressors ("Enhanced VEC Estimation" on page 34).
- Bayesian Time-varying Coefficient Vector Autoregression (BTVCVAR) models ("Bayesian Time-varying Coefficient Vector Autoregression" on page 37).


## Testing and Diagnostics

- Improved cointegration testing, featuring improved support for deterministic regressors ("Cointegration Testing" on page 40).
- New diagnostics in ARDL equations ("Diagnostics in ARDL" on page 41).
- New diagnostics in panel ARDL/PMG equations ("Diagnostics in Panel ARDL/PMG" on page 49).
- Extended VAR/VEC impulse response confidence interval calculation and display ("Enhanced Impulse Response Display" on page 53).


## Data Handling

## Data Handling

- Daily data seasonal adjustment ("Daily Data Seasonal Adjustment" on page 63).
- New and improved Excel file writing engine ("New Excel File Writing Engine" on page 65).
- Enhanced holiday family of functions to return the proportion of an annual event associated with each observation ("Holiday Functions" on page 65).


## Data Sources and File Formats

- Australian Bureau of Statistics SDMX ("SDMX Databases," on page 69).
- Deutsche Bundesbank SDMX ("SDMX Databases," on page 69).
- Insee SDMX ("SDMX Databases," on page 69).
- Trading Economics ("Trading Economics," on page 72).
- World Health Organization ("World Health Organization," on page 77).


## Matrix Language

- Improved data import and export from matrix objects ("New Data import/export engine," on page 81).
- Expanded support for row and column labeling ("Row and Column Label Support," on page 83).
- Improved data access to matrix object data ("Matrix Data Access," on page 86)


## Preliminary Updates to Function Reference

- Preliminary list of new and updated functions ("Preliminary Updates to Function Reference" on page 91).


## Chapter 2. General EViews Interface

EViews 13 features exciting new interface improvements to improve the general EViews interactive and programming environment, and to support complementary external interfaces:

- Alternative graphical user interface ("New Pane and Tab User Interface," on page 5).
- Debugging tools for EViews programs ("Program Debugging" on page 7).
- Program dependency tracking ("Programming Dependency Logging" on page 9).
- Jupyter Notebook Support ("Jupyter Notebook Support" on page 10).


## New Pane and Tab User Interface

The familiar EViews multiple window interface offers users many advantages, especially on large computer screen displays. In some small screen settings, however, it can be more difficult to utilize fully the advantages of having multiple windows open at the same time.

EViews 13 offers a new, alternative user interface mode that employs panes and tabs in places of multiple windows. The built-in organization properties of this interface may be ideally suited to smaller display environments.

To enable or disable the pane and tab mode, click on General Options/Environment/Appearance and select the Use panes \& tabs checkbox to enable the new mode:


In the pane and tab UI mode, different types of windows will appear in docked panes inside the EViews window:

- Workfile and database windows are displayed in a pane on the left-hand side of the window
- Program windows are displayed in a pane on the right-hand side of the window
- Object windows are displayed in a pane in the center of the window


When you open multiple workfiles or multiple programs, previously opened windows will appear as tabs in the corresponding docks. You may bring focus to a specific window by clicking on the corresponding tab.

When you open multiple object windows, the previously opened windows will appear as tabs in the object pane. You may bring focus to an object by clicking on the down arrow at the upper right of the object dock and selecting the desired object, or by selecting Window from the main EViews menu and clicking on the object name in the list of opened objects.

To maximize screen real estate further, you may place the workfile/database and the program panes in drawers. Placing a pane in a drawer temporarily hides the pane while retaining quick, on-demand access to the pane.

- Click on the pin icon at the top of a docked pane to hide it in a drawer on the side of the window. The pane window will close and be replaced by a drawer label on the side of the window. Hovering the cursor over the drawer label will open the drawer and display the pane window. Clicking away from the pane will close the drawer. Click again on the pin to remove the pane from the drawer and open the docked pane window.


## Program Debugging

EViews 13 now offers tools for debugging an EViews program to help you to identify issues or locate the source of problems. The debugging tools allow you to set breakpoints on specific lines, run the program until it hits that breakpoint, and then examine at the state of your workfile or variables at that point in the program execution.

## Setting Breakpoints

Open the EViews program file, then set the breakpoint on a given line by clicking in the left (next to the line number):

| P- Program: CALL_EXEC - (c: \iles\call_exec.prg) |  |  |  |  |  |  |  | 回 | $x$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Run | Debug | Print | Save | SaveAs | Snapshot | Wrap $+/-$ | LineNum+/- | Encrypt |  |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | wfop wfcr dbo fetch fetch <br> inclu <br> for ! <br> seri <br> next <br> seri | tq.w <br> te u 1 <br> n dbte <br> 1 <br> 2 <br> c:Ifil <br> 1 to 1 <br> wli = $j=r n$ |  |  |  |  |  |  |  |

A red dot will appear next to lines with breakpoints set. Clicking on the red dot will clear the breakpoint.

## Starting a Debugging Session

To begin debugging the program, click the Debug button in the toolbar, enter any program arguments, specify whether to Log dependencies and if desired, change the Maximum errors before halting. Click on OK.

EViews will start program execution and will open the debugging pane. There are three areas in the pane: Breakpoints, Watch, and Callstack. Clicking on the name in one of the tabs will display the corresponding area.

The breakpoints window shows all defined breakpoints. You may make a breakpoint temporarily inactive or restore active status by toggling the checkbox next to the breakpoint name.

## Stopped at a Breakpoint

When the program reaches an active breakpoint, the execution will pause and the red dot will be highlighted, in yellow:


At this point you can look at the Callstack or Watch windows for relevant information.

- The Callstack will display information about the active program files and subroutines of the EViews program.
- The Watch window may be used to examine the values of program and replacement variables. Simply enter the desired

Additionally, you may open EViews objects such as series or equations to examine their current states.

## Resume / Step / Run / Stop

To resume the execution, click the Resume button. The EViews program execution will continue until it hits the next breakpoint.

You can also execute just the current line by clicking the Step button.
To run the program to completion without stopping at breakpoints, click the Run button.
You can also cancel the program execution at this point by clicking the Stop button.

## Restrictions during Debugging

During a debugging session, EViews will not allow you to close any windows that were opened by the program as this could negatively affect the program execution.

## Programming Dependency Logging

EViews 13 has a new feature to automatically log a program's external dependencies (e.g. workfiles, databases, and other programs).

This feature is available when you run a program under EViews 13 by checking the Log Dependencies checkbox in the Run Program dialog:


During the run of this program, a new Program Dependencies window will appear, showing all of the external dependencies that have been detected during the run, with information on the source of the dependency and the calling information. For example:

shows dependence on the external program files "xy.prg", "wz.prg" and "z22.prg" along with the workfiles "tq.wf1" and "untitled.wf1", and the database "dbtest.edb". Further, the dependency log shows the program file and line number producing the dependency.

## Jupyter Notebook Support

Jupyter is a web-based interactive development environment (https://jupyter.org/) that allows users to create notebooks for documenting computational workflow. EViews 13 can now be used as a Jupyter kernel. This means you can use Jupyter Notebook to run and organize an EViews program and display results from within the Jupyter Notebook.

## Publishing the EViews Jupyter Kernel

Once you have Jupyter Notebook installed, you can make the EViews kernel available to Jupyter by going to the main EViews menu and selecting Options/General Options/External program interface, then click the Publish Jupyter Notebook button.


This will create a new EViews specific folder under the standard Jupyter Kernels folder location, usually found at "\%AppData\%/Roaming/jupyter/kernels".

## Launching EViews Jupyter Kernel

Once the EViews kernel has been published, you can select it from the Jupyter home page.

## Running EViews Commands

Once the Jupyter Notebook is started, you can type in EViews commands into the textboxes, then SHIFT-ENTER to run them.

## Chapter 3. Graphs and Tables

EViews 13 introduces a number of productivity enhancing improvements to the graph and table presentation toolkit. The following is a brief outline of the most important new features, followed by discussion and links to preliminary documentation.

- Line and shade transparency in graphs ("Graph Line and Shade Transparency," on page 13).
- Custom graph data labels ("Custom Graph Data Labels," on page 17).
- Customizable geomap labels ("Customizable Geomap Labels," on page 20)
- High-low-median colormap preset ("High-Low-Median Colormap Preset," on page 22).
- Fixed row and column display in tables ("Fixing Rows and Columns in Tables," on page 25).


## Graph Line and Shade Transparency

EViews 13 now allows you to customize the opacity (transparency) levels of individual lines and shades in a graph. By exercising fine control over the visibility of stacked graph elements, you can uncover previously hidden features of your data.

When plotting multiple data for series in prior versions of EViews, the graph for data from one series could obscure the data of another. For example, the following area graph shows time series graphs for three series in a group object (ALLIED, DUPONT, and EXXON):


In this graph, data for ALLIED is drawn first, followed by the data for DUPONT, and then by the data for EXXON. Notice that the areas for the later drawn series obscuring the areas for the earlier. Note in particular, that the values of EXXON for observations hide the corresponding values of DUPONT and ALLIED, particularly for observations from 9 to 16.

EViews 13 allows you to adjust the opacity of individual graph elements to improve the visibility of others. To set the opacity for one or more elements, double click on the graph to display the graph options, or select on the Options button on the button bar, then select the Graph Elements node.


Opacity levels may be set for individual Lines and Fill Areas by clicking on an entry in the right hand side of the graph to select a graph element and then entering a number from 0 to 100 in the Opacity \% edit field:

You may set levels for more than one element simultaneously by SHIFT or CTRL clicking to select multiple elements, and then applying the desired setting.

Here is the same graph after setting the Opacity \% of all three of the fill elements to " 50 ":


Note the additional visible detail in the values of the ALLIED and DUPONT series for observations 9 to 16. In particular, the values of ALLIED, the first drawn series were virtually invisible in the earlier graph, but now show an obvious sawtooth pattern.

Similarly, turning down the opacity in a dense scatterplot can show additional detail:


Judicious use of transparency settings makes possible the production of graphs that show data in ways that were not previously possible. The new multiple shaded confidence intervals in combined impulse-response graphs ("Enhanced Impulse Response Display" on page 53) employ these settings and hint at the types of graphs that may be produced:


To set opacity levels by command use the setelem (p. 240) command to select an element and add the lineopacity (for lines) and fillopacity (for fills) keyword to set the opacity value.

Setting the level to 0.0 ( $0 \%$ ) will make the object completely transparent while 1.0 ( $100 \%$ ) will make the object completely opaque.

For example,

```
gr1.setelem(2) lineopacity(.5)
```

In our example, the second line in the graph was set to be 0.5 ( $50 \%$ ) opaque.

## Custom Graph Data Labels

Providing labels for data values can be an important tool for enhancing the information content of graphical presentation of data. EViews 13 offers new automatic tools which make it easy to augment your graphs with informative custom data and observation-based labels.

Recall that in EViews 12 and earlier, EViews offered option to use data values to label each of the observations in a graph.


While often quite useful, this option had limitation. For one, labeling observations with data values was an all or nothing proposition; data values were either displayed for all observations or for none of the observations. Further, customization of the format of the data labels was limited to specifying a size and font of the data label, and showing or not showing an open or closed circle symbol alongside the label.

EViews 13 enhances the ability to label observations to provide you with greater control over your data labeling. You may now:

- Label all observations, no observations, the first observation, or the last observation.
- Specify the content of the data label by using the data value, the observation, the name of the series, or arbitrary text.
- Modify the font, font size and position of the data label.

Data labeling is controlled in the Graph Options dialog. Click on the Options button or dou-ble-click on the graph to display the dialog then select Graph Elements and Lines \& Symbols and select one or more the elements on the right-hand side of the dialog The Symbol/

Data label settings will be displayed when you choose Symbols or Lines \& Symbols to display. To show data labels, select one of the Data label entries in the drop-down control:


Here, we have selected the Data label with a closed circle symbol. By default, when you choose this setting, EViews will display labels for all observations.

EViews 13 introduces the ability to customize the data label. Click on the large Options button under the Data Label entry in the middle of the dialog to display the Label Options dialog:


The first tab provides basic labeling options:

- The radio buttons in the bottom left allow you to choose between displaying labels for All, the First, or the Last observation.
- The Position drop down lets you choose to display the label using Auto positioning, or to the Right of observation, Left of observation, Above observation, Below observation, or Center on observation.
- The Label Format edit field allows you to specify the content of the label. You should enter text for the specification in the edit field. The special functions @xlabel () and @ylabel() instruct EViews to use the corresponding X-values and Y-values of the observations as the label; the function @legend () corresponds to the legend value for the data element. All other text will be used in the label as given. You may use the ENTER key format the label in multiple lines.

The second tab of the dialog specifies font family, size, and color settings, as well as special text effects like underlining and strikethrough.

Below, we use the Last observation setting to label the last data point using the Y-value to form a custom data string "\% growth (actual)", by entering

```
@ylabel()% growth
(actual)
```

in the edit field. Note that the carriage return in the edit field is used to create a multi-line label.


Importantly, if an observation were to be added to end of the graph, the data label would automatic change to reflect the new value as well as move to proper location above new observation.

Similarly, you may use custom data labels in place of a legend. In this example, the legend was disabled and we activate the custom data label for the last observation, using a

## @legend ()

specification in the edit field.


Note that a custom data label differs from a custom text label. Data labels are attached to an observation in the graph, while custom text labels may be placed anywhere in the graph. While custom text labels may be placed so that they appear related to specific data values, it takes a bit of work to tie them to the observation data values. In contrast, a custom data label is designed to be linked to the observations so it may easy to attach label text to the first, last, or all of the observations.

## Customizable Geomap Labels

In EViews 13, you may now use more than one attribute, along with custom text and formatting, to label the shapes in your geomap.

Note that the ability to attach labels to the shapes in a geomap is one of the most important tools for customizing the display of area data. In the simplest example, we may display the name of a geographic region in each area of a geomap display. For example, we may display the county names in our geomap display of the county areas in the State of California, USA.


Here, the county names are one of several attributes of the geomap shapefile data, which pairs each shape are with one or more pieces information. Each shape in the file might have information on "Name", "Population", "Income", "Population \% change", etc.

In EViews 12, you could display a geomap with a single label taken from a single attribute, as in the example above. So while you could instruct EViews to use the "Name" attribute to display county name, or the "Population" attribute to show population, you could not display labels showing both county name and population in each shape.

EViews 13 removes this single attribute restriction, allowing you to use multiple attributes to construct an area label. Furthermore, you may add custom text to your label, and apply custom formatting including, font family and size, and multi-line display.


Should the default position of a label not be ideal, it may be adjusted manually. Simply click and drag a label to its desired location:


Note the improved labeling of the offshore areas in Santa Barbara, Ventura, and Los Angeles counties.

For command support, see

- setjust (p. 259) to set the display justification for multi-line area labels.
- setshapelabel (p. 263) to set which attribute to use or create a custom label to use when labeling shapes.


## High-Low-Median Colormap Preset

EViews supports colormaps where you can define sets of rules that translate numeric values into colors. These colormaps may then be used when setting the text or fill color for series or group spreadsheets, or the fill colors in geomaps.

EViews 12 offers a number of predefined colormaps for common settings, such as the Posi-tive-negative colormap which negative values will be displayed as red and positive values will be displayed as black. In addition to EViews 12 pre-specified presets, EViews 13 includes a new preset which allows users to identify the high, low, and median observations.

To apply this preset to a displayed series spreadsheet, click on Properties then select the Text Color or Fill Color tab as desired:


Choose High-Low-Median in the Type dropdown, and choose the colors appropriately. Click on OK to display the spreadsheet with the colormap applied:

| SJ Series: SERIESO1 |  |  | Workfile SCATTER2:Untitled |  |  |  | $\square \square^{\circ} \square$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc 0 | Object | Properties | Print | Name | Freeze | Defout | $\checkmark$ | Sort | Edit-/ | Smpl-/. | Adjust - / | tat |
| 925.0327 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8/19 | 9/2005 |  | 1197.681 |  |  |  |  |  |  |  |  |  | $\wedge$ |
| 8/26 | 6/2005 |  | 1214.363 |  |  |  |  |  |  |  |  |  |  |
| $9 / 02$ | 2/2005 |  | 1226.443 |  |  |  |  |  |  |  |  |  |  |
| $9 / 09$ | 9/2005 |  | 1259.893 |  |  |  |  |  |  |  |  |  |  |
| 9/16 | 6/2005 |  | 1136.483 |  |  |  |  |  |  |  |  |  |  |
| $9 / 23$ | 3/2005 |  | 1145697 |  |  |  |  |  |  |  |  |  |  |
| $9 / 30$ | 0/2005 |  | 1061.173 |  |  |  |  |  |  |  |  |  |  |
| 10/07 | 7/2005 |  | 1023.376 |  |  |  |  |  |  |  |  |  |  |
| 10/14 | 4/2005 |  | 1013.900 |  |  |  |  |  |  |  |  |  |  |
| 10/21 | 1/2005 |  | 1034.684 |  |  |  |  |  |  |  |  |  |  |
| 10/28 | 8/2005 |  | 1036.761 |  |  |  |  |  |  |  |  |  |  |
| 11/04 | 4/2005 |  | 1012.857 |  |  |  |  |  |  |  |  |  |  |
| 11/11 | 1/2005 |  | 958.9440 |  |  |  |  |  |  |  |  |  |  |
| 11/18 | 8/2005 |  | 925.0327 |  |  |  |  |  |  |  |  |  |  |
| 11/25 | 5/2005 |  | 977.1677 |  |  |  |  |  |  |  |  |  |  |
| 12/02 | 2/2005 |  | 989.6823 |  |  |  |  |  |  |  |  |  |  |
| 12/09 | 9/2005 |  | 980.0910 |  |  |  |  |  |  |  |  |  |  |
| 12/16 | 6/2005 |  | 963.4054 |  |  |  |  |  |  |  |  |  |  |
| 12/23 | 3/2005 |  | 969.5059 |  |  |  |  |  |  |  |  |  |  |
| 12/30 | 0/2005 |  | 947.6603 |  |  |  |  |  |  |  |  |  | $\checkmark$ |
|  |  | 6 |  |  |  |  |  |  |  |  |  | , |  |
| Mean: 925.033 |  |  | Min: 925.033 | P | Std De | v: 0 | Probabille |  |  |  |  |  |  |

For geomaps you will click on Properties, then select the Color tab and choose High-LowMedian in the Type dropdown,


Click on OK to display the geomap with the colormap applied:


- See the existing EViews documentation for discussion of setfillcolor.

Applying a high-low-median colormap is done using the existing setfillcolor proc and the " $\mathrm{t}=$ hilo" option:

```
geomap01.setfillcolor(t=hilo) mapser(POP_TOTAL) min(-352258.35)
        max(7408557.35) highclr(@RGB(255,168,168))
        lowclr(@RGB(255,255,128)) medianclr(@RGB(255,190,96))
```


## Fixing Rows and Columns in Tables

EViews 13 now allows you to fix rows and columns in the display of a table so that these rows and columns are always in view as you scroll the remaining cells of the table.

There are times when it is useful to lock one or more rows or columns in a table so that those rows and columns are always in view. Depending on the contents of the table, the first few rows or columns in your table may contain identifying information about the data cells within the table. When searching for a specific cell or set of cells, fixing the display of these identifiers makes this task much easier. This is especially useful for tables containing a large number of columns or rows.

This situation often occurs, for example, when working with tables where the first row contains column names and the first column contains observation IDs. In the case when the table is scrolled both vertically and horizontally such that the $20^{\text {th }}$ column and the $100^{\text {th }}$ row of a table is in view it may be helpful to see the column names and observation IDs to have provide context to the contents of the cell.

Fixing columns and rows in a table can be done by pressing the Proc button in a table object button bar. Then,

- To fix only the first row, select Fix Row-Column/Fix 1 Row. If more than 1 row is to be fixed, select a cell in the first row after the last desired fixed row and then from the Proc menu select Fix Row-Column/Fix \# Rows. For example, to fix the first 3 rows of table, select a cell in the fourth row and then select Fix Row-Column/Fix 3 Rows.
- The process for fixing columns is the same as fixing rows with the obvious exception of selecting
- Fix Row-Column/Fix \#Columns - Similarly, after navigating through the proc menu, the columns proceeding the current selected cell will be fixed. Note the fixed columns and rows will be denoted in the table by a black cell border.

In the image below, you can see the date column (column A) and the series names (row 1) are in view despite the table being scrolled horizontally 3 columns where column $F$ is the first data column and vertically scrolled 447 rows where row 448 is the first row.

The fixed columns are denoted by a black cell border that extends from the top to bottom of the table and the fixed rows are separated by a similar black border that extends from the left to right of the table.

With fixed headers and rows, we can easily identify the selected cell as being the value for Spain on March $21^{\text {st }}, 2014$.


In this second example, the fixed first two columns and the fixed first row make it easy to identify the LCARPCAP values for Turkey from 1960 to 1978 despite the fact that the displayed values are for the $288^{\text {th }}$ row and $12^{\text {th }}$ column ( M column) of the table.


To unfix any fixed rows or columns, simply press the Proc button and either select Fix RowColumn/Remove Fixed Columns or Remove Fixed Rows. Note: the cell selection is ignored when unfixing rows or columns.

Note that fixed rows and columns are saved with the table when the workfile is saved to disk.

For command support, see:

- fixcol (p. 179)
- fixrow (p. 179)
- fixrowcol (p. 180)

Fixing the beginning set of columns in table is accomplished via the table name followed by the fixcol proc and the number of columns to be fixed. This procedure force the number of specified columns to always be in view regardless of the horizontal scroll position. Once fixed, a black cell border will separate the fixed and unfixed rows and columns.

Similarly, to fix the beginning set of rows in the table use the fixrow keyword followed by the number of rows to be fixed.

For example, the commands

```
tab1.fixcol 2
tab1.fixrow 1
```

will fix the first two columns and the first row of the table.
Alternately you may fix both the row and column using the single command

```
tab1.fixrowcol 1 2
```

You may clear the fixed rows and columns using

```
tab1.fixcol 0
tab1.fixrow 0
```

or

```
tab1.fixrowcol 0 0
```


## Chapter 4. Econometrics and Statistics

EViews 13 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

- Enhanced Autoregressive Distributed Lag (ARDL) support featuring estimation of Nonlinear ARDL models and new diagnostics ("ARDL Estimation" on page 29).
- Improved Pool Mean Group (PMG) estimation featuring expanded deterministic trend support, estimation with Nonlinear ARDL terms, and new diagnostics ("Pool Mean Group (PMG) Estimation" on page 32).
- Difference-in-Difference (DID) estimation and diagnostics ("Difference-in-Difference Estimation" on page 33).
- Enhanced Vector Error Correction (VEC) estimation, featuring improved support for deterministic regressors ("Enhanced VEC Estimation" on page 34).
- Bayesian Time-varying Coefficient Vector Autoregression (BTVCVAR) models ("Bayesian Time-varying Coefficient Vector Autoregression" on page 37).


## Testing and Diagnostics

- Expanded cointegration testing, featuring improved support for deterministic regressors ("Cointegration Testing" on page 40).
- New diagnostics in ARDL equations ("Diagnostics in ARDL" on page 41).
- New diagnostics in panel ARDL/PMG equations ("Diagnostics in Panel ARDL/PMG" on page 49).
- Extended VAR/VEC impulse response confidence interval calculation and display ("Enhanced Impulse Response Display" on page 53).
- Improved forecast sample flexibility ("Enhanced Impulse Response Display" on page 53).


## ARDL Estimation

EViews 13 offers improvements to existing tools for analyzing data using Autoregressive Distributed Lag Models (ARDL), featuring estimation of Nonlinear ARDL (NARDL) models which allow for more complex dynamics, with explanatory variables having differing effects for positive and negative deviations from base values.

The classical ARDL framework assumes that the long-run cointegrating relationship is a symmetric linear combination of regressors. While this is a natural starting assumption, it does not match the behavioral finance and economics literature approach to modeling nonlinearity and asymmetry (Kahneman, Tversky, and Shiller, 1979). In response, Shin (2014) proposes a nonlinear ARDL (NARDL) framework in which short-run and long-run nonlinearities are modeled as positive and negative partial sum decompositions of the explanatory variables.

From the main EViews menu, click on Quick/Estimate Equation... or type the command equation in the command line to open the equation dialog. Then select the ARDL Autoregressive Distributed Lag Models (including NARDL) from the Method dropdown to display the ARDL dialog:


In the Linear dynamic specification, you should a enter a list of variables consisting of the dependent variable followed by any symmetric ARDL distributed lag regressors. At a minimum, the edit field must contain the dependent variable.

Exogenous regressors, including deterministics, may be specified in the Fixed regressors specifications section. Trend regressors corresponding to the five deterministic cases dis-
cussed (None, Restr. constant, Constant, Restr. trend, Trend) may be specified using the Trend specification dropdown. All other exogenous regressors (those apart from the constant and the trend) should be specified in the Fixed regressors edit field.

Asymmetric distributed lag regressors may be listed under Asymmetric dynamic specifications. In particular, the Long-run and short-run edit field may be used to specify regressors which are asymmetric in both the long-run and short-run. Regressors which are asymmetric only in the long-run may be specified in the Long-run only edit field, while those which are asymmetric exclusively in the short-run are specified in the Short-run only edit field.

| Dependent Variable: DLOG(REALCONS) |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Method: ARDL |  |  |  |  |
| Date: 05/04/22 Time: 15:37 |  |  |  |  |
| Sample (adjusted): 1950Q3 2000Q4 |  |  |  |  |
| Included observations: 202 after adjustments |  |  |  |  |
| Max. dependent lags: 1 (Fixed) |  |  |  |  |
| Fixed-lag linear regressors: LOG(REALGDP) |  |  |  |  |
| Fixed-lag dual non-linear regressors: LOG(REALGOVT) |  |  |  |  |
| Deterministics: Restricted constant and no trend (Case 2) |  |  |  |  |
| Selected model: ARDL(1,1,1) |  |  |  |  |
| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| LOG(REALCONS(-1)) | -0.126939 | 0.037213 | -3.411168 | 0.0008 |
| LOG(REALGDP(-1)) | 0.133061 | 0.040618 | 3.275890 | 0.0012 |
| @CUMDP(LOG(REALGOVT(-1))) | -0.004936 | 0.008563 | -0.576444 | 0.5650 |
| @CUMDN(LOG(REALGOVT(-1))) | -0.018402 | 0.020006 | -0.919856 | 0.3588 |
| C | -0.100803 | 0.063927 | -1.576846 | 0.1165 |
| DLOG(REALGDP) | 0.643050 | 0.050018 | 12.85647 | 0.0000 |
| @DCUMDP(LOG(REALGOVT)) | -0.149865 | 0.042682 | -3.511242 | 0.0006 |
| @DCUMDN(LOG(REALGOVT)) | -0.114135 | 0.103098 | -1.107051 | 0.2696 |
| R-squared | 0.473136 | Mean depe | dent var | 0.008782 |
| Adjusted R-squared | 0.454125 | S.D. depen | ent var | 0.008864 |
| S.E. of regression | 0.006549 | Akaike info | riterion | -7.180217 |
| Sum squared resid | 0.008320 | Schwarz crit | rion | -7.049196 |
| Log likelihood | 733.2019 | Hannan-Qu | n criter. | -7.127206 |
| F-statistic | 24.88805 | Durbin-Wat | on stat | 2.584413 |
| Prob(F-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

- For a preliminary version of the new ARDL chapter in User's Guide II, see "Linear and Nonlinear ARDL" on page 315.
- See ardl (p. 112) for updated command documentation.
- See also "Diagnostics in ARDL" on page 41 for a summary of new EViews 13 views and procedures in both Linear and Nonlinear ARDL estimated equations.


## Pool Mean Group (PMG) Estimation

In large sample panel datasets, a popular approach for analyzing dynamic data is the Pooled Mean Group (PMG) estimator of Pesaran, Shin and Smith (1999) (PSS). This model takes the cointegration form of the simple ARDL model and adapts it for a panel setting by allowing the intercepts, short-run coefficients, and cointegrating terms, to differ across cross-sections.

EViews 13 extends the estimation of PMG models to support:

- a greater range of deterministic trend specifications (including those with fully restricted constant and trend terms)
- specifications with asymmetric regressors.

To estimate a Panel ARDL/PMG model in EViews, open the equation dialog by selecting Quick/Estimate Equation..., or by selecting Object/New Object.../Equation and selecting PMG/ARDL from the Method dropdown menu. EViews 13 will then display the new ARDL estimation dialog:


While much of the dialog is familiar from earlier versions of EViews, notably different in the EViews 13 PMG dialog are additional entries in the Trend specification corresponding to restricted intercept and trend specifications, and new edit fields for specifying Asymmetric dynamic specifications. The Trend specification upgrades allow for a wider range of long
and short-term dynamics, while the latter fields support asymmetric variables in the PMG estimator.

- See the discussion of non-panel ARDL estimation in "Background" on page 315 and "Estimating ARDL and NARDL in EViews" on page 320 for additional detail.
- See ardl (p. 112) for updated command documentation.
- See also "Diagnostics in Panel ARDL/PMG" on page 49 for discussion of new EViews 13 views and procedures in panel PMG estimated equations.


## Difference-in-Difference Estimation

Difference-in-difference (DiD) estimation is a popular method of causal inference that allows estimation of the average impact of a treatment on individuals.

EViews 13 offers tools for estimation of the DiD model using the common two-way fixedeffects (TWFE) method, as well as post-estimation diagnostics of the TWFE model, such as those by Goodman-Bacon (2021), Callaway and Sant'Anna (2021), and Borusyak, Jaravel, and Spiess (2021).

To estimate a DiD model in EViews, bring up the equation dialog by clicking on Object/New Object.../Equation or Quick/Estimate Equation... from the main menu bar in your panel workfile. EViews will detect the presence of your panel structure and in place of the standard equation dialog will open the panel equation estimation dialog. Select DiD - Differ-ence-in-Difference in the Method dropdown display the DiD dialog:


In the Equation specification edit field you should enter the dependent variable followed by any exogenous regressors apart from the treatment variable.

The treatment variable should be entered in the Treatment Variable edit field. The treatment series should be a binary variable indicating whether the individual has been treated (i.e., is 1 if the observation in a treatment group which is post-treatment date for that group, and 0 otherwise).

The Options tab contains a single Coefficient name edit field that allows you to change the default coefficient vector.

Click on $\mathbf{O K}$ to perform the difference-in-difference estimation and display the output:

Dependent Variable: L_HOMICIDE
Method: Difference-in-Difference
Date: 05/13/22 Time: 10:56
Periods included: 11
Cross-sections included: 50
Total panel (balanced) observations: 550

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | ---: |
| POST | 0.081812 | 0.064117 | 1.275980 | 0.2026 |
| R-squared | 0.910576 | Mean dependent var | 1.405760 |  |
| Adjusted R-squared | 0.899604 | S.D. dependent var | 0.590154 |  |
| S.E. of regression | 0.186992 | Akaike info criterion | -0.411237 |  |
| Sum squared resid | 17.09842 | Schwarz criterion | 0.066772 |  |
| Log likelihood | 174.0902 | Hannan-Quinn criter. | -0.224439 |  |
| F-statistic | 82.98904 | Durbin-Watson stat | 1.469473 |  |
| Prob(F-statistic) | 0.000000 | Parallel trend stat | 0.820393 |  |
| Prob(P. trend) | 0.411992 |  |  |  |

- See "Difference-in-Difference Estimation" on page 399 for additional discussion.
- See did (p. 143), didcs (p. 144), didgbdecomp (p. 144), didmakeeq (p. 145), didtrends (p. 146) for command documentation.


## Enhanced VEC Estimation

Estimation of Vector Error Correction (VEC) models using reduced rank regression (RRR) has been a staple of EViews for several versions. Previous versions of EViews supported a variety of built-in specifications for deterministic trends, and allowed users to add unrestricted exogenous regressors to account for short-term dynamics.

One important limitation of the existing tools for VEC estimation was the inability of users to add arbitrary exogenous variables to the cointegrating relation since the set of admissible restricted regressors was limited to the endogenous variables and potentially an intercept, and possibly a linear trend term.

EViews 13 removes this limitation on restricted regressors and improves on the prior treatment of exogenous variables in two distinct ways:

- by providing new deterministic trend assumption presets which allow for more flexible specification of restricted and unrestricted deterministic coefficients, and
- by allowing users to add exogenous variables that are restricted to the cointegrating relation, variables that are unrestricted, and variables that in both the short and long-run equations, with an orthogonality assumption used to obtain the different coefficients.

To estimate a VEC using the new features, from the main application menu of an existing var object, click on the Estimate button to open the VAR Specification estimation dialog. Alternately, you may create a new VAR object by selecting Object/New Object... group, then selecting VAR. Once the dialog appears, select Vector Error Correction in the Method dropdown menu to display the VEC estimation dialog:


The Exogenous variables section of the main estimation dialog has fields for variables that are Short-run (outside cointegrating equation) only, variables that are Long-run (inside the cointegrating equation) only, and variables that are Both long-run and short-run.

Further, the Deterministic trend specifications dropdown on the Cointegration tab offers additional predefined deterministic trend specifications.


Selecting a specification in which both intercept and trend are short-run only and clicking on OK produces estimates using the new deterministics:


- See "Vector Error Correction Models (VECMs)" on page 361 for additional discussion.
- See also "Cointegration Testing" on page 40 for related improvements in cointegration testing.
- See ec (p. 151) for updated command documentation.


## Bayesian Time-varying Coefficient Vector Autoregression

The time-varying coefficients vector autoregression, or TVCVAR, is a nonlinear VAR model with coefficients that evolve smoothly over time. EViews 13 supports Bayesian TVCVAR, or BTVCVAR. The BTVCVAR is popular even among those who do not identify as Bayesian because the prior provides a convenient way to induce shrinkage in a model that needs it.

To estimate BTVCVAR in EViews, click on Quick/Estimate VAR... or run var in the command window. This will open the VAR Specification dialog. Select Bayesian TVCVAR from the VAR type dropdown menu. The dialog should now have the Basics, Prior, and Options tabs.


Endogenous variables, lags, exogenous variables, and the estimation sample are set in the Basics tab. Lags are required to be contiguous.

EViews gives users the option of setting aside a subset of the estimation sample for the purpose of specifying a prior distribution. The observations that go towards specifying the prior comprise the prior sample. The remaining observations make up the data sample, which is used to update the prior.

| VAR Specification |  |  |  | $\times$ |
| :---: | :---: | :---: | :---: | :---: |
| Basics Prior | Options |  |  |  |
| Hyperpar <br> T0: <br> tau0: <br> tau1: <br> tau2: <br> nu1: <br> nu2: | ters <br> 0 <br> 5.0 <br> 1.0 <br> 0.01 <br> 5.0 <br> 5.0 | Prior sample size <br> Prior scaling factor for bo <br> Prior scaling factor for $S$ <br> Prior scaling factor for Q <br> Prior dof for S <br> Prior dof for Q |  |  |

Prior hyper-parameters are set inside the Prior tab. To help with setting the prior, EViews maps the BTVCVAR prior hyper-parameters to a set of six scalar quantities. Users set these scalars to specify a prior sample, control the variability of the time-varying coefficients, etc.


There are four categories in the Options tab: Sampler, Display, Smoother, and Stability.
The Sampler options determine how the posterior sample is generated.

- The burn-in size is the number of initial draws to discard. It is specified as a count. The burn-in process gives the underlying Markov chain time to converge to the posterior distribution.
- The posterior sample size is used to determine how many posterior draws are used to carry out post-updating procedures (estimation, forecasting, impulses responses analysis, etc.).
- The thinning size is used to thin the Markov chain. A thinning size of $r$ indicates that every $r$-th draw is stored. For example, no thinning occurs when $r$ is set to 1 , and every other draw is stored when $r$ is set to 2 . By definition, thinning does not apply to burn-in draws.
- The seed field is used to set the random seed for the posterior simulator. EViews will generate a seed automatically if the user does not specify one. Click Clear to clear the seed field.
- The number of subchains field determines how many subchains are used when the posterior sample gets regenerated. Regeneration is typically much faster than initial generation since subchains can be run in parallel.

Display options determine what to report as estimation results. Users can pick either posterior median or posterior mean as their point estimate. The point estimate type selected here will be applied to the coefficients, the observation covariance matrix, and the errors. Users can also display equal-tailed credibility intervals (bands) at one or more credibility levels for the coefficients. To do so, check the box next to Show credibility intervals. Bands use shading by default. To use lines instead, check the box next to Use lines.

A simulation smoother can be selected from the dropdown menu under Smoother. EViews currently supports three simulation smoothers: the Cholesky factor algorithm (CFA), the Kalman filter and smoother (KFS), and the method of McCausland, Miller, \& Pelletier (MMP). For more information, see McCausland, Miller, \& Pelletier (2011) and the references therein.

To enforce stable VAR coefficients at each date within the data sample, select Cogley \& Sargent from the dropdown menu under Stability. The maximum number of attempts threshold ensures that the sampler does not get stuck in an infinite loop in an attempt to obtain stable draws.

Once the BTVCVAR model has been specified, click on $\mathbf{O K}$ to run the posterior simulator. Progress is displayed in the bottom left corner of the EViews window. Once posterior simulation is complete, estimates and other statistics based on the posterior sample are computed.

Estimation results are presented in a spool-like object. The nodes under Output Sections in the left pane are used for navigation. For example, clicking on the Summary node will bring
the summary table into focus. The checkboxes that appear below Display Coefficients are used to show/hide coefficient series that are associated to specific endogenous variables, lags, and exogenous variables. For example, unchecking the box next to $\mathbf{C}$ hides the coefficient series associated to the constant term in all graphs.

- See "Bayesian Time-varying Coefficients VAR Models" on page 423 for a preliminary version of the full documentation.
- See btvcvar (p. 119) for new command documentation.


## Cointegration Testing

The introduction of new deterministic trend settings and exogenous variable support in EViews 13 VEC estimation ("Enhanced VEC Estimation" on page 34) offers corresponding improvements in cointegration testing in both group and var object settings.

To perform a Johansen cointegration test to determine the rank that should be used in estimation of the VEC select View/Cointegration Test/Johansen System Cointegration Test..., from a group window, or Views/Cointegration Test... from an estimated VAR object window using. In the latter case, the test dialog will be pre-filled with the cointegration specification, if applicable:


All new settings allow you to specify new deterministic trend specifications will restricted constants and trends, and now allow you to specify exogenous variables that appear in the long-run cointegrating relation.


- See "Vector Error Correction Models (VECMs)" on page 361 for additional discussion.
- See also "Enhanced VEC Estimation" on page 34 for related improvements in VEC estimation.
- See coint (p. 124) for updated group object and coint (p. 133) for updated var object command documentation.


## Diagnostics in ARDL

In addition to the support for nonlinear ARDL specifications, EViews 13 offers a number of new ARDL diagnostics. These diagnostics are designed to help you examine the dynamic behavior of your variables, the long-run properties of your specification, and the appropriateness of your model:

- Improved display tools make it easier to use the ARDL estimates to examine the short and long-run representations of the distributed lag regressors and cointegrating relationships. ("Enhanced Presentation of ARDL Results" on page 42.)
- EViews now produces cumulative dynamic multipliers (CDM) graphs showing cumulative marginal contribution of an explanatory variable to the dependent variable ("Cumulative Dynamic Multiplier Graphs" on page 45).
- Bounds tests are for cointegration are now available as a view of your ARDL equation ("Bounds Tests" on page 47).
- You may evaluate symmetry restrictions on the coefficients of the asymmetric long and short-run regressors ("Testing for Asymmetry" on page 48).

For a preliminary version of the new ARDL chapter in User's Guide II, see "Linear and Nonlinear ARDL" on page 315.

## Enhanced Presentation of ARDL Results

EViews 13 offers enhanced presentation of ARDL results, with an clearer emphasis on highlighting the error correction results and the cointegrating relationship.

The new error correction spool output (View/ARDL Diagnostics/Error-Correction Results) shows two representations of the coefficients in the cointegrating relationship, allowing you to easily move between viewing the long-run relationship results in the Conditional Error Correction (CEC) and the Error Correction (EC) forms.

The CEC focuses on the natural division between long-run and short-run dynamics, with the former generated by regressors entering in levels/lags, and the latter by regressors in differences

while the EC representation uses the cointegrating relation to highlight the speed of adjustment to long-run equilibrium,


The new cointegrating relation spool view (View/ARDL Diagnostics/Cointegrating Relation) lets you switch between viewing the specification, coefficient results, and a graphical display of the cointegrating relation.


- See cointrel (p.137) and ecresults (p. 154) for command documentation.


## Cumulative Dynamic Multiplier Graphs

To display cumulative dynamic multiplier graphs for each of the explanatory variables in an EViews equation estimated using ARDL, click on View/ARDL Diagnostics/Dynamic Multiplier Graph...

EViews will open a dialog containing display and computation settings:

| $\square$ Dynamic Multipliers |  |
| :--- | :--- |
| Horizon: | 30I |
| Confidence intervals |  |
| $\square$ Show CI |  |
| $\square$ Shade CI band |  |
| Level: | 0.95 |
| Replications: | 999 |
| OK |  |

- You may enter the horizon length $h$ (number of periods to compute the multipliers) in the Horizon edit field.
- For NARDL models, you will be offered the opportunity to display confidence intervals for the computed absolute difference between the positive and negative components for a given regressor. CIs are not available for linear ARDL specifications.

You may check the Show CI to display the CIs, and Shade CI band to display the CIs as bands instead of lines. The Level edit field controls the size of the CI, and the Replications governs how many replications to use in resampling for computing the CI.

Click on OK to continue. EViews will open a spool view, with each node in the spool containing the CDM graph corresponding to one of the explanatory variables.

For symmetric linear models, each graph contains a CDM along with a dashed horizontal line denoting the long-run value.


For asymmetric nonlinear models, each graph will show the positive and negative responses and limit values, along with a line showing the absolute value of the difference between the two, and if requested, a CI for the absolute difference:


- See dynmult (p. 150) for command documentation.


## Bounds Tests

EViews 13 now performs Pesaran's (2001) bounds tests for cointegration that are robust to whether variables of interest are $I(0), I(1)$, or mutually cointegrated. These tests are formulated as standard $F$-test or Wald tests of parameter significance in the cointegrating relationship of the Conditional Error Correction model for each cross-section.

To perform the bounds tests, click on View/ARDL Diagnostics/Bounds Tests. The results of the test performed are presented in the first table of a spool. Below the table of long run coefficient estimates are two additional tables, respectively titled as the $F$-Bounds Test and the $t$-Bounds Test.


## Testing for Asymmetry

The NARDL specification is quite general and can accommodate asymmetries in different combinations of short and long-run dynamics. From an estimated NARDL, you may test long-run symmetry restrictions and/or short-run symmetry restrictions.

To perform the symmetry tests on the estimated NARDL, click on View/ARDL Diagnostics/ Symmetry Test from an estimated equation:

| Coefficient symmetry tests |  |  |  |
| :---: | :---: | :---: | :---: |
| Null hypothesis: Coefficient is symmetric |  |  |  |
| Degrees of freedom (simple tests): $\mathrm{F}(1,194)$, Chi-square(1) |  |  |  |
| Degrees of freedom (joint tests): $\mathrm{F}(2,194)$, Chi-square(2) |  |  |  |
| Equation: EX4 |  |  |  |
| Variable | Statistic | Value | Probability |
| Long-run |  |  |  |
| LOG(REALGOVT) | F-statistic | 0.525997 | 0.4692 |
|  | Chi-square | 0.525997 | 0.4683 |
| Short-run |  |  |  |
| LOG(REALGOVT) | F-statistic | 0.077094 | 0.7816 |
|  | Chi-square | 0.077094 | 0.7813 |
| Joint (Long-Run and Short-Run) |  |  |  |
| LOG(REALGOVT) | F-statistic | 0.493551 | 0.6112 |
|  | Chi-square | 0.987102 | 0.6105 |

- See symmtest (p. 276) for command documentation.


## Diagnostics in Panel ARDL/PMG

In addition to the support for asymmetric ARDL/PMG specifications, EViews 13 offers a number of new PMG diagnostics to help you examine long-run relationships and evaluate the appropriateness of your specification.

- New views make it easier to use the PMG estimates to examine the short and longrun representations of the distributed lag regressors and cointegrating relationships. ("Cross-sectional Cointegration Views" on page 50.)
- Cross-section bounds tests are for cointegration are now available as a view of your ARDL equation ("Cross-sectional Bounds Tests" on page 51).
- You may evaluate symmetry restrictions on the coefficients of asymmetric long and short-run regressors ("Testing for Asymmetry" on page 52).
- You may perform a Hausman-test of the similarity of the PMG estimator to the meangroup (MG) and dynamic fixed effects (DFE) estimators ("Similarity Tests" on page 53).

For a preliminary version of the new ARDL chapter in User's Guide II which discusses a variety of these features in non-panel settings, see "Linear and Nonlinear ARDL" on page 315.

## Cross-sectional Cointegration Views

The Error Correction Results view displays coefficient results for the error correction regressions for each cross-section. Select View/ARDL Diagnostics/Cross-sectional Error-correction Results to display a spool of tables with results for each cross-section:


- See ecresults (p. 154) for command documentation

The Cointegrating Relations Plots view displays information about the error correction term $E C_{t}$ representing the cointegrating relation. Select View/ARDL Diagnostics/Cross-sectional Cointegrating Relations Plots to display a spool of plots for each cross-section:


- See cointrel (p. 137) for command documentation.


## Cross-sectional Bounds Tests

EViews 13 now performs Pesaran's (2001) bounds tests for cointegration that are robust to whether variables of interest are $I(0), I(1)$, or mutually cointegrated. These tests are formulated as standard $F$-test or Wald tests of parameter significance in the cointegrating relationship of the Conditional Error Correction model for each cross-section.

To perform the bounds tests, click on View/ARDL Diagnostics/Cross-sectional Bounds Tests. The results of the test performed on each cross-section are presented in the first table of a spool. Below the table of long run coefficient estimates are two additional tables, respectively titled as the $F$-Bounds Test and the $t$-Bounds Test.


- See boundstest (p. 118) for command documentation


## Testing for Asymmetry

One can test for symmetry formally by performing the usual $t$-test or $F$-test of parameter equality. For example, testing for symmetry for a specific long-run (LR) variable, say $x_{j}$, is equivalent to the following hypothesis:

$$
\begin{array}{ll}
H_{0}: & \lambda_{j}^{+}=\lambda_{j}^{-}  \tag{4.1}\\
H_{1}: & \lambda_{j}^{+} \neq \lambda_{j}^{-}
\end{array}
$$

To perform the symmetry test，select View／ARDL Diagnostics／Symmetry Test from the menu of a nonlinear asymmetric NARDL／PMG equation：
－See symmtest（p．276）for command documentation

## Similarity Tests

We may easily perform a Hausman test on the similarity of the PMG estimator to the mean－ group（MG）and dynamic fixed effects（DFE）estimators．To conduct the test，click on ARDL Diagnostics／PMG Similarity Test：

| Equation：EQ1 Workfile：OECD＿TEST：－panel\ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |
| 四 Hausman Test四 Differences：Mean Group四 Differences：Dynamic Fixed Effects国 Estimation：Mean Group国 Estimation：Dynamic Fixed Effects |  |  |  |  |  | $\wedge$ |
|  | －Hausman Test |  |  |  |  |  |
|  | PMG Hausman Specification Test <br> Null hypothesis：Estimator is statistically similar to the PMG estimator |  |  |  |  |  |
|  | Estimator |  | Stat． | DOF | p －Value |  |
|  | Mean Group |  | 1.575591 | 2 | 0.4548 |  |
|  | Dynamic Fixed Effects |  | 29.823557 | 2 | 0.0000 |  |
|  | －Differences：Mean Group |  |  |  |  |  |
|  | Coefficient Difference Ov | iew：Mean Grown |  |  |  |  |
|  | Variable | MG | PMG | Var（Diff．） | p －Value |  |
|  | LOG（INF） | －0．352900 | －0．464856 | 0.010428 | 0.2729 |  |
|  | LOG（INC） | 0.918134 | 0.904398 | 0.000668 | 0.5951 |  |
|  | －Differences：Dynamic Fixed Effects |  |  |  |  |  |
|  | Coefficient Difference Overview：Dynamic Fixed Effects |  |  |  |  |  |
|  | Variable | DFE | PMG | Var（Diff．） | p －Value |  |
|  | LOG（INF） | －0．266343 | －0．464856 | 0.001440 | 0.0000 |  |
|  | LOG（INC） | 0.912057 | 0.904398 | 0.000174 | 0.5611 |  |

－See similarity（p．273）command documentation

## Enhanced Impulse Response Display

EViews 13 offers considerably more control over the computation and display of impulse response（and variance decomposition）confidence intervals．You may now display multiple confidence intervals with user－specified sizes in a single graph，and employ shading to improve the visibility of these intervals．

Recall that previous versions of EViews imposed important restrictions on the display of impulse response confidence intervals．
－First，for impulse－response confidence intervals computed using analytic or Monte Carlo standard errors，EViews only allowed for the computation of $+/-2$ standard
error bands. There were no options for displaying different sized intervals, let alone for displaying different intervals in the same graph:

| Impulse Responses |  |  |
| :---: | :---: | :---: |
| Display Impulse Definition |  |  |
| Display information <br> Impulses: <br> ip m1 tb3 <br> Responses: <br> ip m1 tb3 <br> Horizon length: $\square$ <br> Accumulate responses: | Response standard errors <br> Method: Analytic (asymptotic) <br> Confidence interval levels: |  |
|  | OK | Cancel |

- Second EViews only allowed you to use dotted lines to display the confidence bands around the impulse-responses:

- Lastly, if you chose to display multiple impulses or responses in a combined graphs, confidence intervals were not available:


Fortunately, EViews 13 removes all three of these limitations.

- First, by default, EViews 13 now uses shaded areas to depict confidence bands:


To display your graphs using the traditional lines, check the box next to Display intervals using lines in the impulse response dialog. If coming from the command line, you may use the uselines keyword in the impulse command options to display in the previous format.

- Second, EViews 13 now lets users display analytic and monte carlo confidence intervals with one or more user-specified sizes:

| Impulse Responses | $\times$ |
| :---: | :---: |
| Display Impulse Definition |  |
| Display information <br> Impulses: $\log (\mathrm{cs}) \log (\mathrm{inc})$ <br> Responses: $\log (\mathrm{cs}) \log (\mathrm{inc}) \mid$ <br> Horizon length: $\square$ 10 $\square$ Accumulate responses | Standard errors and confidence intervals <br> Method for SEs and CIs: <br> Analytic (asymptotic) <br> Confidence interval levels: $0.95$ $\square$ Display intervals using lines |
|  | OK Cancel |

When multiple shaded bands are displayed, EViews automatically uses the new EViews 13 line and shade transparency engine ("Graph Line and Shade Transparency" on page 13) to display the bands using gradients:


Using the selection interface to focus on the topmost left graph, better shows the shading:


- Further, EViews 13 now allows you to display confidence intervals in the impulseresponse combined graphs view:


All of the new shading and multiple band support is also provided for variance decomposition in multiple graph

and combined graph settings,


- See impulse (p. 216) and vdecomp (p. 276), for updated command documentation.


## Forecast Sample Setting Commands

EViews 13 offers a small but surprisingly useful improvement in forecast sample setting.
Previously, forecasting via command generally required the user to first set the global smpl using a smpl statement prior to issuing the forecasting command. For example, the following lines estimated the equation using one sample, and then forecast over two different samples,

```
smpl 1990q1 2010q4
equation eq1.ls y c x1 x2 x3
smpl 2011q1 2020q4
eq1.forecast fcst1
smpl 2008q3 2020q4
eq2.forecast fcst2
```

In this case, setting the global sample prior to equation forecasting was only mildly inconvenient. However, in some settings, particularly those involving non-equation forecasting where you are generating multiple forecasts and data, setting and resetting the global sample was inconvenient at best, and prone to error.

EViews 13 solves this problem by updating most forecast-generating commands to take a forecast sample information.

In this context, there are two types of forecasting procedures.
In the first set of procedures, the forecast sample is set independently of the remainder of the procedure. For these cases, EViews 13 offers an option to allow you to set the sample range pair directly using the option:
forcsmpl $=\quad$ Fit sample (optional). If forecast sample is not provided, smpl the workfile sample will be employed.

This option is available interactively and in commands for:

## Equation Procs

fit static forecast (p. 174). (updated)
forcavg..................average forecasts of a series (p. 182). (updated)
forecast .................dynamic forecast (p. 180). (updated)
The second type of forecasting procedure sets the forecast sample one-period beyond a previously specified estimation sample, and then forecasting continues from that period onward until a forecast endpoint. In this case, EViews 13 allows you to set the forecast sample in two distinct ways: by specifying the number of periods to forecast, or by specifying the forecast end date:

| forclen $=$ int | Number of periods to forecast. |
| :--- | :--- |
| forc $=$ "date" | Specify the date of the forecast end point. |

These options are available interactively and in commands for:

## Series Procs

autoarma $\qquad$ forecast from a series using an ARIMA model with automatic determination of the specification (p. 115). (updated)
ets $\qquad$ perform Error-Trend-Season (ETS) estimation and exponential smoothing (p. 154). (updated)

## Example

Our earlier example is now:

```
smpl 1990q1 2010q4
equation eq1.ls y c x1 x2 x3
```

```
eq1.forecast(forcsmpl="2011q1 2020q4") fcst1
eq1.forecast(forcsmpl="2008q3 2020q4") fcst2
```


## Chapter 5. Data Handling

EViews 13 offers powerful new features for working with data featuring support for the following popular data sources and file formats:

## Data Handling

- Daily data seasonal adjustment ("Daily Data Seasonal Adjustment" on page 63).
- New and improved Excel file writing engine ("New Excel File Writing Engine" on page 65).
- Enhanced holiday family of functions to return the proportion of an annual event associated with each observation ("Holiday Functions" on page 65).


## Data Sources and File Formats

- Australian Bureau of Statistics SDMX ("SDMX Databases," on page 69).
- Deutsche Bundesbank SDMX ("SDMX Databases," on page 69).
- Insee SDMX ("SDMX Databases," on page 69).
- Trading Economics ("Trading Economics," on page 72).
- World Health Organization ("World Health Organization," on page 77).


## Matrix Language Support

- Improved data import and export from matrix objects ("New Data import/export engine," on page 81).
- Expanded support for row and column labeling ("Row and Column Label Support," on page 83).
- Improved data access to matrix object data ("Matrix Data Access," on page 86)


## Data Handling

## Daily Data Seasonal Adjustment

EViews 13 adds the ability to perform seasonal adjustment on daily data, by featuring an extended implementation of the seasonal adjustment of daily time series algorithm of Ollech (2021). Although the original Ollech (2021) algorithm is designed for 7-day week daily data, EViews' implementation handles both 7 and 5-day week daily data.


To perform DSA seasonal adjustment in EViews, open the series and select Proc/Seasonal Adjustment/DSA Daily Seasonal Adjustment... EViews will then open a tree-structured DSA dialog to allow you to set the options for the DSA procedure:


The branches of the tree, on the left, allow you to specify the Basic Options, the ARIMA model, and the three STL Seasonal Adjustment components. Click on the node name in the left to select the node.

- For extensive description of the new DSA procedure, see "Daily Seasonal Adjustment," on page 301.
- See dsa (p. 146) for command documentation


## New Excel File Writing Engine

Previous versions of EViews used the Excel application itself to write Excel 2007 files (.XLSX) so that exporting data from an EViews workfile to an Excel 2007 file required that Excel be installed on your machine. Using Excel to write these files was convenient, but importantly came with performance and capability limitations.

EViews 13 introduces a new Excel 2007 writing engine that no longer requires installation of the Excel application. As a result, exporting EViews workfile data to Excel is now much more efficient, especially when performing a large number of writes. Furthermore, the new engine allows you to write data into an existing Excel file (via command), an operation that was not permitted in earlier versions.

For the most part, using the new engine will be transparent to users for previous versions (apart from the performance improvements). Simply perform your Excel 2007 writes in the usual fashion. There is, however, a new option "mode = update" which is required for writing data into an existing Excel file:

```
tab1.save(t=excelxml, cellfmt=EViews, mode=update) mytable
    range=Country!b5
```

will add the contents of table TAB1 to the preexisting "MyTable.XLSX" Excel file. The data will be written to the 'Country' sheet at cell B5. All preexisting cell formatting in the Excel file will be overwritten using the cell colors and fonts in TAB1.

## Holiday Functions

The @holiday and @holidayset functions return the proportion of an annual event covered by each observation.

Syntax:
@holiday(event[, basis][, flag...])
@holidayset(event[, basis][, flag...])
For the @holiday function, the event argument is a string specifying either a single date, a pair of dates (forming a range), or a single named group of holidays.

For the @holidayset function, the event argument is list of dates and/or named groups of holidays. In either case, each item in the argument has the format

```
"base[~|!][(offset)][[weights]]"
```

For example,

```
@holiday("Jan1")
@holiday("Veterans.us(7)Thanksgiving.us")
@holiday("NewYears[1,2,0]")
@holidayset("Easter! AllSaints! Christmas!")
```

The basecomponent is either a day-of-the-month specification, e.g. "Dec25", a n-th-weekday-of-the-month specification, e.g. "Nov4Thu" (fourth Thursday in November) or "May-1Mon" (last Monday in May), or a named holiday.

## Named Holidays

EViews supports named holidays for common holidays in the G8 countries, including the following ecclesiastical holidays: "epiphany", "easterfriday", "goodfriday", "easter", "eastermonday", "ascension", "pentecost", "whitmonday", "assumption", "allsaints", "immaculate", "christmas", "saintstephen.

Also available are New Years Day ("nyd", "newyear", and "newyears"), Lunar New Year ("cny", "lny", and "lunarnewyear"), International Women’s Day ("women" and "womens"), and International Men’s Day ("men" and "mens").

Named holidays primarily associated with a specific country are suffixed with a locale code following ISO 3166-1 alpha-2, i.e., the Internet country codes for top-level domains such as ".ca" and ".de". These named holidays are:

- Canada -"victoria.ca", "discovery.ca", "canada", "civic.ca", "labour.ca", "thanksgiving.ca" (also"thanks.ca"), "remembrance.ca", "boxing.ca"
- France -"labour.fr", "victory.fr", "bastille.fr", "armistice.fr"
- Germany -"labour.de", "unity.de"
- Italy -"liberation.it", "labour.it", "republic.it"
- Russia -"christmas.ru", "defender.ru", "springlabour.ru", "victory.ru", "russia", "unity.ru"
- United Kingdom -"mayday.uk", "springbank.uk", "summerbank.uk", "boxing.uk"
- United States of America -"mlk.us", "presidents.us", "memorial.us", "independence.us", "labor.us", "columbus.us", "veterans.us", "thanksgiving" (also "thanks")

Note that the named holidays "canada", "russia", and "thanksgiving" do not include a suffix, either to avoid redundancy or maintain compatibility with earlier versions of EViews. Also note that "christmas.ru" is included to reflect the Russian Orthodox Church's use of the Julian calendar.

In general, a suffix may be safely added to any named holiday that does not already include one, e.g."christmas.us" or "nyd.it". Unless noted above, such combinations do not alter the
nominal date of the holiday but may produce different results when combined with a weekend modifier (discussed later).

## Named Groups

Named groups are preset collections of holidays. EViews currently supports a single named group, "bank", for each of the supported locales, i.e., "bank.ca", "bank.fr", "bank.de", etc., allowing easy access to the common bank holidays.

The membership of these groups are:

- Canada, "bank.ca" -"nyd goodfriday victoria discovery canada civic labour thanksgiving remembrance christmas boxing"
- France, "bank.fr" -"nyd eastermonday labour victory ascension whitmonday bastille assumption allsaints armistice christmas"
- Germany, "bank.de" -"nyd goodfriday eastermonday labour ascension whitmonday unity christmas saintstephen"
- Italy, "bank.it" -"nyd epiphany easter eastermonday liberation labour republic assumption allsaints immaculate christmas saintstephen"
- Russia, "bank.ru" -"nyd christmas defender womens springlabour victory russia unity"
- United Kingdom, "bank.uk" -"nyd goodfriday eastermonday mayday springbank summerbank christmas boxing"
- United States of America, "bank.us" -"nyd mlk presidents memorial independence labor columbus veterans thanksgiving christmas"

All named group members have an implied suffix matching the group's suffix. When using a named group, the sum of proportion values within a year will equal the number of group members (ignoring sample effects).

A date or named group may optionally be followed any or all of the following: (1) a weekend modifier, (2) a parenthetical offset, (3) a bracketed list of weights. For a named group, these elements will be applied to all of the individual members of the group.

A weekend modifier character " $\sim$ " or "!" indicate special handling of dates that fall on weekends. If " $\sim$ " is used, then the date will be adjusted to the nearest weekday. A date landing on a Saturday is adjusted to the preceding Friday, and a date landing on a Sunday is adjusted to the following Monday. If "!" is used with a named holiday, then a more sophisticated set of rules is used to determine when the holiday will be observed, reflecting public holiday and bank holiday conventions.

In some locales, holidays are observed according to the simple rule encapsulated by the " $\sim$ " modifier and thus the two modifiers will behave identically. For example, suppose we are evaluating "christmas.us!" for the year 2021. That date lands on a Saturday and the holiday will be observed on the preceding Friday, Dec. 24. However, if evaluating "christmas.uk!" for the same year, weekend holidays are observed on the nearest following weekday in this locale, thus the holiday will be observed on Monday, Dec. 27. Similarly, "boxing.uk!" will be observed on Tuesday, Dec. 28.

A date followed by a parenthetical offset will be adjusted by the given number of days. For example, "christmas (-1)" could be used to specify Christmas Eve. If a weekend modifier is also present, e.g. "christmas $\sim(-1)$ ", the offset is applied after any adjustment made by the modifier.

A date followed by a bracketed list of weights is considered to occur over multiple days. The specified weights determine the relative proportion of the holiday occurring on each day, with the sum of proportions across all days within a year equaling one. The list must contain an odd number of terms, with the middle term corresponding to the nominal date of the holiday (after adjustment from any weekend modifier or offset). For example, evaluating "christmas" for a daily workfile would return the value 1 for the observation on Dec. 25 and the value 0 for all other observations in that year. Evaluating "christmas[1,2,1]" would return the value 0.25 for the observation on Dec. 24 , the value 0.5 for Dec. 25 , and the value 0.25 for Dec. 26, returning the value 0 for all other observations.

Several named weight patterns are available as alternatives to explicit weight lists:

- "rampup(n)" - An increasing integer sequence of length $n$ ending on the date. For example, "[rampup(3)]" is equivalent to " $1,2,3,0,0$ ]".
- "rampdown(n)" - A decreasing integer sequence of length $n$ beginning on the date. For example, "[rampdown(3)]" is equivalent to " $[0,0,3,2,1]$ ".
- "ramp(n)" -An increasing and then decreasing integer sequence centered on the date. For example, " $[\mathrm{ramp}(3)]$ " is equivalent to " $[1,2,3,2,1]$ ".

Note that weights may not be included when a pair of dates is used to specify a range in @holiday. The optional basis parameter may be to specify that only certain day of the week or times of day should be included as part of the holiday. This parameter has the format

```
"start_weekday-end_weekday[, start_time-end_time]"
```

e.g. "mon-thu" or "mon-sun,10am-4pm".

The optional flag parameter supports two options, "binary" and "denorm". If "binary" is specified, any non-zero value that would be returned by @holiday is replaced by one, thus forcing the function to return only zeros and ones. This flag is equivalent to the expression "@holiday (...) > 0". If "denorm" is specified, then the default normalization steps for weighted dates (dividing by the sum of weights) and sets or groups of holidays (dividing by
the number of distinct holidays) are not performed. For example, the previous @holiday("christmas $[1,2,1]$ ") returned values $0.25,0.5$, and 0.25 on sequential observations, whereas @holiday("christmas[1,2,1]", "denorm") would return values 1,2, and 1.

## Data Sources and File Formats

EViews 13 offers direct access to new data sources that offer a large range of publicly available data.

The remainder of this discussion describes interactive access to each of the data sources.

## SDMX Databases

As part of EViews support for SDMX Databases, EViews 13 now offers access to Australian Bureau of Statistics (ABS) SDMX Web Services, Deutsche Bundesbank SDMX, and Insee (L'Institut national de la statistique et des études économiques) SDMX data.

Please note that an internet connection is required to obtain SDMX online data. For more information on the ABS data, see
http://api.data.abs.gov.au
EViews offers a custom interface to SDMX databases. The interface includes a custom browser for navigating and retrieval of available data.

To start, you must open a database window by selecting File/Open Database... from the main EViews menus, then selecting a database from the Database/File type dropdown menu. In addition to the previously supported Eurostat SDMX Database, ECB SDMX Database, UN SDMX Database, IMF SDMX Database, and OECD SDMX Database, you may now select either ABS SDMX Database, Deutsche Bundesbank SDMX Database, or Insee SDMX Database to access the ABS data.

EViews will display a dialog similar to the following:

| Database Specification |  |  | $\times$ |
| :---: | :---: | :---: | :---: |
| Database spedification |  |  |  |
| Database/File type: ABS SOMX Database |  |  |  |
| Server speeification: hhtps://api.data.abs.gov.au |  |  |  |
| User name: |  |  |  |
| Password: |  |  |  |
| Database name: ABS |  |  |  |
| Browse | Browse Registry | Add to Registry |  |
| Open as |  |  |  |
| Database alias (optional short name): |  |  |  |
| OK |  | Cancel |  |

Click OK to open the online database. You will be presented with a standard EViews database representing a connection to data. Click on Browse or Browse-Append to open a custom database.

Here, we see the interface to the ABS data:


The dialog allows you to select data available within datasets. The browser interface to the data provides a way to search through the datasets by typing a keyword in the Filter textbox:


Select a dataset and click the Next button to display a dialog where you can view and select series:


The dialog contains a table with all series matching the search. The interface provides a list of dropdown boxes that contain additional search filter criteria. Clicking a filter drop box, a user can select one or more options. When selection is finished the results of series will be updated to match the filter options.

The interface also provides a graph preview of the series data selected.
Once you have found and selected the series of interest, you may drag-and-drop or click the Export to workfile button to export the series directly into an EViews existing or new workfile in the usual fashion.

- See dbopen (p. 139) for updated command syntax.


## Trading Economics

Trading Economics provides access to a large range of historical and forecast data for economic indicators, stock market, government bonds, exchange rates, and commodity prices.

Please note that an internet connection will be required to obtain Trading Economics online data.

For more information on Trading Economics data subscriptions, please see
https://tradingeconomics.com/analytics/features.aspx
EViews offers a custom interface to the Trading Economics data. To open the Trading Economics database, select File/Open Database... from the main EViews menu, then select Trading Economics from the Database/File type dropdown menu:


When you click on $\mathbf{O K}$, EViews will open a standard database window:


Click on the Browse on the toolbar to open the custom Trading Economics window:


The dialog interface displays five tabs/categories of data: Indicators, Market Currencies, Market Stocks, Market Commodities, and Market Bonds.

## Indicators

If you select the Indicators tab, EViews will offer an option to choose between picking a country/territory, an indicator category, or doing a search:


You can choose Pick Country/Territory and select a country from the dropdown to see historical and forecast indicators for that country:


Alternatively, you may choose Pick Indicator Category and select a category from the dropdown box to see historical and forecast indicators:


Lastly, the user can search for indicators by selecting Search in the dropdown, then selecting Term/Keyword or Symbol and then entering an expression in the search text box:


Whichever method you use to display indicators, the user- interface provides a way to filter through the results. The dialog below displays a table containing the indicator results after the user picked the country Mexico:


You may filter the results by clicking the drop-down box next to each column header to show a set of filter checkboxes. Select one or more checkboxes to display selected values or select All to show all values. Click Reset Filter button to clear all filters.


## Markets

The user may also select any of the markets tabs (Market Currencies, Market Stocks, Market Commodities, Market Bonds). By default, the dialog displays a table with all the stock market symbol results you have permission to see based on your subscription.

The example below shows the selected Market Stocks tab:


You may elect to search for a Term/Keyword or a Symbol.
Once you have found and selected the series of interest, you may drag-and-drop or click the Export to workfile button to export the series directly into an existing or new EViews workfile.

For more information click the Browse Trading Economics Stocks link at the bottom of the dialog to opens the Trading Economics webpage.

- See dbopen (p. 139) for updated command syntax.


## World Health Organization

World Health Organization (WHO) provides access to a large range of health-related data and statistics. Please note that an internet connection will be required to obtain World Health Organization online data.

For more information on data subscription, please see:
https://www.who.int/data/gho
EViews offers a custom interface to the World Health Organization data. To open the World Health Organization database, select File/Open Database... from the main EViews menu, then select World Health Organization from the Database/File type dropdown menu:


When you click on OK, EViews will open a standard database window:


Click on Browse to open the custom World Health Organization window:

| $\square$ WHO |
| :--- | :--- | :--- |
| World Health Organization |
| Q Search By |
| 2 Indicators |
|  |
| Open Browser |

To find data available within the World Health Organization, click on the folder icon Indicators and navigate through the set of nested folders:


```
World Health Organization > Indicators
4.
&% of medicines prescribed in outpatient public health care facilities that are on the EML. (MEDS8 0
$% of public health facilities with canu EM1 (maxn) /MENCR a1 n61
1. % of public health facilities %... of medicines prescribed in outpatient public health care facilities that are c
 2 13-15-years old any alcoholic beverage consumed in past 30 days, (%) (SA_0000001786)
 - 13-15-years old any alcoholic beverage consumed in past 30 days, (%) (SA_0000001786_ARCHN
2. 13-15-years old first drink before age 14, (%) (SA_0000001787)
 & 13-15-years old first drink before age 14, (%) (SA_0000001787_ARCHIVED)
2 15-19 years old abstainers, lifetime (%) (SA_0000001757)
2 15-19 years old abstainers, lifetime (%) (SA 0000001757 ARCHIVED)
 & 15-19 years old former drinkers (%) (SA_0000001758)
4. 15-19 years old former drinkers (%) (SA_0000001758_ARCHIVED)
2. 15-19 years old heavy episodic drinkers (drinkers only). % (SA_0000001756)
A 15-19 years old heavy episodic drinkers (drinkers only). % (SA_0000001756_ARCHIVED)
< >
Open Browser
```

Simply click on any folder to move down into subtopics:


The full path of the active folder will be shown in the header at the top of the window. You can click on any folder within the path to navigate back to that folder. For example, clicking on By country will move back up a single level to show the country choices. Similarly, clicking on the ".." in the window listing will move up a single level.

Alternately you may click on Search By to search for a keyword:


Once you have found and selected the series of interest, you may drag-and-drop (or copy-and-paste) the series directly into an EViews workfile. For more information click the Open Browser link at the bottom of the dialog to open the World Health Organization URL.

- See dbopen (p. 139) for updated command syntax.


## Matrix Language Support

EViews 13 offers improved support for working with matrices, vectors, etc.

## New Data import/export engine

An all new data engine makes it easier than ever to get data into and out of external data sources and offers improved support for different data formats.

## Import Data

You may now read directly into an EViews matrix object (matrix, vector, sym) from text (both ASCII and binary), HTML, Excel XLSX, and Excel 97 XLS files. Excel reads includes support for named ranges and multiple pages. The new engine supports a number of different formats, and features EViews interactive data import wizard which walks you step-bystep through the data import, providing fine-tuned control of the pending import, previews of the final data, and command capture.

To read data into a matrix object open the matrix and select Proc/Import Data... EViews will open the standard file dialog prompting you to select a file:


Double-click to select the file, or highlight the file and click on Open. EViews will open a data import wizard:


Proceed through the steps of the wizard by filling out the desired values and clicking on Next, and click on Finish when ready to import the data. EViews will read the data into the matrix object, resizing the object to match the source size, if possible.

Note that some matrix objects offer some challenges in data import. If you have a vector import from multi-column data,

- See import (p. 192) for updated command syntax for a representative proc (for a matrix object).


## Export Data

When you are ready to export data from a matrix object, EViews 13 allows you to write to a number of formats including the various ASCII, binary, HTML, RTF, and Excel formats, along with LaTeX, Markdown, and PDF files. Importantly, the Excel XLSX export allows you to write the matrix results into existing Excel files, beginning at a specified cell.

To write the contents of the matrix, select Proc/Export Data... from the matrix menu:


Enter a file name in the edit field, or Browse to select a file.
You may change the Data order to transpose the data prior to write, provide options related to the target file type, such as Advanced Excel Options for XLSX,

| Excel options |  | $\times$ |
| :---: | :---: | :---: |
| Settings |  |  |
| File access mode: | Create new file $\quad \checkmark$ |  |
| Destination sheet and cell: | Sheet1! A 1 |  |
|  | (Ex: 'Sheet1!A1') |  |
| Cell formatting: | Overwrite with EViews formats $\checkmark$ |  |
| OK | Cancel |  |

which permit writing into an specific sheet and cell of a new or existing file, with or without formatting.

- See export (p. 161) for updated command syntax for a representative proc (for a matrix object).


## Row and Column Label Support

EViews allows you to attach row and column labels to matrix objects. While prior versions of EViews offered limited ability to define and use these labels, EViews 13 extends the feature to the entire family of matrix objects and offers two additional ways in which labels may be used:

- To label rows and columns in the spreadsheet
- To refer to rows and columns of the matrix when accessing data

By default, there are no row and column labels in matrix objects. When spreadsheets show the contents of the matrix, the rows are labeled as "R1", "R2", etc., and the columns are labeled as "C1", "C2", "C3", etc.

To define row and column labels, you may open a matrix object display the spreadsheet view:


By default, there are no labels defined, but you may use the matrix procs setrowlabels and setcollabels to assign new values:

- See setcollabels (p. 238) and setrowlabels (p. 260) for updated command syntax (a representative entry for the matrix object).
- See clearcollabels (p. 121) and clearrowlabels (p. 122) (for the matrix object).

For example,

```
matO1.setcollabels "First" "Alternate"
```

sets column labels for the first two columns.
By default, these labels will be used in the spreadsheet display of the matrix:


The Row/Collabels + /- toggles on and off the display of the labels.
EViews 13 adds the ability to extract data from the matrices using the row and column labels. For example, with the matrix object, we have matrix data members,
@col(arg) $\qquad$ .Returns the columns defined by arg.
@dropcol(arg) ......Returns the matrix with the columns defined by arg removed
@droprow (arg) .....Returns the matrix with rows defined by arg removed.
@dropsub(arg1, arg2)Returns the matrix with the rows defined by arg1 and columns defined by arg2 removed.
@row(arg) ............Returns the rows defined by arg.
$@ \operatorname{sub}(\arg 1, \arg 2)$..Returns the matrix with rows defined by arg1 and columns with defined by arg2.
that all take args that may be integers, vectors of integers, string, or svectors of strings. Importantly, integer values will correspond to row and column indices. while string values will correspond to previously defined row and column labels.

Thus, in our example from above,

```
vector vec1 = mat01.@col("Alternate")
```

will create the vector VEC1 containing the column labeled with "Alternate" in MATRIX01. This command is equivalent to

```
vector vec1 = mat01.@col(2)
```

See "Matrix Extraction Data Members," on page 86 for additional discussion.

## Matrix Data Access

EViews 13 offers a useful set of tools to facilitate extracting parts of matrices for further use. Some of the functions described below are available in limited form in prior versions of EViews, but the new additions round out the suite of functions for accessing data, allowing for easy-to-use operations that were previously difficult to perform.

## Useful Utility Functions

There are four utility functions that are particularly useful for working with matrix data.
Briefly:

- @fill(n1, n2, n3, ...) - return a numeric vector with the specified values.
- @range ( $n 1, n 2$ ) - return a numeric vector with the sequential integer values from $n 1$ to $n 2$.
- @seq(s,d,n) - return a numeric vector with the arithmetic sequence of $n$ elements beginning with $s$ and incrementing by $d$.
- @sfill("str1", "str2", "str3", ...) - return a svector using the specified double-quote enclosed strings.

Note that the first three functions create vector objects, while the latter creates an svector (string vector) object.

These functions are straightforward in intention. While @fill and @seq will work with arbitrary numeric values, we are interested here in their use in generating vectors of integer values.

```
vector idx1 = @fill(1, 2, 4, 5, 7)
```

creates the vector with elements $\{1,2,4,5,7\}$, while

```
vector idx2 = @range(1, 5)
```

creates the vector with elements $\{1,2,3,4,5\}$, and

```
vector idx3 = @seq(1, 2, 4)
```

creates the vector with elements $\{1,3,5,7\}$, and

```
svector idx4 = @sfill("apple", "pear", "orange")
```

creates the svector with the values \{"apple", "pear", "orange"\}.

## Matrix Extraction Data Members

The best way to extract data from a matrix object is to use matrix object member functions. Some of these member functions were available for selected objects in earlier versions and some are new in EViews 13. Nevertheless, all of the functions have enhanced scope in

EViews 13 along with new functionality for referencing data using matrix row and column labels.

The relevant functions are:

- obj.@col(arg) - returns matrix object containing column(s) of obj associated with arg
- obj.@row(arg) - returns matrix object containing row(s) of obj associated with arg
- obj.@sub(arg1, arg2) - returns matrix object containing row(s) and col(s) of obj associated with arg1 and arg2, respectively
- obj.@dropcol $(\arg )$ - returns matrix object containing column(s) of obj not associated with arg
- obj.@droprow(arg) - returns matrix object containing row(s) of obj not associated with arg
- obj.@dropsub(arg1, arg2) - returns matrix object containing row(s) and col(s) of obj not associated with arg1 and arg2, respectively

For symmetric matrices,

- obj.@sub(arg) - returns sym object containing row(s) and col(s) of obj associated with arg, respectively
- obj.@dropsub(arg) - returns sym object containing row(s) and col(s) of obj not associated with arg, respectively
where
- obj is the name of a matrix object in the workfile
- arg, arg1, arg2 are integers, scalar objects, vectors, or svectors

For cases where args are numeric, the arg values act as row or column indices. Focusing on column functions, we have, for example,

```
vector v1 = x.@col(3)
matrix m1 = x.@col(cid)
```

where X is a matrix and CID is a vector of column indices, and the two lines return the vector V1 and matrix M1 containing the 3rd column of X, and the columns of X referenced in CID, respectively. Note that the elements of CID must be integers from 1 to the number of columns of X .

For cases where args are strings, the arg values are examined to find matches in the corresponding row or column labels ("Row and Column Label Support," on page 83). For example,

```
vector v2 = x.@col("apple")
matrix m2 = x.@col(scid)
```

were SCID is a svector of strings, and the two lines return the vector V1 and matrix M1 containing the 3rd column of X , and the columns of X with labels that match the elements of SCID, respectively. Note that the elements of SCID must be strings that match the column names previously assigned to X .

You may mix the types of arg1, arg2 in data member functions that take two arguments so that, for example,

```
matrix m3 = x.@sub(3, "apple")
```

returns M3 containing the element of X in row 3 and column with label "apple".
Similarly, the commands

```
matrix v1d = x.@dropcol(3)
matrix m1d = x.@dropcol(cid)
matrix v2d = x.@dropcol("apple")
matrix m2d = x.@dropcol(scid)
matrix m3d = x.@dropsub(5, "apple")
```

return the matrix objects

- V1D - the matrix X with column 3 dropped
- M1D - the matrix $X$ with columns referenced by CID dropped
- V2D - the matrix X with the column with label "apple" dropped
- M2D - the matrix X with the columns with labels in SCID dropped
- M3D - the matrix with the row 5 dropped and the column labeled "apple" dropped


## Matrix Extraction with Utility Functions

Combining matrix utility functions ("Useful Utility Functions," on page 86) with the matrix extraction data members offers quite flexible methods for obtaining data from matrices.

Consider, for example, the extraction of multiple columns from the matrix X using the @col data member function:

```
vector xid = @fill(1, 3, 5, 9)
matrix xsub = x.@col(xid)
```

extracts columns $\{1,3,5,9\}$ from the matrix X .
Since the args in the member data extraction functions may themselves be expressions, we may combine the two lines into a single expression:

```
matrix xsub1 = x.@col(@fill(1, 3, 5, 9))
```

Similarly, extracting or dropping the 7 through 9th columns of $X$ may be done using

```
matrix xsub2 = x.@col(@range(7, 9))
matrix xsub3 = x.@dropcol(@range(7, 9))
```

We can perform the same compound extractions in 2-dimensions, as in

```
matrix xsub4 = x.@sub(@range(3, 6), @sfill("apple", "orange"))
matrix xsub5 = x.@dropsub(4, @sfill("apple", "orange"))
```

which create XSUB4 containing rows 3 to 6 and columns with labels matching "apple" and "orange" of X, and XSUB5 containing X after dropping row 4 and the columns with matching labels.

## Preliminary Updates to Function Reference

EViews 13 offers a number of new functions

## Cumulative Statistic Functions

| Function | Name | Description |
| :--- | :--- | :--- |
| @cumdp (x, "date", [,s]) | cumulative sum of <br> positive differences | cumulative (partial sum) process <br> of positive changes around a <br> threshold value at "date". |
| @cumdn (x, "date", $[, s]$ ) | cumulative sum of <br> negative differences | cumulative (partial sum) processes <br> of negative changes around a <br> threshold value at "date". |

## Workfile Functions

@holiday. returns the proportion of an annual event covered by each observation (p. 92). (updated)
@holidayset.......... return the proportion of an annual set of events covered by each observation (p. 92). (updated)

## Support Functions

@makevalidname. string containing an uppercased valid EViews name based on the input (p. 93).
@xtype returns the string describing the type of the active external application (p. 102).

## String Function Summary

## String Functions

@str $\qquad$ returns a string representing the given number (p. 94). (updated)
@val $\qquad$ returns the number that a string represents (p. 99). (updated)
@wreplace $\qquad$ replaces parts of a string based on patterns (p. 102). (updated)

## Matrix Command and Function Summary

## Matrix Utility Functions

@range $\qquad$ Returns a vector holding the sequential integers staring at $l$ and ending at $h$ (p. 98). (new)
@sfill $\qquad$ Returns a svector containing the elements specified by the arguments to the function. (p. 99) (new)

## @holiday

Syntax: @holiday(event[, basis][, flag...])
returns the proportion of an annual event covered by each observation. The event argument is a string specifying either a single date, a pair of dates (forming a range), or a single named group of holidays. Each item in the argument has the format
"base[~|!][(offset)][[weights]]"
The base component is either a day-of-the-month specification, e.g. "Dec 25 ", a n-th-week-day-of-the-month specification, e.g. "Nov4Thu" (fourth Thursday in November) or "May1Mon" (last Monday in May), or a named holiday.

For example,

```
@holiday("Jan1")
@holiday("Veterans.us(7)Thanksgiving.us")
@holiday("NewYears[1, 2,0]")
```


## Cross-references

See "Holiday Functions," on page 65 for extensive discussion and additional syntax. See also @holiday (p. 92).

## @holidayset

Syntax: @holidayset(event[, basis][, flag...])
return the proportion of an annual set of events covered by each observation. The event argument is a list of dates and/or named groups of holidays. Each item in the argument has the format

```
"base[~|!][(offset)][[weights]]"
```

The base component is either a day-of-the-month specification, e.g. "Dec25", a n-th-week-day-of-the-month specification, e.g. "Nov4Thu" (fourth Thursday in November) or "May1Mon" (last Monday in May), or a named holiday.

For example,
@holidayset("Easter! AllSaints! Christmas!")

## Cross-references

See "Holiday Functions," on page 65 for extensive discussion and additional syntax. See also @holiday (p. 92).

## @makevalidname <br> Support Functions

Syntax: @makevalidname(str)

Argument: string, str
Return: string, name
Returns a string containing an uppercased valid EViews name based on str. If $s t r$ is a valid name, then the original string $s t r$ is returned. If $s t r$ is not valid, invalid characters will be replaced in the new string with "_" prior to the return (i.e. the string "re!sult\%" will return "RE_SULT_").

## Cross-references

See also @isvalidname (p. 787) and @getnextname (p. 784).

| @range | Matrix Utility Functions |
| :--- | :--- |
| Syntax: | @range $(x, y)$ |
| Argument 1: | integer, $x$ |
| Argument 2: | integer, $y$ |
| Return: | vector |

Returns a vector holding the sequential integers staring at $x$ and ending at $y$.
Example:

```
@range(1, 10)
```

@range (17, 20)
@range (10, -1)

See also @seqm (p. 756).

| @sfill | Matrix Utility Functions |
| :--- | :--- |
|  | Syntax: |
| Arguments: | @sfill(" $\operatorname{str} 1 ", " \operatorname{str} 2 ", " \operatorname{str} 3 ", .)$. |
| Return: | svector |

Returns a svector containing the elements specified by the arguments to the function. The vector will have length equal to the number of arguments. The maximum number of arguments is 96 .

## Example

```
svector idx4 = @sfill("apple", "pear", "orange")
```

creates the svector with the values \{"apple", "pear", "orange"\}.
See @fill (p. 83), Vector: :fill (p. 1065), Coef::fill (p. 25), Matrix::fill (p. 492), Rowvector::fill (p. 619), and Sym::fill (p. 870) for routines to perform general filling of matrix objects.

| @str | String Functions |
| :--- | :--- |
| Syntax: | @str $(d[, f m t])$ |
| Argument 1: | scalar or vector or series $d$ |
| Argument 2: | numeric format string, fmt |
| Return: | string or svector or alpha series |

Returns a string representing the given number or a vector or alpha series containing the string representations of the values in $d$. You may provide an optional format string. (See also @val (p. 99) to convert strings into numbers.)

EViews offers a variety of ways to write your number as a string. By default, EViews will format the number string using 10 significant digits, with no leading or trailing characters, no thousands separators, and an explicit decimal point.
(The default conversion is equivalent to using @str with the format " g .10 " as described below.)

If you wish to write your number in a different fashion, you must provide an explicit format string. A numeric format string has the format:

$$
\text { [type][t][ }+][(][\$][\#][<|=|>][0][\text { width }][[. \mid . .] \text { precision }][\%][)]
$$

There are a large number of syntax elements in this format but we may simplify matters greatly by dividing them into four basic categories:

- format: [type]
- width: [ $<$ | = | > ][width]
- precision: [precision]
- advanced modifiers: the remaining elements (leading and trailing characters, padding and truncation modifiers, separators, display of negative numbers)

The type, width, and precision components are the basic components of the format so we will focus on them first. We describe the advanced modifiers in "Modified Formats" on page 97.

## Basic Formats

EViews offers formats designed for both real-valued and integer data.

## Basic Real-Value Formats

The basic real-value format is:

```
[type][ < | = | > ][width][.][precision]
```

The type component is a single character indicating the basic type and the width and precision arguments are numbers indicating the number of output characters and the precision at which the numbers should be written. If specified, the precision should be separated from the type and width portion of the format by a "." character (or ".." as we will see in "Modified Formats" on page 97).

If you specify a format with neither width nor precision, EViews will format the number at full precision with matching string width.

The following types are for use with real-valued data:

| g | significant digits <br> z |
| :--- | :--- |
| c | significant digits with trailing zeros <br> fixed characters with single leading space for posi- <br> tive numbers |
| f | fixed decimal places |
| e | scientific/float |
| p | percentage (same as "f" but values are multiplied <br> by 100) |
| s | suppressed decimal point format |
| r | ratio, e.g., "30 $1 / 5 "$ |

The type character may be followed by a width specification, consisting of a width indicating the number of output characters, optionally preceded by a " $>$ ", " $=$ " or " < " modifier.

- If no width is provided, the number will be rendered in a string of the exact length required to represent the value (e.g., the number 1.23450 will return " 1.2345 ", a string of length 6).
- If an unmodified width or one with the " > " modifier is provided, the specified number places a lower-bound on the width of the string: the output will be left-padded to the specified width, if necessary, otherwise the string will be lengthened to accommodate the full output. By default, padding will use spaces, but you may elect to use 0's by providing an advanced modifier ("Modified Formats" on page 97).
- If the" < " modifier is provided along with width, the width places an upper-bound on the width of the string: the output will not be padded to the specified width. If the number of output characters exceeds the width, EViews will return a width-length string filled with the "\#" character.
- If the" = " modifier is provided along with width, the width provides an exact-bound for the width of the string: the output will be padded to specified width, if necessary. If the number of characters exceeds the width, EViews will return a width-length string filled with the "\#" character.

If you specify a precision, the interpretation of the value will vary depending on the format type. For example, precision represents the number of significant digits in the " g " and " z " formats, the number of characters in the " $c$ " format, and the number of digits to the right of the decimal in the " f ", "e", " p ", and " s " formats. For the " r " format, the precision determines maximum denominator for the fractional portion (as a power-of-10).

The following guidelines are used to determine the precision implied by a number format:

- If you specify a format with only a precision specification, the precision will implicitly determine the width at which your numbers are written.
- If you specify a format with only a width specification, the width will implicitly determine the precision at which your numbers are written. Bear in mind that only the modified width specifications " = width" and " < width" impose binding restrictions on the precision.
- If you specify a format using both width and precision, the precision at which your numbers are written will be determined by whichever setting is most restrictive (i.e., " $\mathrm{f}=4.8$ " and " $\mathrm{f}=4.2$ " both imply a formatted number with two digits to the right of the decimal).)


## Basic Integer Formats

The basic integer format is:

$$
\text { [type] }[<|=|>][\text { width] }
$$

The type component is a single character indicating the basic type. The following types are for use with integer data:

| i | integer |
| :--- | :--- |
| h | hexidecimal |
| o | octal |
| b | binary |

If one of the integer formats is used with real-valued data, the non-integer portion of the number will be ignored. You may specify a width using the syntax and rules described in "Basic Real-Value Formats" on page 95.

## Modified Formats

Recall that the syntax of a numeric format string is given by:

$$
\text { [type][t][ }+][(][\$][\#][<|=|>][0][\text { width }][[. \mid . .] \text { precision }][\%][)]
$$

In addition to the basic type, width, and precision specifiers, the formats take a set of modifiers which provide you with additional control over the appearance of your output string:

- You may combine any of the real-value format characters ("g", "e", " f ", etc.) with the letter "t" ("gt", "et", "ft", etc.) to display a thousands separator ("1,234.56"). By default, the separator will be a comma ",", but the character may be changed to a "." using the ".." format as described below.
- You may add a " + " symbol after the format character ("g + ", "et + ", "i +") to display positive numbers with a leading "+".
- To display negative numbers within parentheses, you should enclose the remaining portion of the format in parentheses " $\mathrm{ft}+(\$ 8.2)$ ".
- Add "\$" to the format to display a leading "\$" currency character.
- You should include a "\#" to display a trailing point in scientific notation (e.g., "3.e +34 ").
- The width argument should include a leading zero ("0") if you wish padded numbers to display leading zeros instead of spaces ("g08.2", "i05").
- If you added a " t " character to your real-value format type, you may replace the usual "." width-precision separator with ".." ("ft < 08..2", "e = 7..", "g..9", etc.) to reverse the
thousands and decimal separators so that thousands will be separated by a "." and decimal portion by a "," (" $1.234,56$ ").
- Adding a " \%" character to the end of the format adds the "\%" character to the end of the resulting string.


## Examples

```
string num = @str(1.23)
```

assigns to the string NUM the text " 1.23 ".

```
alpha alpha1 = @str(-123.88)
```

assigns the string "-123.88" to the alpha series ALPHA1.

```
string num = @str(123456,"z.9")
```

returns a string formatted to 9 significant digits with trailing zeros: "123456.000".

```
string num = @str(123456,"z.4")
```

returns a string formatted to 4 significant digits with trailing zeros: " $1.235 \mathrm{e}+05$ ". Note that since the input number has more than 4 significant digits, no trailing zeros are added and the resulting string is identical to one that would have been produced with a " g .4 " format.

```
string num = @str(126.543,"c.7%")
```

returns a string with exactly 7 characters, including a leading space, and a trailing percent sign: " $126.5 \%$ ".

```
string num = @str(126.543,"p.7")
```

converts the number 126.543 into a percentage with 7 decimal places and returns
" 12654.3000000 ". Note no percent sign is included in the string.

```
string num = @str(1.83542,"f$5.4")
```

returns " $\$ 1.8354$ ". The width selection of " 5 " with an implicit " $>$ " modifier is irrelevant, since the precision setting of " 4 ", coupled with the insertion of the " $\$$ " symbol yields a string with more characters than " 5 ".

```
string num = @str(1.83542,"f$8.4")
```

returns " $\$ 1.8354$ ". Here the width selection is binding, and a leading space has been added to the string.

```
string num = @str(1.83542,"f$=5.4")
```

returns " $\$ 1.84$ ". The explicit " = " width modifier causes the width setting of " 5 " to be binding.

```
string num = @str(524.784,"r")
```

converts the number 524.784 into a ratio representation: " $52498 / 125$ ".

```
string num = @str(1784.321,"r=3")
```

will return "\#\#\#", since there is no way to represent 1784.321 as a string with only 3 characters.

```
string num = @str(543,"b")
```

converts the number 543 into a binary representation: "1000011111".
The matrix command

```
svector svec1 = @str(v1)
```

converts the numeric values of vector V1 to strings and puts the results in the svector SVEC1. If the svector SVEC1 exists it will be sized to match the rows of V1 and missing values will be converted to empty strings.

The series command

```
alpha a1 = @val(gdp)
```

creates the alpha series A1 and converts the numeric values of series GDP to string. NA values will be become empty strings.

Format strings may be used to govern the conversion,

```
svector svbin = @str(vec1, "e")
```

converts the numeric values in the vector VEC1 into their strings representation in scientific notation and assigns them to the svector SVBIN.

See @val (p. 99) to convert a string into a number.
$\begin{array}{|l|l|}\hline \text { @val } & \text { String Functions } \\ \hline & \text { Syntax: }\end{array} \quad$ @val (arg $\left.[, f m t]\right)$.

Returns the number that string arg represents, or a vector or series containing the converted values of arg. You may provide an optional numeric format string fmt. (See @str (p. 94) to convert a number into a string.)

In most cases, EViews will be able to convert your string into the corresponding numeric value without additional input. If EViews is unable to perform a conversion, it will return a missing (NA) value.

There are a few important conventions used in the conversion process:

- A leading " $\$$ " in the string will be ignored.
- Strings enclosed in "( )" or with a leading "-" will be treated as negative numbers. All other numeric strings, including those with a leading "+" will be treated as positive numbers. You may not have a leading " + " or "-" inside of the parentheses.
- A trailing "\%" sign instructs EViews to treat the input string as a percentage-the resulting value will be divided by 100 .

There are some situations where you must provide a numeric format string so that EViews can properly interpret your input. The syntax for the format string depends on the type of number the string represents.

## Real-Value Formats

EViews will properly interpret non-delimited decimal and scientific notation numeric input strings as numbers.

If your string uses "," to separate thousands, you should specify the " t " format string to remove "," delimiters prior to conversion. If the string uses "." to separate thousands, you should use "t.." to instruct EViews to remove "." delimiters.

If your input string represents a number with suppressed decimal format, you should include a format string beginning with the letter " s ":
s.precision suppressed decimal point format (precision determines the number of digits to the right of the decimal)

EViews will divide the resulting number by 10 raised to the power of the specified precision. The "s" format specification may be followed by a "t." or a "t.." specification if necessary.

Integer Formats

| r | ratio (e.g., "30 $1 / 5$ "). |
| :--- | :--- |
| i | integer |
| h | hexidecimal |
| o | octal |
| b | binary |

You should use the "r", "h", "o", or "b" formats to indicate that your input is in the specified format. The "i" format is generally not necessary unless you wish to produce a missing value for a non-integer input string.

## Examples

scalar num = @val("\$1.23")
assigns the scalar NUM the numeric value 1.23.

```
series ser1 = @val("-$123.88")
```

returns the value -123.88.

```
scalar sperct = @val("478%")
```

divides the value by 100 , setting the scalar SPERCT to 4.78 .

```
scalar sratio = @val("(321 1/5)", "r")
```

sets the scalar SRATIO equal to -321.2

```
scalar shexa = @val("f01a", "h")
```

treats the string "f01a" as a hexadecimal number, converts it into the decimal equivalent, 61466, and assigns it to the scalar object SHEXA.

```
scalar sbin = @val("11110101", "b")
```

interprets the string " 11110101 " as a binary number, converts it into the decimal equivalent, 245 , and assigns it to the scalar SBIN.

To verify that a value is an integer, you may use the " i " option.

```
scalar sintna = @val("98.32", "i")
scalar sint = @val("96", "i")
```

SINTNA will contain a missing value NA since the input represents a non-integer value, while SINT is set to 96 .

You may use @val to convert values in an svector into a vector. The matrix command,

```
vector v = @val(sv1)
```

converts the string values of svector SV1 to numeric values and returns the values in the svector V. If the vector V exists it will be sized to match the rows of SV1 and non-numeric strings will be converted to NA.

The series command

```
series x = @val(alpha1)
```

converts the string values in the alpha series ALPHA1 to numeric values and returns the values in the series X . Non-numeric strings will be converted to NA.

Format strings may be used to govern the conversion,

```
vector vbin = @val(svbin, "b")
```

interprets the strings in the svector SVBIN as binary numbers, converts it into their decimal equivalents and assigns it to the vector VBIN. If for example, SVBIN contained " 110 " " 001 " and " 010 ", the resultant VBIN will contain 6,1 , and 2.

See also @str (p. 94).

| @wreplace | String Functions |
| :--- | :--- |
| Syntax: | @wreplace_(str_list, "src_pattern", replace_pattern" [, "all"]) |
| Argument 1: | string list, str_list |
| Argument 2: | string pattern, src_pattern |
| Argument 3: | string pattern, replace_pattern |
| Argument 4: | string literal, "all" |
| Return: | string list |

Replaces instances of src_pattern in str_list with replace_pattern. The pattern lists may be made up of any number of "?" (indicates any single character) or "*" (indicates any number of characters). The pattern is case-sensitive and must exactly match the str_list characters to be replaced. Only the first instance of src_pattern within each word of str_list is replaced unless the optional flag "all" is specified (enclosed in quotes), in which case all instances within each word are replaced.

Example:
@wreplace("ABBC AB", "*B*", "*X*")
replaces the first instance of " B " with " X ", returning the string "AXBC AX ".

```
@wreplace("ABBC AB", "*B*", "*X*", "all")
```

replaces all instances of "B" with "X", returning the string "AXXC AX".
@wreplace("ABC DDBC", "??B?", "??X?")
replaces all instances of " B " which have two leading characters and one following character, returning the string "ABC DDXC".

See also @wdrop (p. 687) and @wkeep (p. 690).

| @xtype | Support Functions |
| :--- | :--- |
| Syntax: | @xtype |
| Return: | string |

Returns the string describing the type of the active external application:

- R connections return "rconn".
- Matlab returns " m ".
- Python returns "pyconn".

Returns an empty string if no external connection is active.

## Examples

string y = @xtype
returns "pyconn" if the external application type is Python..
Cross-references

## Preliminary Updates to Command Reference

This chapter contains preliminary documentation for commands that are new or have been updated in EViews 13.

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 13 Commands

Coef
Coef Procs
export ...................save coef as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 158). (new)
import...................imports data from a foreign file into the coef object (p. 186). (updated)

## Coef Values

@droprow (arg) .....Returns the coef with the rows defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@row (arg) ............Returns the rows defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)

## Equations

Equation Methods
ardl.......................least squares with autoregressive distributed lags (p. 112). (updated)
did $\qquad$ estimate a panel equation using the difference-in-difference estimator (p. 143). (new)
ls $\qquad$ equation using least squares or nonlinear least squares (p. 112). (updated)

## Equation Views

boundstest $\qquad$ perform the Pesaran, Shin and Smith (2001) bounds test of long-run relationships from an ARDL estimated equation (p. 118). (updated)
cointrel................. display information about the cointegrating relation specification and the coefficients in ARDL estimated equation (p. 137). (new)
didcs $\qquad$ compute Callaway-Sant'Anna decomposition for difference-in-difference estimation (p. 144). (new)
didgbdecomp perform Goodman-Bacon decomposition for difference-in-difference estimation (p. 144). (new)
didtrends.............. show difference-in-difference trends summary in graphical or tabular form (p. 146). (new)
dynmult $\qquad$ compute dynamic multipliers for long-run regressors in ARDL equations (p. 150). (new)
ecresults $\qquad$ display the conditional error correction (CEC) and error correction (EC) regression results (p. 154). (new)
similarity $\qquad$ compute symmetry test for nonlinear distributed lag variables in nonlinear ARDL models (p. 273). (new)
symmtest. $\qquad$ compute symmetry test for nonlinear distributed lag variables in nonlinear ARDL models (p. 276). (new)

## Equation Procs

didmakeeq $\qquad$ create an equation object with the underlying fixed-effects estimation of a difference-in-difference equation (p. 145). (new)
fit. $\qquad$ static forecast (p. 174). (updated)
forecast dynamic forecast (p. 180). (updated)

## Equation Values

@varselkept $\qquad$ space delimited list of variables kept by model selection. (new)
@varselrejected. space delimited list of the variables dropped by model selection. (new)

## Geomap

## Geomap Procs

setfillcolor $\qquad$ define the fill (background) color used in geomap shapes using values in a series (p. 245). (updated)
setjust $\qquad$ set the display justification for multi-line area labels (p. 259). (new)
setshapelabel set which attribute to use or create a custom label to use when labeling shapes (p. 263). (new)

## Geomap Values

@ids("attr", "val") $\qquad$ space delimited string containing the ID numbers of all the areas which has the matching attribute value for the specified attribute name. (new)

## Graph

## Graph Procs

datelabel ...............controls labeling of the bottom date/time axis in time plots (p. 137). (updated)
setelem .................set individual line, symbol, bar and legend options for each series in the graph (p. 240). (updated)

## Group

## Group Views

coint $\qquad$ test for cointegration between series in a group (p. 124). (updated)

## Group Procs

setfillcolor.............set custom spreadsheet fill coloring for the group (p. 255). (updated)
settextcolor $\qquad$ set custom spreadsheet text coloring for the group (p. 264). (updated)

## Matrix

## Matrix Procs

clearcollabels ........clear the column labels in a matrix object (p. 121). (new)
clearrowlabels .......clear the row labels in a matrix object (p. 122). (new)
export ...................save matrix as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 161). (new)
import..................imports data from a foreign file into the matrix object (p. 192). (updated)
resize $\qquad$ resize the matrix object (p. 233) (new).
setcollabels set the column labels in a matrix object (p. 238). (updated)
setrowlabels .set the row labels in a matrix object (p. 260). (updated)

## Matrix Values

@col(arg) $\qquad$ Returns the columns defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (updated)
@ dropcol(arg) ......Returns the matrix with the columns defined by arg removed. arg may be an integer, vector of integers, string, or s svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (new)
@droprow (arg) .... Returns the matrix with rows defined by arg removed. The arg may be integer, vectors of integers, strings, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@dropsub (arg1, arg2)Returns the matrix with the rows defined by arg1 and columns defined by $\arg 2$ removed. The args may be integers, vectors of integers, string, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@row(arg) $\qquad$ Returns the rows defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (updated)
@sub(arg1, arg2).. Returns the matrix with rows defined by arg1 and columns with defined by arg2. The args may be integers, vectors of integers, strings, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels.
(updated)

## Rowvector

## Rowvector Procs

clearcollabels........ clear the column labels in a rowvector object (p. 121). (new)
clearrowlabels ...... clear the row labels in a rowvector object (p. 123). (new)
export .................. export vector as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 164). (new)
import.................. imports data from a foreign file into the vector object (updated) (p. 198). (updated)
setcollabels........... set the column labels in a rowvector object (p. 238). (new)
setrowlabels ......... set the row labels in a rowvector object (p. 261). (new)

## Rowvector Values

$@ \operatorname{col}(\arg ) . . . . . . . . . . .$. Returns the columns defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (new)
@dropcol(arg) ...... Returns the matrix with the columns defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (new)

## Series

## Series Procs

autoarma ..............forecast from a series using an ARIMA model with automatic determination of the specification (p. 115). (updated)
dsa seasonally adjust daily data using the DSA method (p. 146). (new)
ets perform Error-Trend-Season (ETS) estimation and exponential smoothing (p. 154). (updated)
forcavg average forecasts of a series (p. 182). (updated)
setfillcolor.............define the fill (background) color used in series spreadsheets (p. 255). (updated)
settextcolor $\qquad$ set custom spreadsheet text coloring for the series (p. 269). (updated)
smooth. $\qquad$ exponential smoothing (p. 274). (updated)

## Svector

## Svector Procs

clearcollabels ........clear the column labels in a svector object (p. 122). (new)
clearrowlabels.......clear the row labels in a svector object (p. 123). (new)
fill.........................fill the svector with the specified values (p. 173). (new)
resize....................resize the svector object (p. 234). (new)
setcollabels ............set the column labels in a svector object (p. 239). (new)
setrowlabels ..........set the row labels in a svector object (p. 261). (new)

## Svector Values

@droprow (arg) .....Returns the svector with the rows defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@row(arg) ............Returns the rows defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)

Sym
Sym Procs
clearcollabels ........clear the column labels in a vector object (p. 122). (new)
clearrowlabels.......clear the row labels in a vector object (p. 123). (new)
export $\qquad$ save sym matrix as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 167). (new)
import $\qquad$ imports data from a foreign file into the sym object (updated) (p. 869). (new)
resize resize the sym object (p. 234). (new)
setcollabels $\qquad$ set the column labels in a sym object (p. 239). (new)
setrowlabels set the row labels in a sym object (p. 262). (new)

## Sym Values

$@ \operatorname{col}(\arg ) . . . . . . . . . . .$. Returns the columns defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (new)
@dropcol(arg) ...... Returns the matrix with the columns defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined column labels. (new)
@droprow(arg) .... Returns the matrix with rows defined by arg1 and columns with rows defined by arg2 removed. The args may be integer, vectors of integers, strings, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
$@$ dropsub (arg) ..... Returns the sym with the rows and columns defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@dropsub (arg1, arg2)Returns the matrix with the rows defined by arg1 and columns defined by arg2 removed. The args may be integers, vectors of integers, string, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@row(arg) ........... Returns the rows defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
$@ \operatorname{sub}(\arg ) . . . . . . . . .$. Returns the sym with rows defined by arg1 and columns with rows defined by arg2. The args may be integers, vectors of integers, strings, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
$@$ sub (arg1, arg2) ..Returns the matrix with rows defined by arg1 and columns with defined by arg2. The args may be integers, vectors of integers, strings, or svectors of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (updated)

## Table

## Table Procs

fixcol ....................fixes a set of columns to left of the spreadsheet view so that the leading columns are always in view (p. 179). (new)
fixrow $\qquad$ fixes a set of rows at the top of the spreadsheet view so that the leading rows are always in view (p. 179). (new)
fixrowcol ..............fixes a set of rows at the top and a set of columns to left of a spreadsheet view so that the leading rows and columns are always in view (p. 180). (new)
save save table as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 235). (updated)
setfillcolor $\qquad$ set the fill (background) color of a set of table cells (p. 943). (updated)

## VAR

## Var Methods

btvcvar .................estimate a Bayesian time-varying coefficients VAR specification $x$ (p. 119). (new)
ec $\qquad$ estimate a vector error correction model (p. 151). (updated)

## Var View

coint .....................Johansen cointegration test (p. 133). (updated)
impulse.................impulse response functions (p. 216). (updated)
vdecomp ...............variance decomposition (p. 276). (updated)

## Var Procs

fit produce static forecasts from an estimated VAR (p. 174). (updated) forecast produce dynamic forecasts from an estimated VAR or VEC (p. 180). (updated)

## Vector

## Vector Procs

clearcollabels ........clear the column labels in a vector object (p. 122). (new)
clearrowlabels.......clear the row labels in a vector object (p. 123). (new)
export $\qquad$ export vector as Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, PDF, TEX, or MD file on disk (p. 170). (new)
import................. imports data from a foreign file into the vector object (p. 210). (updated)
resize $\qquad$ resize the vector object (p. 234). (new)
setcollabels set the column label for the vector object (p. 240). (updated)
setrowlabels set the row labels for the vector object (p. 262). (updated)

## Vector Values

@droprow (arg) .... Returns the vector with the rows defined by arg removed. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (new)
@row(arg) ........... Returns the rows defined by arg. arg may be an integer, vector of integers, string, or svector of strings. Integer values correspond to rows and string values correspond to previously defined row labels. (updated)

## Object Container, Data, and File Commands

dbopen $\qquad$ open a database (p. 139). (updated)
pageload.............. load one or more pages into a workfile from a workfile or a foreign data source (p. 228). (updated)
pagesave ............. save page into a workfile or a foreign data source (p. 229). (updated)
wfopen $\qquad$ open workfile or foreign source data as a workfile (p. 278). (updated)
wfsave $\qquad$ save workfile to disk as a workfile or a foreign data source (p. 293). (updated)

## Programming Commands

deleteaddin. $\qquad$ unregister a program file as an EViews Add-in (p. 142). (new)
xopen $\qquad$ open a connection to an external application (p. 298). (updated)

| ardl | Equation Methods |
| :--- | :--- |

Estimate an equation with autoregressive distributed lags using least squares.

## Syntax

equation.ardl(options) linear_regs [@ static_regs] [@asy dual_asymmetric_regs]
[@asylr long_run_asymmetric_regs] [@asysr short_run_asymmetric_regs]

The linear_regs specification is required:

- The linear_regs list should be the dependent variable followed by a list of linear dis-tributed-lag regressors.

The remaining specifications are optional

- static_regs should be a list of static regressors, not including a constant or trend term.
- dual_asymmetric_regs are distributed-lag regressors which are asymmetric both in the short-run and long-run.
- long_run_asymmetric_regs regressors are distributed lag-regressors which are asymmetric in the long-run but symmetric in the short-run.
- short_run_asymmetric_regs are asymmetric regressors which are distributed lagregressors which are asymmetric in the short-run but symmetric in the long-run.

You may specify the lag for an individual distributed-lag variable using the "@fl (variable, lag)" syntax. For instance, if the variable X should use 3 lags, irrespective of the fixed or automatic lag settings, you may specify this by entering "@fl(x, 3)" in the regressor list.

## Options

| $\begin{aligned} & \text { determ = arg } \\ & \text { (default = } \\ & \text { "rconst") } \end{aligned}$ | Johansen deterministic trend type: "none" (no deterministics), "rconst" (restricted constant and no trend), "uconst" (unrestricted constant and no trend), "rtrend" (unrestricted constant and restricted trend, "utrend" (unrestricted constant and unrestricted trend). |
| :---: | :---: |
| $\begin{aligned} & \text { trend }=\text { arg } \\ & \text { (deprecated) } \end{aligned}$ | Johansen deterministic trend type: Note: this is a deprecated s option which handles a subset of cases covered by the "determ = " option: "none" (no deterministics), "const" (restricted constant and no trend, default), "uconst" (unrestricted constant and no trend), "linear" (unrestricted constant and restricted trend, "ulinear" (unrestricted constant and unrestricted trend). |
| fixed | Do not use automatic selection for lag lengths. This option must be used with the "deplags = " and "reglags = " options. |
| $\begin{aligned} & \text { deplags }=\text { int } \\ & (\text { default }=4) \end{aligned}$ | Set the number of lags for the dependent variable to int. If automatic selection is used, this sets the maximum number of possible lags. If fixed lags are used (the fixed option is set), this fixes the number of lags. |


| reglags $=$ int <br> $($ default $=4)$ | Set the number of lags for the explanatory variables <br> (dynamic regressors) to int. If automatic selection is used, <br> this sets the maximum number of possible lags. If fixed <br> lags are used (the fixed option is set), this fixes the number <br> of lags for each regressor. |
| :--- | :--- |
| ic = key (default |  |
| $=$ "aic") | Set the method of automatic model selection. key may take <br> values of "aic" (Akaike information criterion, default), <br> "bic" (Schwarz criterion), "hq" (Hannan-Quinn criterion) <br> or "rbar2" (Adjusted R-squared, not applicable in panel <br> workfiles). |
| nodf | Do not perform degree of freedom corrections in computing <br> coefficient covariance matrix. The default is to use degree <br> of freedom corrections. |
| coef =arg | Specify the name of the coefficient vector (if specified by <br> list); the default behavior is to use the "C" coefficient vec- <br> tor. |
| prompt | Force the dialog to appear from within a program. |
| p Print results. |  |

## Examples

```
wfopen http://www.stern.nyu.edu/~wgreene/Text/Edition7/TableF5-
    2.txt
```

opens example data from Greene (2008, page 685), containing quarterly US macroeconomic variables between 1950 and 2000.

The following command

```
equation eq01.ardl(deplags=8, reglags=8) log(realcons)
    log(realgdp) @ @expand(@quarter, @droplast)
```

creates an equation object and estimates an ARDL model with the log of real consumption as the dependent variable, and the log of real GDP as a dynamic regressor. Quarterly dummy variables are included as static regressors. Automatic model selection is used to determine the number of lags of LOG(REALCONS) and LOG(REALGDP).

The command

```
equation eq02.ardl(deplags=3, reglags=3, fixed) log(realcons)
    log(realgdp) @ @expand(@quarter, @droplast)
```

estimates a second model, replicating Example 20.4 from Greene, with a fixed three lags of the dependent variable and three lags of the regressor.

```
equation eq03.ardl(deplags=1, reglags=1, fixed) log(realcons)
    log(realgdp) @asy log(realgovt)
```

The line above estimates an $\operatorname{ARDL}(1,1,1)$ model with the log of real consumption as the dependent variable, the log of real GDP as a linear regressor, and log of real government expenditures as a dual asymmetric regressor.

```
equation eq04.ardl(deplags=1, reglags=1, fixed) log(realcons)
    log(realgdp) @asy log(realgovt) @asysr log(realinvs)
```

extends the previous model and estimates an $\operatorname{ARDL}(1,1,1,1)$ model by including the log of real investments as a long-run asymmetric regressor.

```
equation eq05.ardl(deplags=1, reglags=1, fixed) log(realcons)
    log(realgdp) @asy log(realgovt) @asysr log(realinvs) @asylr
    log(tbilrate)
```

The line above extends the previous model and estimates an ARDL( $1,1,1,1,1$ ) model by including the log of treasury bill rates as a short-run asymmetric regressor.

```
wfopen oecd.wf1
equation eq06.ardl(fixed, deplags=1, reglags=1) log(cons) log(inf)
    log(inc)
```

This example estimates a panel ARDL model using the workfile "OECD.wf1". This model replicates that given in the original Pesaran, Shin and Smith 1999 paper. Model selection is not used to choose the optimal lag lengths, rather a fixed single lag of both the dependent variable and the regressor are used.

## Cross-references

See "Autoregressive Distributed Lag (ARDL) Models," beginning on page 321 of User's Guide II for further discussion.

| autoarma | Series Procs |
| :--- | :--- |

Forecast from a series using an ARIMA model with the specification of the model selected automatically.

## Syntax

series.autoarma(options) forecast_name [exogenous_regressors]

## Options

| tform = arg | Specify the type of dependent variable transformation. <br> arg may be "auto" (automatically decide between log or no <br> transformation, default), "none" (perform no transforma- <br> tion), "log" (perform a log transformation), and "bc" (per- <br> form the Box-Cox transformation. |
| :--- | :--- |
| bc = int | Set the power of the Box-Cox transformation. Only applica- <br> ble if the tform = bc option is used. |
| diff = int | Set the maximum level of differencing to test for. Default <br> value is 2. |
| maxar = int | Set the maximum number of AR terms. Default value is 4. |
| maxma = int | Set the maximum number of MA terms. Default value is 4. |
| maxsar = int | Set the maximum number of seasonal AR terms. Default <br> value is 0. |
| maxsma = int | Set the maximum number of seasonal MA terms. Default <br> value is 0. |
| periods = int | Set the periodicity of the seasonal ARMA terms. This <br> defaults to the number of observations in a year, based on <br> current workfile frequency. |
| avg = key | Use forecast averaging, rather than model selection. key <br> sets the type of averaging to perform, and may take values <br> of "aic" (SAIC weights), "sic" (BMA weights) or "uni" (uni- <br> form weights). |
| mselen = key | Set the model selection criteria. key make take values of <br> "aic" (Akaike Information Criteria, default), "sic" (Schwarz <br> Information Criteria), "hq" (Hannan-Quinn criteria) or <br> "mse" (Mean Square Error criteria). This option is ignored <br> if the "avg = " option is used. |
| msenconv | Allow non-converged models to be used in model selection <br> or forecast averaging. |
| Set the percentage of the estimation sample to be used for <br> mSE calculation. key may take values of "5", "10", "15"" or <br> "20". This option is only applicable if the "select = mse" <br> option is used. |  |
| key | Set the type of forecast to use when calculating MSE. key <br> may either be "dyn" (dynamic, default), or an integer, n, <br> between 1 and 12 indicating that an $n$-step static forecast <br> should be performed. This option is only applicable if the <br> "select = mse" option is used. |
| mser |  |


| kpsssig = key | Set the significance level of the KPSS test when determin- <br> ing the appropriate level of differencing for the dependent <br> variable. key may take values of "1", "5" (default) or "10". |
| :--- | :--- |
| fgraph | Output a forecast comparison graph. |
| atable | Output a selection criteria comparison table |
| agraph | Output a selection criteria comparison graph. |
| etable | Output a final equation output table. Not applicable if the <br> "avg =" option is used. |
| eqname = name | Create an equation object in the workfile with the same <br> specification as the final selected equation. Not applicable <br> if the "avg = " option is used. |
| seed = num | Set the random number generator seed for random starting <br> values. <br> 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 sample.

## Example

The commands

```
wfopen elecdmd.wf1
elecdmd.autoarma(maxsar=1, maxsma=1, noconv, forclen=20, agraph,
    atable, fgraph) elecdmd_f @expand(@month) realgdp tempf
```

open the workfile "elecdmd.WF1" and then perform automatic forecasting on the series ELECDMD. The forecasts will be stored in a series called ELECDMD_F. The ARIMAX model includes exogenous regressors of REALGDP, TEMPF and a set of monthly dummy variables, created with the @expand keyword.

The number of maximum SAR terms and SMA terms are set to 1 (instead of the default 0 ). Model selection is used to determine the best ARMA model, with non-converged models included in the selection process.

The forecast covers 20 periods, and upon completion, EViews will display a graph of the Akaike information criteria of each of the ARMA models considered, as well as a table of each of the selection criteria, and a graph of the each of the forecasts.

## Cross-references

See "Automatic ARIMA Forecasting" on page 536 of User's Guide I for additional discussion.

| boundstest | Equation Views |
| :--- | :--- |

Perform the Pesaran, Shin and Smith (2001) bounds test of long-run relationships from an ARDL estimated equation.

This view displays a spool object with the ARDL bounds test diagnostics. The first table is a summary of the test along with statistic values. The second table summarizes the bound test critical values associated with the $F$-statistic. When appropriate (the deterministic case does not include a restricted constant (cases 3 and 5), a third table summarizes the bound test critical values associated with the $t$-statistic.

## Syntax

eq_name.boundstest(options)

## Options

p Print output.

## Examples

```
wfopen http://www.stern.nyu.edu/~wgreene/Text/Edition7/TableF5-
    2.txt
equation eq02.ardl(deplags=3, reglags=3, fixed) log(realcons)
    log(realgdp) @ @expand(@quarter, @droplast)
show eq02.boundstest
```

This example uses data from Greene (2008, page 685), containing quarterly US macroeconomic variables between 1950 and 2000. The first line of this example downloads the data set, the second line creates an equation object and estimates an ARDL model with the log of real consumption as the dependent variable. Three lags of the dependent variable, and three lags of the log of real GDP are used as dynamic regressors. Quarterly dummy variables are included as static regressors.

The final line performs the Pesaran, Shin and Smith (2001) bounds test to test for a long-run relationship between the log of real consumption and the log of real GDP.

## Cross-references

See "Autoregressive Distributed Lag (ARDL) Models," beginning on page 321 of User's Guide II for further discussion.

| btvcvar | Var Methods |
| :--- | :--- |

Estimate a Bayesian time-varying coefficients VAR, or BTVCVAR, model.

## Syntax

var_name.btvcvar(options) lag_pairs endog_list [@ exog_list]
btvcvar estimates a Bayesian time-varying coefficients VAR. The order of the VAR is specified using a lag pair, followed by a list of series or groups for endogenous variables. Exogenous variables can be included using the @-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, use the noconst option.

## Options

## Prior hyper-parameters

| $\begin{aligned} & \mathrm{T} 0=\text { int } \\ & (\text { default }=0) \end{aligned}$ | Set prior sample size $T_{0}$. A prior sample is not used if T0 is set to 0 . To use a prior sample, T 0 must be set to an integer larger than the number of coefficients per equation in a standard VAR version of the model. |
| :---: | :---: |
| $\begin{aligned} & \text { tau0 }=\text { num } \\ & (\text { default }=5.0) \end{aligned}$ | Set prior scaling parameter for the initial state $b_{0}$. |
| $\begin{aligned} & \text { tau1 = num } \\ & (\text { default }=1.0) \end{aligned}$ | Set prior scaling parameter for the observation covariance $S$. |
| $\begin{aligned} & \text { nu1 }=\text { num } \\ & (\text { default }=5.0) \end{aligned}$ | Set prior dof parameter for the observation covariance $S$. |
| $\begin{aligned} & \text { tau2 }=\text { num } \\ & (\text { default }=0.01) \end{aligned}$ | Set prior scaling parameter for the process covariance $Q$. |
| $\begin{aligned} & \text { nu2 }=\text { num } \\ & (\text { default }=5.0) \end{aligned}$ | Set prior dof parameter for the process covariance $Q$. |

## Display options

usemean
showci
cilevels $=$ arg
(default = "0.95")
uselines

## MCMC options

burn
$($ default $=5000)$
size $=$ int
(default $=5000)$
thin $=$ int
(default $=1$ )
nsub $=$ int
seed $=$ int
rng $=$ arg
(default $=$ "kn" or method set via rndseed)

Use posterior mean as the point estimate. The posterior median is used if usemean is not included in the options list.

Show credibility intervals (bands).
Set credibility levels. For multiple levels, enter a spacedelimited list of values surrounded by quotation marks, e.g., " 0.30 .50 .8 ".

Use lines instead of shading for credibility intervals.

Set burn-in size.

Set posterior sample size.

Set thinning size. A thinning size of $r$ indicates that every $r$-th draw after the burn-in period is stored.

Set the number of subchains.
Set the random seed. EViews will generate a seed if one is not specified.

Set random number generator type. Available types are: 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

| smoother = arg <br> (default = "CFA") | Set simulation smoothing method. Available methods are <br> "CFA" (Cholesky factor algorithm), "KFS" (Kalman filter <br> and smoother), and "MMP" (McCausland, Miller, \& Pel- <br> letier, 2011). |
| :--- | :--- |
| stable | Use the method of Cogley \& Sargent for obtaining stable <br> draws. |
| maxattempts $=$ int <br> $($ default $=100)$ | Set the maximum number of attempts for the sampler to <br> draw stable VAR coefficients for all dates in the data sam- <br> ple. |
| noconst | Do not include a constant in the exogenous regressors list. |

## Examples

To declare and estimate a BTVCVAR named MYVAR with endogenous variables DGP and UNEMP, a constant, the first lag, and a prior sample of size 40, run

```
var myvar.btvcvar(t0=40) 1 1 gdp unemp
```

in the command window. Running the command

```
var myvar.btvcvar(t0=40, showci, cilevels="0.3 0.5") 1 1 gdp unemp
```

will also display shaded $30 \%$ and $50 \%$ credibility bands. For reproducible results, also set the number of subchains ( $n s u b$ ), the random seed (seed), and the random number generator type (rng):

```
var myvar.btvcvar(t0=40, nsub=12, seed=342458900, rng=kn, showci,
    cilevels="0.3 0.5") 1 1 gdp unemp
```

Command capture will always show the nsub, seed, and rng options.

## Cross-references

See Chapter 46. "Bayesian Time-varying Coefficients VAR Models," on page 927 of User's Guide II for details.

| clearcollabels | Matrix Procs |
| :--- | :--- |

Clear the column labels in a matrix object.

## Syntax

matrix_name.clearcollabels
Examples
mat1.clearcollabels
clears the custom column labels from the matrix MAT1.
Cross-references

| Clearcollabels | Rowvector Procs |
| :--- | :--- |

Clear the column labels in a rowvector object.

## Syntax

rowvector_name.clearcollabels

## Examples

rvec1.clearcollabels
clears the custom column labels from the rowvector RVEC1.

## Cross-references

| clearcollabels | Svector Procs |
| :--- | :--- |

Clear the column label in a svector object.

## Syntax

svector_name.clearcollabels
Examples

```
svec1.clearcollabels
```

clears the custom column label from the svector SVEC1.
Cross-references

| Clearcollabels | Vector Procs |
| :--- | :--- |

Clear the column label in a vector object.

## Syntax

vector_name.clearcollabels

## Examples

```
vec1.clearcollabels
```

clears the custom column label from the vector VEC1.

## Cross-references

| Clearrowlabels | Matrix Procs |
| :--- | :--- |

Clear the row labels in a matrix object.

## Syntax

matrix_name.clearrowlabels

## Examples

```
mat1.clearrowlabels
```

clears the custom row labels from the matrix MAT1.

## Cross-references

## clearrowlabels <br> Rowvector Procs

Clear the row label in a rowvector object.

## Syntax

matrix_name.clearrowlabels
Examples
rvec1.clearrowlabels
clears the custom row label from the rowvector RVEC1.
Cross-references

| clearrowlabels | Svector Procs |
| :--- | :--- |

Clear the row labels in a vector object.

## Syntax

svector_name.clearrowlabels
Examples
svec1.clearrowlabels
clears the custom row labels from the svector SVEC1.
Cross-references

| clearrowlabels | Vector Procs |
| :--- | :--- |

Clear the row labels in a vector object.

## Syntax

vector_name.clearrowlabels

## Examples

vec1.clearrowlabels
clears the custom row labels from the vector VEC1.

## Cross-references

| coint | Group Views |
| :--- | :--- |

Perform either (1) Johansen's system cointegration test, (2) Engle-Granger or PhillipsOuliaris single equation cointegration testing, or (3) Pedroni, Kao, or Fisher panel cointegration testing for the series in the group.

There are three forms for the coint command depending on which form of the test you wish to perform.

## Johansen Cointegration Test Syntax

 group_name.coint(options) [lag_spec] [@ x1 x2 x3 ...] [@exogsr sx1 sx2 sx3 ...] [@exoglr lx1 lx2 lx3 ...] [@exogboth bx1 bx2 bx3 ...]uses the coint keyword followed by options, and optionally,

- a lag_spec consisting of one or more pairs of lag intervals, where the lag orders are for the differences in the error correction representation of the VEC, not the levels representation of the VAR.
- an "@"-sign or "@exogsr" followed by a list of exogenous variables in the short-run equation only
- "@exoglr" followed by a list of exogenous variables in the long-run relation only
- "@exogboth" followed by a list of exogenous variables in both the long-run relation and the short-run equations
(This type of cointegration testing may be used in a non-panel workfile except when performing Fisher combined testing using the Johansen framework.)

Note that the output for Johansen cointegration tests displays $p$-values for the rank test statistics. These $p$-values are computed using the response surface coefficients as estimated in MacKinnon, Haug, and Michelis (1999). The 0.05 critical values are also based on the response surface coefficients from MacKinnon-Haug-Michelis. Note: the reported critical values assume no exogenous variables other than an intercept and trend.

## Options for the Johansen Test

## Deterministic Trend Option

There are 8 different deterministic trend assumptions that you may specify using the "determ = arg" option.

These cases correspond to whether the intercept (" c ") and the trend (" t ") are either

- not included ("n")
- in the long-run cointegrating relation only ("l")
- in the short-run equation only ("s")
- in both the long and short-run equations ("b")

The values of arg are text shortcuts formed by joining a text shortcut for the intercept specification with a text shortcut for the trend specification.

The individual intercept and trend specifications are formed by joining the " c " and the " t " with the appropriate letter describing inclusion in the long and short-run equations.

For example,

- "cb" indicates that the constant is in both the long and short-run equation
- "tl" indicates that the trend is in the long-run cointegrating equation only
so that
- "cbtl" indicates that the constant is in both the long and short-run and the trend is in the long-run only

Using this convention (along with a special "none" option), we may easily describe options arguments for all 8 deterministic cases:

| cntn, none | Case 1: No deterministic terms. <br> Corresponding VAR model has no deterministic terms. |
| :--- | :--- |
| cltn | Case 2: Restricted constant. <br> Constant only in the cointegrating relations. <br> Corresponding VAR has a constant. |
| cbtn (default) | Case 3 (JHJ): Unrestricted constant <br> Constant included both in the short-run equation and (arti- <br> ficially) in the cointegrating relations via orthogonalization. <br> Corresponding VAR has a constant and trend. |
| cstn | Case 3: Unrestricted constant <br> Constant only in the short-run equation. <br> Corresponding VAR has a trend. |


| cbtl | Case $4(\mathrm{JHJ}):$ Unrestricted constant and restricted trend <br> Constant included both in the short-run equation and <br> (artificially) in the cointegrating relations via <br> orthogonalization, and trend included only in the cointe- <br> grating relations. <br> Corresponding VAR has a constant and trend. <br> Case 4: Unrestricted constant and restricted trend <br> Constant only in the short-run equation, and trend only in <br> the cointegrating relation. <br> Corresponding VAR has a trend. |
| :--- | :--- |
| cbtb | Case 5 (JHJ): Unrestricted constant and trend <br> Constant and trend both included in the short-run equation <br> and (artificially) in the cointegrating relations via <br> orthogonalization. <br> Corresponding VAR has a constant, linear, and quadratic <br> trend. |
| csts | Case 5: Unrestricted constant and trend <br> Constant and trend both included in the short-run equa- <br> tion. <br> Corresponding VAR has a linear and quadratic trend. |

or you may use the "determsummary" option to compute tests under all deterministic assumptions.

## Other Johansen Options

determsummary Summarize all deterministic trend cases.
restrict Impose restrictions as specified by the "restspec = " option.
restspec $=$ "spec" $\quad$ Define the restricted VEC specification where spec is a space a space delimited list of VEC coefficient restrictions.
$m=$ integer $\quad \quad$ Maximum number of iterations for restricted estimation maxit $=$ integer (only valid if you choose the restrict option).
$\mathrm{c}=$ scalar $\quad$ Convergence criterion for restricted estimation. (only valid
$\mathrm{cvg}=$ scalar $\quad$ if you choose the restrict option).

| save = mat_name | Stores test statistics as a named matrix object. The save $=$ option stores a $(k+1) \times 4$ matrix, where $k$ is the number of endogenous variables in the VAR. The first column contains the eigenvalues, the second column contains the maximum eigenvalue statistics, the third column contains the trace statistics, and the fourth column contains the log likelihood values. The $i$-th row of columns 2 and 3 are the test statistics for rank $i-1$. The last row is filled with NAs, except the last column which contains the log likelihood value of the unrestricted (full rank) model. |
| :---: | :---: |
| cvtype $=$ ol | Display 0.05 and 0.01 critical values from OsterwaldLenum (1992). <br> This option reproduces the output from version 4 . The default is to display critical values based on the response surface coefficients from MacKinnon-Haug-Michelis (1999). Note that the argument on the right side of the equals sign are letters, not numbers 0-1). |
| $\begin{aligned} & \text { cvsize }=\text { arg } \\ & \text { (default }=0.05 \text { ) } \end{aligned}$ | Specify the size of MacKinnon-Haug-Michelis (1999) critical values to be displayed. The size must be between 0.0001 and 0.9999 ; values outside this range will be reset to the default value of 0.05 . This option is ignored if you set "cvtype = ol". |
| prompt | Force the dialog to appear from within a program. |
|  | Print results. |

## Single Equation Test Syntax

group_name.coint(method = arg, options) [@determ determ_spec] [@regdeterm regdeterm_spec]
where
method = arg Test method: Engle-Granger residual test ("eg"), PhillipsOuliaris residual test ("po").

Cointegrating equation specifications that include a constant, linear, or quadratic trends, should use the "trend = " option to specify those terms. If any of those terms are in the stochastic regressors equations but not in the cointegrating equation, they should be specified using the "regtrend = " option.

Deterministic trend regressors that are not covered by the list above may be specified using the keywords @determ and @regdeterm. To specify deterministic trend regressors that enter into the regressor and cointegrating equations, you should add the keyword @determ followed by the list of trend regressors. To specify deterministic trends that enter in the regres-
sor equations but not the cointegrating equation, you should include the keyword @regdeterm followed by the list of trend regressors.

Note that the $p$-values for the test statistics are based on simulations, and do not account for any user-specified deterministic regressors.

This type of cointegration testing may be used in a non-panel workfile. The remaining options for the single equation cointegration tests are outlined below.

## Options for Single Equation Tests

## Options for the Engle-Granger Test

The following options determine the specification of the Engle-Granger test (Augmented Dickey-Fuller) equation and the calculation of the variances used in the test statistic.

| $\begin{aligned} & \text { trend = arg } \\ & \text { (default ="const") } \end{aligned}$ | Specification for the powers of trend to include in the cointegrating equation: None ("none"), Constant ("const"), Linear trend ("linear"), Quadratic trend ("quadratic"). |
| :---: | :---: |
|  | Note that the specification implies all trends up to the specified order so that choosing a quadratic trend instructs EViews to include a constant and a linear trend term along with the quadratic. |
| $\begin{aligned} & \text { regtrend = arg } \\ & \text { (default = "none") } \end{aligned}$ | Additional trends to include in the regressor equations (but not the cointegrating equation): None ("none"), Constant ("const"), Linear trend ("linear"), Quadratic trend ("quadratic"). Only trend orders higher than those specified by "trend $=$ " will be considered. |
|  | Note that the specification implies all trends up to the specified order so that choosing a quadratic trend instructs EViews to include a constant and a linear trend term along with the quadratic. |
| $\begin{aligned} & \text { lag }=\arg \\ & (\text { default }=" \mathrm{a} ") \end{aligned}$ | Method of selecting the lag length (number of first difference terms) to be included in the regression: "a" (automatic information criterion based selection), or integer (user-specified lag length). |
| $\begin{aligned} & \text { lagtype = arg } \\ & (\text { default }=\text { "sic") } \end{aligned}$ | Information criterion or method to use when computing automatic lag length selection: "aic" (Akaike), "sic" (Schwarz), "hqc" (Hannan-Quinn), "msaic" (Modified Akaike), "msic" (Modified Schwarz), "mhqc" (Modified Hannan-Quinn), "tstat" ( $t$-statistic). |
| maxlag $=$ integer | Maximum lag length to consider when performing automatic lag-length selection $\text { default }=\operatorname{int}\left(\min ((T-k) / 3,12) \cdot(T / 100)^{1 / 4}\right)$ <br> where $k$ is the number of coefficients in the cointegrating equation. Applicable when "lag = a ". |


| lagpval = number <br> $($ default $=0.10)$ | Probability threshold to use when performing automatic <br> lag-length selection using a $t$-test criterion. Applicable <br> when both "lag = a" and "lagtype = tstat". |
| :--- | :--- |
| nodf | Do not degree-of-freedom correct estimates of the vari- <br> ances. |
| prompt | Force the dialog to appear from within a program. <br> p |

## Options for the Phillips-Ouliaris Test

The following options control the computation of the symmetric and one-sided long-run variances in the Phillips-Ouliaris test.

## Basic Options

| trend =arg |
| :--- |
| (default = "const") |


| Specification for the powers of trend to include in the |
| :--- |
| cointegrating equation: None ("none"), Constant ("const"), |
| Linear trend ("linear"), Quadratic trend ("quadratic"). |
| Note that the specification implies all trends up to the spec- |
| ified order so that choosing a quadratic trend instructs |
| EViews to include a constant and a linear trend term along |
| with the quadratic. |

(default = "none") | Additional trends to include in the regressor equations (but |
| :--- |
| not the cointegrating equation): None ("none"), Constant |
| ("const"), Linear trend ("linear"), Quadratic trend ("qua- |
| dratic"). Only trend orders higher than those specified by |
| "trend=" will be considered. |

HAC Whitening Options

```
lag= arg (default = 0) Lag specification: integer (user-specified lag value), "a"
    (automatic selection).
infosel = arg Information criterion for automatic selection: "aic"
(default = "aic") (Akaike), "sic" (Schwarz), "hqc" (Hannan-Quinn) (if
    "lag=a").
```

$$
\begin{array}{ll}
\text { maxlag }=\text { integer } & \text { Maximum lag-length for automatic selection (optional) (if } \\
\text { "lag=a"). The default is an observation-based maximum. }
\end{array}
$$

## HAC Kernel Options

| $\begin{aligned} & \text { kern = arg } \\ & \text { (default = "bart") } \end{aligned}$ | Kernel shape: "none" (no kernel), "bart" (Bartlett, default), "bohman" (Bohman), "daniell" (Daniel), "parzen" (Parzen), "parzriesz" (Parzen-Riesz), "parzgeo" (ParzenGeometric), "parzcauchy" (Parzen-Cauchy), "quadspec" (Quadratic Spectral), "trunc" (Truncated), "thamm" (Tukey-Hamming), "thann" (Tukey-Hanning), "tparz" (Tukey-Parzen). |
| :---: | :---: |
| $\begin{aligned} & \text { bw = arg } \\ & \text { (default = "nwfixed") } \end{aligned}$ | Bandwidth: "fixednw" (Newey-West fixed), "andrews" (Andrews automatic), "neweywest" (Newey-West automatic), number (User-specified bandwidth). |
| nwlag $=$ integer | Newey-West lag-selection parameter for use in nonparametric bandwidth selection (if "bw = neweywest"). |
| $\begin{aligned} & \text { bwoffset }=\text { integer } \\ & (\text { default }=0) \end{aligned}$ | Apply integer offset to bandwidth chosen by automatic selection method ("bw = andrews" or "bw = neweywest"). |
| bwint | Use integer portion of bandwidth chosen by automatic selection method ("bw = andrews" or "bw = neweywest"). |

## Panel Test Syntax

```
group_name.coint(option)
```

The coint command tests for cointegration among the series in the group. This form of the command should be used with panel structured workfiles.

## Options for the Panel Tests

For panel cointegration tests, you may specify the type using one of the following keywords:
Pedroni (default) Pedroni (1994 and 2004).
Kao Kao (1999)
Fisher Fisher - pooled Johansen
Depending on the type selected above, the following may be used to indicate deterministic trends:

$$
\begin{array}{ll}
\text { const (default) } & \text { Include a constant in the test equation. } \\
& \text { Applicable to Pedroni and Kao tests. }
\end{array}
$$

\(\left.$$
\begin{array}{ll}\text { trend } & \begin{array}{l}\text { Include a constant and a linear time trend in the test equa- } \\
\text { tion. }\end{array}
$$ <br>

Applicable to Pedroni tests.\end{array}\right]\)| Do not include a constant or time trend. |
| :--- |
| none |
| Applicable to Pedroni tests. |

## Additional Options:

| $\begin{aligned} & \text { hac = arg } \\ & \text { (default ="bt") } \end{aligned}$ | Method of estimating the frequency zero spectrum: "bt" (Bartlett kernel), "pr" (Parzen kernel), "qs" (Quadratic Spectral kernel). <br> Applicable to Pedroni and Kao tests. |
| :---: | :---: |
| $\begin{aligned} & \mathrm{bw}=\text { arg } \\ & \text { (default }=\text { "nw") } \end{aligned}$ | Method of selecting the bandwidth, where arg may be "nw" (Newey-West automatic variable bandwidth selection), or a number indicating a user-specified common bandwidth. <br> Applicable to Pedroni and Kao tests. |
| $\operatorname{lag}=\arg$ | For Pedroni and Kao tests, the method of selecting lag length (number of first difference terms) to be included in the residual regression. For Fisher tests, a pair of numbers indicating lag. |
| $\begin{aligned} & \text { infosel = arg } \\ & \text { (default = "sic") } \end{aligned}$ | Information criterion to use when computing automatic lag length selection: "aic" (Akaike), "sic" (Schwarz), "hqc" (Hannan-Quinn). <br> Applicable to Pedroni and Kao tests. |
| maxlag $=$ int | Maximum lag length to consider when performing automatic lag length selection, where int is an integer. The default is $\operatorname{int}\left(\min \left(T_{i} / 3,12\right) \cdot\left(T_{i} / 100\right)^{1 / 4}\right)$ <br> where $T_{i}$ is the length of the cross-section. Applicable to Pedroni and Kao tests. |
| $\begin{aligned} & \text { disp }=\text { arg } \\ & (\text { default }=500) \end{aligned}$ | Maximum number of individual results to be displayed. |
| prompt | Force the dialog to appear from within a program. |
| p | Print results. |

## Examples

## Johansen Test

```
gr1.coint(determsummary) 1 4
```

summarizes the results of the Johansen cointegration test for the series in the group GR1 for all specifications of trend. The test equation uses lags of up to order four.

## Engle-Granger Test

```
gr1.coint(method=eg)
```

performs the default Engle-Granger test on the residuals from a cointegrating equation which includes a constant. The number of lags is determined using the SIC criterion and an observation-based maximum number of lags.

```
gr1.coint(method=eg, trend=linear, lag=a, lagtype=tstat,
    lagpval=.15, maxlag=10)
```

employs a cointegrating equation that includes a constant and linear trend, and uses a sequential $t$-test starting at lag 10 with threshold probability 0.15 to determine the number of lags.

```
gr1.coint(method=eg, lag=5)
```

conducts an Engle-Granger cointegration test on the residuals from a cointegrating equation with a constant, using a fixed lag of 5 .

## Phillips-Ouliaris Test

```
gr1.coint(method=po)
```

performs the default Phillips-Ouliaris test on the residuals from a cointegrating equation with a constant, using a Bartlett kernel and Newey-West fixed bandwidth.

```
gr1.coint(method=po, bw=andrews, kernel=quadspec, nodf)
```

estimates the long-run covariances using a Quadratic Spectral kernel, Andrews automatic bandwidth, and no degrees-of-freedom correction.

```
gr1.coint(method=po, trend=linear, lag=1, bw=4)
```

estimates a cointegrating equation with a constant and linear trend, and performs the Phil-lips-Ouliaris test on the residuals by computing the long-run covariances using AR(1) prewhitening, a fixed bandwidth of 4, and the Bartlett kernel.

## Panel Tests

For a panel structured workfile,

```
grp1.coint(pedroni,maxlag=3,infosel=sic)
```

performs Pedroni's residual-based panel cointegration test with automatic lag selection with a maximum lag limit of 3 . Automatic selection based on Schwarz criterion.

## Cross-references

See Chapter 55. "Cointegration Testing," on page 1245 of User's Guide II for details on the various cointegration tests. See also Equation::coint (p. 71).

| coint | Var Views |
| :--- | :--- |

Johansen's cointegration test for the series in the var object.

## Syntax

var_name.coint(options) [lag_spec] [@ x1 x2 x3 ...] [@exogsr sx1 sx2 sx3 ...] [@exoglr lx1 lx2 lx3 ...] [@exogboth bx1 bx2 bx3 ...]
uses the coint keyword followed by options, and optionally,

- a lag_spec consisting of one or more pairs of lag intervals, where the lag orders are for the differences in the error correction representation of the VEC, not the levels representation of the VAR.
- an "@"-sign or " @exogsr" followed by a list of exogenous variables in the short-run equation only
- "@exoglr" followed by a list of exogenous variables in the long-run relation only
- "@exogboth" followed by a list of exogenous variables in both the long-run relation and the short-run equations

The coint command tests for cointegration among the series in the var object using the lag spec, exogenous variables, and if relevant, deterministic spec and VEC restrictions specified in estimation.

- You may provide explicit lag_spec to override the one used in estimation.

Note that if the estimation lags were for a VAR specification in levels, the default lag_spec will be the original spec adjusted for the error correction differences. Thus, if the original estimation was for a " 14 " VAR, the default lag_spec will be " 13 ".

- You may provide a "determ = " option to override an existing VEC deterministic trend specification.
- You may explicitly list any type of exogenous variable to override the entire existing specification for the exogenous variables.

The output for cointegration tests displays $p$-values for the rank test statistics. These $p$-values are computed using the response surface coefficients as estimated in MacKinnon, Haug, and Michelis (1999). The 0.05 critical values are also based on the response surface coefficients from MacKinnon-Haug-Michelis. Note: the reported critical values assume no exogenous variables other than an intercept and trend.

## Options

## Deterministic Trend Option

There are 8 different deterministic trend assumptions that you may specify using the "determ = arg" option.

These cases correspond to whether the intercept (" c ") and the trend (" t ") are either

- not included ("n")
- in the long-run cointegrating relation only ("l")
- in the short-run equation only ("s")
- in both the long and short-run equations ("b")

The values of $a r g$ are text shortcuts formed by joining a text shortcut for the intercept specification with a text shortcut for the trend specification.

The individual intercept and trend specifications are formed by joining the " c " and the " t " with the appropriate letter describing inclusion in the long and short-run equations.

For example,

- "cb" indicates that the constant is in both the long and short-run equation
- "tl" indicates that the trend is in the long-run cointegrating equation only
so that
- "cbtl" indicates that the constant is in both the long and short-run and the trend is in the long-run only

Using this convention (along with a special "none" option), we may easily describe options arguments for all 8 deterministic cases:

| cntn, none | Case 1: No deterministic terms. <br> Corresponding VAR model has no deterministic terms. |
| :--- | :--- |
| cltn | Case 2: Restricted constant. <br> Constant only in the cointegrating relations. <br> Corresponding VAR has a constant. |
| cbtn (default) | Case 3 (JHJ): Unrestricted constant <br> Constant included both in the short-run equation and (arti- <br> ficially) in the cointegrating relations via orthogonalization. <br> Corresponding VAR has a constant and trend. |
| cstn | Case 3: Unrestricted constant <br> Constant only in the short-run equation. <br> Corresponding VAR has a trend. |


| cbtl | Case 4 (JHJ): Unrestricted constant and restricted trend <br> Constant included both in the short-run equation and <br> (artificially) in the cointegrating relations via <br> orthogonalization, and trend included only in the cointe- <br> grating relations. <br> Corresponding VAR has a constant and trend. <br> Case 4: Unrestricted constant and restricted trend <br> Constant only in the short-run equation, and trend only in <br> the cointegrating relation. <br> Corresponding VAR has a trend. <br> cbtl <br> Case 5 (JHJ): Unrestricted constant and trend <br> Constant and trend both included in the short-run equation <br> and (artificially) in the cointegrating relations via <br> orthogonalization. <br> Corresponding VAR has a constant, linear, and quadratic <br> trend. |
| :--- | :--- |
| csts | Case 5: Unrestricted constant and trend <br> Constant and trend both included in the short-run equa- <br> tion. <br> Corresponding VAR has a linear and quadratic trend. |

or you may use the "determsummary" option to compute tests under all deterministic assumptions.

## Other Options

| determsummary | Summarize all deterministic trend cases. |
| :--- | :--- |
| restrict | Impose restrictions as specified by the Var : : append <br> (p. 995) proc, or the "restspec = " option. |
| restspec = "spec" | Define the restricted VEC specification where spec is a <br> space a space delimited list of VEC coefficient restrictions. |
| $\mathrm{m}=$ integer, | Maximum number of iterations for restricted estimation <br> maxit $=$ integer <br> (only valid if you choose the restrict option). |
| $\mathrm{c}=$ scalar, | Convergence criterion for restricted estimation. (only valid <br> $\mathrm{cvg}=$ scalar |
| if you choose the restrict option). |  |

\(\left.$$
\begin{array}{ll}\text { save }=\text { mat_name } & \begin{array}{l}\text { Stores test statistics as a named matrix object. The save }= \\
\\
\text { option stores a }(k+1) \times 4 \text { matrix, where } k \text { is the num- } \\
\\
\text { ber of endogenous variables in the VAR. The first column } \\
\text { contains the eigenvalues, the second column contains the }\end{array}
$$ <br>
\& maximum eigenvalue statistics, the third column contains <br>
the trace statistics, and the fourth column contains the log <br>
likelihood values. The i -th row of columns 2 and 3 are the <br>

test statistics for rank i-1 . The last row is filled with\end{array}\right\}\)| NAs, except the last column which contains the log likeli- |
| :--- |
| hood value of the unrestricted (full rank) model. |

## Examples

```
var1.coint(determ=cbtl) 1 12 @
```

carries out the Johansen test for the series in the var object named VAR1 using lags 1 to 12 . The " @"-sign without a list of exogenous variables ensures that the test does not include any exogenous variables in VAR1.

## Cross-references

See "Johansen Cointegration Test" on page 1245 of User's Guide II for details on the Johansen test.

See also : :ec (p. 151).

| Cointrel | Equation Views |
| :--- | :--- |

Display information about the cointegrating relation specification and the coefficients in an ARDL estimated equation.

## Syntax

eq_name.cointrel(options)
Options
p Print output.

## Example

```
ardl_eq.cointrel
```

displays a spool object with the table and graph showing the cointegrating relation.
Cross-references

| datelabel | Graph Procs |
| :--- | :--- |

Control labeling of the bottom date/time axis in time plots.
datelabel sets options that are specific to the appearance of time/date labeling. Many of the options that also affect the appearance of the date axis are set by the Graph: :axis (p. 317) command with the "bottom" option. These options include tick control, label and font options, and grid lines.

## Syntax

graph_name.datelabel option_list

## Options

format("datestring")
interval(step_size
[,steps][,align_date])
span(arg)
end / -end
duallevel / -duallevel
datestring should be one of the supported data formats describing how the date should appear. The datestring argument should be enclosed in double-quotes. For example, "yy:mm" specifies two-digit years followed by a colon delimited and then two-digit months.
You may use the special single space datestring " " to indicate automatic formatting.
You may also add " $\backslash \mathrm{n}$ " to denote a new line providing the option to make the date string 2 lines. For example, "Month $\backslash$ nyear" will place the month on the first line and the year on the second. Note: there is a 2 line maximum. A second " $\backslash \mathrm{n}$ " will therefore create an error.
EViews provides considerable flexibility in formatting your dates. See "Date Formats" on page 97 of the Command and Programming Reference for a complete description.
where step_size takes one of the following values: "auto" (steps and align_date are ignored), "ends" (only label endpoints; steps and align_date are ignored), "all" (label every point; the steps and align_date options are ignored), "obs" (steps are one observation), "year" (steps are one year), " $m$ " (steps are one month), " q " (steps are one quarter).
steps is a number (default $=1$ ) indicating the number of steps between labels.
align_date is a date specified to receive a label.
Note, the align_date should be in the units of the data being graphed, but may lie outside the current sample or workfile range.

Specify date label spanning: "auto" (automatic determination), "on" (turn spanning on; label start of period, tick on obs.), "between" (center label on period), "trimbetween" (center label on period, trim spaces at axis ends).
Consider the case of a yearly label with monthly ticks. If span is on, the label is centered on the 12 monthly ticks. If the span option is off, year labels are put on the first quarter or month of the year.
[Use / Do not use] end-of-period labeling.
[Allow / Do not allow] two row date labels on the observation axis.

## Examples

will display dates using four-digit years followed by the default delimiter ":" and a two-digit month (e.g. - "1974:04").

```
graph1.datelabel format(yy[q]mm)
```

will display a two-digit year followed by a " q " separator and then a two-digit month (e.g. "74q04")

```
graph1.datelabel interval(y, 2, 1951)
```

specifies labels every two years on odd numbered years.

```
graph1.datelabel format("Month dd\nYYYY")
```

specifies time axis label will have 2 lines. The first line will contain the full month name and day and the second line will contain the 4 digit year.

## Cross-references

See Chapter 16. "Graph Objects," on page 835 of User's Guide I for a discussion of graph options.

See also Graph: : axis (p. 317), Graph: :options (p. 340), and : :setelem (p. 240).

## dbopen

Object Container, Data, and File Commands

Open an existing database.

## Syntax

dbopen(options) [path\Jdb_name [as shorthand_name]
Follow the dbopen keyword with the name of a database. You should include a path name to open a database not in the default path. The opened database will become the default database.

You do not need to specify a database name when opening a Datastream or FRED connection ("type = datastream" or "type = fred") as EViews will automatically connect to the proper location.

You may use the "as" keyword to provide an optional shorthand_name or a short text label which is used to refer to the open database in commands and programs. If you leave this field blank, a default shorthand_name will be assigned automatically.

See "Database Shorthands" on page 333 of User's Guide I for additional discussion.
By default, EViews will use the extension of the database file to determine type. For example, files with the extension ".EDB" will be opened as an EViews database, while files with the extension ".IN7" will be opened as a GiveWin database. You may use the "type = " option to specify an explicit type.

## Options

$$
\text { type }=\arg , \mathrm{t}=\arg \quad \text { Specify the database type: (see table below) } .
$$

The following table summaries the various database formats, along with the corresponding "type = " keywords:

|  | Option "type = " keywords | Notes |
| :---: | :---: | :---: |
| Australian Bureau of Statistics SDMX | "abs" | (b) |
| AREMOS Bank | "aremos |  |
| AREMOS TSD | "a", "tsd" |  |
| Bloomberg | "bloom" | (a), (b) |
| Bureau of Economic Analysis | "bea" | (b) |
| Bureau of Labor Statistics | "bls" | (b) |
| CEIC | "ceic" | (a), (b) |
| Datastream | "datastream" | (a), (b) |
| DBnomics | "dbnomics" | (b) |
| Deutsche Bundesbank SDMX | "bbk" | (b) |
| DRIPro Link | "dripro" | (b) |
| DRI DDS | "dds" |  |
| ECB (European Central Bank) | ecb" | (b) |
| EIA Bulk File | "eiabulk" | (a), (c) |
| EIA (U.S. Energy Information Administration) | "eia" | (a), (b) |
| Eurostat SDMX | "eurostat" | (b) |
| EViews | "e", "eviews" |  |
| FAME | "f", "fame" | (a) |
| FRED | "fred" | (b) |
| FRED v1 | "fredv1" | (b) |
| GiveWin/PcGive | "g", "give" |  |
| Haver | "h", "haver" | (a) |
| IHS Global Insight | "ihs global insight" | (a), (b) |
| IHS Magellan | "magellan" | (a), (b) |
| IHSMarkit API | "ihsmarkit" | (a), (b) |
| IMF (International Monetary Fund) SDMX | "imf" | (b) |


| INSEE (National Institute of Statis- <br> tics and Economic Studies) SDMX | "insee" | (b) |
| :--- | :---: | :---: |
| Moody's Economy.com | "economy" | (a), (b) |
| NOAA (National Oceanic And <br> Atmospheric Administration) | "noaa" | (b) |
| OECD (Organization for Economic <br> Cooperation and Development) <br> SDMX | "oecd" | (b) |
| RATS 4.x | "r", "rats" |  |
| RATS Portable / TROLL | "l", "trl" |  |
| SDMX_ML | "sdmx" | (c) |
| Trading Economics | "tradingeconomics" | (b) |
| TSP Portable | "t", "tsp" |  |
| UN (United Nations) | "uscensus" | (b) |
| US Census | "who" | (b) |
| WHO (World Health Organization) | "worldbank" | (b) |
| World Bank | (b) |  |

- (a) You must have EViews Enterprise Edition to access these databases.
- (b) You must have an active connection to the internet to access these databases.
- (c) You must have internet access to download these file databases prior to opening them with EViews.

In addition, specific types may require installation of additional software. For details see, "Notes on Particular Formats" on page 360 in User's Guide I.

The following options may be required when connecting to a remote server:

```
s = server_id, Server name
server = server_id
u = user, Username
username = user
p = pwd, Password
password = pwd
```


## Examples

```
dbopen c:\data\us1
```

opens a database named US1 in the C: $\backslash$ DATA directory. The command:
dbopen us1
opens a database in the default path. If the specified database does not exist, EViews will issue an error message. You should use db (p. 345) or dbcreate (p. 348) to create a new database.

## Cross-references

See Chapter 10. "EViews Databases," on page 329 of User's Guide I for a discussion of EViews databases.

See also db (p. 345) and dbcreate (p. 348).

| deleteaddin | Programming Commands |
| :--- | :--- |

## Unregister a program file as an EViews Add-in.

## Syntax

deleteaddin(options) [path \]prog_name

unregisters the specified program file as an EViews Add-in.
If you do not provide the optional path specification, EViews looks for the program file in the default EViews Add-ins directory.

Explicit path specifications containing ". $\$ " and "..\" (to indicate the current level and one directory level up) are evaluated relative EViews default directory.

You may use the special " <addins > " directory keyword in your path specification.

## Options

| type $=$ arg | Specify the Add-ins type, where arg is the name of a <br> EViews object type. The default is to create a global Add- <br> in. |
| :--- | :--- |
| proc $=\arg$ | User-defined command/procedure name. If omitted, the <br> Add-in will not have a command form. |

## Examples

```
deleteaddin .\myaddin.prg
```

unregisters the Add-in associated with file "Myaddin.prg".
Alternatively,

```
deleteaddin(proc="myaddin")
```

unregisters the Add-in whose proc name matches "myaddin". Note that this name may not match the program name.

```
deleteaddin(type="graph", proc="recshade")
```

unregisters the graph "Recshade" specific Add-in. In cases, where more than 1 Add-in has the same proc name, the type is useful to differentiate which is to be unregistered.

## Cross-references

See Chapter 8. "Add-ins," on page 191 for a detailed discussion of Add-ins.

| did | Equation Methods |
| :--- | :--- |

Estimate a panel equation using the difference-in-difference estimator. This estimation method is only available for equations estimated in panel workfiles.

## Syntax

equation.did(options) $y[x 1]$ [@ treatment]
List the dependent variable, followed by an optional list of exogenous regressors, followed by an "@" and then the binary treatment variable. You should not include a constant in the specification.

## Options

| coef $=\arg$ | Specify the name of the coefficient vector. The default <br> behavior is to use the "C" coefficient vector. |
| :--- | :--- |
| prompt | Force the dialog to appear from within a program. |
| p | Print results. |

## Examples

```
equation eq1.did asmrs @ post
```

estimates an equation by difference-in-difference with ASMRS as the outcome variable, and POST as the treatment variable.

```
equation eq2.did lemp lpop @ treated
```

estimates an equation by difference-in-difference with LEMP as the outcome variable, TREATED as the treatment variable, and LPOP as an exogenous regressor.

## Cross-references

| didcs | Equation Views |
| :--- | :--- |

Display the Callaway-Sant'Anna decomposition for difference-in-difference estimation.
For panel equations estimated using the difference-in-difference method.

## Syntax

```
eq_name.didcs(options) [additional_regs]
```

You should follow the didcs keyword by an optional list of additional regressors added to the Callaway-Sant'Anna estimation.

## Options

| notyet | Use observations where an individual is not yet <br> treated as the comparison group. Default is to only <br> use individuals that are never treated as the com- <br> parison. |
| :--- | :--- |
| both | Use both observations where an individual is never <br> treated or has not yet been treated as the compari- <br> son group. Default is to only use individuals that are <br> never treated as the comparison. |
| p | Print output. |

## Example

```
equation eq1.did lemp @ treated
eq1.didcs lpop
```

estimates an equation by difference-in-difference with LEMP as the outcome variable, and TREATED as the treatment variable, and then displays the Callaway-Sant'Anna decomposition, adding LPOP as an additional exogenous regressor.

## Cross-references

| didgbdecomp | Equation Views |
| :--- | :--- |

Display the Goodman-Bacon decomposition for difference-in-difference estimation.
For panel equations estimated using the difference-in-difference method.

## Syntax

eq_name.didgbdecomp(options)

## Options

p Print output.

## Example

```
equation eq1.did asmrs @ post
eq1.didgbdecomp
```

estimates an equation by difference-in-difference with ASMRS as the outcome variable, and POST as the treatment variable, and then displays the Goodman-Bacon decomposition.

Cross-references

## didmakeeq

Equation Procs

Create an equation object with the underlying fixed-effects estimation of a difference-in-difference equation.

For panel equations estimated using the difference-in-difference method.

## Syntax

eq_name.didmakeeq new_eqname
You should follow the didmakeeq keyword with the name of the new estimated equation with an equivalent specification to be created in the workfile.

## Example

```
equation eq1.did asmrs @ post
eq1.didmakeeq eq_underlying
```

estimates an equation by difference-in-difference with ASMRS as the outcome variable, and POST as the treatment variable, and then creates the underlying fixed effects estimation in the equation object EQ_UNDERLYING

## Cross-references

## didtrends

## Equation Views

Display difference-in-difference trends summary in graph or tabular form.
Syntax
eq_name.didtrends(options)
For panel equations estimated using the difference-in-difference method.

## Options

t Display results in a table.
p Print output.

## Example

```
equation eq1.did asmrs @ post
eq1.didtrends
eq1.didtrends(t)
```

estimates an equation by difference-in-difference with ASMRS as the outcome variable, and POST as the treatment variable, and then displays the trend summary graph, then as a table.

Cross-references

| dsa | Series Procs |
| :--- | :--- |

Seasonally adjust daily series using the DSA method.

## Syntax

series_name.dsa(options) seas_name [@fa factor_name] [@trnd trend_name]
You may follow the dsa keyword with a name to save the seasonally adjusted series. Further, you may use the @fa and @trnd keywords to provide names for the saved seasonal factors and the trend series.

## Options

| forcend = arg | Specify the end date of the forecast. If not specified, the <br> last observation in the workfile is used. The forecast begins <br> at the observation following the current workfile sample <br> (note, if the workfile sample is equal to the workfile range, <br> no forecasting is performed). |
| :--- | :--- |
| extendfri | For 5-day week data, interpolate to 7-day weeks by repeat- <br> ing the Friday value for Saturday and Sunday. Default is to <br> perform 5-day DSA instead of converting to 7-day. |
| interwkend | For 5-day week data, interpolate to 7-day weeks by using <br> linear interpolation between the Friday value and Monday <br> values for Saturday and Sunday. Default is to perform 5- <br> day DSA instead of converting to 7-day. |
| fixedarima | Use a fixed ARIMA model. Default is to use model selection <br> to determine the ARIMA model. |
| nodiff | Set the level of differencing in the ARIMA model to 0. <br> Default is 1 if using a fixed ARIMA model, or a choice <br> between 0 and 1 if using automatic selection. |
| maxar = integer | If using fixed ARIMA model (see the fixedarima option), <br> specify the AR order. If using automatic selection, specify <br> the maximum AR order. |
| maxma = integer | If using fixed ARIMA model (see the fixedarima option), <br> specify the MA order. If using automatic selection, specify |
| the maximum MA order. |  |

\(\left.\left.$$
\begin{array}{ll}\begin{array}{l}\text { olcvalue }=\text { arg } \\
\text { oldelta }=\text { arg }\end{array} & \begin{array}{l}\text { Specify the critical value for the outlier detection process. } \\
\text { olinits = integer }\end{array} \\
\begin{array}{l}\text { Specify the delta value for the TC outlier detection process. }\end{array} \\
\text { oloutits = integer } & \begin{array}{l}\text { Specify number of inner iterations in the outlier detection } \\
\text { process. }\end{array} \\
\text { Specify number of outer iterations in the outlier detection } \\
\text { process. }\end{array}
$$\right\} \begin{array}{l}When forecasting day-of-week factors, repeat the last week <br>
of actual data throughout the forecast period. Default is to <br>

use exponential smoothing to forecast the factors.\end{array}\right\}\)| Force the dialog to appear from within a program. |
| :--- |
| prompt |

## STL options

## Day-of-week

| weeksp $=$ integer | Specify the seasonal polynomial degree. Default is 0 . |
| :---: | :---: |
| weektp $=$ integer | Specify the trend polynomial degree. Default is 1 . |
| weekfp $=$ integer | Specify the filter polynomial degree. Default is 1. |
| weeksl $=$ integer | Specify the length of the seasonal smoothing window (odd integers only). Default is 151 . |
| weektl $=$ integer | Specify the length of the trend smoothing window (odd integers only). Default is based upon the seasonal smoothing window length. |
| weekfl $=$ integer | Specify the length of the filter smoothing window (odd integers only). Default is 1 . |
| weekinits $=$ integer | Specify number of inner iterations. Default is 1. |
| weekoutits $=$ integer | Specify the number of outer iterations. Default is 15 . |
| ay-of-month |  |
| monthsp $=$ integer | Specify the seasonal polynomial degree. Default is 0 . |
| monthtp $=$ integer | Specify the trend polynomial degree. Default is 1 . |
| monthfp $=$ integer | Specify the filter polynomial degree. Default is 1 . |
| monthsl $=$ integer | Specify the length of the seasonal smoothing window (odd integers only). Default is 51 . |
| monthtl $=$ integer | Specify the length of the trend smoothing window (odd integers only). Default is based upon the seasonal smoothing window length. |

```
monthfl = integer Specify the length of the filter smoothing window (odd
    integers only). Default is 1.
monthinits = integer Specify number of inner iterations. Default is 1.
monthoutits = integer Specify the number of outer iterations. Default is 15.
```


## Day-of-year

| yearsp $=$ integer | Specify the seasonal polynomial degree. Default is 0. <br> yeartp $=$ integer |
| :--- | :--- |
| yearfp $=$ integer | Specify the trend polynomial degree. Default is 1. <br> yearsl $=$ integer |
| Specify the filter polynomial degree. Default is 1. |  |
| Seartl $=$ integer | Specify the length of the seasonal smoothing window (odd <br> integers only). Default is 13. |
| Specify the length of the trend smoothing window (odd <br> integers only). Default is based upon the seasonal smooth- <br> ing window length. |  |
| yearfl $=$ integer | Specify the length of the filter smoothing window (odd <br> integers only). Default is 1. |
| yearinits $=$ integer | Specify number of inner iterations. Default is 1. |
| yearoutits $=$ integer | Specify the number of outer iterations. Default is 15. |

## Example

```
elecdmd.dsa(forcend=2015/6/30) elecdmd_adjusted
```

Performs daily seasonal adjustment on the ELECDMD series, specifying that the forecast end point should be 30 June 2015, and that the final adjusted series should be named ELECDMD_ADJUSTED.

```
elecdmd.dsa(fixedtrig, nodom, nodoy) elecdmd_adjusted @fa
    elecdmd_factors
```

Performs daily seasonal adjustment on ELECDMD, using a fixed number of trigonometric terms in the ARIMA step, and without using day-of-month or day-of-year STL. As well as saving the final adjusted series as ELECDMD_ADJUSTED, the final seasonal factor are also saved under ELECDMD_FACTORS.

## Cross-references

## dynmult

Dynamic multipliers for long-run regressors in ARDL equations.
Displays a spool object with the cumulative dynamic multiplier curve for each of the longrun regressors. The argument is a positive integer denoting the horizon length, and defaults to 15 .

## Syntax

eq_name.dynmult(options) [horizon]
horizon is a positive integer denoting the horizon length, and defaults to 15 .

## Options

| noci | Do not generate confidence intervals for asymmetric <br> regressors. Note that confidence intervals can only be <br> generated for asymmetric regressors. |
| :--- | :--- |
| noshade | Display confidence interval using lines instead of <br> shaded bands. |
| level = number <br> $($ default $=0.95)$ | Number between 0 and 1 representing the confidence <br> interval level. |
| reps $=$ integer <br> $($ default $=999)$ | Number of Monte Carlo repetitions used in the genera- <br> tion of confidence intervals (if applicable). |
| $\mathrm{f}=$ number | Fraction of failed repetitions before stopping. Only <br> applicable if a se_pattern is provided. |
| prompt | Force the dialog to appear from within a program. |
| p | Print output. |

## Example

```
ardl_eq.dynmult
```

generates cumulative dynamic multiplier curves for each long-run regressor. The horizon length is 15 , and the $95 \%$ confidence intervals (if they exist), are shaded, and derived from 999 Monte Carlo replications.

```
ardl_eq.dynmult(noshade) 30
```

generates cumulative dynamic multiplier curves for each long-run regressor. The horizon length is 30 , and the $95 \%$ confidence intervals (if they exist), are not shaded.

```
ardl_eq.dynmult(noci)
```

produces cumulative dynamic multiplier curves for each long-run regressor. The horizon length is 15 , and no confidence intervals are displayed.

```
ardl_eq.dynmult(level=0.99, reps=499) 10
```

shows cumulative dynamic multiplier curves for each long-run regressor. The horizon length is 10 , and the $99 \%$ confidence intervals (if they exist), are shaded, and derived from 499 Monte Carlo replications.

## Cross-references

| ec | Var Methods |
| :--- | :--- |

Estimate a vector error correction model (VEC).

## Syntax

var_name.ec(options) lag_pairs endog_list [@x1 x2 x3 ...] [@exogsr sx1 sx2 sx3 ...] [@exoglr lx1 lx2 lx3 ...] [@exogboth bx1 bx2 bx3 ...]

Specify the order of the VEC by entering lag_pairs consisting of one or more pairs of lag intervals, and then list the series or groups to be used as endogenous variables.

Note that the lag orders are for the differences in the error correction representation of the VEC, not the levels representation of the VAR. If you are comparing results obtained elsewhere, you should be certain that the specifications for the lag orders are comparable.

In addition, you may optionally provide:

- an "@"-sign or "@exogsr" followed by a list of exogenous variables in the short-run equation only
- "@exoglr" followed by a list of exogenous variables in the long-run relation only
- "@exogboth" followed by a list of exogenous variables in both the long-run relation and the short-run equations

Do not include an intercept or trend in the VEC specification, these deterministic trend terms should be specified using the "determ = " option, as described below.

## Options

## Deterministic Trend Option

There are 8 different deterministic trend assumptions that you may specify using the "determ = arg" option.

These cases correspond to whether the intercept (" c ") and the trend (" t ") are either

- not included ("n")
- in the long-run cointegrating relation only ("l")
- in the short-run equation only ("s")
- in both the long and short-run equations ("b")

The values of arg are text shortcuts formed by joining a text shortcut for the intercept specification with a text shortcut for the trend specification.

The individual intercept and trend specifications are formed by joining the " c " and the " t " with the appropriate letter describing inclusion in the long and short-run equations.

For example,

- "cb" indicates that the constant is in both the long and short-run equation
- "tl" indicates that the trend is in the long-run cointegrating equation only
so that
- "cbtl" indicates that the constant is in both the long and short-run and the trend is in the long-run only

Using this convention (along with a special "none" option), we may easily describe options arguments for all 8 deterministic cases:

| cntn, none | Case 1: No deterministic terms. <br> Corresponding VAR model has no deterministic terms. <br> cltn <br> Case 2: Restricted constant. <br> Constant only in the cointegrating relations. <br> Corresponding VAR has a constant. |
| :--- | :--- |
| cstn | Case 3 (JHJ): Unrestricted constant <br> Constant included both in the short-run equation and (arti- <br> ficially) in the cointegrating relations via orthogonalization. <br> Corresponding VAR has a constant and trend. |
| cbtl | Case 3: Unrestricted constant <br> Constant only in the short-run equation. <br> Corresponding VAR has a trend. |
| Case 4 (JHJ): Unrestricted constant and restricted trend <br> Constant included both in the short-run equation and <br> (artificially) in the cointegrating relations via <br> orthogonalization, and trend included only in the cointe- <br> grating relations. <br> Corresponding VAR has a constant and trend. |  |


| cstl | Case 4: Unrestricted constant and restricted trend |
| :--- | :--- |
| Constant only in the short-run equation, and trend only in |  |
| the cointegrating relation. |  |
| Corresponding VAR has a trend. |  |
| Cbtb | Case $5(\mathrm{JHJ}):$ Unrestricted constant and trend |
| Constant and trend both included in the short-run equation |  |
| and (artificially) in the cointegrating relations via |  |
| orthogonalization. |  |
| Corresponding VAR has a constant, linear, and quadratic |  |
| trend. |  |
| csts | Case 5: Unrestricted constant and trend |
| Constant and trend both included in the short-run equa- |  |
| tion. |  |
| Corresponding VAR has a linear and quadratic trend. |  |

## Other Options

| rank $=$ integer <br> $($ default $=1)$ | Number of cointegrating relationships. |
| :--- | :--- |
| restrict | Impose restrictions as specified by the Var : : append <br> (p. 995) proc, or the "restspec =" option. |
| restspec = "spec" | Define the restricted VEC specification where spec is a <br> space a space delimited list of VEC coefficient restrictions. |
| $\mathrm{m}=$ integer, | Maximum number of iterations for restricted estimation <br> (only valid if you choose the restrict option). |
| $\mathrm{c}=$ scalar, |  |
| $\mathrm{cvg}=$ scalar | Convergence criterion for restricted estimation. (only valid <br> if you choose the restrict option). |
| prompt | Force the dialog to appear from within a program. |
| p | Print the results view. |

## Examples

```
var macro1.ec 1 4 m1 gdp tb3
```

declares a var object MACRO1 and estimates a VEC with four lagged first differences, three endogenous variables and one cointegrating equation using the default trend option " $c$ ".

```
var term.ec(determ=cstl, rank=2) 1 2 4 4 tb1 tb3 tb6 @ d2 d3 d4
```

declares a var object TERM and estimates a VEC with lagged first differences of order 1, 2, 4, three endogenous variables, three exogenous variables, and two cointegrating equations
using deterministic trend option "determ = cstl" for a model with a constant in the short-run equation, and a trend in the long-run cointegrating relation.

```
var macro1.ec(determ=cstl, rank=2) 1 2 4 4 tb1 tb3 tb6 @exogsr
    exog1 @exoglr exog2 @exogdual exog3
```

The line above declares a VAR object MACRO01 with the same basic specification as TERM, but with an additional short-run exogenous variable EXOG1, a long-run exogenous variable EXOG2, and a dual exogenous variable EXOG3.

## Cross-references

See "Vector Error Correction (VEC) Models" on page 883 of User's Guide II for a discussion of VECs.

See Var: : ls (p. 1029) and Var: :bvar (p. 1002) for estimation of ordinary VARs and Bayesian VAR models. See also, Var: :coint (p. 1006) and Var: :append (p. 995).

| ecresults | Equation Views |
| :--- | :--- |

Display a spool object showing tables with the conditional error correction (CEC) and error correction (EC) regression results.

## Syntax

eq_name.ecresults(options)

## Options

p Print output.

## Example

```
ardl_eq.ecresults
```

displays a spool object with the CEC and EC regressions from the ARDL equation ARDL_EQ.

## Cross-references

| 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

$$
\begin{aligned}
& \mathrm{e}=\arg \quad \text { Set error type: "a" (additive), "m" (multiplicative), "e" } \\
& \text { (default = "a") (auto). } \\
& \mathrm{t}=\arg \quad \text { Set trend type. key can be: " } \mathrm{n} \text { " (none), "a" (additive), } \\
& \text { (default }=\text { " } n \text { ") " } m \text { " (multiplicative), "ad" (additive dampened), " } m \text { " } \\
& \text { (multiplicative dampened), "e" (auto). } \\
& \mathrm{s}=\arg \quad \text { Set season type. key can be: "n" (none), "a"(additive), } \\
& \text { (default = "n") "m" (multiplicative), "e" (auto). } \\
& \text { modsel }=\text { arg Model selection method: "aic" (Akaike information cri- } \\
& \text { (default }=\text { "aic") terion), "bic" (Bayesian information criterion/Schwartz } \\
& \text { criterion), "hq" (Hannan-Quinn information criterion), } \\
& \text { "amse" (average mean squared errors). } \\
& \text { alpha }=\arg \quad \text { Specify fixed value for level parameter } \alpha . \\
& \text { beta }=\arg \quad \text { Specify fixed value for trend parameter } \beta \text { in models }
\end{aligned}
$$

| gamma $=\arg$ | Specify fixed value for seasonal parameter $\gamma$ in models <br> with a seasonal component. |
| :--- | :--- |
| phi $=\arg$ | Specify fixed value for dampening parameter $\phi$ in mod- <br> els with dampened trends. |
| nomult | Do not allow multiplicative trend or seasonal terms. <br> Only applies if the $t=e$ or $s=e$ options are set. |

## Optimization options

amse
namse $=$ integer
$\mathrm{c}=$ number
$\mathrm{m}=$ integer
ustart
noi

Set Average Mean Square Error (AMSE) as the objective function (The default is log-likelihood as the objective function).

Specify the AMSE length-the number of observations over which to calculate AMSE if "amse" is selected.

Set the convergence criteria.
Set the maximum number of iterations.
Employ user-supplied starting values (taken from the C vector in the workfile).

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). |
| :---: | :---: |
| $\begin{aligned} & \text { dgopt = arg } \\ & \text { (default }=\text { "m") } \end{aligned}$ | 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 " 1 " (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 " 1 " (likelihood comparison). |

$$
\begin{array}{ll}
\text { level = name } & \begin{array}{l}
\text { Save the level component as a separate series in the } \\
\text { workfile. }
\end{array} \\
\text { trend = name } & \begin{array}{l}
\text { Save the trend component as a separate series in the } \\
\text { workfile (if applicable). }
\end{array} \\
\text { season = name } & \begin{array}{l}
\text { Save the seasonal component as a separate series in the } \\
\text { workfile (if applicable). }
\end{array}
\end{array}
$$

## 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".

```
t.b3.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 "Exponential Smoothing" on page 551 of User's Guide I for a discussion of exponential smoothing methods.

See also : : smooth (p. 274).

| export | Coef Procs |
| :--- | :--- |

Export coef vector to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

coef_name.export(options) [path\Jfile_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel 2007 files is:
coef_name.export(options) [path \Jfile_name [table_description]
where the table_description may contain:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the Excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$$
\begin{array}{ll}
\mathrm{t}=\text { file_type } \\
(\text { default = "csv") }
\end{array} \quad \begin{aligned}
& \text { Specifies the file type, where file_type may be one of: } \\
& \text { "excelxml" (Excel 2007 (xml)),"ccs" (CSV - comma-sepa- } \\
& \text { rated), "rtf" (Rich-text format), "txt" (tab-delimited text), } \\
& \text { "html" (HTML - Hypertext Markup Language), "emf" } \\
& \text { (Enhanced Metafile), "pdf" (PDF - Portable Document For- } \\
& \text { mat), "tex" (LaTeX), or "md" (Markdown). } \\
& \text { Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", } \\
& \text { ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- } \\
& \text { tively. }
\end{aligned} \quad \begin{aligned}
& \text { Scale size, where arg is from 5 to 200, representing the per- } \\
& \text { centage of the original table size (only valid for HTML or } \\
& \text { RTF files). }
\end{aligned}
$$

## PDF Options

landscape
Save in landscape mode (the default is to save in portrait mode).
size $=$ arg
Page size: "letter", "legal", "a4", and "custom".
(default = "letter")
width = number Page width in inches if "size = custom".
(default $=8.5$ )
height $=$ number
(default = 11)
leftmargin $=$ number
Left margin width in inches.
(default $=0.5$ )
rightmargin $=$ number $\quad$ Right margin width in inches.
(default $=0.5$ )
topmargin $=$ number
(default = 1)
bottommargin $=\quad$ Bottom margin width in inches.
number $($ default $=1)$
Page height in inches if "size = custom".

## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

mode $=\arg$
cellfmt $=$ arg $\quad$ Specify whether to use EViews, pre-existing, or remove cell formatting (colors, font, number formatting when possible, column widths and row heights) for the written range.
arg may be "eviews" (replace current formatting in the file with the same cell formatting in EViews), "preserve" (leave current cell formatting already in the Excel file), or "clear" (remove current formatting and do not replace).

## Examples

The command:
coef1.export myvector
exports data in COEF1 to a CSV file named "myvector.CSV" in the default directory.

```
coef1.export(h,t=csv, \(n=" N a N ") ~ m y v e c t o r\)
```

saves the contents of COEF1 along with the column and row headers to a CSV (comma separated value) file named "myvector.CSV" and writes all NA values as "NaN".

```
coef1.export (h,t=html, \(s=50\) ) myvector
```

writes the data of COEF1 along with the column and row headers to a HTML file named "myvector.HTM" at half of the original size.

```
coef1.export(n=".", r=B) myvector
```

exports the data in the second column to a CSV file named "myvector.CSV", and writes all NA values as ".".

```
coef1.export(t=excelxml, cellfmt=clear, mode=update) myvector
    range=Country!b5
```

writes the data in COEF1 to the preexisting "myvector.XLSX" Excel file to the "Country" sheet at cell B5, where all cell formatting is cleared.

Cross-references

| export | Matrix Procs |
| :--- | :--- |

Export matrix to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

matrix_name.export(options) [path\Jfile_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel 2007 files is:
matrix_name.export(options) [path <br>file_name [table_description]
where the table_description may contain:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the Excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$\mathrm{t}=$ file_type

(default = "csv") $\quad$| Specifies the file type, where file_type may be one of: |
| :--- |
| "excelxml" (Excel 2007 (xml)),"csv" (CSV - comma-sepa- |
| rated), "rtt" (Rich-text format), "txt" (tab-delimited text), |
| "html" (HTML - Hypertext Markup Language), "emf" |
| (Enhanced Metafile), "pdf" (PDF - Portable Document For- |
| mat), "tex" (LaTeX), or "md" (Markdown). |
| Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", |
| ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- |
| tively. |

## PDF Options

landscape
size $=$ arg
(default = "letter")
width $=$ number
(default $=8.5$ )
(default = 11)
leftmargin = number
(default $=0.5$ )
rightmargin $=$ number
(default $=0.5$ )
topmargin $=$ number $\quad$ Top margin width in inches.
(default = 1)
bottommargin $=$ number (default $=1$ )
height = number $\quad$ Page height in inches if "size = custom".
Save in landscape mode (the default is to save in portrait mode).

Page size: "letter", "legal", "a4", and "custom".

Page width in inches if "size= custom".

Left margin width in inches.

Right margin width in inches.

Bottom margin width in inches.

## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

$$
\begin{array}{ll}
\text { mode }=\text { arg } & \begin{array}{l}
\text { Specify whether to create a new file, overwrite an existing } \\
\text { file, or update an existing file. arg may be "create" (create } \\
\text { new file only; error on attempt to overwrite) or "update" } \\
\text { (update an existing file, only overwriting the area specified } \\
\text { by the range = table_description). } \\
\text { If the "mode = " option is not used, EViews will create a } \\
\text { new file, unless the file already exists in which case it will } \\
\text { overwrite it. }
\end{array} \\
\text { cellfmt = arg } & \begin{array}{l}
\text { Specify whether to use EViews, pre-existing, or remove cell } \\
\text { formatting (colors, font, number formatting when possible, } \\
\text { column widths and row heights) for the written range. } \\
\text { arg may be "eviews" (replace current formatting in the file } \\
\text { with the same cell formatting in EViews), "preserve" (leave } \\
\text { current cell formatting already in the Excel file), or "clear" } \\
\text { (remove current formatting and do not replace). }
\end{array}
\end{array}
$$

## Examples

The command:

```
matrixl.export mymatrix
```

exports the data in MATRIX1 to a CSV file named "mymatrix.CSV" in the default directory.

```
matrix1.export(h, t=csv, n="NaN") mymatrix
```

saves the contents of MATRIX1 along with the column and row headers to a CSV (comma separated value) file named "mymatrix.CSV" and writes all NA values as "NaN".

```
matrixl.export(h, t=html, s=50) mymatrix
```

exports the data in MATRIX1 along with the column and row headers to a HTML file named "mymatrix.HTM" at half of the original size.

```
matrix1.save(n=".", r=B) mymatrix
```

saves the data in the second column to a CSV file named "mymatrix.CSV", and writes all NA values as ".".

```
matrixl.export(t=excelxml, cellfmt=clear, mode=update) mymatrix
    range=Country!b5
```

writes the data in MATRIX1 to the preexisting "mymatrix.XLSX" Excel file to the "Country" sheet at cell B5, where all cell formatting is cleared.

## Cross-references

| export | Rowvector Procs |
| :--- | :--- |

Export rowvector to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

vector_name.export(options) [path<br>file_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel 2007 files is:
vector_name.export(options) [path \Jfile_name [table_description]
where the table_description may contain:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the Excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$$
\begin{array}{ll}
\mathrm{t}=\text { file_type } \\
\text { (default = "csv") }
\end{array} \quad \begin{aligned}
& \text { Specifies the file type, where file_type may be one of: } \\
& \text { "excelxml" (Excel 2007 (xml)),"c"sv" (CSV - comma-sepa- } \\
& \text { rated), "rft" (Rich-text format), "txt" (tab-delimited text), } \\
& \text { "html" (HTML - Hypertext Markup Language), "emf" } \\
& \text { (Enhanced Metafile), "pdf" (PDF - Portable Document For- } \\
& \text { mat), "tex" (LaTeX), or "md" (Markdown). } \\
& \text { Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", } \\
& \text { ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- } \\
& \text { tively. }
\end{aligned} \quad \begin{aligned}
& \text { Scale size, where arg is from 5 to 200, representing the per- } \\
& \text { centage of the original table size (only valid for HTML or } \\
& \text { RTF files). }
\end{aligned}
$$

## PDF Options

landscape
Save in landscape mode (the default is to save in portrait mode).
size $=$ arg
Page size: "letter", "legal", "a4", and "custom".
(default = "letter")
width = number Page width in inches if "size = custom".
(default $=8.5$ )
height $=$ number
(default = 11)
leftmargin $=$ number
Left margin width in inches.
(default $=0.5$ )
rightmargin $=$ number $\quad$ Right margin width in inches.
(default $=0.5$ )
topmargin $=$ number $\quad$ Top margin width in inches.
(default = 1)
bottommargin $=\quad$ Bottom margin width in inches.
number $($ default $=1)$
Page height in inches if "size = custom".

## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

mode $=\arg$
cellfmt $=$ arg $\quad$ Specify whether to use EViews, pre-existing, or remove cell formatting (colors, font, number formatting when possible, column widths and row heights) for the written range.
arg may be "eviews" (replace current formatting in the file with the same cell formatting in EViews), "preserve" (leave current cell formatting already in the Excel file), or "clear" (remove current formatting and do not replace).

## Examples

The command:
rvector1.export myvector
exports data in VECTOR1 to a CSV file named "myvector.CSV" in the default directory.
rvector1.export(h,t=csv, $n=" N a N ")$ myvector
saves the contents of RVECTOR1 along with the column and row headers to a CSV (comma separated value) file named "myvector.CSV" and writes all NA values as "NaN".

```
rvector1.export(h,t=html, s=50) myvector
```

writes the data of VECTOR1 along with the column and row headers to a HTML file named "myvector.HTM" at half of the original size.

```
rvector1.export(n=".", r=B) myvector
```

exports the data in the second column to a CSV file named "myvector.CSV", and writes all NA values as ".".

```
rvectorl.export(t=excelxml, cellfmt=clear, mode=update) myvector
    range=Country!b5
```

writes the data in VECTOR1 to the preexisting "myvector.XLSX" Excel file to the "Country" sheet at cell B5, where all cell formatting is cleared.

Cross-references

| export | Sym Procs |
| :--- | :--- |

Export sym to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

sym_name.export(options) [path \ffile_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel 2007 files is:
sym_name.export(options) [path<br>file_name [table_description]
where the table_description may contain:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the Excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$\mathrm{t}=$ file_type

(default = "csv") $\quad$| Specifies the file type, where file_type may be one of: |
| :--- |
| "excelxml" (Excel 2007 (xml)),"csv" (CSV - comma-sepa- |
| rated), "rtt" (Rich-text format), "txt" (tab-delimited text), |
| "html" (HTML - Hypertext Markup Language), "emf" |
| (Enhanced Metafile), "pdf" (PDF - Portable Document For- |
| mat), "tex" (LaTeX), or "md" (Markdown). |
| Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", |
| ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- |
| tively. |

## PDF Options

landscape
size $=$ arg
(default = "letter")
width $=$ number
(default $=8.5$ )
(default = 11)
leftmargin $=$ number
(default $=0.5$ )
rightmargin $=$ number
(default $=0.5$ )
topmargin $=$ number $\quad$ Top margin width in inches.
(default = 1)
bottommargin $=$ number (default $=1$ )
height = number $\quad$ Page height in inches if "size = custom".
Save in landscape mode (the default is to save in portrait mode).

Page size: "letter", "legal", "a4", and "custom".

Page width in inches if "size= custom".

Left margin width in inches.

Right margin width in inches.

Bottom margin width in inches.

## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

> mode $=$ arg $\begin{aligned} & \text { Specify whether to create a new file, overwrite an existing } \\ & \text { file, or update an existing file. arg may be "create" (create } \\ & \text { new file only; error on attempt to overwrite) or "update" } \\ & \text { (update an existing file, only overwriting the area specified } \\ & \text { by the range = table_description). } \\ & \text { If the "mode = " option is not used, EViews will create a } \\ & \text { new file, unless the file already exists in which case it will } \\ & \text { overwrite it. }\end{aligned}$ Specify whether to use EViews, pre-existing, or remove cell formatting (colors, font, number formatting when possible, column widths and row heights) for the written range. arg may be "eviews" (replace current formatting in the file with the same cell formatting in EViews), "preserve" (leave current cell formatting already in the Excel file), or "clear" (remove current formatting and do not replace).

## Examples

The command:

```
sym1.export mysym
```

exports the data in SYM1 to a CSV file named "mysym.CSV" in the default directory.

```
sym1.export(h, t=csv, n="NaN") mysym
```

saves the contents of SYM1 along with the column and row headers to a CSV (comma separated value) file named "mysym.CSV" and writes all NA values as "NaN".

```
sym1.export(h, t=html, s=50) mysym
```

exports the data in SYM1 along with the column and row headers to a HTML file named "mysym.HTM" at half of the original size.

```
sym1.save(n=".", r=B) mysym
```

saves the data in the second column to a CSV file named "mysym.CSV", and writes all NA values as ".".

```
sym1.export(t=excelxml, cellfmt=clear, mode=update) mysym
    range=Country!b5
```

writes the data in SYM1 to the preexisting "mysym.XLSX" Excel file to the "Country" sheet at cell B5, where all cell formatting is cleared.

## Cross-references

| export | Vector Procs |
| :--- | :--- |

Export vector to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

vector_name.export(options) [path<br>file_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel 2007 files is:
vector_name.export(options) [path <br>file_name [table_description]
where the table_description may contain:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the Excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$$
\begin{array}{ll}
\mathrm{t}=\text { file_type } \\
(\text { default = "csv") }
\end{array} \quad \begin{aligned}
& \text { Specifies the file type, where file_type may be one of: } \\
& \text { "excelxml" (Excel 2007 (xml)),"ccs" (CSV - comma-sepa- } \\
& \text { rated), "rtf" (Rich-text format), "txt" (tab-delimited text), } \\
& \text { "html" (HTML - Hypertext Markup Language), "emf" } \\
& \text { (Enhanced Metafile), "pdf" (PDF - Portable Document For- } \\
& \text { mat), "tex" (LaTeX), or "md" (Markdown). } \\
& \text { Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", } \\
& \text { ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- } \\
& \text { tively. }
\end{aligned} \quad \begin{aligned}
& \text { Scale size, where arg is from 5 to 200, representing the per- } \\
& \text { centage of the original table size (only valid for HTML or } \\
& \text { RTF files). }
\end{aligned}
$$

## PDF Options

landscape
Save in landscape mode (the default is to save in portrait mode).
size $=$ arg
Page size: "letter", "legal", "a4", and "custom".
(default = "letter")
width = number Page width in inches if "size = custom".
(default $=8.5$ )
height $=$ number
(default = 11)
leftmargin $=$ number
Left margin width in inches.
(default $=0.5$ )
rightmargin $=$ number $\quad$ Right margin width in inches.
(default $=0.5$ )
topmargin $=$ number
(default = 1)
bottommargin $=\quad$ Bottom margin width in inches.
number $($ default $=1)$
Page height in inches if "size = custom".

## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

$$
\begin{array}{ll}
\text { mode }=\text { arg } & \begin{array}{l}
\text { Specify whether to create a new file, overwrite an existing } \\
\text { file, or update an existing file. arg may be "create" (create } \\
\text { new file only; error on attempt to overwrite) or "update" } \\
\text { (update an existing file, only overwriting the area specified } \\
\text { by the range = table_description). } \\
\text { If the "mode }=\text { " option is not used, EViews will create a } \\
\text { new file, unless the file already exists in which case it will } \\
\text { overwrite it. }
\end{array} \\
\text { cellfmt }=\arg \quad \begin{array}{l}
\text { Specify whether to use EViews, pre-existing, or remove cell } \\
\text { formatting (colors, font, number formatting when possible, } \\
\text { column widths and row heights) for the written range. } \\
\text { arg may be "eviews" (replace current formatting in the file } \\
\text { with the same cell formatting in EViews), "preserve" (leave } \\
\text { current cell formatting already in the Excel file), or "clear" } \\
\text { (remove current formatting and do not replace). }
\end{array}
\end{array}
$$

## Examples

The command:
vector1.export myvector
exports data in VECTOR1 to a CSV file named "myvector.CSV" in the default directory.
vector1.export(h,t=csv, $n=" N a N ")$ myvector
saves the contents of VECTOR1 along with the column and row headers to a CSV (comma separated value) file named "myvector.CSV" and writes all NA values as "NaN".

```
vector1.export(h,t=html, s=50) myvector
```

writes the data of VECTOR1 along with the column and row headers to a HTML file named "myvector.HTM" at half of the original size.

```
vector1.export(n=".", r=B) myvector
```

exports the data in the second column to a CSV file named "myvector.CSV", and writes all NA values as ".".

```
vectorl.export(t=excelxml, cellfmt=clear, mode=update) myvector
    range=Country!b5
```

writes the data in VECTOR1 to the preexisting "myvector.XLSX" Excel file to the "Country" sheet at cell B5, where all cell formatting is cleared.

## Cross-references

## fill

 Svector ProcsFill a svector with the specified values.

## Syntax

svector_name.fill(options) s1[s2 s3 ...]
Follow the keyword with a list of strings to place in the svector object. Each value should be surrounded by double quotes if necessary, and values should separated by a space. Running out of values before the object is completely filled is not an error; the remaining cells or observations will be unaffected, unless the "l" option is specified to enable looping. If, however, you list more values than the object can hold, EViews will return an error message.

## Options

| 1 | Loop repeatedly over the list of values as many times as it <br> takes to fill the vector. |
| :--- | :--- |
| $0=$ integer <br> $($ default $=1)$ | Fill the svector starting from the specified element. Default <br> is the first element. |

## Examples

```
sv1.fill a B C
```

sets the first element of SV1 to "a", the second to "B" and the third to " $c$ ".

```
sv1.fill(o=2) a "Hello World" name
```

sets the second element of SV1 to "a", the third to "Hello World" and the fourth to "name".

```
sv1.fill(o=4, l) first "" Last
```

sets the fourth element of SV1 to "first", the fifth to be an empty string, and the sixth value to "Last", then repeats the same three values for the remaining rows, so that the seventh element is set to "first", the eight element is empty, the ninth set to "Last" and so on.

```
sv1.fill(l) ""
```

clears all of the values in the svector SV1.

## Cross-references

See Chapter 11. "Matrix Language," on page 261 of the Command and Programming Reference for a detailed discussion of vector and matrix manipulation in EViews.

| fit | Equation Procs |
| :--- | :--- |

Compute static forecasts or fitted values from an estimated equation.
When the regressor contains lagged dependent values or ARMA terms, fit uses the actual values of the dependent variable instead of the lagged fitted values. You may instruct $f$ it to compare the forecasted data to actual data, and to compute forecast summary statistics.

Not available for equations estimated using ordered methods; use Equation: :makemodel (p. 153) to create a model using the ordered equation results (see example below).

## Syntax

```
eq_name.fit(options) yhat [y_se]
eq_name.fit(options) yhat [y_se y_var]
```

Following the fit keyword, you should type a name for the forecast series and, optionally, a name for the series containing the standard errors. For ARCH specifications, you may use the second form of the command, and optionally include a name for the conditional variance series.

Forecast standard errors are currently not available for binary, censored, and count models.

## Options

## Basic Options

d In models with implicit dependent variables, forecast the entire expression rather than the normalized variable.
u Substitute expressions for all auto-updating series in the equation.
g Graph the fitted values together with the $\pm 2$ standard error bands.
ga
e Produce the forecast evaluation table.
i Compute the fitted values of the index. Only for binary, censored and count models.

S
n
Ignore ARMA terms and use only the structural part of the equation to compute the fitted values.

Ignore coef uncertainty in standard error calculations that use them.

| forcsmpl $=$ <br> smpl | Fit sample (optional). If forecast sample is not provided, <br> the workfile sample will be employed. |
| :--- | :--- |
| $\mathrm{f}=$ arg <br> $($ default $=$ <br> "actual") | Out-of-fit-sample fill behavior: "actual" (fill observations <br> outside the fit sample with actual values for the fitted vari- <br> able), "na" (fill observations outside the fit sample with <br> missing values). |
| prompt | Force the dialog to appear from within a program. |
| p | Print view. |

## Stochastic Options

Options for forecasting from a functional coefficients estimated equation.

| stochastic = arg <br> (default $=$ <br> "none") | Stochastic method: "none" (none), "mca" (Monte Carlo - <br> asymptotic), "mcbs" (Monte Carlo - bootstrap), "bs" <br> (bootstrap). |
| :--- | :--- |
| reps $=$ integer <br> (default $=999)$ | Number of stochastic replications |
| lhr $=$ arg <br> $($ default $=0.1)$ | Lower historical range (number between 0 and upper his- <br> torical range). |
| uhr $=$ arg <br> (default $=0.9)$ | Upper historical range (number between lower historical <br> range and 1). |
| bsdep | Bootstrap only the dependent variable (not the functional <br> coefficient variable). |

## Examples

```
equation eql.ls cons c cons(-1) inc inc(-1)
eq1.fit c_hat c_se
genr c_up=c_hat+2*c_se
genr c_low=c_hat-2*c_se
line cons c_up c_low
```

The first line estimates a linear regression of CONS on a constant, CONS lagged once, INC, and INC lagged once. The second line stores the static forecasts and their standard errors as C_HAT and C_SE. The third and fourth lines compute the + /-2 standard error bounds. The fifth line plots the actual series together with the error bounds.

```
equation eq2.binary(d=l) y c wage edu
eq2.fit yf
eq2.fit(i) xbeta
genr yhat = 1-@clogit(-xbeta)
```

The first line estimates a logit specification for $Y$ with a conditional mean that depends on a constant, WAGE, and EDU. The second line computes the fitted probabilities, and the third line computes the fitted values of the index. The fourth line computes the probabilities from the fitted index using the cumulative distribution function of the logistic distribution. Note that YF and YHAT should be identical.

Note that you cannot fit values from an ordered model. You must instead solve the values from a model. The following lines generate fitted probabilities from an ordered model:

```
equation eq3.ordered y c x z
eq3.makemodel (oprob1)
solve oprob1
```

The first line estimates an ordered probit of Y on a constant, X , and Z . The second line makes a model from the estimated equation with a name OPROB1. The third line solves the model and computes the fitted probabilities that each observation falls in each category.

## Cross-references

To perform dynamic forecasting, use : :forecast (p. 180). See Equation: : makemodel (p. 153) and Model::solve (p. 553) for forecasting from systems of equations or ordered equations.

See Chapter 25. "Forecasting from an Equation," on page 167 of the User's Guide II for a discussion of forecasting in EViews and Chapter 31. "Discrete and Limited Dependent Variable Models," on page 363 of the User's Guide II for forecasting from binary, censored, truncated, and count models.

| fit | Var Procs |
| :--- | :--- |

Computes (n-period ahead) static forecasts of the VAR or VEC equation.
fit computes the static forecast of variables and all observations in a specified sample. In some settings, you may instruct forecast to compare the forecasted data to actual data, and to compute summary statistics.

## Syntax

var_name.fit(options) f_pattern [se_pattern]
You should enter a naming suffix for the forecast series and, optionally, a naming suffix for the series containing the standard errors. Standard errors are currently only available for non-Bayesian VARs, and are computed via simulation.

Not currently available for switching VARs

## Options

## General Options

| g | Graph the forecasts in individual graphs - one per depen- <br> dent variable. |
| :--- | :--- |
| m | Graph the forecasts in a combined graph. |
| $\mathrm{e}=\arg$ | Produce the forecast evaluation table. |
| Out-of-forecast-sample fill behavior: "actual" (fill observa- <br> "actual") | tions outside the forecast sample with actual values for the <br> fitted variable), "na" (fill observations outside the forecast <br> sample with missing values). |
| prompt | Force the dialog to appear from within a program. |
| p | Print view. |

## Non-Bayesian Options

| streps = integer | Number of simulation repetitions. Only applicable if a <br> se_pattern is provided. |
| :--- | :--- |
| $\mathrm{f}=$ number | Fraction of failed repetitions before stopping. Only applica- <br> ble if a se_pattern is provided. |

## BVAR Options

| classical | Perform classical forecasting - forecast based upon the pos- <br> terior means of the coefficients as if they were calculated <br> from a classical VAR. If omitted Bayesian sampling is used. |
| :--- | :--- |

If "classical" is not specified, the following Bayesian forecasting options are available:
mean Store the mean of the draws from the sampler. If omitted the median is stored.
draws $=$ integer $\quad$ Number of draws.
(default $=$ 100000)
burn $=\arg \quad$ Proportion of initial draws to discard.
(default $=0.1$ )
seed $=$ integer $\quad$ Random number seed.
dropunstable Drop any draws that produce unstable coefficients.
dgraph Produce distribution graphs.
fangraph Produce fan graphs.
page $=\arg \quad$ Store the individual draws in a new page.

## BTVCVAR Options

| usemean | Use posterior mean as the point estimate. The posterior median is used if usemean is not included in the options list. |
| :---: | :---: |
| showci | Show credibility intervals (bands). |
| $\begin{aligned} & \text { cilevels = arg } \\ & (\text { default = } \\ & " 0.95 ") \end{aligned}$ | Set credibility levels. For multiple levels, enter a spacedelimited list of values surrounded by quotation marks, e.g., "0.3 0.50 .8 ". |
| uselines | Use lines instead of shading for credibility intervals. |
| seed $=$ int | Set the random seed. EViews will generate a seed if one is not specified. |
| $\begin{aligned} & \text { rng }=\text { arg } \\ & \text { (default }=\text { "kn" } \\ & \text { or method set } \\ & \text { via rndseed) } \end{aligned}$ | Set random number generator type. Available types are: 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"). |

## Examples

The following lines:

```
smpl 1970q1 1990q4
var varl.ls 1 3 con inc
smpl 1991q1 1995q4
var1.fit(m) _f _se
```

estimate a VAR over the period 1970Q1-1990Q4, and then computes static forecasts for the period 1991Q1-1995Q4, and plots the forecasts as line graphs.

Cross-references
See "Forecasting" on page 868 of User's Guide II for a discussion of forecasting from VARs variance decompositions

See also : :forecast (p. 180).

| fixcol | Table Procs |
| :--- | :--- |

Fixes a set of columns to left of the spreadsheet view of a table object so that the leading columns are always in view.

## Syntax

table_name.fixcol cols
where cols is the number of columns to be fixed

## Example

```
tab1.fixcol 3
```

fixes the first 3 columns of the table TAB1 such that they are always in view despite the horizontal scroll position.

```
tab1.fixcol 0
```

removes any fixed columns in table TAB1.
Cross-references

| fixrow | Table Procs |
| :--- | :--- |

Fixes a set of rows at the top of the spreadsheet view of a table object so that the leading rows are always in view.

## Syntax

table_name.fixrow rows
where rows is the number of rw to be fixed

## Example

tab1.fixrow 2
fixes the first 2 rows of the table TAB1 such that they are always in view despite the vertical scroll position.

```
tab1.fixrow 0
```

removes any fixed rows in table TAB1.

## Cross-references

| fixrowcol | Table Procs |
| :--- | :--- |

Fixes a set of rows at the top and a set of columns to left of a spreadsheet view of a table object so that the leading rows and columns are always in view.

## Syntax

table_name.fixrowcol rows cols
where rows is the number of rows to be fixed and cols is the number of columns to be fixed.

## Example

tab1.fixrowcol 14
fixes the first row and the first 4 columns of the table TAB1 such that they are always in view despite the horizontal and vertical scroll position of the table.
tab1.fixrowcol 00
removes all fixed rows and columns in table TAB1.
tab1.fixrowcol 04
in table TAB1 removes all fixed rows but fixes the first 4 columns.

## Cross-references

| forecast | Equation Procs |
| :--- | :--- |

Computes ( $n$-period ahead) dynamic forecasts of an estimated equation.
forecast computes the forecast for all observations in a specified sample. In some settings, you may instruct forecast to compare the forecasted data to actual data, and to compute summary statistics.

## Syntax

eq_name.forecast(options) yhat [y_se]
eq_name.forecast(options) yhat [y_se y_var]
Enter a name for the forecast series and, optionally, a name for the series containing the standard errors. For ARCH specifications, you may use the second form of the command, and optionally enter a name for the conditional variance series. Forecast standard errors are currently not available for binary or censored models. forecast is not available for models estimated using ordered methods.

## Options

| d | In models with implicit dependent variables, forecast the entire expression rather than the normalized variable. |
| :---: | :---: |
| u | Substitute expressions for all auto-updating series in the equation. |
| g | Graph the forecasts together with the $\pm 2$ standard error bands. |
| ga | Graph the forecasts along with the actuals (if available). |
| e | Produce the forecast evaluation table. |
| i | Compute the forecasts of the index. Only for binary, censored and count models. |
| s | Ignore ARMA terms and use only the structural part of the equation to compute the forecasts. |
| n | Ignore coef uncertainty in standard error calculations that use them. |
| $\mathrm{b}=\arg$ | MA backcast method: "fa" (forecast available). Only for equations estimated with MA terms. This option is ignored if you specify the "s" (structural forecast) option. The default method uses the estimation sample. |
| forcsmpl $=$ smpl | Forecast sample (optional). If forecast sample is not provided, the workfile sample will be employed |
| $\begin{aligned} & \mathrm{f}=\text { arg } \\ & \text { (default }= \\ & \text { "actual") } \end{aligned}$ | 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 forecas sample with missing values). |
| stochastic | Perform stochastic simulation for dynamic equations estimated using least squares. |
| $\begin{aligned} & \text { streps }=\text { integer } \\ & (\text { default }=1000) \end{aligned}$ | Number of stochastic repetitions (for threshold regression or stochastic simulation). |
| stfrac $=$ number <br> (default $=.02$ ) | Fraction of failed repetitions before stopping (for threshold regression or stochastic simulation). |
| prompt | Force the dialog to appear from within a program. |
| p | Print view. |

## Examples

The following lines:

```
equation eq1.ls con c con(-1) inc
smpl 1991q1 1995q4
eq1.fit con_s
eq1.forecast con_d
plot con_s con_d
```

estimate a linear regression over the period 1970Q1-1990Q4, compute static (fitted) and dynamic forecasts for the period 1991Q1-1995Q4, and plot the two forecasts in a single graph.

```
equation eq1.ls m1 gdp ar(1) ma(1)
eq1.forecast m1_bj bj_se
eq1.forecast(s) m1_s s_se
plot bj_se s_se
```

estimates an ARMA $(1,1)$ model, computes the forecasts and standard errors with and without the ARMA terms, and plots the two forecast standard errors.

Cross-references
To perform static forecasting with equation objects see : :fit (p. 174). For multiple equation forecasting, see Equation: :makemodel (p. 153), and Model::solve (p. 553).

For more information on equation forecasting in EViews, see Chapter 25. "Forecasting from an Equation," on page 167 of the User's Guide II.

| forcavg | Series Procs |
| :--- | :--- |

Average different forecasts of a series.

## Syntax

series.forcavg(options) forecast_data
You should specify the forecast data to be averaged by entering a list of objects as forecast_data. The list may be a list of series objects, a group object, a series naming pattern (such as " f " to indicate all series starting with the letter " F "), or a list of equation objects.

If a list of equations is entered, EViews will automatically forecast from those equation objects over the forecast sample (the current workfile sample).

## Options

| wgt = "key" | Set the type of averaging to use. key can be "mean" <br> (default), "trmean" (trimmed-mean)," "med" (median), <br> "ols" (least squares weights), "mse" (mean square error <br> weights), "ranks", (MSE ranks), "aic" (Smoothed AIC <br> weights), or "sic" (BMA weights). "aic" and "sic" are only <br> available if a list of equations is provided as the forecast_- <br> data. |
| :--- | :--- |
| trim = num | Set the level of trimming for the Trimmed mean method. <br> Num should be a number between 1 and 100. Only applica- <br> ble if the "trmean" option is used. |
| msepwr = int | Set the power to which the MSE values are raised in the <br> MSE ranks method. Only applicable if the "mseranks" <br> option is used. |
| s | Use a static (rather than dynamic) forecast when comput- <br> ing the forecasts over the training sample. Only applicable <br> if forecast_data is a list of equation objects. |
| forcsmpl = arg | Forecast sample (optional). If forecast sample is not pro- <br> vided, the workfile sample will be employed. |
| trainsmpl = arg | Specify the sample used for calculating the averaging <br> weights. Only applicable if the "ols", "mse", "mseranks", <br> "aic" or "sic" options are used. |
| name = arg | Set the name of the final averaged series. |
| wgtname = arg | Save the weights into a vector in the workfile with the <br> name wgtname. |

## Example

The commands

```
wfopen elecdmd.wf1
elecdmd.forcavg(trainsmpl="2012M1 2012M12", wgt=mse) elecf_fe*
```

open the workfile elecdmd.wf1 and then perform forecast averaging using the actual series ELECDMD, and the forecast series specified by the naming pattern ELECF_FE*.

The averaging method MSE is used. A training sample of 2012M1 to 2012 M 12 is used to calculate the weights in the MSE and MSE Ranks methods.

See "Forecast Averaging" on page 545 of User's Guide I for additional discussion.
See also Series: forceval (p. 683).
$\square$
forecast Var Procs

Computes (n-period ahead) dynamic forecasts of the VAR or VEC equation.
forecast computes the forecast for all variables and all observations in a specified sample. In some settings, you may instruct forecast to compare the forecasted data to actual data, and to compute summary statistics.

## Syntax

var_name.forecast(options) f_pattern [se_pattern]
You should enter a naming suffix for the forecast series and, optionally, a naming suffix for the series containing the standard errors. Forecast standard errors are currently only available for non-Bayesian VARs, and are computed via simulation.

Not currently available for switching VARs

## Options

## General Options

g Graph the forecasts in individual graphs - one per dependent variable.
m Graph the forecasts in a combined graph.
e Produce the forecast evaluation table.
$\mathrm{f}=\arg \quad$ Out-of-forecast-sample fill behavior: "actual" (fill observa(default $=\quad$ tions outside the forecast sample with actual values for the "actual") fitted variable), "na" (fill observations outside the forecast sample with missing values).
prompt Force the dialog to appear from within a program.
p Print view.

## Non-Bayesian Options

| streps = integer | Number of simulation repetitions. Only applicable if a <br> se_pattern is provided. |
| :--- | :--- |
| $\mathrm{f}=$ number | Fraction of failed repetitions before stopping. Only applica- <br> ble if a se_pattern is provided. |

## BVAR Options

classical Perform classical forecasting - forecast based upon the posterior means of the coefficients as if they were calculated from a classical VAR. If omitted Bayesian sampling is used.

If "classical" is not specified, the following Bayesian forecasting options are available:
mean Store the mean of the draws from the sampler. If omitted the median is stored.
draws $=$ integer $\quad$ Number of draws.
(default =
100000)
burn $=\arg \quad$ Proportion of initial draws to discard.
(default $=0.1$ )
seed $=$ integer $\quad$ Random number seed.
dropunstable Drop any draws that produce unstable coefficients.
dgraph Produce distribution graphs.
fangraph Produce fan graphs.
page $=$ arg $\quad$ Store the individual draws in a new page.

## BTVCVAR Options

| usemean | Use posterior mean as the point estimate. The posterior <br> median is used if usemean is not included in the options <br> list. |
| :--- | :--- |
| showci | Show credibility intervals (bands). |
| cilevels = arg <br> (default $=$ <br> "0.95") | Set credibility levels. For multiple levels, enter a space- <br> delimited list of values surrounded by quotation marks, <br> e.g., "0.3 0.5 0.8". |
| uselines | Use lines instead of shading for credibility intervals. |
| seed = int | Set the random seed. EViews will generate a seed if one is <br> not specified. |
| rng = arg | Set random number generator type. Available types are: <br> (default $=$ "kn" <br> improved Knuth generator ("kn"), improved Mersenne <br> or method set <br> via rndseed) |
| Twister ("mt"), Knuth’s (1997) lagged Fibonacci generator <br> used in EViews 4 ("kn4"), L'Ecuyer's (1999) combined <br> multiple recursive generator ("le"), Matsumoto and |  |
| Nishimura's (1998) Mersenne Twister used in EViews 4 |  |
| ("mt4"). |  |

## Examples

The following lines:

```
smpl 1970q1 1990q4
var varl.ls 1 3 con inc
smpl 1991q1 1995q4
var1.forecast(m) _f _se
```

estimate a VAR over the period 1970Q1-1990Q4, and then computes dynamic forecasts for the period 1991Q1-1995Q4, and plots the forecasts as line graphs.

```
smpl 1970q1 1990q4
var var2.bvar(prior=inw) 1 3 con inc
smpl 1991q1 1995q4
var1.forecast(m, draws=50000, burn=.05, dgraph, page=draws) _f
```

estimates a Bayesian VAR with an independent normal-Wishart prior over the same period, and then forecasts that VAR taking 50,000 draws of a Gibbs sampler, discarding the first 2,500 draws, producing a distribution graph of the forecasts and storing the draws into a new panel page called DRAWS.

## Cross-references

See "Forecasting" on page 868 of User's Guide II for a discussion of forecasting from VARs and VECS.

See also : : fit (p. 174).

| import | Coef Procs |
| :--- | :--- |

Imports data from a foreign file into the coef object.

## Syntax

coef_name.import([type = ]) source_description import_specification

- source_description should contain a description of the file from which the data is to be imported. The specification of the description is usually just the path and file name of the file, however you can also specify more precise information. See wfopen (p. 539) of the Command and Programming Reference for more details on the specification of source_description.
- The optional "type = " option may be used to specify a source type. For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename. The following table summaries the various source formats and along with the corresponding "type = " keywords:

|  | Option Keywords |
| :--- | :---: |
| Excel (through 2003) | "excel" |
| Excel 2007 (xml) | "excelxml" |
| HTML | "html" |
| Text / ASCII | "text" |

- import_specification can be used to provide additional information about the file to be read. The details of import_specification will depend upon the type of file being imported.


## Excel Files

The syntax for reading Excel files is:
coef_name.import(type = excel[xml]) source_description [table_description]
[variables_description]
The following table_description elements may be used when reading Excel data:

- "range $=a r g "$, where $a r g$ is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the data (default is 1). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
coef_obj.import "c:\data files\data.xls"
```

loads the active sheet of "data.XLSX" into the VEC_NAME vector object.

```
coef_obj.import "c:\data files\data.xls" range="GDP data"
```

reads the data contained in the "GDP data" sheet of "data.XLS" into the COEF_OBJ object.

## HTML Files

The syntax for reading HTML pages is:
> coef_name.import(type = html) source_description [table_description] [variables_description]

The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which HTML table to read in an HTML file/page containing multiple tables.

When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.

- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 " "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
coef_obj.import "c:\data.html"
```

loads into the COEF_OBJ the data located in the HTML file "Data.HTML" located on the $\mathrm{C}: \backslash$ drive

```
coef_obj.import(type=html)
    "http://www.tradingroom.com.au/apps/mkt/forex.ac" colhead=3
```

loads into a coef object called COEF_OBJ the data with the given URL located on the website site "http://www.tradingroom.com.au". The column header is set to three rows.

## Text and Binary Files

The syntax for reading text or binary files is:
coef_name.import(type = arg) source_description [table_description] [variables_description]

If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype = [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file:
"crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting.
"fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines = int", number of lines to use in forming each row when "rectype = crlf" (default is 1 ).
- "reclen $=$ int", number of bytes to use in forming each row when "rectype $=$ fixed".
- "recfields = int", number of fields to use in forming each row when "rec-
type $=$ streamed".
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes $=$ [single $\mid$ double|both|none]", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " t " for tab, " s " for space and " a " for any character.
- "mult $=[$ on $\mid$ off $]$ ", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian $=$ [big|little]", selects the endianness of numeric fields contained in binary files.
- "string = [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra
zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column. When importing a CSV file, lines which have the delimiter as the last character (for example: "name, description, date"), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns. Note this is not the same as a line containing "name, description, date". In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:
fformat = ([n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)
where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E-scientific
A - alphanumeric
X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor (default $=1$ ). More complicated Fortran compatible variations on this format are possible.
- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)"
where optional type is " $\$$ " for string or " $\#$ " for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.

Text and Binary File Examples (.txt, .csv, etc.)
coef_obj.import c:\data.csv skip=5
reads "Data.CSV" into a coef_obj, skipping the first 5 rows.

```
coef_obj.import(type=text) c:\date.txt delim=comma
```

loads the comma delimited data "Date.TXT" into the COEF_OBJ matrix object.

| import | Matrix Procs |
| :--- | :--- |

Imports data from a foreign file into the matrix object.

## Syntax

matrix_name.import([type = ]) source_description import_specification

- source_description should contain a description of the file from which the data is to be imported. The specification of the description is usually just the path and file name of the file, however you can also specify more precise information. See wfopen (p. 539) of the Command and Programming Reference for more details on the specification of source_description.
- The optional "type = " option may be used to specify a source type. For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename. The following table summaries the various source formats and along with the corresponding "type = " keywords:

|  | Option Keywords |
| :--- | :---: |
| Excel (through 2003) | "excel" |
| Excel 2007 (xml) | "excelxml" |
| HTML | "html" |
| Text / ASCII | "text" |

- import_specification can be used to provide additional information about the file to be read. The details of import_specification will depend upon the type of file being imported.


## Excel Files

The syntax for reading Excel files is:
matrix_name.import(type = excel[xml]) source_description [table_description] [variables_description]

The following table_description elements may be used when reading Excel data:

- "range = arg", where arg is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
matrix_name.import "c:\data files\data.xls"
```

loads the active sheet of DATA.XLSX into the MATRIX_NAME matrix object.

```
matrix_name.import "c:\data files\data.xls" range="GDP data"
```

reads the data contained in the "GDP data" sheet of "Data.XLS" into the MATRIX_NAME object.

## HTML Files

The syntax for reading HTML pages is:
matrix_name.import(type = html) source_description [table_description] [variables_description]

The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which HTML table to read in an HTML file/page containing multiple tables.
When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.
- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int|all]", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
mat1.import "c:\data.html"
```

loads into the MAT1 matrix object the data located in the HTML file "Data.HTML" located on the $\mathrm{C}: \backslash$ drive

```
mat1.import(type=html)
    "http://www.tradingroom.com.au/apps/mkt/forex.ac" colhead=3
```

loads into a matrix object MAT1 the data with the given URL located on the website site "http://www.tradingroom.com.au". The column header is set to three rows.

## Text and Binary Files

The syntax for reading text or binary files is:

$$
\begin{aligned}
& \text { matrix_name.import(type = arg) source_description [table_description] } \\
& \text { [variables_description] }
\end{aligned}
$$

If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype = [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file: "crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting. "fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines $=$ int", number of lines to use in forming each row when "rectype $=$ crlf" (default is 1 ).
- "reclen = int", number of bytes to use in forming each row when "rectype = fixed".
- "recfields = int", number of fields to use in forming each row when "rectype = streamed".
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes $=$ [single $\mid$ double $\mid$ both $\mid$ none]", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " t " for tab, " s " for space and "a" for any character.
- "mult $=[$ on $\mid$ off $]$ ", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian $=$ [big|little]", selects the endianness of numeric fields contained in binary files.
- "string $=$ [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column. When importing a CSV file, lines which have the delimiter
as the last character (for example: "name, description, date"), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns. Note this is not the same as a line containing "name, description, date". In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:
fformat = ([n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)
where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E - scientific
A - alphanumeric
X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor (default $=1$ ). More complicated Fortran compatible variations on this format are possible.
- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)"
where optional type is " $\$$ " for string or " $\#$ " for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.

Text and Binary File Examples (.txt, .csv, etc.)
mat2.import c:\data.csv skip=5
reads "Data.CSV" into a MAT2, skipping the first 5 rows.

```
mat2.import(type=text) c:\date.txt delim=comma
```

loads the comma delimited data "Date.TXT" into the MAT2 matrix object.

| import | Rowvector Procs |
| :--- | :--- |

Imports data from a foreign file into the rowvector object.

## Syntax

rowvector_name.import([type = ]) source_description import_specification

- source_description should contain a description of the file from which the data is to be imported. The specification of the description is usually just the path and file name of the file, however you can also specify more precise information. See wfopen (p. 539) of the Command and Programming Reference for more details on the specification of source_description.
- The optional "type = " option may be used to specify a source type. For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename. The following table summaries the various source formats and along with the corresponding "type = " keywords:

|  | Option Keywords |
| :--- | :---: |
| Excel (through 2003) | "excel" |
| Excel 2007 (xml) | "excelxml" |
| HTML | "html" |
| Text / ASCII | "text" |

- import_specification can be used to provide additional information about the file to be read. The details of import_specification will depend upon the type of file being imported.


## Excel Files

The syntax for reading Excel files is:

$$
\begin{aligned}
& \text { rowvector_name.import }(\text { type }=\text { excel }[x \mathrm{xll}]) \text { source_description [table_description] } \\
& \text { [variables_description] }
\end{aligned}
$$

The following table_description elements may be used when reading Excel data:

- "range = arg", where arg is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
rowvec_obj.import "c:\data files\data.xls"
```

loads the active sheet of "Data.XLSX" into the ROWVEC_OBJ matrix object.

```
rowvec_obj.import "c:\data files\data.xls" range="GDP data"
```

reads the data contained in the "GDP data" sheet of "Data.XLS" into the ROWVEC_OBJ object.

## HTML Files

The syntax for reading HTML pages is:

```
rowvector_name.import(type = html) source_description [table_description]
    [variables_description]
```

The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which HTML table to read in an HTML file/page containing multiple tables.

When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.

- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 " "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
rvec.import "c:\data.html"
```

loads into the RVEC object the data located in the HTML file "Data.HTML" located on the $\mathrm{C}: \backslash$ drive
rvec.import (type=html)
"http://www.tradingroom.com.au/apps/mkt/forex.ac" colhead=3
loads into a rowvector RVEC the data with the given URL located on the website site "http://www.tradingroom.com.au". The column header is set to three rows.

## Text and Binary Files

The syntax for reading text or binary files is:

$$
\begin{aligned}
& \text { rowvector_name.import(type = arg) source_description [table_description] } \\
& \text { [variables_description] }
\end{aligned}
$$

If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype $=$ [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file:
"crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting.
"fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines = int", number of lines to use in forming each row when "rectype = crlf" (default is 1 ).
- "reclen = int", number of bytes to use in forming each row when "rectype = fixed".
- "recfields = int", number of fields to use in forming each row when "rectype $=$ streamed $"$.
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes $=[$ single $\mid$ double $\mid$ both $\mid$ none $]$ ", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " t " for tab, " s " for space and "a" for any character.
- "mult $=[$ on $\mid$ off $]$ ", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian $=$ [big|little]", selects the endianness of numeric fields contained in binary files.
- "string $=$ [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column.

When importing a CSV file, lines which have the delimiter as the last character (for example: 'name,description,date,'), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns. Note this is not the same as a line containing 'name,description,date'. In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:
fformat $=([n 1] T y p e[$ Width][.Precision], [n2]Type[Width][.Precision], ...)
where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E - scientific
A - alphanumeric
X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor (default $=1$ ). More complicated Fortran compatible variations on this format are possible.
- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)"
where optional type is " $\$$ " for string or " $\#$ " for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int|all]", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.

Text and Binary File Examples (.txt, .csv, etc.)

```
rvec2.import c:\data.csv skip=5
```

reads "Data.CSV" into a RVEC2, skipping the first 5 rows.

```
rvec2.import(type=text) c:\date.txt delim=comma
```

loads the comma delimited data "Date.TXT" into the RVEC2 matrix object.

| import | Sym Procs |
| :--- | :--- |

Imports data from a foreign file into the sym object.

## Syntax

sym_name.import([type = ]) source_description import_specification

- source_description should contain a description of the file from which the data is to be imported. The specification of the description is usually just the path and file name of the file, however you can also specify more precise information. See wfopen (p. 539) of the Command and Programming Reference for more details on the specification of source_description.
- The optional "type = " option may be used to specify a source type. For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename. The following table summaries the various source formats and along with the corresponding "type = " keywords:

|  | Option Keywords |
| :--- | :---: |
| Excel (through 2003) | "excel" |
| Excel 2007 (xml) | "excelxml" |
| HTML | "html" |
| Text / ASCII | "text" |

- import_specification can be used to provide additional information about the file to be read. The details of import_specification will depend upon the type of file being imported.


## Excel Files

The syntax for reading Excel files is:
sym_name.import(type = excel[xml]) source_description [table_description]
[variables_description]
The following table_description elements may be used when reading Excel data:

- "range $=$ arg", where $\arg$ is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the "na=" argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same rows. If the first five rows are not enough to correctly determine the data format, use the "scan = " argument to instruct EViews to look at more rows. In addition, you may want to specify a the "na= " value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the data (default is 1). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
sym_obj.import "c:\data files\data.xls"
```

loads the active sheet of DATA.XLSX into the SYM_NAME sym object.

```
sym_obj.import "c:\data files\data.xls" range="GDP data"
```

reads the data contained in the "GDP data" sheet of "Data.XLS" into the SYM_OBJ object.

## HTML Files

The syntax for reading HTML pages is:
sym_name.import(type $=\mathrm{html})$ source_description [table_description] [variables_description]

The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which HTML table to read in an HTML file/page containing multiple tables.
When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.
- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = " $\arg 1 "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the "na=" argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same rows. If the first five rows are not enough to correctly determine the data format, use the "scan = " argument to instruct EViews to look at more rows. In addition, you may want to specify a the "na= " value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
sym01.import 01"c:\data.html"
```

loads into the SYM01 matrix object the data located in the HTML file "Data.HTML" located on the $\mathrm{C}: \backslash$ drive

## Text and Binary Files

The syntax for reading text or binary files is:
sym_name.import(type = arg) source_description [table_description] [variables_description]

If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype = [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file:
"crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting.
"fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines $=$ int", number of lines to use in forming each row when "rectype $=$ crlf" (default is 1 ).
- "reclen $=$ int", number of bytes to use in forming each row when "rectype $=$ fixed".
- "recfields = int", number of fields to use in forming each row when "rec-
type $=$ streamed".
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes $=$ [single $\mid$ double|both|none]", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " $t$ " for tab, " $s$ " for space and "a" for any character.
- "mult $=[$ on $\mid$ off $]$ ", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian = [big|little]", selects the endianness of numeric fields contained in binary files.
- "string $=$ [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column.

When importing a CSV file, lines which have the delimiter as the last character (for example: 'name,description,date,'), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns.

Note this is not the same as a line containing 'name,description,date'. In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:

$$
\text { fformat }=([n 1] T y p e[\text { Width][.Precision], [n2]Type[Width][.Precision], ...) }
$$

where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E - scientific
A - alphanumeric
X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor $($ default $=1)$. More complicated Fortran compatible variations on this format are possible.

- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)"
where optional type is " $\$$ " for string or "\#" for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "types = ("arg1","arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), " f " (numeric data), "d" (dates), or "w" (EViews automatic detection). This option is rarely used.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the "na=" argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same
rows. If the first five rows are not enough to correctly determine the data format, use the "scan = " argument to instruct EViews to look at more rows. In addition, you may want to specify a the "na=" value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.

Text and Binary File Examples (.txt, .csv, etc.)

```
sym2.import c:\data.csv skip=5
```

reads "Data.CSV" into SYM2, skipping the first 5 rows.

```
sym01.import(type=text) c:\date.txt delim=comma
```

loads the comma delimited data "Date.TXT" into the SYM01 matrix object.

| import | Vector Procs |
| :--- | :--- |

Imports data from a foreign file into the vector object.

## Syntax

vector_name.import([type = ]) source_description import_specification

- source_description should contain a description of the file from which the data is to be imported. The specification of the description is usually just the path and file name of the file, however you can also specify more precise information. See wfopen (p. 539) of the Command and Programming Reference for more details on the specification of source_description.
- The optional "type = " option may be used to specify a source type. For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename. The following table summaries the various source formats and along with the corresponding "type = " keywords:

|  | Option Keywords |
| :--- | :---: |
| Excel (through 2003) | "excel" |
| Excel 2007 (xml) | "excelxml" |
| HTML | "html" |
| Text / ASCII | "text" |

- import_specification can be used to provide additional information about the file to be read. The details of import_specification will depend upon the type of file being imported.


## Excel Files

The syntax for reading Excel files is:

$$
\begin{aligned}
& \text { vector_name.import(type = excel[xml]) source_description [table_description] } \\
& \text { [variables_description] }
\end{aligned}
$$

The following table_description elements may be used when reading Excel data:

- "range = arg", where arg is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "types = ("arg1","arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), "f" (numeric data), "d" (dates), or "w" (EViews automatic detection). This option is rarely required.
- "na = " $\arg 1 "$ ", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the data (default is 1). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
vec_obj.import "c:\data files\data.xls"
```

loads the active sheet of DATA.XLSX into the VEC_NAME vector object.

```
vec_obj.import "c:\data files\data.xls" range="GDP data"
```

reads the data contained in the "GDP data" sheet of "Data.XLS" into the VEC_OBJ object.

## HTML Files

The syntax for reading HTML pages is:
vector_name.import(type $=\mathrm{html}$ ) source_description [table_description] [variables_description]

The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which HTML table to read in an HTML file/page containing multiple tables.

When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.

- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan $=[$ int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
vec1.import "c:\data.html"
```

loads into the VEC1 matrix object the data located in the HTML file "Data.HTML" located on the $\mathrm{C}: \backslash$ drive

```
vec1.import(type=html)
    "http://www.tradingroom.com.au/apps/mkt/forex.ac" colhead=3
```

loads into a vector object called VEC1 the data with the given URL located on the website site "http://www.tradingroom.com.au". The column header is set to three rows.

## Text and Binary Files

The syntax for reading text or binary files is:

```
vector_name.import(type = arg) source_description [table_description]
    [variables_description]
```

If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype $=$ [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file:
"crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting.
"fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines = int", number of lines to use in forming each row when "rectype = crlf" (default is 1 ).
- "reclen = int", number of bytes to use in forming each row when "rectype = fixed".
- "recfields = int", number of fields to use in forming each row when "rectype $=$ streamed $"$.
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes $=$ [single $\mid$ double $\mid$ both $\mid$ none]", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " t " for tab, " s " for space and "a" for any character.
- "mult $=[$ on $\mid$ off $]$ ", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian $=$ [big|little]", selects the endianness of numeric fields contained in binary files.
- "string $=$ [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column.

When importing a CSV file, lines which have the delimiter as the last character (for example: 'name,description,date,'), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns. Note this is not the same as a line containing 'name,description,date'. In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:
fformat $=([n 1] T y p e[$ Width][.Precision], [n2]Type[Width][.Precision], ...)
where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E - scientific
A - alphanumeric
X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor (default $=1$ ). More complicated Fortran compatible variations on this format are possible.
- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)"
where optional type is " $\$$ " for string or " $\#$ " for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "types = ("arg1","arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), "f" (numeric data), "d" (dates), or "w" (EViews automatic detection). This option is rarely used.
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan $=[$ int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file).
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.

Text and Binary File Examples (.txt, .csv, etc.)

```
vec2.import c:\data.csv skip=5
```

reads "Data.CSV" into a VEC2, skipping the first 5 rows.

```
vec01.import(type=text) c:\date.txt delim=comma
```

loads the comma delimited data DATE.TXT into the VEC01 matrix object.

| impulse | Var Views |
| :--- | :--- |

Display impulse response functions of var object with an estimated VAR or VEC.

## Syntax

var_name.impulse( $n$, options) ser1 [ser2 ser3 ...] [@ shock_series [@ ordering_series]]
You must specify the number of periods $n$ over which to compute the impulse response functions.

List the series names in the var whose responses you would like to compute. You may optionally specify the sources of shocks. To specify the shocks, list the series after an "@". By default, EViews computes the responses to all possible sources of shocks using the ordering in the Var.

If you are using impulses from the Cholesky factor, you may change the Cholesky ordering by listing the order of the series after a second "@".

## Options

## General Options

| g | Display combined graphs, with impulse responses of one <br> variable to all shocks shown in one graph. |
| :--- | :--- |
| m (default) | Display multiple graphs, with impulse response to each <br> shock shown in separate graphs. |
| t | Tabulate the impulse responses. |
| a | Accumulate the impulse responses. |


| $\begin{aligned} & \text { imp }=\text { arg } \\ & \text { (default }=\text { "chol") } \end{aligned}$ | Type of factorization for the decomposition: unit impulses, ignoring correlations among the residuals ("imp = unit"), non-orthogonal, ignoring correlations among the residuals ("imp = nonort"), Cholesky with d.f. correction <br> ("imp = chol"), Cholesky without d.f. correction <br> ("imp = mlechol"), Generalized ("imp = gen"), structural <br> ("imp = struct"), or user specified ("imp = user"). |
| :---: | :---: |
|  | The structural factorization is based on the estimated structural VAR. To use this option, you must first estimate the structural decomposition; see Var: : svar (p. 1045). |
|  | For user-specified impulses, you must specify the name of the vector/matrix containing the impulses using the "fname = " option. |
|  | The option "imp = mlechol" is provided for backward compatibility with EViews 3.x and earlier. |
| fname $=$ name | Specify name of vector/matrix containing the impulses. The vector/matrix must have $k$ rows and 1 or $k$ columns, where $k$ is the number of endogenous variables. |
| $\mathrm{se}=\arg$ | $\begin{aligned} & \text { Standard error calculations: "se = a" (analytic), } \\ & \text { "se = mcarlo" (Monte Carlo), "se = boot" (bootstrap). } \end{aligned}$ |
|  | If selecting Monte Carlo or bootstrap, you must specify the number of replications with the "rep = " option. |
|  | Note the following: |
|  | (1) Analytic standard errors are currently not available for (a) VECs and (b) structural decompositions identified by long-run restrictions. The "se = a " option will be ignored for these cases. |
|  | (2) Monte Carlo standard errors are currently not available for (a) VECs and (b) structural decompositions. The "se = mcarlo" option will be ignored for these cases. |
|  | (3) VECs only compute bootstrap standard errors so this option will be ignored. |
| rep $=$ integer | Number of Monte Carlo or bootstrap replications to be used in computing the standard errors. Must be used with the "se = mcarlo" and "se = boot" options. |
| $\begin{aligned} & \text { bs =arg (default } \\ & =\text { "hp") } \end{aligned}$ | Bootstrap method: "sp" (standard percentile), "hp" (Hall's percentile), "hs" (Hall's studentized), "ku" (Killian's unbiased). |
| dbsrep (default $=$ 499) | Number of double bootstrap replications. Must be used with the "bs = hs" and "bs = ku" options unless the "fdb" option is specified, in which case this option will be ignored. |


| fdb | Approximate the double bootstrap computation using fast double bootstrap routines. |
| :---: | :---: |
| $\begin{aligned} & \text { cilevels = arg } \\ & (\text { default }= \\ & " 0.95 ") \end{aligned}$ | Confidence interval coverage: space limited list of numbers between 0 and 1 . |
| uselines | Use lines instead of shading for confidence intervals. |
| matbys = name | Save responses ordered by shocks (impulses) in a named matrix. The first column is the response of the first variable to the first shock, the second column is the response of the second variable to the first shock, and so on. The response and shock orderings correspond to the ordering of variables in the VAR. |
| matbyr $=$ name | Save responses ordered by response series in a named matrix. The first column is the response of the first variable to the first shock, the second column is the response of the first variable to the second shock, and so on. The response and shock orderings correspond to the ordering of variables in the VAR. |
| smat $=$ name | Save responses ordered by shocks (impulses) in a named matrix (akin to the "matbys = " option). The shocks and responses are ordered according to the user-specified order given by the "@ shock_series" and "@ ordering_series" specifications. |
| $\mathrm{rmat}=$ name | Save responses ordered by response series in a named matrix (akin to the "matbyr =" option). The shocks and responses are ordered according to the user-specified order given by the "@ shock_series" and "@ ordering_series" specifications. |
| cimat $=$ name | Save matrix consisting of confidence intervals (lower-upper pairs for each impulse-response combination). |
| rcimat $=$ name | Save matrix consisting of impulse responses and associated confidence intervals (lower-upper pairs for each impulseresponse combination). |
| prompt | Force the dialog to appear from within a program. |
|  | Print the results. |

## BVAR Options

$$
\begin{array}{ll}
\text { bvartype = arg } & \text { Impulse method: Bayesian sampling ("bayes"), classical } \\
\text { (default }= & \text { impulse response analysis using the posterior residual } \\
\text { "bayes") } & \text { covariance matrix ("classpost"), classical impulse response } \\
& \begin{array}{l}
\text { analysis using the empirical residual covariance matrix } \\
\text { ("classemp"). }
\end{array}
\end{array}
$$

If you are using Bayesian sampling, the following Bayesian options are available:

```
draws = integer Number of draws.
(default =
100000)
burn = arg Proportion of initial draws to discard.
(default=0.1)
seed = integer Random number seed.
dropunstable Drop any draws that produce unstable coefficients.
dgraph Produce distribution graphs.
page = arg Store the individual draws in a new page.
```


## BTVCVAR Options

| starts $=$ arg | Set impulse dates. For multiple dates, enter a space-delim- <br> ited list of values surrounded by quotation marks, e.g., <br> "1980q1 2000q1 2020q1". |
| :--- | :--- |
| usemean | Use posterior mean as the point estimate. The posterior <br> median is used if usemean is not included in the options |
| list. |  |

## Examples

```
var varl.ls 1 4 m1 gdp cpi
var1.impulse(10,m) gdp @ m1 gdp cpi
```

The first line declares and estimates a VAR with three variables. The second line displays multiple graphs of the impulse responses of GDP to shocks to the three series in the VAR using the ordering as specified in VAR1.

```
var1.impulse(10,m) gdp @ m1 @ cpi gdp m1
```

displays the impulse response of GDP to a one standard deviation shock in M1 using a different ordering.

## Cross-references

See Chapter 44. "Vector Autoregression and Error Correction Models," on page 835 of User's Guide II for a discussion of impulse responses in VARs.

See also : :vdecomp (p. 276).

| Is | Equation Methods |
| :--- | :--- |

Estimation by linear or nonlinear least squares regression.
When the current workfile has a panel structure, is also estimates cross-section weighed least squares, feasible GLS, and fixed and random effects models.

## Syntax

eq_name.ls(options) y x1 [x2 x3 ...]
eq_name.ls(options) specification
For linear specifications, list the dependent variable first, followed by a list of the independent variables. Use a "C" if you wish to include a constant or intercept term; unlike some programs, EViews does not automatically include a constant in the regression. You may add AR, MA, SAR, and SMA error specifications, a D fractional differencing term, and PDL specifications for polynomial distributed lags. If you include lagged variables, EViews will adjust the sample automatically, if necessary.

Both dependent and independent variables may be created from existing series using standard EViews functions and transformations. EViews treats the equation as linear in each of the variables and assigns coefficients $\mathrm{C}(1), \mathrm{C}(2)$, and so forth to each variable in the list.

Linear or nonlinear single equations may also be specified by explicit equation. You should specify the equation as a formula. The parameters to be estimated should be included explicitly: " $\mathrm{C}(1)$ ", " $\mathrm{C}(2)$ ", and so forth (assuming that you wish to use the default coefficient vector "C"). You may also declare an alternative coefficient vector using coef and use these coefficients in your expressions.

## Options

## Non-Panel LS Options

indicator
$\mathrm{w}=\arg$
wtype = arg (default = "istdev")
wscale $=$ arg

Z
optmethod $=\arg$

Include indicator saturation detection as part of estimation routine.

Weight series or expression.
Note: we recommend that, absent a good reason, you employ the default settings Inverse std. dev. weights ("wtype = istdev") with EViews default scaling ("wscale = eviews") for backward compatibility with versions prior to EViews 7.

Weight specification type: inverse standard deviation ("istdev"), inverse variance ("ivar"), standard deviation ("stdev"), variance ("var").

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.

Turn off backcasting in ARMA models where "arma = cls".
Optimization method for nonlinear least squares and ARMA: "bfgs" (BFGS); "newton" (Newton-Raphson), "opg" or "bhhh" (OPG or BHHH), "kohn" (Kohn-Ansley for ARMA estimated by ML or GLS), or "legacy" (EViews legacy for nonlinear least squares and ARMA estimated by CLS).
Gauss-Newton is the default method.

```
optstep = arg Step method for nonlinear least squares and ARMA: "mar
    quardt" (Marquardt); "dogleg" (Dogleg); "linesearch" (Line
    search).
    Marquardt is the default method.
cov = arg Covariance method: "ordinary" (default method based on
    inverse of the estimated information matrix), "huber" or
    "white" (Huber-White sandwich method available for non-
    linear least squares or ARMA estimated by CLS), "hac"
    (Newey-West HAC, available for nonlinear least squares or
    ARMA estimated by CLS)..
covinfo = arg Information matrix method: "opg" (OPG); "hessian"
        (observed Hessian).
    (Applicable when non-legacy "optmethod = ".)
```

| nodf | Do not perform degree of freedom corrections in computing coefficient covariance matrix. The default is to use degree of freedom corrections. |
| :---: | :---: |
| $\begin{aligned} & \text { covlag }=\text { arg } \\ & (\text { default }=1) \end{aligned}$ | Whitening lag specification: integer (user-specified lag value), "a" (automatic selection). |
| $\begin{aligned} & \text { covinfosel = arg } \\ & (\text { default }=\text { "aic") } \end{aligned}$ | 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 $T^{1 / 3}$. |
| $\begin{aligned} & \text { covkern }=\text { arg } \\ & \text { (default }=\text { "bart") } \end{aligned}$ | Kernel shape: "none" (no kernel), "bart" (Bartlett, default), "bohman" (Bohman), "daniell" (Daniel), "parzen" (Parzen), "parzriesz" (Parzen-Riesz), "parzgeo" (ParzenGeometric), "parzcauchy" (Parzen-Cauchy), "quadspec" (Quadratic Spectral), "trunc" (Truncated), "thamm" (Tukey-Hamming), "thann" (Tukey-Hanning), "tparz" (Tukey-Parzen). |
| $\begin{aligned} & \text { covbw }=\text { arg } \\ & \text { (default }=\text { "fixednw" } \end{aligned}$ | Kernel Bandwidth: "fixednw" (Newey-West fixed), "andrews" (Andrews automatic), "neweywest" (NeweyWest 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. |
| $\mathrm{m}=$ integer | Set maximum number of iterations. |
| $\mathrm{c}=$ scalar | Set convergence criterion. The criterion is based upon the maximum of the percentage changes in the scaled coefficients. The criterion will be set to the nearest value between $1 \mathrm{e}-24$ and 0.2. |
| $\operatorname{arma}=\arg$ | ARMA estimation method: "ml" (maximum likelihood); "gls" (generalized least squares), "cls" (conditional least squares). |
|  | Not applicable to ARFIMA models which always estimate using maximum likelihood. |
| armastart $=$ arg | ARMA coefficient starting values: "auto" (automatic) "fixed" (legacy EViews fixed); "random" (random draw); "user" (user-specified). |
|  | Applicable when "arma $=\mathrm{ml}$ " or "arma $=$ gls". |

Use the current coefficient values in estimator coefficient vector as starting values for equations specified by list with AR or MA terms when "arma = cls" (see also param (p. 455) of the Command and Programming Reference).

| $\mathrm{s}=$ number | Determine starting values for equations specified by list with AR or MA terms when "arma = cls". Specify a number between zero and one representing the fraction of preliminary least squares estimates computed without AR or MA terms to be used. Note that out of range values are set to " $s=1$ ". Specifying " $s=0$ " initializes coefficients to zero. By default EViews uses " $\mathrm{s}=1$ ". <br> Does not apply to coefficients for AR and MA terms which are set to EViews determined default values. |
| :---: | :---: |
| numericderiv / -numericderiv | [Do / do not] use numeric derivatives only. If omitted, EViews will follow the global default. |
| fastderiv / -fastderiv | [Do / do not] use fast derivative computation. If omitted, EViews will follow the global default. <br> Available only for legacy estimation ("optmeth = legacy"). |
| $\operatorname{cov}=\arg$ | Covariance method: "ordinary" (default method based on inverse of the estimated information matrix), "huber" or "white" (Huber-White sandwich method available for nonlinear least squares or ARMA estimated by CLS), "hac" (Newey-West HAC, available for nonlinear least squares or ARMA estimated by CLS)., "hc" (extended heteroskedasticity consistent), "hcuser" (user-specified heteroskedasticity), "cr" (cluster robust). <br> The extended "hc" methods are only available for linear specifications. |
| $\begin{aligned} & \text { hctype }=\arg (\text { default } \\ & \text { "hc2") } \end{aligned}$ | Extended heteroskedasticity consistent method: "hc0" (no d.f. adjustment), "hc1" (d.f. adjusted), "hc2", "hc3", "hc4", "hc4m", "hc5", when "cov = hc". |
| userwt $=$ arg | Name of series containing user-diagonal weights (if "cov = hcuser") |
| $\begin{aligned} & \text { crtype = arg (default } \\ & \text { "cr1") } \end{aligned}$ | Cluster robust weighting method: "cr0" (no finite sample correction), "cr1" (finite sample correction), "hc2", "hc3", "hc4", "hc4m", "hc5", when "cov = cr". |
| crname $=$ arg | Cluster robust series name, when "cov=cr". |
| $\begin{aligned} & \mathrm{k}=\arg \\ & (\text { default }=0.7) \end{aligned}$ | Parameter for "cov = hc, hctype = hc5" or "cov = cr, crtype = cr5" . |
| $\begin{aligned} & \mathrm{k} 1=\text { arg } \\ & (\text { default }=1.0) \end{aligned}$ | Parameter for "cov = hc, hctype $=$ hc4m" or "cov = cr, crtype $=$ cr4m". |


| k2 = arg <br> $($ default $=1.5)$ | Parameter for "cov = hc, hctype = hc4m" or "cov = cr, <br> crtype $=$ cr4m". |
| :--- | :--- |
| showopts / | [Do / do not] display the starting coefficient values and <br> estimation options in the estimation output. |
| -showopts |  |
| coef = arg | Specify the name of the coefficient vector (if specified by <br> list); the default behavior is to use the "C" coefficient vec- <br> tor. |
| prompt | Force the dialog to appear from within a program. <br> p |
| Print estimation results. |  |

Note: not all options are available for all equation methods. See the User's Guide II for details on each estimation method.

## Non-Panel Indicator Saturation Options

For use if "indicator" option is specified.

| noiis | Do not search for impulse terms. |
| :--- | :--- |
| sis | Search for step-shift terms. |
| trend | Search for trend terms. |
| pval $=$ number <br> $($ default $=0.05)$ | Set the terminal condition $p$-value used to determine the <br> stopping point of each search path |
| nolm | Do not perform AR LM diagnostic test. |
| arpval $=$ number <br> $($ default $=0.025)$ | Set $p$-value used in AR LM diagnostic test. |
| arlags $=$ int $($ default <br> $=1)$ | Set number of lags used in AR LM diagnostic test. |
| noarch | Do not perform ARCH LM diagnostic test. |
| archpval $=$ number <br> $($ default $=0.025)$ | Set $p$-value used in ARCH LM diagnostic test. |
| archlags $=$ int <br> $($ default $=1)$ | Set number of lags used in ARCH LM diagnostic test. |
| nojb | Do not perform Jarque-Bera normality diagnostic test. |
| jbpval $=$ number <br> $($ default $=0.025)$ | Set $p$-value used in Jarque-Bera normality diagnostic test. |
| nopet | Do not perform Parsimonious Encompassing diagnostic <br> test. |
| petpval $=$ number <br> $($ default $=0.025)$ | Set $p$-value used in Parsimonious Encompassing diagnostic <br> test. |

$\left.\begin{array}{ll}\text { nogum } & \begin{array}{l}\text { Do not include the general model as a candidate for model } \\ \text { selection. }\end{array} \\ \text { noempty } & \begin{array}{l}\text { Do not include the empty model as a candidate for model } \\ \text { selection. }\end{array} \\ \text { ic }=a r g & \begin{array}{l}\text { Set the information criterion used in model selection: "AIC" } \\ \text { (Akaike information criteria, default), "BIC" (Schwarz }\end{array} \\ \text { information criteria), "HQ" (Hannan-Quin criteria). }\end{array}\right\}$

## Panel LS Options

$\mathrm{cx}=\arg$
per $=\arg$
$\mathrm{wgt}=\arg$
$\operatorname{cov}=\arg$
keepwgts

$$
\begin{aligned}
& \text { rancalc }=\text { arg } \\
& \text { (default }=" \mathrm{sa"})
\end{aligned}
$$

nodf

Cross-section effects: (default) none, fixed effects ("cx=f"), random effects ("cx=r").

Period effects: (default) none, fixed effects ("per $=\mathrm{f}$ "), random effects ("per = r").

GLS weighting: (default) none, cross-section system weights ("wgt = cxsur"), period system weights ("wgt = persur"), cross-section diagonal weighs ("wgt = cxdiag"), period diagonal weights ("wgt = perdiag").

Coefficient covariance method: (default) ordinary, White cross-section system (period clustering) robust ("cov = cxwhite" or "cov = percluster"), White period system (cross-section clustering) robust ("cov= perwhite" or "cov = cxcluster"), White heteroskedasticity robust ("cov = stackedwhite"), White two-way cluster robust (cov = bothcluster"), Cross-section system robust/PCSE ("cov = cxsur"), Period system robust/PCSE ("cov = persur"), Cross-section heteroskedasticity robust/PCSE ("cov = cxdiag"), Period heteroskedasticity robust/PCSE ("cov = perdiag").

Keep full set of GLS weights used in estimation with object, if applicable (by default, only small memory weights are saved).

Random component method: Swamy-Arora ("rancalc $=$ sa"), Wansbeek-Kapteyn ("rancalc $=$ wk"), WallaceHussain ("rancalc $=$ wh").

Do not perform degree of freedom corrections in computing coefficient covariance matrix. The default is to use degree of freedom corrections.

| coef $=$ arg | Specify the name of the coefficient vector (if specified by list); the default behavior is to use the " $C$ " coefficient vector. |
| :---: | :---: |
| $\begin{aligned} & \text { iter }=\arg (\text { default }= \\ & \text { "onec") } \end{aligned}$ | Iteration control for GLS specifications: perform one weight iteration, then iterate coefficients to convergence ("iter = onec"), iterate weights and coefficients simultaneously to convergence ("iter = sim"), iterate weights and coefficients sequentially to convergence ("iter = seq"), perform one weight iteration, then one coefficient step ("iter = oneb"). <br> Note that random effects models currently do not permit weight iteration to convergence. |
| unbalsur | Compute SUR factorization in unbalanced data using the subset of available observations for a cluster. |
| $\mathrm{m}=$ integer | Set maximum number of iterations. |
| $\mathrm{c}=$ scalar | Set convergence criterion. The criterion is based upon the maximum of the percentage changes in the scaled coefficients. The criterion will be set to the nearest value between $1 \mathrm{e}-24$ and 0.2 . |
| s | Use the current coefficient values in estimator coefficient vector as starting values for equations specified by list with AR terms (see also param (p. 455) of the Command and Programming Reference). |
| $\mathrm{s}=$ number | Determine starting values for equations specified by list with AR terms. Specify a number between zero and one representing the fraction of preliminary least squares estimates computed without AR terms to be used. Note that out of range values are set to " $s=1$ ". Specifying " $s=0$ " ini tializes coefficients to zero. By default EViews uses " $\mathrm{s}=1$ " Does not apply to coefficients for AR terms which are instead set to EViews determined default values. |
| numericderiv / -numericderiv | [Do / do not] use numeric derivatives only. If omitted, EViews will follow the global default. |
| fastderiv / -fastderiv | [Do / do not] use fast derivative computation. If omitted, EViews will follow the global default. |
| showopts / <br> -showopts | [Do / do not] display the starting coefficient values and estimation options in the estimation output. |
| prompt | Force the dialog to appear from within a program. |
|  | nt basic estimation results. |

## Examples

```
equation eq1.ls m1 c uemp inf(0 to -4) @trend(1960:1)
```

estimates a linear regression of M1 on a constant, UEMP, INF (from current up to four lags), and a linear trend.

```
equation eq2.ls(z) d(tbill) c inf @seas(1) @seas(1)*inf ma(2)
```

regresses the first difference of TBILL on a constant, INF, a seasonal dummy, and an interaction of the dummy and INF, with an MA(2) error. The " $z$ " option turns off backcasting.

```
coef(2) beta
param beta(1) . 2 beta(2) . 5 c(1) 0.1
equation eq3.ls(cov=white) q = beta(1)+beta(2)*(l^c(1) + k^(1-
    c(1)))
```

estimates the nonlinear regression starting from the specified initial values. The "cov = white" option reports heteroskedasticity consistent standard errors.

```
equation eq4.ls r = c(1)+c(2)*r(-1)+div(-1)^c(3)
sym betacov = eq4.@cov
```

declares and estimates a nonlinear equation and stores the coefficient covariance matrix in a symmetric matrix called BETACOV.

```
equation eq5.ls(cx=f, per=f) n w k ys c
```

estimates the equation EQ5 in the panel workfile using both cross-section and period fixed effects.

```
equation eq6.ls(cx=f, wgt=cxdiag) n w k ys c
```

estimates the equation EQ6 in a panel workfile with cross-section weights and fixed effects.

## Cross-references

Chapter 19. "Basic Regression Analysis," on page 5 and Chapter 20. "Additional Regression Tools," on page 23 of the User's Guide II discuss the various regression methods in greater depth.

Chapter 16. "Special Expression Reference," on page 601 of the Command and Programming Reference describes special terms that may be used in ls specifications.

See Chapter 17. "Panel Estimation," on page 645 of the User's Guide II for a discussion of panel equation estimation.

## pageload Object Container, Data, and File Commands

Load one or more new pages in the default workfile.

## Syntax


pageload(options) source_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@selectif condition]
pageload(options)source_description table_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@selectif condition]

The basic syntax for pageload follows that of wfopen (p. 278). The difference between the two commands is that pageload creates a new page in the default workfile, rather than opening or creating a new workfile. If a page is loaded with a name that matches an existing page, EViews will rename the new page to the next available name (e.g., "INDIVID" will be renamed "INDIVID1".

If a workfile is provided as the source file, EViews will, by default, open all pages in the source workfile. Specific pages may be loaded by providing their names.

## Examples

```
pageload "c:\my documents\data\panel1"
```

loads the workfile PANEL1.WF1 from the specified directory. All of the pages in the workfile will be loaded as new pages into the current workfile.

```
pageload f.wf1 mypage
```

loads the page "mypage" in the workfile F.WF1 located in the default directory.
See the extensive set of examples in wfopen (p. 278).

## Cross-references

See "Creating a Page by Loading a Workfile or Data Source" on page 90 of User's Guide I for discussion.

See also wfopen (p. 278) and pagecreate (p. 450).

## pagesave

Object Container, Data, and File Commands

Save the active page in the default workfile as an EViews workfile (.WF1 file) or as a foreign data source.

## Syntax

pagesave(options) [path $\backslash$ filename
pagesave(options) source_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
pagesave(options) source_description table_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]

The command saves the active page in the specified directory using filename. By default, the page is saved as an EViews workfile, but options may be used to save all or part of the page in a foreign file or data source.

When saving to a foreign data file, the basic specification consists of a "type = " option and source_description and table_description arguments which specify the format of the foreign data file. See below for details on source_description and table_description.

The remaining optional elements specify the actual elements to be saved.

## Options

| type = arg, $\mathrm{t}=$ arg | Optional type specification: (see table below). <br> Note that ODBC support is provided only in the EViews <br> Enterprise Edition. |
| :--- | :--- |
| mode = arg | Specify whether to create a new file, overwrite an existing <br> file, or update an existing file. arg may be "create" (create <br> new file only; error on attempt to overwrite) or "update" <br> (update an existing file, only overwriting the area specified <br> by the range = table_description). <br> If a "mode =" option is not provided, EViews will create a <br> new file, unless the file already exists in which case it will <br> overwrite it. |
| Note that the "mode = update" option is only available for: <br> 1) Excel versions through 2003, if Excel is installed, and 2) <br> Excel 2007 (xml). |  |
| maptype =arg $\quad$Write selected maps as: numeric (" n "), character ("c"), <br> both numeric and character ("b"). |  |


| nomapval | Do not write mapped values for series with attached value <br> labels (the default is to write mapped values) |
| :--- | :--- |
| noid | Do not write observation identifiers to foreign data files (by <br> default, EViews will include a column with the date or <br> observation identifier). |
| nonames | Do not export variable names. |
| $\operatorname{attr}$ | Include object attributes (if the output type supports it). <br> When specified, the first column will contain attribute <br> names and each attribute value will be displayed after the <br> name row. |

The following table summaries the various foreign formats, along with the corresponding "type = " keywords:

|  | Type Keywords | Supports Attributes |
| :--- | :---: | :---: |
| Access | "access" |  |
| Aremos-TSD | "a", "aremos", "tsd" |  |
| Binary | "binary" |  |
| dBASE | "dbase" |  |
| Excel (through 2003) | "excel" | Yes |
| Excel 2007 (xml) | "excelxml" | Yes |
| EViews Workfile | --- |  |
| Gauss Dataset | "gauss" |  |
| GiveWin/PcGive | "g", "give" |  |
| HTML | "html" |  |
| JSON** | json |  |
| Lotus 1-2-3 | "lotus" |  |
| ODBC Dsn File | "odbc" |  |
| ODBC Data Source | "dos", "microtsp" |  |
| MicroTSP Workfile | "mac" |  |
| MicroTSP Mac Workfile | "r", "rats" |  |
| RATS 4.x | "l", "trl" |  |
| RATS Portable / TROLL | "sasprog" |  |
| SAS Program | "sasxport" |  |
| SAS Transport | "spss" |  |
| SPSS | "spssport" |  |
| SPSS Portable |  |  |


| Stata (Version 7 Format) | "stata" |  |
| :--- | :---: | :---: |
| Tableau Data Extract | "tde" |  |
| Text / ASCII | "text" | Yes |
| TSP Portable | "t", "tsp" |  |

Note that if you wish to save your Excel 2007 XML file with macros enabled, you should specify the explicit filename extension ".XLSM".

## Foreign Data Descriptions

When saving to a foreign data format the base specification consists of a basic specification of source_description and table_description which specify the exact details of the format.

The command for saving as foreign data formats is
pagesave(options) source_description [table_description] [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
where the syntax of the table_description and variables_description differs slightly depending on the type of file.

- Note that the JSON type will ignore any @keep, @drop, and @smpl arguments.


## Excel Files

The base syntax for writing Excel files is:
pagesave(options) source_description [table_description]
where source_description is the path and name of the Excel file to be saved, and where the following table_description elements may be employed:

- "range $=$ arg", where $\arg$ is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].
If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.
- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.


## HTML Files

The base syntax for saving HTML files is:
pagesave(options) source_description [table_description]
where source_description is the path and name of the file to be saved, and where the following table_description element may be employed:

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.


## Text and Binary and Other Files

The base syntax for saving other files is:
pagesave(options) source_description
where source_description is the path and name of the file to be saved.

## Examples

EViews Workfile Examples

```
pagesave new_wf
```

saves the current EViews workfile page as "New_wf.WF1" in the default directory.

```
pagesave "c:\documents and settings\my data\consump"
```

saves the current workfile page as "Consump.WF1" in the specified path.

```
pagesave macro @keep gdp unemp
```

saves the two series GDP and UNEMP in a separate workfile, "macro.WF1" in the default directory.

```
pagesave macro @dropmap gdp*
```

saves all of the series in the current workfile, other than those that match the name pattern "gdp*" in a workfile, "macro.WF1" in the default directory.

The command:

```
pagesave "<mydropboxdrive>"\folder\nipa.wf1"
```

will save the file to the cloud location MYDROPBOXDRIVE.

## Foreign Data Examples

```
pagesave(type=excelxml, mode=update) macro.xlsx
```

saves the current workfile page as a modern Excel ".XLSX" file.

```
pagesave(type=excelxml, mode=update) macro.xlsx range="Sheet2!a1"
    byrow @keep gdp unemp
```

will save the two series GDP and UNEMP into the existing Excel file "macro.XLSX", specifying that the series should be written by row, starting in cell A1 on sheet Sheet2.

To save the latter file in a macro-enabled Excel 2007 file, you should specify the explicit filename extension ".XLSM":

```
pagesave(type=excelxml, mode=update) macro.xlsm range="Sheet2!a1"
    byrow @keep gdp unemp
```

Alternately,

```
pagesave(type=excelxml, noid) macro.xlsx range="Sheet2!a1"
```

will save the current workfile page as the Excel file "macro.XLSX" but will not include a column of dates.

If you wish to save a column of dates in a specific date format, you can do so by first creating an alpha series in the workfile with the specified format, then saving the file with the "noid" option including that alpha series:

```
alpha mydates = @datestr(@date, "YYYY-MM-DD")
pagesave(type=excelxml, noid) macro.xlsm range="Sheet2!a1" @keep
    mydates gdp unemp
```

Will save the series GDP and UNEMP into the Excel file "macro.XLSM" along with a date series with the format "YYYY-MM-DD".

## Cross-references

See "Saving a Workfile" on page 77 in the User's Guide I.
See also wfopen (p. 278) and wfsave (p. 293).

## resize

Matrix Procs

## Resize the matrix object.

## Syntax

matrix_name.resize rows cols

## Examples

```
mat1.resize 3 5
```

resizes the matrix MAT1 to 3 rows and 5 columns, retaining the contents of any existing elements and initializing new elements to 0 .

## Cross-references

| resize | Svector Procs |
| :--- | :--- |

Resize the svector object.
Syntax
svector_name.resize rows

## Examples

```
svec1.resize 20
```

resizes the svector SVEC1 to 20 rows, retaining the contents of any existing elements and initializing new elements to the empty string "".

## Cross-references

| resize | Sym Procs |
| :--- | :--- |

Resize the sym object.

## Syntax

sym_name.resize rows/cols
Examples
sym1.resize 20
resizes the sym SYM1 to 20 rows/columns, retaining the contents of any existing elements and initializing new elements to 0 .

## Cross-references

| resize | Vector Procs |
| :--- | :--- |

Resize the vector object.

## Syntax

vector_name.resize rows

## Examples

vec1.resize 20
resizes the vector VEC1 to 20 rows, retaining the contents of any existing elements and initializing new elements to 0 .

Cross-references

| save | Table Procs |
| :--- | :--- |

Save table to disk as an Excel 2007 XLSX, CSV, tab-delimited ASCII text, RTF, HTML, Enhanced Metafile, LaTeX, PDF, or Markdown file.

## Syntax

table_name.save(options) [path<br>file_name
Follow the keyword with a name for the file. file_name may include the file type extension, or the file type may be specified using the " $\mathrm{t}=$ " option.

If an explicit path is not specified, the file will be stored in the default directory, as set in the File Locations global options.

The base syntax for writing Excel files is:
table_name.save(options) [path \Jfile_name [table_description]
where the following table_description elements may be employed:

- "range = arg", where arg is top left cell of the destination Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].

If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.

## Options

$\mathrm{t}=$ file_type

(default = "csv") | Specifies the file type, where file_type may be one of: |
| :--- |
| "excelxml" (Excel 2007 (xml)),"csv", (CSV - comma-sepa- |
| rated), "rtf" (Rich-text format), "txt" (tab-delimited text), |
| "html" (HTML - Hypertext Markup Language), "emf" |
| (Enhanced Metafile), "pdf" (PDF - Portable Document For- |
| mat), "tex" (LaTeX), or "md" (Markdown). |
| Files will be saved with the ".xlsx", ".csv", ".rtf", ".txt", |
| ".htm", ".emf", ".pdf", ".tex", or ".md" extensions, respec- |
| tively. |

## PDF Options

landscape

```
size = arg
(default = "letter")
```

width $=$ number
(default $=8.5$ )
height $=$ number $\quad$ Page height in inches if "size = custom".
(default = 11)
leftmargin $=$ number
(default $=0.5$ )
rightmargin $=$ number $\quad$ Right margin width in inches.

Save in landscape mode (the default is to save in portrait mode).

Page size: "letter", "legal", "a4", and "custom".

Page width in inches if "size = custom".

Left margin width in inches.

```
topmargin = number Top margin width in inches.
(default = 1)
bottommargin = Bottom margin width in inches.
number (default = 1)
```


## LaTeX Options

texspec / -texspec
[Include / Do not include] the full LaTeX documentation specification in the LaTeX output. The default behavior is taken from the global default settings.

## Excel Options

$$
\begin{array}{ll}
\text { mode }=\text { arg } & \begin{array}{l}
\text { Specify whether to create a new file, overwrite an existing } \\
\text { file, or update an existing file. arg may be "create" (create } \\
\text { new file only; error on attempt to overwrite) or "update" } \\
\text { (update an existing file, only overwriting the area specified }
\end{array} \\
\text { by the range = table_description). } \\
\text { If the "mode = " option is not used, EViews will create a } \\
\text { new file, unless the file already exists in which case it will } \\
\text { overwrite it. }
\end{array}
$$

## Examples

The command:
tab1.save mytable
saves TAB1 to a CSV file named "mytable.CSV" in the default directory.

```
tab1.save(t=csv, n="NaN") mytable
```

saves TAB1 to a CSV (comma separated value) file named "mytable.csv" and writes all NA values as "NaN".

```
tab1.save(r=B2:C10, t=html, s=50) mytable
```

saves from data from the second row, second column, to the tenth row, third column of TAB1 to a HTML file named "mytable.HTM" at half of the original size.

```
tab1.save(f, n=".", r=B) mytable
```

saves the contents of the second column of the table in full precision to a CSV file named "mytable.CSV", and writes all NA values as ".".

```
tab1.save(t=excelxml, cellfmt=eviews, mode=update) mytable
    range=Country!b5
```

adds TAB1 to the preexisting "mytable.XLSX" Excel file to the "Country" sheet at cell B5, where the cell colors and fonts in TAB1 will also be copied.

## Cross-references

For additional discussion of table commands see Chapter 3. "Working with Tables and Spreadsheets," on page 57 of the Command and Programming Reference.

See Chapter 17. "Table and Text Objects," beginning on page 873 of User's Guide I for a discussion of tables.

## setcollabels

## Matrix Procs

Set the column labels in a matrix object.

## Syntax

matrix_name.setcollabels label1 label2 label3....
Follow the setcollabels command with a space delimited list of column labels. Note that each column label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are columns, EViews will keep the corresponding default column names ("C11", "C12", etc...).

## Examples

mat1.setcollabels USA UK FRANCE
sets the column label for the first column in matrix MAT1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

## setcollabels

Rowvector Procs

Set the column labels in a rowvector object.

## Syntax

rowvector_name.setcollabels label1 label2 label3....

Follow the setcollabels command with a space delimited list of column labels. Note that each column label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are columns, EViews will keep the corresponding default column names ("C11", "C12", etc...).

## Examples

```
rvec1.setcollabels USA UK FRANCE
```

sets the column label for the first column in rowvector MAT1 to USA, the second to UK, and the third to FRANCE.

Cross-references

| setcollabels | Svector Procs |
| :--- | :--- |

Set the column label in a svector object.

## Syntax

svector_name.setcollabels label1
Follow the setcollabels command with the column label. Note that the column label should not contain spaces unless it is enclosed in quotes.

## Examples

```
svec1.setcollabels MyResults
```

sets the column label to "MyResults".
Cross-references

| setcollabels | Sym Procs |
| :--- | :--- |

Set the column labels in a sym object.

## Syntax

sym_name.setcollabels label1 label2 label3....
Follow the setcollabels command with a space delimited list of column labels. Note that each column label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are columns, EViews will keep the corresponding default column names ("C11", "C12", etc...).

Examples
sets the column label for the first column in symmetric matrix SYM1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

| setcollabels | Vector Procs |
| :--- | :--- |

Set the column label in a vector object.
Syntax
vector_name.setcollabels label1
Follow the setcollabels command with the column label. Note that the column label should not contain spaces unless it is enclosed in quotes.

## Examples

vec1.setcollabels MyResults
sets the column label to "MyResults".
Cross-references

| setelem | Graph Procs |
| :--- | :--- |

Set individual line, bar and legend options for each series in the graph.

## Syntax

graph_name.setelem(graph_elem) argument_list
where graph_elem is the identifier for the graph element whose options you wish to modify:

| integer | Index for graph element (for non-boxplot graphs). For <br> example, if you provide the integer "2", EViews will modify <br> the second line in the graph. |
| :--- | :--- |
| box_elem | Boxplot element to be modified: box ("b"), median <br> ("med"), mean ("mean"), near outliers ("near" or "no"), <br> far outliers ("far" or "fo"), whiskers ("w"), staples ("s"). <br> For boxplot graphs only. |

The argument list for setelem may contain one or more of the following:
symbol（arg）

| （1）circle | $\bigcirc \bigcirc-0-0-$ |
| :---: | :---: |
| （2）filledcircle | － －－－－ |
| （3）transcircle | （1）$\bigcirc \bigcirc \bigcirc$ |
| （4）star | 米 米 |
| （5）diagcross | $\times \times \rightarrow \rightarrow$ |
| （6）cross |  |
| （7）filledsquare | ■ ■－ |
| （8）square | ㅁ ロ－－－－ |
| （9）filledtriup |  |
| （10）triup | $\triangle \Delta-4-4$ |
| （11）filledtridown | F 7 － 7 |
| （12）tridown | $\nabla \nabla-7-7-$ |
| （13）obslabel | Data Label |
| （14）dotobslabel | －Data Label |
| （15）dotobslabelcircle | $\bigcirc$ Data Label |
| （16）dash | －－ |
| （17）downbrack |  |
| （18）upbrack | ப ப |

Sets the drawing symbol：arg can be an integer from 1－18， or one of the matching keywords．＂obslabel＂and＂dotob－ slabel＂，and＂dotobslabelcircle＂use the observation label as the symbol．
Selecting a symbol automatically turns on symbol use．
The＂none＂option turns off symbol use．
symbolsize（arg）， symsize（arg）

Sets the symbol size．arg may be an integer between 1－8， where 1 is the smallest symbol and 8 is the largest，or one of the keywords：＂XS＂（X－Small），＂S＂（Small），＂M＂ （Medium），＂L＂（Large），＂XL＂（X－Large），＂2XL＂（2X－Large）， ＂3XL＂（3X－Large），＂4XL＂（4X－Large）．
linecolor(arg), lcolor(arg)
linewidth(n1), lwidth (n1)
linepattern(arg), lpat(arg)

Sets the line and symbol color. arg may be one of the predefined color keywords, or it may be specified using individual red-green-blue (RGB) components using the "@RGB" or "@HEX" functions. The arguments to the $@$ RGB function are a set of three integers from 0 to 255 , representing the RGB values of the color. The arguments to the "@HEX" function are a set of six characters representing the RGB values of the color in hexadecimal. Each two character set represents a red, green or blue component in the range ' 00 ' to ' FF '. For a description of the available color keywords see "Color definitions" on page 244.

Sets the line and symbol width: $n 1$ should be a number between ". 25 " and " 5 ", indicating the width in points.

Sets the line pattern to the type specified by arg. arg can be an integer from 112 or one of the matching keywords.
Note that the option interacts with the graph options for "color", "lineauto", "linesolid", "linepat" (see Graph::options
(1) solid
(2) dash1
(3) dash2
(4) dash3
(5) dash4
(6) dash5
(7) dash6
(8) dash7
(9) dash8
(10) dash9
(11) dash10 - -
(12) none (p. 340), for details). You may need to set the graph option for "linepat" to enable the display of line patterns. See

Graph: :options
(p. 340).

Note also that the patterns with index values 7-11 have been modified since version 5.0. In particular, the "none" option has been moved to position 12 .
The "none" option turns off lines and uses only symbols.
lineopacity $(\arg )$, Sets the line opacity to the value specified by arg. arg can be an value from 0 to 1 .
Setting the level to 0.0 will make the object completely transparent ( $0 \%$ opacity) while a value of 1.0 will make the object completely opaque ( $100 \%$ opacity).
fillcolor (arg), fcolor(arg)
fillgray( $n 1$ ), $\operatorname{gray}(n 1)$
fillhatch (arg), hatch (arg)

Sets the hatch characteristics for bars and pies: arg can be an integer from 17 , or one of the matching keywords.
Sets the gray scale for bars and pies: $n 1$ should be an integer from $1-15$ corresponding to one of the predefined gray scale settings (from lightest to darkest).

$$
15
$$ one of the predefined color keywords, or it may be specified using individual red-green-blue (RGB) components using the "@RGB" function or "@HEX" functions. The arguments to the @RGB function are a set of three integers from 0 to 255 , representing the RGB values of the color. The arguments to the "@HEX" function are a set of six characters representing the RGB values of the color in hexadecimal. Each two character set represents a red, green or blue component in the range ' 00 ' to ' FF '. For a description of the available color keywords see "Color definitions" on page 244.



Sets the fill opacity to the value specified by arg. arg can be an value from 0 to 1 .
Setting the level to 0.0 will make the object completely transparent ( $0 \%$ opacity) while a value of 1.0 will make the object completely opaque ( $100 \%$ opacity).
$\left.\begin{array}{ll}\text { preset }(n 1) & \begin{array}{l}\text { Sets line and fill characteristics to the specified EViews pre- } \\ \text { set values, where n1 is an integer from 1-30. Simultane- } \\ \text { ously sets "linecolor", "linepattern", "linewidth", } \\ \text { "symbol", "fillcolor", "fillgray", and "fillhatch" to the } \\ \text { EViews predefined definitions for graph element } n 1 .\end{array} \\ \text { When applied to boxplots, the line color of the specified } \\ \text { element will be applied to the box, whiskers, and staples. }\end{array}\right\}$

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special "@RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | $@ h e x(0000 f f)$ |
| :--- | :--- | :--- |
| red | $@ \operatorname{rgb}(255,0,0)$ | $@ h e x(f f 0000)$ |
| ltred | $@ \operatorname{rgb}(255,168,168)$ | $@ h e x(f f a 8 a 8)$ |
| green | $@ \operatorname{rgb}(0,128,0)$ | $@ h e x(008000)$ |
| black | $@ \operatorname{rgb}(0,0,0)$ | $@ h e x(000000)$ |
| white | $@ \operatorname{rgb}(255,255,255)$ | $@ h e x(f f f f f)$ |
| purple | $@ \operatorname{rgb}(128,0,128)$ | $@ h e x(800080)$ |
| range | @rgb $(255,128,0)$ | $@ h e x(f f 8000)$ |
| yellow | $@ \operatorname{rgb}(255,255,0)$ | $@ h e x(f f f f 00)$ |


| gray | $@ \operatorname{rgb}(128,128,128)$ | $@ \operatorname{hex}(808080)$ |
| :--- | :--- | :--- |
| ltgray | $@ \operatorname{rgb}(192,192,192)$ | $@ h e x(c 0 c 0 c 0)$ |

## Examples

```
graph1.setelem(2) lcolor(blue) lwidth(2) symbol(circle)
```

sets the second line of GRAPH1 to be a blue line of width 2 with circle symbols.

```
graph1.setelem(1) lcolor(blue)
graph1.setelem(1) linecolor(0, 0, 255)
```

are equivalent methods of setting the linecolor to blue.

```
graph1.setelem(1) fillgray(6)
```

sets the gray-scale color for the first graph element.
The lines:

```
graph1.setelem(1) scale(l)
graph1.setelem(2) scale(l)
graph1.setelem(3) scale(r)
```

create a dual scale graph where the first two series are scaled together and labeled on the left axis, and the third series is scaled and labeled on the right axis.

```
graph1.setelem(2) legend("gross domestic product")
```

sets the legend for the second graph element.

## Cross-references

See Chapter 16. "Graph Objects," on page 835 of User's Guide I for a discussion of graph options in EViews.

See also Graph: :axis (p. 317), : : datelabel (p. 137) and Graph: :options (p. 340).

| setfillcolor | Geomap Procs |
| :--- | :--- |

Define the fill (background) color used in geomap shapes using values in a series.

## Syntax

geomap_name.setfillcolor( $\mathbf{t}=$ type ) fillcolor_args
where:

```
type = arg Type of fill coloring for spreadsheet cells: "single" (single
    color), "posneg" (positive-negative single threshold),
    "range" (single range coloring), "hilo" (high-low-median),
    "custom" (custom coloring).
```


## General Arguments

To specify the series or expression whose values will determine the background color:

- mapser (spec)
where spec is a series name or expression.
To specify the minimum and maximum values where the coloring begins and ends:
- min(color_arg)
- max (color_arg)

To set the missing value (NA) background color:

- naclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the color defaults to "white".


## Type-specific Arguments

There are optional type-specific arguments that correspond to each of the type choices:

## Single color

To set the single background color:
clr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the color defaults to "white".

## Positive-negative single threshold

You may set the color for both the non-negative (posclr) and the negative (negclr) values posclr(color_arg)
negclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the nonnegative color defaults to "white" and the negative color defaults to light-red.

## Single range

To specify the range, you must specify the range endpoints:
range(lower_val, upper_val[, range_def)
where range_def specifies the range endpoints:

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

By default, the range will be open on the lower and closed on the upper threshold limits.
You should provide a color specification for the inside range color (inclr) and outside range color (outclr):
inclr(color_arg)
outclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the interior color defaults to light-red, and the exterior defaults to white.
High-Low-Median
When "type = hilo" you may specify the high, low, and median coloring values:
highclr(color_arg)
lowclr(color_arg)
medianclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the colors default to light-red.

## Custom

When "type = custom" you may specify custom coloring options.
You may optionally set a base background color, and then add one or more custom threshold or range color specifications. Multiple threshold and range specifications will layer, with the first applied first, followed by the second, and so on.

## Custom Base Color

To set the base color (optional):
clr(color_arg)
as described below in "Color definitions" on page 249. If omitted, the color defaults to "white".

## Custom Threshold

To add a threshold specification:
thresh(limit(threshold_value, threshold_spec), lowclr(below_arg), highclr(above_arg), threshold_name])
where threshold_spec is one of

| cright | closed on the right |
| :--- | :--- |
| cleft | closed on the left |

and the below_arg and above_arg are one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg) | gradient using color specification |
| @trans | transparent |

and color_arg are as described below in "Color definitions" on page 249. If omitted, the color defaults to "white".

The optional threshold_name argument may be used to attach a name to the corresponding definition.

## Custom Range

To add a range specification:
range(limit(low_value, high_value, range_spec), inclr(inside_arg), outclr(outside_arg)[, range_name])
where range_spec is one of

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

inside_arg is one of
color_arg solid color specification
$@ \operatorname{grad}($ color_arg1, gradient using color specification, where col-
color_arg2) or_arg1 and color_arg2 are the low and high colors, respectively.
@trans transparent
outside_arg is one of
color_arg solid color specification
@grad(color_arg) gradient using color specification
@trans transparent
color_arg1 and color_arg2 are as described below in "Color definitions" on page 249.
The optional range_name argument may be used to attach a name to the corresponding definition.

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special " @RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | $@ h e x(0000 f f)$ |
| :--- | :--- | :--- |
| red | $@ \operatorname{rgb}(255,0,0)$ | $@ h e x(f f 0000)$ |
| ltred | $@ \operatorname{rgb}(255,168,168)$ | $@ h e x(f f a 8 a 8)$ |
| green | $@ \operatorname{rgb}(0,128,0)$ | $@ h e x(008000)$ |
| black | $@ \operatorname{rgb}(0,0,0)$ | $@ h e x(000000)$ |
| white | $@ \operatorname{rgb}(255,255,255)$ | $@ h e x(f f f f f f)$ |
| purple | $@ \operatorname{rgb}(128,0,128)$ | $@ h e x(800080)$ |
| orange | $@ \operatorname{rgb}(255,128,0)$ | $@ h e x(f f 8000)$ |
| yellow | $@ \operatorname{rgb}(255,255,0)$ | $@ h e x(f f f f 00)$ |
| gray | $@ \operatorname{rgb}(128,128,128)$ | $@ h e x(808080)$ |
| ltgray | @rgb $(192,192,192)$ | $@ h e x(c 0 c 0 c 0)$ |

## Examples

To set a gray fill color for the shapes, you may use:

```
gmap.setfillcolor(type=single) clr(gray)
```

To set a fill color for negative values, you may use

```
gmap.setfillcolor(type=posneg) mapser(ser1)
```

which sets the fill color to white for non-negative values and light red for negative values of SER1.

Similarly,
gmap.setfillcolor(type=posneg) mapser(serl) posclr(@rgb(10, 20, 30)) negclr(purple)
sets the background sheet fill color to @ $\operatorname{rgb}(10,20,30)$ for non-negative values and purple for negative values of SER1.

Range coloring may be specified using the "type = range" option. The command

```
gmap.setfillcolor(type=range) mapser(ser1) clr(ltgray) range(10,
    20, cleft) inclr(@rgb(128, 0, 128)) outclr(ltred) naclr(green)
```

sets the background fill to $@ \operatorname{rgb}(128,0,128)$ for values between 10 and 20 , light-red to values outside of the range 10 to 20 , and green, for missing values.

Custom coloring allows you to construct more complex background filling:

```
gmap.setfillcolor(type=custom) mapser(ser1) clr(@rgb(10, 0, 0))
    range(limit(-10, 10, oboth), inclr(green), outclr(white)))
    thresh(limit(-1, oleft), highclr(grey), lowclr(@trans))
```


## Cross-references

See "Geomaps" on page 667 and of User's Guide I for a discussion of geomaps. See "ValueBased Text and Fill Coloring" on page 182 of User's Guide I for discussion of color settings.

See also Geomap: : options (p. 287) for options to control the shape border color and legend.

## setfillcolor

 Group ProcSet the fill (background) color used in the group spreadsheet using values in the spreadsheet or in a different series.

## Syntax

group_name.setfillcolor(spec) fill_color_args
where the required spec is one of the following:

$$
\begin{array}{ll}
\text { type }=\text { arg } & \begin{array}{l}
\text { Type of fill coloring for spreadsheet cells: "single" (single } \\
\text { color), "posneg" (positive-negative single threshold), } \\
\text { "range" (single range coloring), "hilo" (high-low-median), } \\
\text { "custom" (custom coloring). }
\end{array} \\
\mathrm{s}=\arg & \begin{array}{l}
\text { Colormap source: "none" (do not use a colormap and } \\
\text { therefore do not color) or "series" (use the same colormap } \\
\text { used by the individual series. }
\end{array}
\end{array}
$$

The first form specifies a colormap for all of the series in the group. The second form, uses individual colormaps obtained from the individual series.

## General Arguments

To specify the series or expression whose values will determine the background color:

- mapser(spec)
where spec is a series name or expression.
To specify the minimum and maximum values where the coloring begins and ends:
- min(color_arg)
- max(color_arg)

To set the missing value (NA) background color:

- naclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the color defaults to "white".


## Type-specific Arguments

There are optional type-specific arguments that correspond to each of the type choices:

## Single color

To set the single background color:
clr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the color defaults to "white".
Positive-negative single threshold
You may set the color for both the non-negative (posclr) and the negative (negclr) values

$$
\begin{aligned}
& \text { posclr(color_arg) } \\
& \text { negclr(color_arg) }
\end{aligned}
$$

where color_arg is described below in "Color definitions" on page 249. If omitted, the nonnegative color defaults to "white" and the negative color defaults to light-red.

## Single range

To specify the range, you must specify the range endpoints:
range(lower_val, upper_val[, range_def)
where range_def specifies the range endpoints:

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

By default, the range will be open on the lower and closed on the upper threshold limits.

You should provide a color specification for the inside range color (inclr) and outside range color (outclr):

```
inclr(color_arg)
```

outclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the interior color defaults to light-red, and the exterior defaults to white.

## High-Low-Median

When "type = hilo" you may specify the high, low, and median coloring values:
highclr(color_arg)
lowclr(color_arg)
medianclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the colors default to light-red.

## Custom

When "type = custom" you may specify custom coloring options.
You may optionally set a base background color, and then add one or more custom threshold or range color specifications. Multiple threshold and range specifications will layer, with the first applied first, followed by the second, and so on.

## Custom Base Color

To set the base color (optional):
clr(color_arg)
as described below in "Color definitions" on page 249. If omitted, the color defaults to "white".

## Custom Threshold

To add a threshold specification:
thresh(limit(threshold_value, threshold_spec), lowclr(below_arg), highclr(above_arg), threshold_name])
where threshold_spec is one of

| cright | closed on the right |
| :--- | :--- |
| cleft | closed on the left |

and the below_arg and above_arg are one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg) | gradient using color specification |
| @trans | transparent |

and color_arg are as described below in "Color definitions" on page 249. If omitted, the color defaults to "white".

The optional threshold_name argument may be used to attach a name to the corresponding definition.

## Custom Range

To add a range specification:
range(limit(low_value, high_value, range_spec), inclr(inside_arg), outclr(outside_arg)[, range_name])
where range_spec is one of

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

inside_arg is one of
color_arg solid color specification
@grad(color_arg1, gradient using color specification, where col-
color_arg2) or_arg1 and color_arg2 are the low and high colors, respectively.
@trans transparent
outside_arg is one of
color_arg solid color specification
@grad(color_arg) gradient using color specification
@trans transparent
color_arg1 and color_arg2 are as described below in "Color definitions" on page 249.
The optional range_name argument may be used to attach a name to the corresponding definition.

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special "@RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | $@ h e x(0000 f f)$ |
| :--- | :--- | :--- |
| red | $@ \operatorname{rgb}(255,0,0)$ | $@ h e x(f f 0000)$ |
| green | $@ \operatorname{rgb}(0,128,0)$ | $@ h e x(f f a 8 a 8)$ |
| black | $@ \operatorname{rgb}(0,0,0)$ | $@ h e x(008000)$ |
| white | $@ \operatorname{rgb}(255,255,255)$ | $@ h e x(000000)$ |
| purple | $@ \operatorname{rgb}(128,0,128)$ | $@ h e x(f f f f f f)$ |
| range | $@ \operatorname{rgb}(255,128,0)$ | $@ h e x(800080)$ |
| yellow | $@ \operatorname{rgb}(255,255,0)$ | $@ h e x(f f 8000)$ |
| gray | $@ \operatorname{rgb}(128,128,128)$ | $@ h e x(f f f f 00)$ |
| ltgray | $@ \operatorname{rgb}(192,192,192)$ | $@ h e x(808080)$ |

## Examples

To set a gray background color for all cells in the spreadsheet, you may use:

```
grp1.setfillcolor(type=single) clr(gray)
```

To set a background color for negative values, you may use

```
grp1.setfillcolor(type=posneg) mapser(ser1)
```

which sets the background sheet fill color to white for non-negative values and light red for negative values of SER1.

Similarly,

```
grp1.setfillcolor(type=posneg) mapser(ser1) posclr(@rgb(10, 20,
    30)) negclr(purple)
```

sets the background sheet fill color to @rgb $(10,20,30)$ for non-negative values and purple for negative values of SER1.

Range coloring may be specified using the "type = range" option. The command

```
grp1.setfillcolor(type=range) mapser(ser1) clr(ltgray) range(10,
    20, cleft) inclr(@rgb(128, 0, 128)) outclr(ltred) naclr(green)
```

sets the background fill to @ $\operatorname{rgb}(128,0,128)$ for values between 10 and 20 , light-red to values outside of the range 10 to 20 , and green, for missing values.

Custom coloring allows you to construct more complex background filling:

```
grp1.setfillcolor(type=custom) mapser(ser1) clr(@rgb(10, 0, 0))
    range(limit(-10, 10, oboth), inclr(green), outclr(white)))
    thresh(limit(-1, oleft), highclr(grey), lowclr(@trans))
```


## Cross-references

See "Value-Based Text and Fill Coloring" on page 182 of User's Guide I.
See also Group: :settextcolor (p. 430).

| setfillcolor | Series Procs |
| :--- | :--- |

Set the fill (background) color used in the series spreadsheet using values in the spreadsheet or in a different series.

## Syntax

series_name.setfillcolor(t = type) fill_color_args
where:

$$
\begin{array}{ll}
\text { type }=\text { arg } & \begin{array}{l}
\text { Type of fill coloring for spreadsheet cells: "single" (single } \\
\text { color), "posneg" (positive-negative single threshold), } \\
\text { "range" (single range coloring), "hilo" (high-low-median), } \\
\text { "custom" (custom coloring). }
\end{array}
\end{array}
$$

## General Arguments

To specify the series or expression whose values will determine the background color:

- mapser(spec)
where spec is a series name or expression.
To specify the minimum and maximum values where the coloring begins and ends:
- min(color_arg)
- max(color_arg)

To set the missing value (NA) background color:

- naclr(color_arg)
where color_arg is described below in "Color definitions" on page 258. If omitted, the color defaults to "white".


## Type-specific Arguments

There are optional type-specific arguments that correspond to each of the type choices:

## Single color

To set the single background color:

```
clr(color_arg)
```

where color_arg is described below in "Color definitions" on page 258. If omitted, the color defaults to "white".

Positive-negative single threshold
You may set the color for both the non-negative (posclr) and the negative (negclr) values
posclr(color_arg)
negclr(color_arg)
where color_arg is described below in "Color definitions" on page 258. If omitted, the nonnegative color defaults to "white" and the negative color defaults to light-red.

## Single range

To specify the range, you must specify the range endpoints:
range(lower_val, upper_val[, range_def)
where range_def specifies the range endpoints:

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

By default, the range will be open on the lower and closed on the upper threshold limits.
You should provide a color specification for the inside range color (inclr) and outside range color (outclr):
inclr(color_arg)
outclr(color_arg)
where color_arg is described below in "Color definitions" on page 258. If omitted, the interior color defaults to light-red, and the exterior defaults to white.

## High-Low-Median

When "type = hilo" you may specify the high, low, and median coloring values:
highclr(color_arg)
lowclr(color_arg)
medianclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the colors default to light-red.

## Custom

When "type = custom" you may specify custom coloring options.
You may optionally set a base background color, and then add one or more custom threshold or range color specifications. Multiple threshold and range specifications will layer, with the first applied first, followed by the second, and so on.

## Custom Base Color

To set the base color (optional):
clr(color_arg)
as described below in "Color definitions" on page 258. If omitted, the color defaults to
"white".

## Custom Threshold

To add a threshold specification:
thresh(limit(threshold_value, threshold_spec), lowclr(below_arg), highclr(above_arg), threshold_name])
where threshold_spec is one of

| cright | closed on the right |
| :--- | :--- |
| cleft | closed on the left |

and the below_arg and above_arg are one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg) | gradient using color specification |
| @trans | transparent |

and color_arg are as described below in "Color definitions" on page 258. If omitted, the color defaults to "white".

The optional threshold_name argument may be used to attach a name to the corresponding definition.

## Custom Range

To add a range specification:
range(limit(low_value, high_value, range_spec), inclr(inside_arg), outclr(outside_arg)[, range_name])
where range_spec is one of

| cright | closed on the right only |
| :---: | :---: |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |
| g is one of |  |
| color_arg | solid color specification |
| @grad(color_arg1, color_arg2) | gradient using color specification, where color_arg1 and color_arg2 are the low and high colors, respectively. |
| @trans | transparent |

outside_arg is one of

| color_arg | solid color specification |
| :--- | :--- |
| $@ \operatorname{grad}($ color_arg $)$ | gradient using color specification |
| $@$ trans | transparent |

color_arg1 and color_arg2 are as described below in "Color definitions" on page 258.
The optional range_name argument may be used to attach a name to the corresponding definition.

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special "@RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | @hex(0000ff) |
| :---: | :---: | :---: |
| red | $@ \operatorname{rgb}(255,0,0)$ | @hex(ff0000) |
| green | $@ \operatorname{rgb}(0,128,0)$ | @hex(ffa8a8) |
| black | $@ \operatorname{rgb}(0,0,0)$ | @hex(008000) |
| white | $@ \mathrm{rgb}(255,255,255)$ | @hex(000000) |
| purple | @rgb (128, 0, 128) | @hex(ffffff) |
| orange | $@ \operatorname{rgb}(255,128,0)$ | @hex(800080) |
| yellow | @rgb (255, 255, 0 ) | @hex(ff8000) |


| gray | $@ \operatorname{rgb}(128,128,128)$ | $@ h e x(f f f f 00)$ |
| :--- | :--- | :--- |
| ltgray | $@ \operatorname{rgb}(192,192,192)$ | $@ h e x(808080)$ |

## Examples

To set a gray background color for all cells in the spreadsheet, you may use:

```
myser.setfillcolor(type=single) clr(gray)
```

To set a background color for negative values, you may use

```
myser.setfillcolor(type=posneg) mapser(ser1)
```

which sets the background sheet fill color to white for non-negative values and light red for negative values of SER1.

Similarly,

```
myser.setfillcolor(type=posneg) mapser(ser1) posclr(@rgb(10, 20,
    30)) negclr(purple)
```

sets the background sheet fill color to $@ \operatorname{rgb}(10,20,30)$ for non-negative values and purple for negative values of SER1.

Range coloring may be specified using the "type = range" option. The command

```
myser.setfillcolor(type=range) mapser(ser1) clr(ltgray) range(10,
    20, cleft) inclr(@rgb(128, 0, 128)) outclr(ltred) naclr(green)
```

sets the background fill to $@ \operatorname{rgb}(128,0,128)$ for values between 10 and 20 , light-red to values outside of the range 10 to 20 , and green, for missing values.

Custom coloring allows you to construct more complex background filling:

```
myser.setfillcolor(type=custom) mapser(ser1) clr(@rgb(10, 0, 0))
    range(limit(-10, 10, oboth), inclr(green), outclr(white)))
    thresh(limit(-1, oleft), highclr(grey), lowclr(@trans))
```


## Cross-references

"Value-Based Text and Fill Coloring" on page 182 of User's Guide I.
See also : :settextcolor (p. 269).

| setjust | Geomap Procs |
| :--- | :--- |

Set the display justification for multi-line area labels.
The justification setting has no effect on single-line labels.

## Syntax

geomap_name.setjust format_arg
where format_arg is a set of arguments used to specify format settings.
The format_arg may be formed using the following:
left / center / right Horizontal justification settings

## Examples

```
geomap1.setjust center
```

centers the labels for areas in the GEOMAP1 geomap object.
geomap1.setjust left
Left justifies the lines in the labels for the areas in the GEOMAP1 geomap object.

## Cross-references

See "Geomaps" on page 667 and of User's Guide I for a discussion of geomaps.

| setrowlabels | Matrix Procs |
| :--- | :--- |

Set the row labels in a matrix object.

## Syntax

matrix_name.setrowlabels label1 label2 label3....
Follow the setrowlabels command with a space delimited list of row labels. Note that each row label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are rows, EViews will use the corresponding default row names ("R11", "R12", etc...).

## Examples

```
mat1.setrowlabels USA UK FRANCE
```

sets the row label for the first row in matrix MAT1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

## setrowlabels

## Rowvector Procs

Set the row label in a rowvector object.

## Syntax

rowvector_name.setrowlabels label.
Follow the setrowlabels command with a space delimited list of row labels. Note that each row label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are rows, EViews will use the corresponding default row names ("R11", "R12", etc...).

## Examples

rvec1.setrowlabels MyResults
sets the row label for the rowvector RVEC1 to "MyResults".
Cross-references

| setrowlabels | Svector Procs |
| :--- | :--- |

Set the row labels in a svector object.
Syntax
svector_name.setrowlabels label1 label2 label3....
Follow the setrowlabels command with a space delimited list of row labels. Note that each row label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are rows, EViews will keep the corresponding default row names ("R11", "R12", etc...).

## Examples

```
svec1.setrowlabels USA UK FRANCE
```

sets the row label for the first row in svector SVEC1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

| setrowlabels | Sym Procs |
| :--- | :--- |

Set the row labels in a sym object.

## Syntax

sym_name.setrowlabels label1 label2 label3....
Follow the setrowlabels command with a space delimited list of row labels. Note that each row label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are rows, EViews will use the corresponding default row names ("R11", "R12", etc...).

## Examples

sym1.setrowlabels USA UK FRANCE
sets the row label for the first row in sym SYM1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

| setrowlabels | Vector Procs |
| :--- | :--- |

Set the row labels in a vector object.

## Syntax

vector_name.setrowlabels label1 label2 label3....
Follow the setrowlabels command with a space delimited list of row labels. Note that each row label should not contain spaces unless it is enclosed in quotes. If you provide fewer labels than there are rows, EViews will keep the corresponding default row names ("R11", "R12", etc...).

## Examples

```
vecl.setrowlabels USA UK FRANCE
```

sets the row label for the first row in vector VEC1 to USA, the second to UK, and the third to FRANCE.

## Cross-references

## setshapelabel <br> Geomap Procs

Set which attribute to use or create a custom label to use when labeling shapes in the geomap.

## Syntax

geomap_name.setshapelabel(attribute_name) custom_label
where attribute_name is the name of an attributed in the geomap. In the case where attribute_name is equal to "custom", the custom_label will be used. In all other cases, the command will be ignored.

## Examples

geomap1.setshapelabel (name)
will label the areas in the GEOMAP1 geomap object using areas name attribute.
geomap1.setshapelabel (none)
turn off all area labels in the GEOMAP1 geomap object.

```
geomap1.setshapelabel (fillvalues)
```

will label the areas in the GEOMAP1 geomap object using the numerical values used for determining area fill color. This only applicable after a colormap has been applied. All values will otherwise be NA.

```
geomap1.Setshapelabel(custom)
    Area:[county],[state]\nPop:[fillvalues]
```

will create a custom 2 line label for the areas in the GEOMAP1 geomap object. The first line of the label will read "Area:" followed by the areas county attribute, a comma, and then the areas state. The first line will look similar to "Area:Suffolk,NY". The second line of the label will "Pop:" followed by the value used to color the area.Examples

## Cross-references

See "Geomaps" on page 667 and of User's Guide I for a discussion of geomaps.
$\square$
settextcolor

## Group Proc

Set the text color used in the group spreadsheet using values in the spreadsheet or in a different series.

## Syntax

group_name.settextcolor(spec) fill_color_args
where the required spec is one of the following:

```
type = arg Type of fill coloring for spreadsheet cells: "single" (single
    color), "posneg" (positive-negative single threshold),
    "range" (single range coloring), "hilo" (high-low-median),
    "custom" (custom coloring).
s = arg Colormap source: "none" (do not use a colormap and
    therefore do not color) or "series" (use the same colormap
    used by the individual series.
```

The first form specifies a colormap for all of the series in the group. The second form, uses individual colormaps obtained from the individual series.

## General Arguments

To specify the series or expression whose values will determine the background color:

- mapser (spec)
where spec is a series name or expression.
To specify the minimum and maximum values where the coloring begins and ends:
- min(color_arg)
- max(color_arg)

To set the missing value (NA) background color:

- naclr(color_arg)
where color_arg is described below in "Color definitions" on page 267. If omitted, the color defaults to "black".


## Type-specific Arguments

There are optional type-specific arguments that correspond to each of the type choices:

## Single color

To set the single text color:
clr(color_arg)
where color_arg is described below in "Color definitions" on page 267. If omitted, the color defaults to "black".

Positive-negative single threshold
You may set the color for both the non-negative (posclr) and the negative (negclr) values
posclr(color_arg)
negclr(color_arg)
where color_arg is described below in "Color definitions" on page 267. If omitted, the nonnegative color defaults to "black" and the negative color defaults to "red".
Single range
To specify the range, you must specify the range endpoints:
range(lower_val, upper_val[, range_def)
where range_def specifies the range endpoints:

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

By default, the range will be open on the lower and closed on the upper threshold limits.
You should provide a color specification for the inside range color (inclr) and outside range color (outclr):
inclr(color_arg)
outclr(color_arg)
where color_arg is described below in "Color definitions" on page 267. If omitted, the interior color defaults to light-red, and the exterior defaults to white.
High-Low-Median
When "type = hilo" you may specify the high, low, and median coloring values:
highclr(color_arg)
lowclr(color_arg)
medianclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the colors default to light-red.

## Custom

When "type = custom" you may specify custom coloring options.
You may optionally set a base text color, and then add one or more custom threshold or range color specifications. Multiple threshold and range specifications will layer, with the first applied first, followed by the second, and so on.

## Custom Base Color

To set the base color (optional):
clr(color_arg)
as described below in "Color definitions" on page 267. If omitted, the color defaults to "white".

## Custom Threshold

To add a threshold specification:
thresh(limit(threshold_value, threshold_spec), lowclr(below_arg), highclr(above_arg), threshold_name])
where threshold_spec is one of

| cright | closed on the right |
| :--- | :--- |
| cleft | closed on the left |

and the below_arg and above_arg are one of
color_arg solid color specification
$@$ grad(color_arg) gradient using color specification
@trans transparent
and color_arg are as described below in "Color definitions" on page 267. If omitted, the color defaults to "white".

The optional threshold_name argument may be used to attach a name to the corresponding definition.

## Custom Range

To add a range specification:
range(limit(low_value, high_value, range_spec), inclr(inside_arg), outclr(outside_arg)[, range_name])
where range_spec is one of

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | solid color specification |
| rg is one of | gradient using color specification, where col- <br> or_arg1 and color_arg2 are the low and high <br> colors, respectively. |
| @grad $\left(c o l o r \_a r g 1, ~\right.$ <br> color_arg2) | transparent |
| @trans |  |

outside_arg is one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg) | gradient using color specification |
| @trans | transparent |

color_arg1 and color_arg2 are as described below in "Color definitions" on page 267.
The optional range_name argument may be used to attach a name to the corresponding definition.

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special "@RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | $@ h e x(0000 f f)$ |
| :--- | :--- | :--- |
| red | $@ \operatorname{rgb}(255,0,0)$ | $@ h e x(f f 0000)$ |
| ltred | $@ \operatorname{rgb}(255,168,168)$ | $@ h e x(f f a 8 a 8)$ |
| green | $@ \operatorname{rgb}(0,128,0)$ | $@ h e x(008000)$ |
| black | $@ \operatorname{rgb}(0,0,0)$ | $@ h e x(000000)$ |
| white | $@ \operatorname{rgb}(255,255,255)$ | $@ h e x(f f f f f f)$ |
| purple | $@ \operatorname{rgb}(128,0,128)$ | $@ h e x(800080)$ |
| orange | $@ \operatorname{rgb}(255,128,0)$ | $@ h e x(f f 8000)$ |


| yellow | $@ \operatorname{rgb}(255,255,0)$ | $@ h e x(f f f f 00)$ |
| :--- | :--- | :--- |
| gray | $@ \operatorname{rgb}(128,128,128)$ | $@ h e x(808080)$ |
| ltgray | $@ \operatorname{rgb}(192,192,192)$ | $@ h e x(c 0 c 0 c 0)$ |

## Examples

To set a gray text color for all cells in the spreadsheet, you may use:

```
grp1.settextcolor(type=single) clr(gray)
```

To set a text color for negative values, you may use

```
grp1.settextcolor(type=posneg) mapser(ser1)
```

which sets the text color to black for non-negative values and red for negative values of SER1.

Similarly,

```
grp1.settextcolor(type=posneg) mapser(ser1) posclr(@rgb(10, 20,
    30)) negclr(purple)
```

sets the text color to @rgb(10, 20,30) for non-negative values and purple for negative values of SER1.

Range coloring may be specified using the "type = range" option. The command

```
grp1.settextcolor(type=range) mapser(ser1) clr(ltgray) range(10,
    20, cleft) inclr(@rgb(128, 0, 128)) outclr(ltred) naclr(green)
```

sets the text to @rgb(128, 0, 128) for values between 10 and 20 , light-red to values outside of the range 10 to 20 , and green, for missing values.

Custom coloring allows you to construct more complex text coloring:

```
grp1.settextcolor(type=custom) mapser(ser1) clr(@rgb(10, 0, 0))
    range(limit(-10, 10, oboth), inclr(green), outclr(white)))
    thresh(limit(-1, oleft), highclr(grey), lowclr(@trans))
```


## Cross-reference

See "Value-Based Text and Fill Coloring" on page 182 of User's Guide I.
See also Group: :setfillcolor (p. 420).

| settextcolor | Series Procs |
| :--- | :--- |

Set the text color used in the series spreadsheet using values in the spreadsheet or in a different series.

## Syntax

series_name.settextcolor(t = type) text_color_args
where:

```
type = arg Type of fill coloring for spreadsheet cells: "single" (single
    color), "posneg" (positive-negative single threshold),
    "range" (single range coloring), "hilo" (high-low-median),
    "custom" (custom coloring).
```


## General Arguments

To specify the series or expression whose values will determine the background color:

- mapser(spec)
where spec is a series name or expression.
To specify the minimum and maximum values where the coloring begins and ends:
- min(color_arg)
- max(color_arg)

To set the missing value (NA) background color:

- naclr(color_arg)
where color_arg is described below in "Color definitions" on page 272. If omitted, the color defaults to "black".


## Type-specific Arguments

There are optional type-specific arguments that correspond to each of the type choices:

## Single color

To set the single text color:
clr(color_arg)
where color_arg is described below in "Color definitions" on page 272. If omitted, the color defaults to "black".

## Positive-negative single threshold

You may set the color for both the non-negative (posclr) and the negative (negclr) values posclr(color_arg)
negclr(color_arg)
where color_arg is described below in "Color definitions" on page 272. If omitted, the nonnegative color defaults to "black" and the negative color defaults to "red".

## Single range

To specify the range, you must specify the range endpoints:
range(lower_val, upper_val[, range_def)
where range_def specifies the range endpoints:

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

By default, the range will be open on the lower and closed on the upper threshold limits.
You should provide a color specification for the inside range color (inclr) and outside range color (outclr):
inclr(color_arg)
outclr(color_arg)
where color_arg is described below in "Color definitions" on page 272. If omitted, the interior color defaults to light-red, and the exterior defaults to white.

## High-Low-Median

When "type = hilo" you may specify the high, low, and median coloring values:
highclr(color_arg)
lowclr(color_arg)
medianclr(color_arg)
where color_arg is described below in "Color definitions" on page 249. If omitted, the colors default to light-red.

## Custom

When "type = custom" you may specify custom coloring options.
You may optionally set a base text color, and then add one or more custom threshold or range color specifications. Multiple threshold and range specifications will layer, with the first applied first, followed by the second, and so on.

## Custom Base Color

To set the base color (optional):
clr(color_arg)
as described below in "Color definitions" on page 272. If omitted, the color defaults to "white".

## Custom Threshold

To add a threshold specification:
thresh(limit(threshold_value, threshold_spec), lowclr(below_arg), highclr(above_arg), threshold_name])
where threshold_spec is one of
cright closed on the right
cleft closed on the left
and the below_arg and above_arg are one of
color_arg solid color specification
@grad(color_arg) gradient using color specification
@trans transparent
and color_arg are as described below in "Color definitions" on page 272. If omitted, the color defaults to "white".

The optional threshold_name argument may be used to attach a name to the corresponding definition.

Custom Range
To add a range specification:
range(limit(low_value, high_value, range_spec), inclr(inside_arg), outclr(outside_arg)[, range_name])
where range_spec is one of

| cright | closed on the right only |
| :--- | :--- |
| cboth | closed on both sides |
| cleft | closed on the left only |
| oboth | open on both sides |

inside_arg is one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg1, | gradient using color specification, where col- <br> or_arg1 and color_arg2 are the low and high <br> color_arg2) <br> colors, respectively. |
| @trans | transparent |

outside_arg is one of

| color_arg | solid color specification |
| :--- | :--- |
| @grad(color_arg) | gradient using color specification |
| @trans | transparent |

color_arg1 and color_arg2 are as described below in "Color definitions" on page 272.
The optional range_name argument may be used to attach a name to the corresponding definition.

## Color definitions

color_arg specifies the color to be employed in the arguments above. The color may be specified using predefined color names, by specifying the individual red-green-blue (RGB) components using the special " @RGB" function, or by specifying the individual red-green-blue (RGB) components in hexadecimal using the special "@HEX" function.

The predefined colors are given by the keywords (with their RGB and HEX equivalents):

| blue | $@ \operatorname{rgb}(0,0,255)$ | @ hex(0000ff) |
| :---: | :---: | :---: |
| red | @rgb (255, 0, 0) | @hex(ff0000) |
| ltred | $@ \operatorname{rgb}(255,168,168)$ | @hex(ffa8a8) |
| green | $@ \operatorname{rgb}(0,128,0)$ | @hex(008000) |
| black | $@ \mathrm{rgb}(0,0,0)$ | @ hex(000000) |
| white | $@ \operatorname{rgb}(255,255,255)$ | @hex(ffffff) |
| purple | $@ \operatorname{rgb}(128,0,128)$ | @ hex(800080) |
| orange | $@ \operatorname{rgb}(255,128,0)$ | @hex(ff8000) |
| yellow | $@ \operatorname{rgb}(255,255,0)$ | @ hex(ffff00) |
| gray | $@ \operatorname{rgb}(128,128,128)$ | @ hex(808080) |
| ltgray | $@ \operatorname{rgb}(192,192,192)$ | @hex(c0c0c0) |

## Examples

To set a gray text color for all cells in the spreadsheet, you may use:

```
myser.settextcolor(type=single) clr(gray)
```

To set a text color for negative values, you may use

```
myser.settextcolor(type=posneg) mapser(ser1)
```

which sets the text color to black for non-negative values and red for negative values of SER1.

Similarly,

```
myser.settextcolor(type=posneg) mapser(ser1) posclr(@rgb(10, 20,
    30)) negclr(purple)
```

sets the text color to @rgb $(10,20,30)$ for non-negative values and purple for negative values of SER1.

Range coloring may be specified using the "type = range" option. The command

```
myser.settextcolor(type=range) mapser(ser1) clr(ltgray) range(10,
    20, cleft) inclr(@rgb(128, 0, 128)) outclr(ltred) naclr(green)
```

sets the text to @rgb(128, 0, 128) for values between 10 and 20 , light-red to values outside of the range 10 to 20 , and green, for missing values.

Custom coloring allows you to construct more complex text coloring:

```
myser.settextcolor(type=custom) mapser(ser1) clr(@rgb(10, 0, 0))
    range(limit(-10, 10, oboth), inclr(green), outclr(white)))
    thresh(limit(-1, oleft), highclr(grey), lowclr(@trans))
```


## Cross-references

"Value-Based Text and Fill Coloring" on page 182 of User's Guide I.
See also : :setfillcolor (p. 255).

| similarity | Equation Views |
| :--- | :--- |

Compute the PMG Hausman test for similarity against mean-group and dynamic fixed effects estimators (in panel equations estimated by ARDL/PMG)

## Syntax

eq_name.similarity(options)
Options
p Print output.

## Example

```
pmg_eq.similarity
```

displays a spool object with several tables containing the results of the Hausman test, comparisons of results, and auxiliary estimation results employed in computing the test statistic.

## Cross-references

| smooth | Series Procs |
| :--- | :--- |

## Exponential smoothing.

Forecasts a series using one of a number of exponential smoothing techniques. By default, smooth estimates the damping parameters of the smoothing model to minimize the sum of squared forecast errors, but you may specify your own values for the damping parameters.
smooth automatically calculates in-sample forecast errors and puts them into the series RESID.

## Syntax

series_name.smooth(method) smooth_name [freq]
You should follow the smooth keyword with a name for the smoothed series. You must also specify the smoothing method in parentheses. The optional freq may be used to override the default for the number of periods in the seasonal cycle. By default, this value is set to the workfile frequency (e.g. -4 for quarterly data). For undated data, the default is 5 .

## Options

## Smoothing method options

| $\mathrm{s}[, x]$ | Single exponential smoothing for series with no trend. You <br> may optionally specify a number $x$ between zero and one <br> for the mean parameter. |
| :--- | :--- |
| $\mathrm{d}[, x] \quad$ | Double exponential smoothing for series with a trend. You <br> may optionally specify a number $x$ between zero and one <br> for the mean parameter. |


| $\mathrm{n}[, x, y]$ | Holt-Winters without seasonal component. You may <br> optionally specify numbers $x$ and $y$ between zero and one <br> for the mean and trend parameters, respectively. |
| :--- | :--- |
| $\mathrm{a}[, x, y, z]$ | Holt-Winters with additive seasonal component. You may <br> optionally specify numbers $x, y$, and $z$, between zero and <br> one for the mean, trend, and seasonal parameters, respec- <br> tively. |
| $\mathrm{m}[, x, y, z] \quad$ | Holt-Winters with multiplicative seasonal component. You <br> may optionally specify numbers $x, y$, and $z$, between zero <br> and one for the mean, trend, and seasonal parameters, <br> respectively. |

## Other Options:

| forcsmpl $=$ <br> arg | Forecast sample (optional). If forecast sample is not pro- <br> vided, the workfile sample will be employed. |
| :--- | :--- |
| prompt | Force the dialog to appear from within a program. |
| p | Print a table of forecast statistics. |

If you wish to set only some of the damping parameters and let EViews estimate the other parameters, enter the letter "e" where you wish the parameter to be estimated.

If the number of seasons is different from the frequency of the workfile (an unusual case that arises primarily if you are using an undated workfile for data that are not monthly or quarterly), you should enter the number of seasons after the smoothed series name. This optional input will have no effect on forecasts without seasonal components.

## Examples

```
sales.smooth(s) sales_f
```

smooths the SALES series by a single exponential smoothing method and saves the smoothed series as SALES_F. EViews estimates the damping (smoothing) parameter and displays it with other forecast statistics in the SALES series window.

```
tb3.smooth(n,e,.3) tb3_hw
```

smooths the TB3 series by a Holt-Winters no seasonal method and saves the smoothed series as TB3_HW. The mean damping parameter is estimated while the trend damping parameter is set to 0.3 .

```
smpl @first @last-10
order.smooth(m,.1,.1,.1) order_hw
smpl @all
graph gra1.line order order_hw
show gral
```

smooths the ORDER series by a Holt-Winters multiplicative seasonal method leaving the last 10 observations. The damping parameters are all set to 0.1 . The last three lines plot and display the actual and smoothed series over the full sample.

## Cross-references

See "Exponential Smoothing" on page 551 of User's Guide I for a discussion of exponential smoothing methods. See also : :ets (p. 154).

## symmtest

Compute symmetry test for distributed lag variables in an equation estimated with a nonlinear ARDL (NARDL) specification.

This view displays a table object with the NARDL symmetry test. The top part of the table is a is a summary of the test. This is followed by three additional sections with test statistics and corresponding $p$-values for relevant regressors tests for: 1) long-run asymmetry, 2) short-run asymmetry, 3 ) both long and short-run asymmetry.

## Syntax

eq_name.symmtest(options)

## Options

p Print output.

## Example

ardl_eq.symmtest
computes the NARDL symmetry tests for relevant regressors.

## Cross-references

| vdecomp | Var Views |
| :--- | :--- |

Variance decomposition in VARs.

## Syntax

var_name.vdecomp(n, options) series_list [@ @ ordering]

List the series names in the VAR whose variance decomposition you would like to compute. You may optionally specify the ordering for the factorization after two "@"-signs.

You must specify the number of periods $n$ over which to compute the variance decompositions.

## Options

g
m
t (default)
imp = arg
(default = "chol")
se = mcarlo
rep $=$ integer
cilevels $=$ arg
(default = "0.95")
uselines
matbys = name
matbyr = name
prompt
p

Display combined graphs, with the decompositions for each variable shown in a graph.

Display multiple graphs, with each response-shock pair shown in a separate graph.

Show numerical results in table.
Type of factorization for the decomposition: "chol" (Cholesky with d.f. correction), "mlechol" (Cholesky without d.f. correction), "struct" (structural).
The structural factorization is based on the estimated structural VAR. To use this option, you must first estimate the structural decomposition; see Var: :svar (p. 1045).
The option "imp = mlechol" is provided for backward compatibility with EViews 3.x and earlier.

Monte Carlo standard errors. You must specify the number of replications with the "rep = " option.
Currently available only when you have specified the Cholesky factorization (using the "imp = chol" option).

Number of Monte Carlo replications to be used in computing the standard errors. Must be used with the "se = mcarlo" option.

Confidence interval coverage: space limited list of numbers between 0 and 1 .

Use lines instead of shading for confidence intervals.
Save responses by shocks (impulses) in named matrix. The first column is the response of the first variable to the first shock, the second column is the response of the second variable to the first shock, and so on.

Save responses by response series in named matrix. The first column is the response of the first variable to the first shock, the second column is the response of the first variable to the second shock, and so on.

Force the dialog to appear from within a program.
Print results.

If you use the "matbys = " or "matbyr = " options to store the results in a matrix, two matrices will be returned. The matrix with the specified name contains the variance decompositions, while the matrix with "_FSE" appended to the name contains the forecast standard errors for each response variable. If you have requested Monte Carlo standard errors, there
will be a third matrix with "_SE" appended to the name which contains the variance decomposition standard errors.

## Examples

```
var varl.ls 1 4 m1 gdp cpi
var1.vdecomp(10,t) gdp
```

The first line declares and estimates a VAR with three variables and lags from 1 to 4 . The second line tabulates the variance decompositions of GDP up to 10 periods using the ordering as specified in VAR1.

```
var1.vdecomp(10,t) gdp @ @ cpi gdp m1
```

performs the same variance decomposition as above using a different ordering.

## Cross-references

See "Variance Decomposition" on page 864 of User's Guide II for additional details.
See also : : impulse (p. 216).

| wfopen | Object Container, Data, and File Commands |
| :--- | :--- |

Open a workfile. Reads in a previously saved workfile from disk, or reads the contents of a foreign data source into a new workfile.

The opened workfile becomes the default workfile; existing workfiles in memory remain on the desktop but become inactive.

## Syntax

wfopen [path $\backslash$ ]source_name
wfopen(options) source_description [table_description] [variables_description]
wfopen(options) source_description [table_description] [dataset_modifiers]
where path is an optional local path or URL.
There are three basic forms of the wfopen command:

- the first form is used by EViews native ("EViews and MicroTSP" on page 281) and time series database formats ("Time Series Formats" on page 281).
- the second form used for raw data files-Excel, Lotus, ASCII text, and binary files ("Raw Data Formats" on page 282).
- the third form is used with the remaining source formats, which we term dataset formats, since the data have already been arranged in named variables ("Datasets" on page 291).
(See "Options" on page 280 for a description of the supported source formats and corresponding types.)

In all three cases, the workfile or external data source should be specified as the first argument following the command keyword and options.

- In most cases, the external data source is a file, so the source_description will be the description of the file (including local path or URL information, if necessary). Alternatively, the external data source may be the output from a web server, in which case the URL should be provided. Similarly, when reading from an ODBC query, the ODBC DSN (data source name) should be used as the source_description.

If the source_description contains spaces, it must be enclosed in (double) quotes.
For raw and dataset formats, you may use table_description to provide additional information about the data to be read:

- Where there is more than one table that could be formed from the specified external data source, a table_description may be provided to select the desired table. For example, when reading from an Excel file, an optional cell range may be provided to specify which data are to be read from the spreadsheet. When reading from an ODBC data source, a SQL query or table name must be used to specify the table of data to be read.
- In raw data formats, the table_description allows you to provide additional information regarding names and descriptions for variables to be read, missing values codes, settings for automatic format, and data range subsetting.
- When working with text or binary files, the table_description must be used to describe how to break up the file into columns and rows.

For raw and non-EViews dataset formats, you may use the dataset_modifiers specification to select the set of variables, maps (value labels), and observations to be read from the source data. The dataset_modifiers consists of the following keyword delimited lists:
[@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@selectif condition]

- The @keep and @drop keywords, followed by a list of names and patterns, are used to specify variables to be retain or dropped. Similarly, the @keepmap and @dropmap keywords followed by lists of name patterns controls the reading of value labels. The keyword @selectif, followed by an if condition (e.g., "if age > 30 and gender = 1") may be used to select a subset of the observations in the original data. By default, all variables, value labels, and observations are read.

By default, all variables, maps and observations in the source file will be read.

## Options

$$
\begin{array}{ll}
\text { type =arg, } \mathrm{t}=\text { arg } & \begin{array}{l}
\text { Optional type specification: (see table below). } \\
\text { Note that ODBC support is provided only in the EViews } \\
\text { Enterprise Edition. }
\end{array} \\
\text { link } & \begin{array}{l}
\text { Link the object to the source data so that the values can be } \\
\text { refreshed at a later time. }
\end{array} \\
\mathrm{wf}=w f \_n a m e & \begin{array}{l}
\text { Optional name for the new workfile. }
\end{array} \\
\text { page = page_name } & \begin{array}{l}
\text { Optional name for the page in the new workfile. }
\end{array} \\
\text { prompt } & \text { Force the dialog to appear from within a program. }
\end{array}
$$

For the most part, you should not need to specify a "type = " option as EViews will automatically determine the type from the filename.

The following table summaries the various source formats and types, along with the corresponding "type = " keywords:

| Access | dataset | "access" |
| :--- | :---: | :---: |
| Aremos-TSD | time series database | "a", "aremos", "tsd" |
| Binary | raw data | "binary" |
| dBASE | dataset | "dbase" |
| Excel (through 2003) | raw data | "excel" |
| Excel 2007 (xml) | raw data | "excelxml" |
| EViews Workfile | native | --- |
| Gauss Dataset | dataset | "gauss" |
| GiveWin/PcGive | time series database | "g", "give" |
| HTML | raw data | "html" |
| Lotus 1-2-3 | dataset | "lotus" |
| ODBC Dsn File | dataset | "msquery" |
| ODBC Query File | dataset | "odbc" |
| ODBC Data Source | native | "dos", "microtsp" |
| MicroTSP Workfile | native | "mac" |
| MicroTSP Mac Workfile | "rats" |  |
| RATS 4.x | time series database | "sasprog" |
| RATS Portable / TROLL | time series database | dataset |


| SAS Transport | dataset | "sasxport" |
| :--- | :---: | :---: |
| SPSS | dataset | "spss" |
| SPSS Portable | dataset | "spssport" |
| Stata | dataset | "stata" |
| Text / ASCII | raw data | "text" |
| TSP Portable | time series database | "t","tsp" |

## EViews and MicroTSP

The syntax for EViews and MicroTSP files is:
wfopen [path $\backslash$ ]workfile_name
where path is an option local path or URL.

## Examples

```
wfopen c:\data\macro
```

loads a previously saved EViews workfile "Macro.WF1" from the "data" directory in the C drive.

```
wfopen c:\tsp\nipa.wf
```

loads a MicroTSP workfile "Nipa.WF". If you do not use the workfile type option, you should add the extension ".WF" to the workfile name when loading a DOS MicroTSP workfile. An alternative method specifies the type explicitly:

```
wfopen(type=dos) nipa
```

The command:

```
wfopen "<mydropboxdrive>\folder\nipa.wf1"
```

will open the file from the cloud location MYDROPBOXDRIVE.

## Time Series Formats

The syntax for time series format files (Aremos-TSD, GiveWin/PcGive, RATS, RATS Portable/TROLL, TSP Portable) is:
wfopen(options) [path $\backslash$ ]source_name
where path is an optional local path or URL.
If the source files contain data of multiple frequencies, the resulting workfile will be of the lowest frequency, and higher frequency data will be converted to this frequency. If you wish to obtain greater control over the workfile creation, import, or frequency conversion processes, we recommend that you open the file using dbopen (p. 139) and use the database tools to create your workfile.

## Aremos Example

```
wfopen dlcs.tsd
wfopen(type=aremos) dlcs.tsd
```

open the AREMOS-TSD file DLCS.

## GiveWin/PcGive Example

```
wfopen "f:\project\pc give\data\macrodata.in7"
wfopen(type=give) "f:\project\pc give\data\macrodata"
```

open the PcGive file MACRODATA.

## Rats Examples

```
wfopen macrodata.rat"
wfopen macrodata.trl
```

read the native RATS 4.x file MACRODATA.RAT and the RATS Portable/TROLL file "Macrodata.TRL".

## TSP Portable Example

wfopen macrodata.tsp
reads the TSP portable file "Macrodata.TSP".

## Raw Data Formats

The command for reading raw data (Excel 97-2003, Excel 2007, HTML, ASCII text, Binary, Lotus 1-2-3) is
wfopen(options) source_description [table_description] [variables_description] [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@selectif condition]
where the syntax of the table_description and variables_description differs slightly depending on the type of file.

## Excel and Lotus Files

The syntax for reading Excel and Lotus files is:
wfopen(options) source_description [table_description] [variables_description]
The following table_description elements may be used when reading Excel and Lotus data:

- "range = arg", where arg is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].
If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automati-
cally to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.
- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "namepos = [first|firstatt|last|lastatt|all|none|attonly|discard|custom]", which row(s) of the column headers should be used to form the column name, and also how to use the rest. The setting "first" (or "last") refers to the object name being in the first (or last) column header row, and all other rows as the object's description. Similarly, "firstatt" (or "lastatt") will use the first (or last) row as the name field, but will use all others as a custom attribute. The setting "all" will concatenate all column header fields into the object's name. "none" will concatenate all column header fields into the object's description. "attonly" will save all column header fields into the object's custom attributes. "discard" will skip all header rows altogether, and "custom" will allow you to specify explicitly how to treat each column header row using the "colheadnames = " argument. The default setting is "all" if no "colheadnames = " is specified, otherwise "custom".
- "colheadnames = ("arg1", "arg2")", required when "namepos = custom". Specifies the name \& type of each column header row. "Name" will be mapped to the object name, "Description" to the object's description field, and the rest will be stored as custom object attributes. Any blank name will cause that column header row to be skipped.
- "nonames", the file does not contain a column header (same as "colhead $=0$ ").
- "names = ("arg1","arg2",...)", user specified column names, where $\arg 1, \arg 2, \ldots$ are names of the first series, the second series, etc. when names are provided, these override any names that would otherwise be formed from the column headers.
- "descriptions = ("arg1","arg2",...)", user specified descriptions of the series. If descriptions are provided, these override any descriptions that would otherwise be read from the data.
- "types = ("arg1","arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), " f " (numeric data), "d" (dates), or "w" (EViews automatic detection). Note that the types appear without quotes: e.g., "types = (a,a,a)".
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int $\mid$ all $]$ ", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the "na=" argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same rows. If the first five rows are not enough to correctly determine the data format, use the "scan =" argument to instruct EViews to look at more rows. In addition, you may want to specify a the " $n a=$ " value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Excel Examples

```
wfopen "c:\data files\data.xls"
```

loads the active sheet of DATA.XLS into a new workfile.

```
wfopen(page=mypage) "c:\data files\data.xls" range="GDP data"
    @drop X
```

reads the data contained in the "GDP data" sheet of "Data.XLS" into the MYPAGE page of a new workfile. The data for the series X is dropped, and the name of the new workfile page is "GDP".

To load the Excel file containing US Macro Quarterly data from Stock and Watson's Introduction to Econometrics you may use the command:

```
wfopen
    http//wps.aw.com/wps/media/objects/3254/3332253/datasets2e/dat
    asets/USMacro_Quarterly.xls
```

which will load the Excel file directly into EViews from the publisher's website (as of 08/2009).

## HTML Files

The syntax for reading HTML pages is:
wfopen(options) source_description [table_description] [variables_description]
The following table_description elements may be used when reading an HTML file or page:

- "table = arg", where arg specifies which table to read in an HTML file/page containing multiple tables.

When specifying arg, you should remember that tables are named automatically following the pattern "Table01", "Table02", "Table03", etc. If no table name is specified, the largest table found in the file will be chosen by default. Note that the table numbering may include trivial tables that are part of the HTML content of the file, but would not normally be considered as data tables by a person viewing the page.

- "skip = int", where int is the number of rows to discard from the top of the HTML table.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.

The optional variables_description may be formed using the elements:

- "colhead = int", number of table rows to be treated as column headers.
- "namepos $=$ [first|firstatt|last|lastatt|all|none|attonly|discard|custom]", which row(s) of the column headers should be used to form the column name, and also how to use the rest. The setting "first" (or "last") refers to the object name being in the first (or last) column header row, and all other rows as the object's description. Similarly, "firstatt" (or "lastatt") will use the first (or last) row as the name field, but will use all others as a custom attribute. The setting "all" will concatenate all column header fields into the object's name. "none" will concatenate all column header fields into the object's description. "attonly" will save all column header fields into the object's custom attributes. "discard" will skip all header rows altogether, and "custom" will allow you to specify explicitly how to treat each column header row using the "colheadnames = " argument. The default setting is "all" if no "colheadnames = " is specified, otherwise "custom".
- "colheadnames = ("arg1", "arg2")", required when "namepos = custom". Specifies the name \& type of each column header row. "Name" will be mapped to the object name, "Description" to the object's description field, and the rest will be stored as custom object attributes. Any blank name will cause that column header row to be skipped.
- "nonames", the file does not contain a column header (same as "colhead $=0$ ").
- "names = ("arg1"," $\arg 2 ", \ldots) "$, user specified column names, where $\arg 1, \arg 2, \ldots$ are names of the first series, the second series, etc. when names are provided, these override any names that would otherwise be formed from the column headers.
- "descriptions = ("arg1","arg2",...)", user specified descriptions of the series. If descriptions are provided, these override any descriptions that would otherwise be read from the data.
- "types = ("arg1", "arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), " f " (numeric data), "d" (dates), or " $w$ "(EViews automatic detection). Note that the types appear without quotes: e.g., "types = (a,a,a)".
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int|all]", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the "na=" argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same rows. If the first five rows are not enough to correctly determine the data format, use the "scan = " argument to instruct EViews to look at more rows. In addition, you may want to specify a the "na= " value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs $=$ int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## HTML Examples

```
wfopen "c:\data.html"
```

loads into a new workfile the data located on the HTML file "Data.HTML" located on the C: $\backslash$ drive

```
wfopen(type=html)
    "http://www.tradingroom.com.au/apps/mkt/forex.ac" colhead=3,
    namepos=first
```

loads into a new workfile the data with the given URL located on the website site "http://www.tradingroom.com.au". The column header is set to three rows, with the first row used as names for columns, and the remaining two lines used to form the descriptions.

## Text and Binary Files

The syntax for reading text or binary files is:
wfopen(options) source_description [table_description] [variables_description]
If a table_description is not provided, EViews will attempt to read the file as a free-format text file. The following table_description elements may be used when reading a text or binary file:

- "ftype $=$ [ascii|binary]" specifies whether numbers and dates in the file are stored in a human readable text (ASCII), or machine readable (Binary) form.
- "rectype $=$ [crlf|fixed|streamed]" describes the record structure of the file:
"crlf", each row in the output table is formed using a fixed number of lines from the file (where lines are separated by carriage return/line feed sequences). This is the default setting.
"fixed", each row in the output table is formed using a fixed number of characters from the file (specified in "reclen = arg"). This setting is typically used for files that contain no line breaks.
"streamed", each row in the output table is formed by reading a fixed number of fields, skipping across lines if necessary. This option is typically used for files that contain line breaks, but where the line breaks are not relevant to how rows from the data should be formed.
- "reclines $=$ int", number of lines to use in forming each row when "rectype = crlf" (default is 1 ).
- "reclen = int", number of bytes to use in forming each row when "rectype = fixed".
- "recfields = int", number of fields to use in forming each row when "rectype = streamed".
- "skip = int", number of lines (if rectype is "crlf") or bytes (if rectype is not "crlf") to discard from the top of the file.
- "comment = string", where string is a double-quoted string, specifies one or more characters to treat as a comment indicator. When a comment indicator is found, everything on the line to the right of where the comment indicator starts is ignored.
- "emptylines = [keep|drop]", specifies whether empty lines should be ignored ("drop"), or treated as valid lines ("keep") containing missing values. The default is to ignore empty lines.
- "tabwidth = int", specifies the number of characters between tab stops when tabs are being replaced by spaces (default $=8$ ). Note that tabs are automatically replaced by spaces whenever they are not being treated as a field delimiter.
- "fieldtype = [delim|fixed|streamed|undivided]", specifies the structure of fields within a record:
"Delim", fields are separated by one or more delimiter characters
"Fixed", each field is a fixed number of characters
"Streamed", fields are read from left to right, with each field starting immediately after the previous field ends.
"Undivided", read entire record as a single series.
- "quotes = [single|double|both|none]", specifies the character used for quoting fields, where "single" is the apostrophe, "double" is the double quote character, and "both" means that either single or double quotes are allowed (default is "both"). Characters contained within quotes are never treated as delimiters.
- "singlequote", same as "quotes = single".
- "delim = [comma|tab|space|dblspace|white|dblwhite]", specifies the character(s) to treat as a delimiter. "White" means that either a tab or a space is a valid delimiter. You may also use the abbreviation " $\mathrm{d}=$ " in place of "delim = ".
- "custom = "arg1"", specifies custom delimiter characters in the double quoted string. Use the character " t " for tab, " s " for space and "a" for any character.
- "mult = [on|off]", to treat multiple delimiters as one. Default value is "on" if "delim" is "space", "dblspace", "white", or "dblwhite", and "off" otherwise.
- "endian $=$ [big|little]", selects the endianness of numeric fields contained in binary files.
- "string $=$ [nullterm|nullpad|spacepad]", specifies how strings are stored in binary files. If "nullterm", strings shorter than the field width are terminated with a single zero character. If "nullpad", strings shorter than the field width are followed by extra zero characters up to the field width. If "spacepad", strings shorter than the field width are followed by extra space characters up to the field width.
- "byrow", transpose the incoming data. This option allows you to import files where the series are contained in rows (one row per series) rather than columns.
- "lastcol", include implied last column. For lines that end with a delimiter, this option adds an additional column. When importing a CSV file, lines which have the delimiter as the last character (for example: "name, description, date"), EViews normally determines the line to have 3 columns. With the above option, EViews will determine the line to have 4 columns. Note this is not the same as a line containing "name, description, date". In this case, EViews will always determine the line to have 3 columns regardless if the option is set.

A central component of the table_description element is the format statement. You may specify the data format using the following table descriptors:

- Fortran Format:
fformat = ([n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...)
where Type specifies the underlying data type, and may be one of the following,
I - integer
F - fixed precision
E - scientific


## A - alphanumeric

X - skip
and $n 1, n 2, \ldots$ are the number of times to read using the descriptor $($ default $=1$ ). More complicated Fortran compatible variations on this format are possible.

- Column Range Format:
rformat = "[n1]Type[Width][.Precision], [n2]Type[Width][.Precision], ...) "
where optional type is " $\$$ " for string or "\#" for number, and $n 1, n 2, n 3, n 4$, etc. are the range of columns containing the data.
- C printf/scanf Format:
cformat = "fmt"
where fmt follows standard C language (printf/scanf) format rules.
The optional variables_description may be formed using the elements:
- "colhead = int", number of table rows to be treated as column headers.
- "namepos $=$ [first|firstatt|last|lastatt|all|none|attonly|discard|custom]", which row(s) of the column headers should be used to form the column name, and also how to use the rest. The setting "first" (or "last") refers to the object name being in the first (or last) column header row, and all other rows as the object's description. Similarly, "firstatt" (or "lastatt") will use the first (or last) row as the name field, but will use all others as a custom attribute. The setting "all" will concatenate all column header fields into the object's name. "none" will concatenate all column header fields into the object's description. "attonly" will save all column header fields into the object's custom attributes. "discard" will skip all header rows altogether, and "custom" will allow you to specify explicitly how to treat each column header row using the "colheadnames $=$ " argument. The default setting is "all" if no "colheadnames $=$ " is specified, otherwise "custom".
- "colheadnames = ("arg1", "arg2")", required when "namepos = custom". Specifies the name \& type of each column header row. "Name" will be mapped to the object name, "Description" to the object's description field, and the rest will be stored as custom object attributes. Any blank name will cause that column header row to be skipped.
- "nonames", the file does not contain a column header (same as "colhead $=0$ ").
- "names = ("arg1", "arg2",...)", user specified column names, where $\arg 1, \arg 2, \ldots$ are names of the first series, the second series, etc. when names are provided, these override any names that would otherwise be formed from the column headers.
- "descriptions = ("arg1", "arg2",...)", user specified descriptions of the series. If descriptions are provided, these override any descriptions that would otherwise be read from the data.
- "types = ("arg1","arg2",..)", user specified data types of the series. If types are provided they will override the types automatically detected by EViews. You may use any of the following format keywords: "a" (character data), "f" (numeric data), "d" (dates), or "w" (EViews automatic detection). Note that the types appear without quotes: e.g., "types = (a,a,a)".
- "na = "arg1"", text used to represent observations that are missing from the file. The text should be enclosed on double quotes.
- "scan = [int|all]", number of rows of the table to scan during automatic format detection ("scan = all" scans the entire file). Note: If a "range = " argument is not specified, then EViews will only scan the first five rows of data to try and determine the data format for each column. Likewise, if the " $n a=$ " argument is not specified, EViews will also try to determine possible NA values by looking for repeated values in the same rows. If the first five rows are not enough to correctly determine the data format, use the "scan =" argument to instruct EViews to look at more rows. In addition, you may want to specify a the " $n a=$ " value to override any dynamic NA value that EViews may determine on its own.
- "firstobs = int", first observation to be imported from the table of data (default is 1 ). This option may be used to start reading rows from partway through the table.
- "lastobs = int", last observation to be read from the table of data (default is last observation of the file). This option may be used to read only part of the file, which may be useful for testing.


## Text and Binary File Examples (.txt, .csv, etc.)

```
wfopen c:\data.csv skip=5, names=(gdp, inv, cons)
```

reads "Data.CSV" into a new workfile page, skipping the first 5 rows and naming the series GDP, INV, and CONS.

```
wfopen(type=text) c:\date.txt delim=comma
```

loads the comma delimited data DATE.TXT into a new workfile.

```
wfopen(type=raw) c:\data.txt skip=8, rectype=fixed,
    format=(F10,X23,A4)
```

loads a text file with fixed length data into a new workfile, skipping the first 8 rows. The reading is done as follows: read the first 10 characters as a fixed precision number, after that, skip the next 23 characters (X23), and then read the next 4 characters as strings (A4).

```
wfopen(type=raw) c:\data.txt rectype=fixed, format=2(4F8,2I2)
```

loads the text file as a workfile using the specified explicit format. The data will be a repeat of four fixed precision numbers of length 8 and two integers of length 2 . This is the same description as "format $=(\mathrm{F} 8, \mathrm{~F} 8, \mathrm{~F} 8, \mathrm{~F} 8, \mathrm{I} 2, \mathrm{I} 2, \mathrm{~F} 8, \mathrm{~F} 8, \mathrm{~F} 8, \mathrm{~F} 8, \mathrm{I} 2, \mathrm{I} 2)$ ".

```
wfopen(type=raw) c:\data.txt rectype=fixed, rformat="GDP 1-2 INV 3
    CONS 6-9"
```

loads the text file as a workfile using column range syntax. The reading is done as follows: the first series is located at the first and second character of each row, the second series occupies the 3rd character, the third series is located at character 6 through 9 . The series will named GDP, INV, and CONS.

## Datasets

The syntax for reading data from the remaining sources (Access, Gauss, ODBC, SAS program, SAS transport, SPSS, SPSS portable, Stata) is:
wfopen(options) source_description table_description [@keep keep_list] [@drop drop_list] [@selectif condition]

Note that for this purpose we view Access and ODBC as datasets.

## ODBC or Microsoft Access

The syntax for reading from an ODBC or Microsoft Access data source is wfopen(options) source_description table_description [@keep keep_list] [@drop drop_list] [@selectif condition]

When reading from an ODBC or Microsoft Access data source, you must provide a table_description to indicate the table of data to be read. You may provide this information in one of two ways: by entering the name of a table in the data source, or by including an SQL query statement enclosed in double quotes.

Note that ODBC support is provided only in the EViews Enterprise Edition.

## ODBC Examples

```
wfopen c:\data.dsn CustomerTable
```

opens in a new workfile the table named CUSTOMERTABLE from the ODBC database described in the DATA.DSN file.

```
wfopen(type=odbc) "my server" "select * from customers where id>30"
    @keep p*
```

opens in a new workfile with SQL query from database using the server "MY SERVER", keeping only variables that begin with $P$. The query selects all variables from the table CUSTOMERS where the ID variable takes a value greater than 30 .

## Other Dataset Types

The syntax for reading data from the remaining sources (Gauss, SAS program, SAS transport, SPSS, SPSS portable, Stata) is:
wfopen(options) source_description [@keep keep_list] [@drop drop_list] [@selectif condition]

Note that no table_description is required.

## SAS Program Example

If a data file, "Sales.DAT", contains the following space delimited data:
AZ 1101002
CA 2002003
NM 90908
OR 120708
WA 1131123
UT 98987
then the following SAS program file can be read by EViews to open the data:

```
Data sales;
    infile sales.dat';
    input state $ price sales;
run;
```


## SAS Transport Examples

```
wfopen(type=sasxport) c:\data.xpt
```

loads a SAS transport file "data.XPT" into a new workfile.

```
wfopen c:\inst.sas
```

creates a workfile by reading from external data using the SAS program statements in "Inst.SAS". The program may contain a limited set of SAS statements which are commonly used in reading in a data file.

## Stata Examples

To load a Stata file "Data.DTA" into a new workfile, dropping maps MAP1 and MAP2, you may enter:

```
wfopen c:\data.dta @dropmap map1 map2
```

To download the sports cards dataset from Stock and Watson's Introduction to Econometrics you may use the command:

```
wfopen
    http://wps.aw.com/wps/media/objects/3254/3332253/datasets2e/da
    tasets/Sportscards.dta
```

which will load the Stata dataset directly into EViews from the publisher's website (as of 08/2009).

## Cross-references

See Chapter 3. "Workfile Basics," on page 41 of User's Guide I for a discussion of workfiles.
See also pageload (p. 228), read (p. 478), fetch (p. 363), wfsave (p. 293), wfclose (p. 531) and pagesave (p. 229).

## wfsave

Object Container, Data, and File Commands

Save the default workfile as an EViews workfile (.wf1 file) or as a foreign file or data source.

## Syntax

wfsave(options) [path $\backslash$ ]filename
wfsave(options) source_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
wfsave(options) source_description table_description [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
saves the active workfile in the specified directory using filename. By default, the workfile is saved as an EViews workfile, but options may be used to save all or part of the active page in a foreign file or data source.

When saving to a foreign data file, the basic specification consists of a "type = " option and source_description and table_description arguments which specify the format of the foreign data file. See below for details on source_description and table_description.

The remaining optional elements specify the actual elements to be saved.

## Options

Workfile (WF1) Save Options
1 Save using single precision.
2 Save using double precision.
c Save compressed workfile (not compatible with EViews versions prior to 5.0).

## Workfile (WF2) Save Options

| jf | Save JSON with formatting making it easier to read. <br> Increases file size. |
| :--- | :--- |
| nojf | Saves JSON without any formatting. Minimizes file size. |
| gzip | Saves JSON as a compressed gzip file. Minimizes file size. |
| nogzip | Saves JSON without any gzip compression (i.e., simple text <br> file). Increases file size. |

The default workfile save settings use the global options.

## Foreign Source Save Options

These options only apply when saving your file to a format other than an EViews workfile. note that some of the options only apply to specific file types.

| type = arg, t = arg | Optional type specification: (see table below). <br> Note that ODBC support is provided only in the EViews <br> Enterprise Edition. |
| :--- | :--- |
| mode = arg | Specify whether to create a new file, overwrite an existing <br> file, or update an existing file. arg may be "create" (create <br> new file only; error on attempt to overwrite) or "update" <br> (update an existing file, only overwriting the area specified <br> by the range = table_description). <br> If a "mode =" option is not provided, EViews will create a <br> new file, unless the file already exists in which case it will <br> overwrite it. |
| Note that the "mode = update" option is only available for: <br> 1) Excel versions through 2003, if Excel is installed, and 2) <br> Excel 2007 (xml). |  |
| maptype =arg | Write selected maps as: numeric ("n"), character ("c"), <br> both numeric and character ("b"). |
| nomapval | Do not write mapped values for series with attached value <br> labels (the default is to write the mapped values if avail- <br> able). |
| noidDo not write observation identifiers to foreign data files (by <br> default, EViews will include a column with the date or <br> observation identifier). |  |


| nonames | Do not write variable names (only applicable to file for- <br> mats that support writing raw data without variable <br> names). |
| :--- | :--- |
| na $=\arg$ | String value to be used for NAs. |
| attr | Include object attributes (if the output type supports it). <br> When specified, the first column will contain attribute <br> names and each attribute value will be displayed after the <br> name row. |

The following table summaries the various formats along with the corresponding "type = " keywords:

|  | Type Keywords | Supports Attributes |
| :---: | :---: | :---: |
| Access | "access" |  |
| Aremos-TSD | "a", "aremos", "tsd" |  |
| Binary | "binary" |  |
| dBASE | "dbase" |  |
| Excel (through 2003) | "excel" | Yes |
| Excel 2007 (xml) | "excelxml" | Yes |
| EViews Workfile | --- |  |
| Gauss Dataset | "gauss" |  |
| GiveWin/PcGive | "g", "give" |  |
| HTML | "html" |  |
| JSON | "json" |  |
| JSON (legacy output generated by EViews prior to EViews 12) | "jsonlegacy" |  |
| EViews workfile (WF1) | "wf1" |  |
| EViews workfile (WF2) | "wf2" |  |
| Lotus 1-2-3 | "lotus" |  |
| ODBC Dsn File | "dsn" |  |
| ODBC Data Source | "odbc" |  |
| MicroTSP Workfile | "dos", "microtsp" |  |
| MicroTSP Mac Workfile | "mac" |  |
| RATS 4.x | "r", "rats" |  |
| RATS Portable / TROLL | "l", "trl" |  |
| SAS Program | "sasprog" |  |


| SAS Transport | "sasxport" |
| :--- | :---: |
| SPSS | "spss" |
| SPSS Portable | "spssport" |
| Stata (Version 7 Format) | "stata" |
| Tableau Data Extract | tde |
| Text / ASCII | "text" |
| TSP Portable | "t", "tsp" |

Note that if you wish to save your Excel 2007 XML file with macros enabled, you should specify the explicit filename extension ".XLSM".

## Foreign Data Descriptions

When saving to a foreign data format the base specification consists of a basic specification of source_description and table_description which specify the exact details of the format.

The command for saving as foreign data formats is
wfsave(options) source_description [table_description] [@keep keep_list] [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
where the syntax of the table_description and variables_description differs slightly depending on the type of file.

- Note that saving as a foreign data file, with the exception of JSON, will save the current workfile page only.
- The JSON type will save all series objects from all pages of the current working, ignoring any @keep, @drop, and @smpl arguments.


## Excel Files

The base syntax for writing Excel files is:
wfsave(options) source_description [table_description]
where source_description is the path and name of the Excel file to be saved, and where the following table_description elements may be employed:

- "range = arg", where arg is a range of cells to read from the Excel workbook, following the standard Excel format [worksheet!][topleft_cell[:bottomright_cell]].
If the worksheet name contains spaces, it should be placed in single quotes. If the worksheet name is omitted, the cell range is assumed to refer to the currently active sheet. If only a top left cell is provided, a bottom right cell will be chosen automatically to cover the range of non-empty cells adjacent to the specified top left cell. If only a sheet name is provided, the first set of non-empty cells in the top left corner of the chosen worksheet will be selected automatically. As an alternative to specifying
an explicit range, a name which has been defined inside the excel workbook to refer to a range or cell may be used to specify the cells to read.
- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.


## HTML Files

The base syntax for saving HTML files is:
wfsave(options) source_description [table_description]
where source_description is the path and name of the file to be saved, and where the following table_description element may be employed:

- "byrow", transpose the incoming data. This option allows you to read files where the series are contained in rows (one row per series) rather than columns.


## Text and Binary and Other Files

The base syntax for saving other files is:
wfsave(options) source_description
where source_description is the path and name of the file to be saved.

## Examples

## EViews Workfile Examples

wfsave new_wf
saves the current EViews workfile as "New_wf.WF1" in the default directory.

```
wfsave "c:\documents and settings\my data\consump"
```

saves the current workfile as "Consump.WF1" in the specified path.
wfsave macro @keep gdp unemp
saves the two series GDP and UNEMP in a separate workfile, "macro.WF1" in the default directory.

```
wfsave macro @dropmap gdp*
```

saves all of the series in the current workfile, other than those that match the name pattern "gdp*" in a workfile, "macro.WF1" in the default directory.

The command:

```
wfsave "<mydropboxdrive>"\folder\nipa.wf1"
```

will save the file to the cloud location MYDROPBOXDRIVE.

## Foreign Data Examples

```
wfsave(type=excelxml, mode=update) macro.xlsx
```

saves the current workfile page as a modern Excel ".XLSX" file.

```
wfsave(type=excelxml, mode=update) macro.xlsx range="Sheet2!a1"
    byrow @keep gdp unemp
```

will save the two series GDP and UNEMP into the existing Excel file "macro.XLSX", specifying that the series should be written by row, starting in cell A1 on sheet Sheet2.

To save the latter file in a macro-enabled Excel 2007 file, you should specify the explicit filename extension ".XLSM",

```
wfsave(type=excelxml, mode=update) macro.xlsm range="Sheet2!a1"
    byrow @keep gdp unemp
```

Alternately,

```
wfsave(type=excelxml, noid) macro.xlsx range="Sheet2!a1"
```

will save the current workfile page as the Excel file "macro.XLSX" but will not include a column of dates.

If you wish to save a column of dates in a specific date format, you can do so by first creating an alpha series in the workfile with the specified format, then saving the file with the "noid" option including that alpha series:

```
alpha mydates = @datestr(@date, "YYYY-MM-DD")
wfsave(type=excelxml, noid) macro.xlsm range="Sheet2!a1" @keep
    mydates gdp unemp
```

will save the series GDP and UNEMP into the Excel file "macro.XLSM" along with a date series with the format "YYYY-MM-DD".

## Cross-references

See Chapter 3. "Workfile Basics," on page 41 of the User's Guide I for a discussion of workfiles.

See also pagesave (p. 229), wfopen (p. 278), and pageload (p. 228).

| Xopen | External Interface Commands |
| :--- | :--- |

Open a connection to an external application.

## Syntax

xopen(options)
xopen is used to start a COM session with an external application, either R or MATLAB. EViews can only have a single session open at a time; a session must be closed (see xopen (p. 298)) before a new session can be opened.

## Options

$$
\begin{array}{ll}
\text { type = arg } & \begin{array}{l}
\text { Set the type of connection to be opened. arg may be "r" (R) } \\
\text { or "m" (MATLAB). }
\end{array} \\
\text { keepcurrent } & \begin{array}{l}
\text { If "type =" is the same as a currently open connection, } \\
\text { keep original connection since it is already open. } \\
\text { (optional) Set the version of MATLAB or statconnDCOM to } \\
\text { which EViews connects when opening a session. If not } \\
\text { specified, EViews will use the default ProgID specified in } \\
\text { the global options. }
\end{array} \\
\text { nolog } & \begin{array}{l}
\text { Do not open a session log window. }
\end{array} \\
\text { case }=\text { arg } & \begin{array}{l}
\text { Specify the default case for objects in R or MATLAB using } \\
\text { xput (p. 572). If "case =" is not provided, the default case } \\
\text { specified in the global options will be assumed. Note that } \\
\text { once a connection has been opened, the case option cannot } \\
\text { be changed; you may however, use the "name = "option } \\
\text { when using xput (p. 572) to provide an explicit name. }
\end{array}
\end{array}
$$

Note that the MATLAB ProgIDs may be of particular interest as MATLAB (R2008a and later) offers several distinct ways in which to connect to the server. The relevant ProgIDs are:

1. "MATLAB.Application" - this ProgID starts a command window version of MATLAB that was most recently used as a server (might not be the latest installed version of MATLAB).
2. "MATLAB.Application.Single"- same as (1) but starts a dedicated server so that other programs looking to use MATLAB cannot connect to your instance.
3. "MATLAB.Autoserver"-starts a command window server using the most recent version of MATLAB.
4. "MATLAB.Autoserver.Single"-same as (3) but starts a dedicated server.
5. "MATLAB.Desktop.Application"-starts the full desktop MATLAB as a server using the most recent version of MATLAB.

Each ProgID may be amended to indicate a specific version of MATLAB. For example, using the ProgID:

MATLAB.Desktop.Application.7.6
instructs EViews to use the full desktop MATLAB GUI for version R2008a (v7.6) as the Automation server.

## Examples

```
xopen(type=m)
```

opens a connection to MATLAB.

```
xopen(type=r, case=lower)
```

opens a connection to R and sets the default name-case to lower.

```
xopen(type=m, progid=MATLAB.Desktop.Application.7.9)
```

opens a connection to MATLAB 7.9 running with the full desktop GUI.

## Cross-references

See "EViews COM Automation Client Support (MATLAB, R, Python)," beginning on page 179 for discussion. See also "External Program Interface" on page 959 of User's Guide I for global options setting.

See xclose (p. 565), xget (p. 565), xput (p. 572), xrun (p. 574), and xlog (p. 568).

## Daily Seasonal Adjustment

Daily Seasonal Adjustment (DSA) models daily time series data that contains three distinct seasonalities-a day-of-the-week (DoW) effect, a day-of-the-month (DoM) effect, and a day-of-the-year (DoY) effect, as well as potential holiday or event/calendar effects.

EViews offers an implementation of the seasonal adjustment of daily time series algorithm of Ollech (2021).

## Background

Although the original Ollech (2021) algorithm is designed for 7-day week daily data, EViews' implementation handles both 7 and 5-day week daily data.

## 7-day Seasonal Adjustment

The seasonal adjustment process of 7-day weekly data with DSA can be broken down into five stages:

- Stage 1: Adjusting for the day-of-the-week effect using STL decomposition.
- Stage 2: Remove holiday and calendar effects using ARIMA modeling.
- Stage 3: Adjusting for the day-of-the-month effect using STL decomposition.
- Stage 4: Adjusting for the day-of-the-year effect using STL decomposition.
- Stage 5: Combining the adjustments and performing any out-of-sample forecasts.

Each of these stages may further be broken down into individual steps.

## Stage 1: Day-of-the-week adjustment

The first stage adjusts for the DoW effect by performing STL decomposition on the original data with a periodicity set to 7 .

## Stage 2: Holiday and calendar effects

The holiday and calendar effects are modeled using a seasonal ARIMA process. This can both model the impact of known events, or detect and model unknown effects using outlier detection.

Following Ollech (2021), EViews approximates the yearly seasonal ARIMA terms using trigonometric functions composed of a series of sines and cosines as exogenous regressors in an ARIMA model:

$$
\begin{equation*}
\hat{S}_{t}=\sum_{j=1}^{J}\left(\left(\hat{\beta}_{1, j} \cdot \sin \left(\frac{2 \pi j G(t)}{365}\right)\right)+\left(\hat{\beta}_{2, j} \cdot \cos \left(\frac{2 \pi j G(t)}{365}\right)\right)\right) \tag{0.1}
\end{equation*}
$$

Where $S$ denotes the exogenous terms in an ARIMA, and $G(t)$ is a indicator function denoting the day of the year, i.e., cycles through $1, \ldots, 365$. The optimal number of trigonometric terms, $J$, is determine by information criteria. Note that this approximation assumes that there are exactly 365 days in a year, which requires special handling of leap years.

The order of the ARIMA model can also be determined by information criteria.
The full set of steps in Stage 2 are:
a. Remove all February 29 (leap year) observations from the DoW adjusted data produced in Stage 1.
b. Determine the optimal order of an ARIMA model using information criteria on the data created in step a), with an initial large set of trigonometric terms and any userspecified holiday, calendar or other variables as exogenous variables.
c. Determine the optimal number of trigonometric terms using information criteria based on an ARIMA model with fixed order, as determined in step b).
d. Having determined the optimal ARIMA order and number of trigonometric terms, detect outliers in the ARIMA process using the time-series outlier detection method of Chen and Liu (1993) (see below for details).
e. Estimate a final ARIMA model including effects for the detected outliers in step d) and estimate fitted values to produce a series with calendar and holiday effects removed.
f. If forecasting is required, forecast from the ARIMA model over the forecasting period.

## Stage 3: Day-of-the-month adjustment

This stage takes the final data from Stage 2 and performs an STL decomposition with a periodicity of 31 to adjust for the DoM effect. This process requires that every month has exactly 31 days, and so for months with fewer than 31 days, interpolated values are inserted to extend the month to 31. The steps in Stage 3 are then:
a. Expand the adjusted data from Stage 2 so that every month has 31 days.
b. Interpolate, with a cubic spline, the inserted observations in step a).
c. Perform STL with a 31 periodicity to produce DoM adjusted data.
d. Remove the inserted observations from step a).

## Stage 4: Day-of-the-year adjustment

The final seasonal adjustment performs an STL decomposition on the adjusted data from Stage 3 with a periodicity of 365 to adjust for the DoY effect.

## Stage 5: Combining and forecasting

The final stage brings together the results from the previous stages and performs any forecasting. The steps in this stage are:
a. Re-insert observations for February 29, and use a cubic spline to interpolate the adjusted values from Stage 4.
b. Any holiday or event/calendar effects removed in Stage 2 are added back into the adjusted data. This produces the final seasonally adjusted (and forecasted data).
c. The final trend estimate is produced by LOESS estimation of the final adjusted data against a simple time trend.
d. The in-sample final seasonal factors are computed as the original data minus the adjusted data.
e. To produce forecasts of the seasonal factors, a forecast of the original data is required. This is computed from the ARIMA forecast in Stage 2(f), adjusted to add back the holiday/calendar effects, and the day-of-the-week seasonal factors (which are themselves forecasted using exponential smoothing, or by extending the last week of calculated seasonal factors repeated throughout the forecast).
f. Individual DoW, DoM and DoY trends and factors can be retrieved from the corresponding STL decompositions.

## Outlier Detection

Stage 2 includes automatic outlier detection and correction in the ARIMA model based upon the method given in Chen and Liu (1993). In this paper, four different types of outlier effects are identified: Innovational Outlier (IO), Additive Outlier (AO), Level Shift (LS), and Temporary Change (TC).

For precise details of the procedure, we refer the reader to the original article, however as a short outline, the procedure follows these steps:
a. Calculate residuals from an initial ARIMA estimation.
b. At each time period of the estimation sample, calculate standardized statistics for each type of outlier effect.
c. If the maximum value of the statistics is greater than some critical value (cvalue), remove the effect of that outlier from the residuals, and then re-calculate the statistics in step b) using the adjusted residuals.
d. Iterate through step c) until no more outlier effects are detected at the critical value.
e. Re-estimate the ARIMA model, including outlier effects for the types and dates identified in step d) then return to step b).
f. Iterate through step e) until no new outliers are detected.

## 5-Day Adjustment

The original DSA algorithm is designed to handle data that are reported seven days (D7) a week. However much economic data is only reported for five days (D5) of the week, with no data available on weekends. Indeed, the application of DSA in Ollech (2021) uses five day data, and expands out the data to seven day by repeating the Friday value for Saturday and Sunday before performing DSA.

The implementation of DSA in EViews offers three alternatives for handling D5 data:

- Extend the D5 data to be D7 and use the Friday value as the value for Saturday/Sunday (as per Ollech (2021)).
- Extend the D5 data to be D7 using interpolation between Friday and Monday to fill in Saturday/Sunday values.
- Perform DSA on the D5 data itself where the daily STL has a periodicity of 5, the monthly STL has a periodicity of 23 and the yearly STL is 262 . For months/years with fewer than $23 / 262$ observations, insert observations and interpolate the missing values.


## Performing Daily Seasonal Adjustment in EViews

To perform DSA seasonal adjustment in EViews, open the series and select Proc/Seasonal Adjustment/DSA Daily Seasonal Adjustment... EViews will then open a tree-structured DSA dialog to allow you to set the options for the DSA procedure:


The branches of the tree, on the left, allow you to specify the Basic Options, the ARIMA model, and the three STL Seasonal Adjustment components. Click on the node name in the left to select the node.

## Basic Options

Within the Basic Options node:

- You may use the Forecast end edit field to specify the end of the forecast period. The start of the forecast period will be the first observation after the current workfile sample.

If this field is blank, EViews will perform seasonal adjustment of the series over the current workfile sample, and will not forecast beyond the sample.

If the field is filled the Day-of-week factor forecast options will be enabled, which specify how the day of the week factor should be forecasted (using either exponential smoothing, or simply extending the last week of data).

- The Output series names edit field may be used to specify the names of the output series from the procedure. If this edit field is left blank, EViews will not output the respective series.

If you workfile is structured as Daily 5-day week, the Five day week options dropdown can be used to specify how EViews will treat weekends.

- Selecting Ignore weekends will perform DSA seasonal adjustment as though there are 262 days in each year and 23 days in each month (using interpolation to create observations for years/months with fewer days).
- Selecting either Expand weekends using Friday value or Expand weekends using interpolation will convert the data to a Daily 7-day week format, and fill-in the weekend values using either the value from the previous Friday, or will interpolate between the Friday and the following Monday value.


## ARIMA

The ARIMA node offers options for the ARIMA model estimated in Stage 2. The options are provided in separate dialog nodes: Specification, Variables, Outliers.

## ARIMA Specification

The Specification node controls the specification of the estimated ARIMA model:


The Model Specification radio button allows you to choose between whether to tell EViews to use Automatic selection to determine the appropriate order of the ARIMA model, or whether to use the Fixed model with user-specified order.

- The Max. difference, Max. AR and Max. MA dropdown menus specify the maximum orders to test if performing automatic selection, or the specified order if a fixed model is selected.
- When performing automatic selection, the Criteria dropdown allows you to specify the information criteria used to determine the most appropriate ARIMA number of trigonometric terms.


## ARIMA Variables

The variables node is used to specify exogenous regressors used in the ARIMA estimation.


- The Trig. seasonal terms section selects the number of trigonometric seasonal dummies to include in the ARIMA estimation, as defined in Equation (0.1). You may select whether to determine automatically the number of terms using an information criterion, or you can fix the number at a specified value. Note the criterion employed for selecting the number of seasonal terms is specified in the ARIMA specification tab ("ARIMA Specification" on page 306).
- The User-supplied regressors box is used to enter any additional regressors you wish to include in the ARIMA estimation. Typically, variables modeling holiday patterns are used within the ARIMA estimation, and the built-in EViews function @holiday can be used to create sophisticated holiday variables.


## ARIMA Outliers

Options for automatic detection of outliers in the ARIMA model can be set on the Outliers node.


The Outlier types check boxes select which types of outlier to detect and model in the ARIMA, whereas the Parameters edit fields specify the parameters used during the detection process. The Critical value field specifies the critical value used to determine whether an observation contains an outlier, Delta is used in the specification of a Temporary change outlier, and the two iterations fields are used to specify the maximum number of iterations of the inner and outlier loops of the procedure.

## Seasonal Adjustment

The three Seasonal Adjustment nodes specify options for the Day-of-week, Day-of-month, and Day-of-year STL decompositions. The dialog for all three types is the same:


The Perform day-of-week adjustment (or equivalent for the other month or year nodes) specifies whether to include seasonal adjustment at this periodicity. The Polynomial degree, Smoothing window, and Iteration control sections specify options for the STL procedure, see the STL section of UG1 for details.

The Output section allows you to save the outputs of the individual STL procedures as separate series. Select the desired output and then enter a name for the output series.

## Example

As an example of using daily seasonal adjustment, we will work with a time-series containing daily electricity demand data for England and Wales between 2005 and 2014. The data are contained in the workfile "Elecdmd.wf1", which contains the single series ELECDMD. The workfile extends until the end of 2015, leaving 20 months of NAs within the ELECDMD series.

Viewing a graph of this series, we can see that there are strong seasonal patterns to these data:


Not only does there appear to be a monthly pattern to electricity demand (with higher demand in the winter, as electricity is used for heating), but, if we use the slider on the bottom of the window to zoom in on a few months of data, we can see there is also a day of the week pattern:


We will use the DSA tools to remove both the seasonal and the weekly patterns. We begin by re-setting the workfile sample to begin and the workfile start and end on April 30, 2014 by issuing the command (assuming "Month/Day/Year" data handling):

```
smpl @first 04/30/2014
```

Next we open the ELECDMD series and click on Proc/Seasonal Adjustment/DSA Daily Seasonal Adjustment to open the DSA dialog. EViews will specify a forecast end point of December 31, 2015, which is the last date in our workfile. We keep this and the remaining options in the dialog at their default values. Click OK to perform the adjustment.

Further, the output from the DSA will be displayed in a spool. The output is lengthy, containing detailed information on each part of the adjustment process.


For example, the Day-of-Week STL displays a Trend-Seasonal-Adjusted graph for the Day-ofWeek seasonal adjustment,

while the Auto-ARMA Order node displays a graph of the ARMA selection criterion,
Order Quality Ranking

and the Outliers and ARIMA Estimates show information on the outlier identification process, and the estimated ARIMA process:


Lastly, the output series will be saved in the workfile, in the case, in the series ELECDMADJ. The series will contain the DSA adjusted data, which when plotted against the original ELECDMD for the whole sample shows the effect of removing the monthly seasonality,

and when looking at a smaller time-frame, the effects of removing within-week seasonality


## References

Chen, C. and L.-M. Liu (1993) "Joint Estimation of Model Parameters and Outlier Effects in Time Series," Journal of the American Statistical Association, 88(421), 284-297.
Ollech, Daniel (2021) "Seasonal Adjustment of Daily Time Series," Journal of Time Series Econometrics, 13(2), 235-264.

## Linear and Nonlinear ARDL

Autoregressive Distributed Lag (ARDL) models, are linear time series models (Pesaran, 1998 and 2001) in which the dependent and independent variables are related contemporaneously and across historical (lagged) values.

EViews offers powerful time-saving tools for estimating and examining the properties of Autoregressive Distributed Lag (ARDL) models. ARDLs are standard least squares regressions that include lags of both the dependent variable and explanatory variables as regressors (Greene, 2008). Although ARDL models have been used in econometrics for decades, they have gained popularity in recent years as a method of examining cointegrating relationships between variables through the work of Pesaran and Shin (PS 1998) and Pesaran, Shin and Smith (PSS 2001).

While it is possible to use a standard least squares procedure to estimate an ARDL, the specialized ARDL estimator in EViews offers a number of useful features including model selection and the computation of post-estimation diagnostics.

## Background

If $y_{t}$ is the dependent (autoregressive) variable, $x_{1, t}, \ldots, x_{k, t}$ are $k$ distributed-lag explanatory variables, and $d_{1, t}, \ldots, d_{m, t}$ are $m$ exogenous, potentially deterministic variables, the Intertemporal Dynamics (ITD) representation of an $\operatorname{ARDL}\left(p, q_{1}, \ldots, q_{k}\right)$ model is given by:

$$
\begin{equation*}
y_{t}=\sum_{j=1}^{p} \psi_{j} y_{t-j}+\sum_{r=1}^{k} \sum_{j=0}^{p_{r}} \beta_{r, j} x_{r, t-j}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t} \tag{0.2}
\end{equation*}
$$

where $\epsilon_{t}$ are the innovations, and $\alpha_{s}, \psi_{j}$, and $\beta_{r, j}$ are the coefficients associated with the exogenous variables, $p$ lags of $y_{t}$, and $p_{r}$ lags of the $k$ distributed lag regressors $x_{r}$, respectively.

Let $L$ be the usual lag operator and define the lag polynomials:

$$
\begin{align*}
& \psi(L)=1-\sum_{j=1}^{p} \psi_{j} L^{j}  \tag{0.3}\\
& \beta_{r}(L)=\sum_{j=0}^{p_{r}} \beta_{r, j} L^{j}
\end{align*}
$$

Substituting into Equation (0.2) yields:

$$
\begin{equation*}
\psi(L) y_{t}=\sum_{r=1}^{k} \beta_{r}(L) x_{r, t}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t} \tag{0.4}
\end{equation*}
$$

Noting that any series $z_{t}$ may be written as $z_{t}=z_{t-1}+\Delta z_{t}$ and performing a BeveridgeNelson decomposition on both $\psi(L)$ and the $\beta_{r}(L)$ in Equation (0.4) produces the Conditional Error Correction (CEC) representation of the ARDL,

$$
\begin{align*}
& \Delta y_{t}=-\psi(1) y_{t-1}+(1-L) \tilde{\psi}(L) y_{t}+\sum_{r=1}^{k} \beta_{r}(1) x_{r, t-1}+  \tag{0.5}\\
& \sum_{r=1}^{k}\left(\beta_{r}(1)+\tilde{\beta}_{r}(L)\right) \Delta x_{r, t}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

which, with a bit of manipulation, may be rewritten as

$$
\begin{align*}
& \Delta y_{t}=-\phi y_{t-1}+\sum_{r=1}^{k} \lambda_{r} x_{r, t-1}+\sum_{j=1}^{p} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k} \eta_{r} \Delta x_{r, t}+  \tag{0.6}\\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}} \delta_{r, j} \Delta x_{r, t-j}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

where

$$
\begin{align*}
\lambda_{r} & =\beta_{r}(1)=\sum_{j=0}^{p_{r}} \beta_{r, j} \\
\delta_{r, m} & =-\sum_{j=m+1}^{p_{r}} \beta_{r, j}  \tag{0.7}\\
\tilde{\beta}_{r}(L) & =\sum_{j=0}^{p_{r}-1} \delta_{r, j} L^{j}
\end{align*}
$$

and

$$
\begin{gather*}
\eta_{j}=\lambda_{j}+\delta_{j, 0} \\
\phi=\psi(1)=\sum_{j=1}^{p} \psi_{j} \\
\gamma_{m}=-\sum_{j=m+1}^{p-1} \psi_{j}  \tag{0.8}\\
\tilde{\psi}(L)=\sum_{j} \gamma_{j} L^{j}
\end{gather*}
$$

Since CEC Equation (0.6) and Equation (0.9) are derived from ITD Equation (0.2), there is an obvious one-to-one correspondence between the two. As with the vector error correction (VEC) form of a VAR, the CEC form offers easy identification of a cointegrating relationship between the dependent variable and the explanatory variables in the ARDL. We discuss this parallel in greater depth in "Relationship to Vector Error Correction (VEC) Models," on page 318 .

Rearrange terms, we may re-write Equation (0.9) as

$$
\begin{aligned}
& \Delta y_{t}=-\phi\left(y_{t-1}-\sum_{r=1}^{k} \frac{\lambda_{r}}{\phi} x_{r, t-1}\right)+\sum_{j=1}^{p} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k} \eta_{r} \Delta x_{r, t}+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}} \delta_{r, j} \Delta x_{r, t-j}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{aligned}
$$

If we define the equilibrium error correction term,

$$
\begin{equation*}
E C_{t}=y_{t-1}-\sum_{r=1}^{k} \frac{\lambda_{r}}{\phi} x_{r, t-1} \tag{0.10}
\end{equation*}
$$

then Equation (0.9) may be written in Error Correction (EC) form:

$$
\begin{aligned}
& \Delta y_{t}=-\phi E C_{t}+\sum_{j=1}^{p} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k} \eta_{r} \Delta x_{r, t}+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}} \delta_{r, j} \Delta x_{r, t-j}+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{aligned}
$$

where $\phi$ is the error correction parameter, and the long-term equilibrium parameters for the explanatory variables are given by $\lambda_{r} / \phi$, for $r=1, \ldots, k$.

Conveniently, the coefficients in both the ITD and the CEC representations of the ARDL model may be estimated via least squares.

## Relationship to Vector Error Correction (VEC) Models

Assuming the same lag across the distributed-lag regressors $x_{r, t}$ and that the deterministics $d_{m, t}$ consist of a simple constant and linear trend, Pesaran (2001) demonstrates that the ARDL CEC representation in Equation (0.9) is in fact the $\operatorname{CEC}$ of the $\operatorname{VAR}(p)$ model:

$$
\begin{equation*}
\Phi(L)\left(z_{t}-\mu-\xi t\right)=u_{t} \tag{0.12}
\end{equation*}
$$

where
$z_{t}=\left(y_{t}, x_{1, t}, \ldots, x_{k, t}\right)^{\prime}$ is a $k+1$ vector of endogenous variables, $\mu$ and $\xi$ are the $k+1$ vectors of intercept and trend coefficients, respectively, and

$$
\begin{equation*}
\Phi(L)=I_{k+1}-\sum_{j=1}^{p} \Phi_{j} L^{j} \tag{0.13}
\end{equation*}
$$

is the $k+1$ matrix lag polynomial.
Invoking the BN decomposition on $\Phi(L)$ and with following some rearrangement, the CEC representation of this VEC may be written as

$$
\begin{align*}
\Delta z_{t} & =-\boldsymbol{\Phi}(1) z_{t-1}+\sum_{j=1}^{p-1} \tilde{\boldsymbol{\Phi}}_{j} \Delta z_{t-j}+\boldsymbol{\Phi}(1) \mu+\left(\sum_{j=1}^{p} i \boldsymbol{\Phi}_{j}\right) \xi+\boldsymbol{\Phi}(1) \xi t+u_{t} \\
& =\rho z_{t-1}+\sum_{j=1}^{p-1} \tilde{\boldsymbol{\Phi}}_{j} \Delta z_{t-j}+\alpha_{0}+\alpha_{1} t+u_{t} \tag{0.14}
\end{align*}
$$

where

$$
\begin{align*}
\Phi(1) & =I_{k+1}-\sum_{j=1}^{p} \Phi_{j} \\
\rho & =-\Phi(1) \\
\tilde{\boldsymbol{\Phi}}_{m} & =-\sum_{j=m+1}^{p} \Phi_{j}  \tag{0.15}\\
\alpha_{0} & =\Phi(1) \mu+\left(\sum_{j=1}^{p} i \boldsymbol{\Phi}_{j}\right) \xi \\
\alpha_{1} & =\Phi(1) \xi
\end{align*}
$$

which is equivalent to Equation (0.6).

## Nonlinear (asymmetric) ARDL

The classical ARDL framework assumes that the long-run relationship $E C_{t}$ is a symmetric linear combination of regressors. While this is a natural starting assumption, it does not match the behavioral finance and economics literature approach to modeling nonlinearity and asymmetry (Kahneman, Tversky, and Shiller, 1979). In response, Shin (2014) proposes a nonlinear ARDL (NARDL) framework in which short-run and long-run nonlinearities are modeled as positive and negative partial sum decompositions of the explanatory variables.

Consider the partial sum decomposition of a variable $z_{t}$ as $z_{t}=z^{0}+z_{t}^{+}+z_{t}^{-}$where $z_{t}^{+}$ and $z_{t}$ are the partial sum processes of positive and negative changes in $z_{t}$, respectively, around a threshold of $z^{0}$ :

$$
\begin{align*}
& z_{t}^{+}=\sum_{s=1}^{t} \max \left(\Delta z_{s}, z^{0}\right)  \tag{0.16}\\
& z_{t}^{-}=\sum_{s=1}^{t} \min \left(\Delta z_{s}, z^{0}\right)
\end{align*}
$$

where $z^{0}$ is the initial value of $z$.
The ITD representation of a $\operatorname{NARDL}\left(p, q_{1}, \ldots, q_{k}\right)$ model is given by:

$$
\begin{equation*}
y_{t}=\sum_{j=1}^{p} \psi_{j} y_{t-j}+\sum_{r=1}^{k} \sum_{j=0}^{p_{r}}\left(\beta_{r}^{0} x_{r}^{0}+\beta_{r, j}^{+} x_{r, t-j}^{+}+\beta_{r, j}^{-} \overline{x_{r, t-j}}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t} \tag{0.17}
\end{equation*}
$$

where $\beta_{r}^{0}$ are coefficients for the initial conditions, and where $\beta_{r, j}^{+}$and $\beta_{r, j}^{-}$are coefficients associated with the asymmetric distributed-lag variables.

We may an obtain a CEC representation of the ITD NARDL model,

$$
\begin{align*}
& \Delta y_{t}=-\phi y_{t-1}+\sum_{r=1}^{k}\left(\lambda_{r}^{+} x_{r, t-1}^{+}+\lambda_{r}^{-} x_{r, t-1}^{-}\right)+  \tag{0.18}\\
& \sum_{j=1}^{p-1} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k}\left(\eta_{r}^{+} \Delta x_{r, t}^{+}+\eta_{r}^{-} \Delta x_{r, t}^{-}\right)+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}-1}\left(\delta_{r, j}^{+} \Delta x_{r, t-j}^{+}+\delta_{r, j}^{-} \Delta \overline{x_{r, t-j}^{-}}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

where the $\lambda_{r}^{+}, \lambda_{r}^{-}, \eta_{r}^{+}, \eta_{r}^{-}, \delta_{r, j}^{+}, \delta_{r, j}^{-}$are asymmetric analogues of the coefficients in Equation (0.7).

We may rearrange terms so that Equation (0.18) becomes

$$
\begin{align*}
& \Delta y_{t}=-\phi\left(y_{t-1}-\sum_{r=1}^{k}\left(\frac{\lambda_{r}^{+}}{\phi} x_{r, t-1}^{+}+\frac{\lambda_{r}^{-}}{\phi} x_{r, t-1}^{-}\right)\right)+  \tag{0.19}\\
& \sum_{j=1}^{p-1} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k}\left(\eta_{r}^{+} \Delta x_{r, t}^{+}+\eta_{r}^{-} \Delta x_{r, t}^{-}\right)+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}-1}\left(\delta_{r, j}^{+} \Delta x_{r, t-j}^{+}+\delta_{r, j}^{-} \Delta x_{r, t-j}^{-}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

Then, define the asymmetric equilibrium error correction term,

$$
\begin{equation*}
E C A_{t}=y_{t-1}-\sum_{r=1}^{k}\left(\frac{\lambda_{r}^{+}}{\phi} x_{r, t-1}^{+}+\frac{\lambda_{r}^{-}}{\phi} x_{r, t-1}^{-}\right) \tag{0.20}
\end{equation*}
$$

so that the CEC Equation (0.19) may be written in EC form:

$$
\begin{align*}
& \Delta y_{t}=-\phi E C A_{t}+\sum_{j=1}^{p-1} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k}\left(\eta_{r}^{+} \Delta x_{r, t}^{+}+\eta_{r}^{-} \Delta x_{r, t}^{-}\right)+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}-1}\left(\delta_{r, j}^{+} \Delta x_{r, t-j}^{+}+\delta_{r, j}^{-} \Delta x_{r, t-j}^{-}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

where $\phi$ is the error correction parameter, the long-term equilibrium parameters for the explanatory variables are given by $\lambda_{r}^{+} / \phi$ and $\lambda_{r}^{-} / \phi$ for $r=1, \ldots, k$. The short-run parameters for the explanatory variables are given by the $\eta_{r}^{+}, \eta_{r}^{-}, \delta_{r, j}^{+}, \delta_{r, j}^{-}$.

Notice that because the CEC representation decomposes the effect of the distribution lag variables into short and long-run components, it allows for asymmetries in various combinations of short and long-run dynamics. This flexibility does not exist in the ITD representation.

As with their symmetric counterparts, NARDL models may be estimated via least squares. This result is appealing since nonlinear models often require iterative estimation routines. Furthermore, bounds testing procedures ("Bounds Test View" on page 328) remain valid and require no meaningful adjustments.

## Estimating ARDL and NARDL in EViews

EViews provides an powerful interface for ARDL and NARDL estimation.
From the main EViews menu, click on Quick/Estimate Equation... or type the command equation in the command line to open the equation dialog. Then select the ARDL -

Autoregressive Distributed Lag Models (including NARDL) from the Method dropdown to display the Specification tab of the ARDL dialog:


- In the first edit field under Linear dynamic specification, you should a enter a list of variables consisting of the dependent variable followed by any symmetric ARDL distributed lag regressors. At a minimum, the edit field must contain the dependent variable.
- Exogenous regressors, including deterministics, may be specified in the Fixed regressors specifications section. Trend regressors corresponding to the five deterministic cases discussed in "Bounds Test View" on page 328 (None, Restr. constant, Constant, Restr. trend, Trend) may be specified using the Trend specification dropdown. All other exogenous regressors (those apart from the constant and the trend) should be specified in the Fixed regressors edit field.
- Asymmetric distributed lag regressors may be listed under Asymmetric dynamic specifications. In particular, the Long-run and short-run edit field may be used to specify regressors which are asymmetric in both the long-run and short-run. Regressors which are asymmetric only in the long-run may be specified in the Long-run
only edit field, while those which are asymmetric exclusively in the short-run are specified in the Short-run only edit field.
- The Lag selection section specifies the lags for the dependent variable and the distributed lag regressors. By default, EViews uses automatic lag selection for both, following the PSS(1999), the lag structure of a ARDL model is chosen optimally using standard model selection criteria. You may use the Model Selection Criteria dropdown menu on the Options page to select your criterion, choosing between using Akaike (AIC), Schwarz (BIC), Hannan-Quinn (HQ), or the adjusted $R^{2}$. Alternately, you may select the radio button for Fixed to provide user-specified lags. Given your choice of method, you may then use the dropdown menus to specify the actual Max Lags or Lags to be used for the Dependent variable or the distributed lag Regressors.

For automatic lag selection, if $p$ and $q$ are the maximum number of lags of the dependent and explanatory variables, and $k$ is the total number of distributed-lag regressors, the total number of model evaluations is $p(q+1)^{k}$; the number of combinations of the set of numbers $1, \ldots, p$ and $k$ additional numbers from $0, \ldots, q$. For example, with 2 distributed-lag regressors and $p=q=4$, the total number of considered models is 100 .

When specifying the maximum number of lags, bear in mind that the ARDL model selection process will use the same sample for each estimation so that observations will be dropped from each candidate estimation based on the specified maximum. Once the lags are chosen, the final estimation output will use all observations available for the selected model. Thus, the final estimates will generally employ more observations than the model that was estimated during selection, and will be different than the selection model results.

- Additionally, you can override the global lag specification for individual variables. You may specify the lag for an individual variable using the "@fl (variable, lag)" syntax. For instance, if the variable X should use 3 lags, irrespective of the fixed or automatic lag settings, you may specify this by entering " $@ \mathrm{fl}(\mathrm{x}, 3)$ " in the regressor list.
- The Options tab of the estimation dialog allows you to specify the type of model selection to be used if you chose automatic lag selection. You may choose between the Akaike Information Criterion (AIC), Schwarz Criterion (SC), Hannan-Quinn Criterion (HQ), or the Adjusted R-squared objective.
- You may also select the type of covariance matrix to use in the final estimates, using the Coefficient covariance matrix dropdown. You may choose between Ordinary, White, and HAC (Newey-West) covariance estimation, and specify whether or not to perform a d.f. Adjustment. Note that these settings do not affect the model selection criteria.

By default, linear ARDL estimation results are displayed using the IDT representation Equation (0.2) while nonlinear ARDL estimates are displayed using the Conditional Error

Correction (CEC) form Equation (0.18). You may display the CEC and EC representations of a linear model using the "Error Correction Output View" on page 325 as described below.

For example, in these linear ARDL results, note that the dependent variable LOG(REALCONS) in the IDT representation is in levels,

```
Dependent Variable: LOG(REALCONS)
Method: ARDL
Date: 05/04/22 Time: 15:37
Sample (adjusted): 1951Q2 2000Q4
Included observations: }199\mathrm{ after adjustments
Dependent lags: }8\mathrm{ (Automatic)
Automatic-lag linear regressors (8 max. lags): LOG(REALGDP)
Static regressors: @EXPAND(@QUARTER, @DROPLAST)
Deterministics: Restricted constant and no trend (Case 2)
Model selection method: Akaike info criterion (AIC)
Number of models evaluated: }7
Selected model: ARDL(5,1)
```

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |  |  |
| :---: | ---: | :--- | ---: | ---: | :---: | :---: |
| LOG(REALCONS(-1)) | 0.854510 | 0.064428 | 13.26300 | 0.0000 |  |  |
| LOG(REALCONS(-2)) | 0.258776 | 0.082121 | 3.151153 | 0.0019 |  |  |
| LOG(REALCONS(-3)) | -0.156598 | 0.071521 | -2.189542 | 0.0298 |  |  |
| LOG(REALCONS(-4)) | -0.194069 | 0.070465 | -2.754106 | 0.0065 |  |  |
| LOG(REALCONS(-5)) | 0.169457 | 0.048486 | 3.494951 | 0.0006 |  |  |
| LOG(REALGDP) | 0.547615 | 0.048246 | 11.35042 | 0.0000 |  |  |
| LOG(REALGDP(-1)) | -0.475684 | 0.051091 | -9.310547 | 0.0000 |  |  |
| @QUARTER=1 | -0.000348 | 0.001176 | -0.295813 | 0.7677 |  |  |
| @QUARTER=2 | -0.000451 | 0.001165 | -0.386775 | 0.6994 |  |  |
| @QUARTER=3 | 0.000854 | 0.001171 | 0.729123 | 0.4668 |  |  |
| C | -0.058209 | 0.027842 | -2.090705 | 0.0379 |  |  |
| R-squared | 0.999873 | Mean dependent var |  |  |  | 7.902158 |
| Adjusted R-squared | 0.999867 | S.D. dependent var | 0.502623 |  |  |  |
| S.E. of regression | 0.005805 | Akaike info criterion | -7.406420 |  |  |  |
| Sum squared resid | 0.006336 | Schwarz criterion | -7.224378 |  |  |  |
| Log likelihood | 747.9388 | Hannan-Quinn criter. | -7.332743 |  |  |  |
| F-statistic | 148407.0 | Durbin-Watson stat | 1.865392 |  |  |  |
| Prob(F-statistic) | 0.000000 |  |  |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

Alternately for a nonlinear ARDL the CEC results are for a dependent variable in differences:

| Dependent Variable: DLOG(REALCONS) |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Method: ARDL |  |  |  |  |
| Date: 05/04/22 Time: 15:37 |  |  |  |  |
| Sample (adjusted): 1950Q3 2000Q4 |  |  |  |  |
| Included observations: 202 after adjustments |  |  |  |  |
| Max. dependent lags: 1 (Fixed) |  |  |  |  |
| Fixed-lag linear regressors: LOG(REALGDP) |  |  |  |  |
| Fixed-lag dual non-linear regressors: LOG(REALGOVT) |  |  |  |  |
| Deterministics: Restricted constant and no trend (Case 2) |  |  |  |  |
| Selected model: ARDL(1,1,1) |  |  |  |  |
| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| LOG(REALCONS(-1)) | -0.126939 | 0.037213 | -3.411168 | 0.0008 |
| LOG(REALGDP(-1)) | 0.133061 | 0.040618 | 3.275890 | 0.0012 |
| @CUMDP(LOG(REALGOVT(-1))) | -0.004936 | 0.008563 | -0.576444 | 0.5650 |
| @CUMDN(LOG(REALGOVT(-1))) | $-0.018402$ | 0.020006 | -0.919856 | 0.3588 |
| C | -0.100803 | 0.063927 | -1.576846 | 0.1165 |
| DLOG(REALGDP) | 0.643050 | 0.050018 | 12.85647 | 0.0000 |
| @DCUMDP(LOG(REALGOVT)) | -0.149865 | 0.042682 | -3.511242 | 0.0006 |
| @DCUMDN(LOG(REALGOVT)) | -0.114135 | 0.103098 | -1.107051 | 0.2696 |
| R -squared | 0.473136 | Mean depen | ent var | 0.008782 |
| Adjusted R-squared | 0.454125 | S.D. depend | nt var | 0.008864 |
| S.E. of regression | 0.006549 | Akaike info | terion | -7.180217 |
| Sum squared resid | 0.008320 | Schwarz crit |  | -7.049196 |
| Log likelihood | 733.2019 | Hannan-Qui | criter. | -7.127206 |
| F-statistic | 24.88805 | Durbin-Wats | stat | 2.584413 |
| $\operatorname{Prob}(\mathrm{F}$-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

Note that the nonlinear results include entries for the special cumulative positive and negative difference transformations (@cumdp and @cumdn) of the nonlinear regressor LOG(REALGOVT). For brevity, the initialization date for the @cumdp and @cumdn cumulative difference functions is not displayed in the output. You may view this date using the full representation of the equation by clicking on View/Representation from the estimated equation.

## Views and Procs of ARDL

Since ARDL and NARDL models are estimated by simple least squares, all of the views and procedures available to equation objects estimated by least squares are also available for ARDL models. There there are a few ARDL specific issues, view, and procs that require additional discussion.

## Model Selection Summary

The Model Selection Summary item on the View menu allows you to view either a Criteria Graph or a Criteria Table. The graph shows the model selection value for the twenty "best"
models. If you use either the Akaike Information Criterion (AIC), the Schwarz Criterion (BIC), or the Hannan-Quinn (HQ) criterion, the graph will show the twenty models with the lowest criterion value. If you choose the Adjusted R-squared as the model selection criteria, the graph will show the twenty models with the highest Adjusted R-squared. The table form of the view shows the log-likelihood value, the AIC, BIC and HQ values, and the Adjusted Rsquareds of the top twenty models in tabular form.

## Error Correction Output View

The Conditional Error Correction (CEC) and Error Correction (EC) representations of the ARDL specification (Equation (0.6) and Equation (0.11)) offer easy-to-visualize representations of the cointegrating relationship between the dependent variable and the explanatory variables.

By default, linear ARDL estimation results are displayed using the IDT representation Equation ( 0.2 ) while by default nonlinear ARDL estimates are displayed using the Conditional Error Correction (CEC) and Error Correction (EC) form Equation (0.6).

The Error Correction Output view displays the estimation results in the error corrections forms. Select View/ARDL Diagnostics/Error Correction Output from the menu of an estimated ARDL equation. EViews displays a spool with two tables.

The first node in the spool corresponds to the Conditional Error Correction (CEC) representation (Equation (0.6) or Equation (0.18)) of the equation:


The second node in the spool corresponds to results for the Error Correction (EC) representation of the equation (Equation (0.11) or Equation (0.21)), highlighting the speed of adjustment to equilibrium in the cointegrating relationship. The results show the least squares estimates for the equation which employs the equilibrium error correction term in place of the individual cointegrating series:


Here, the error correction term $E C_{t}$ given by Equation (0.10) and Equation (0.20) is included among the regressors and is denoted as "COINTEQ". The coefficient associated with this regressor is the speed of adjustment to equilibrium in each period. If variables are indeed cointegrated, we typically expect this coefficient to be negative and highly significant.

Note that the name of the lag of the dependent variable and the COINTEQ term in these two tables are always followed by a single asterisk, The single asterisk indicates that the $p$-value associated with the variable is incompatible with the $t$-Bounds distribution in Theorem 3.2 in PSS(2001).

The names of other variables may be followed by a double asterisk. A double asterisk indicates that the variable is a dynamic regressor with an optimal lag of zero so that EViews does not include lags and differences of the variables in the specification.

## Cointegrating Relation View

The Cointegrating Relation view displays information about the error correction term $E C_{t}$ representing the cointegrating relation. Select View/ARDL Diagnostics/Cointegrating Relation from the menu of an estimated ARDL equation to display a spool containing two tables and a graph.


- The Cointegrating Specification table shows the assumptions underlying the estimate of the cointegrating relationship, and the equation specification for $E C_{t}$.
- The Cointegrating Coefficients table shows coefficient estimates for the underlying variables inside the cointegrating relationship. These are the $\lambda_{r} / \phi$ coefficients in Equation (0.9) and the $\lambda_{r}^{+} / \phi$ and $\lambda_{r}^{-} / \phi$ coefficients in Equation (0.19).
- The Cointegrating Series graph displays the values of the $E C_{t}$ for every observation in the estimation sample.


## Bounds Test View

The traditional cointegration tests of Engle (1987), Phillips (1990), or Johansen (1995) require all variables in a VAR system to be $I(1)$. This requirement not only requires pretesting for the presence of unit roots in each of the endogenous variables, but is also subject to misclassification.

In contrast, Pesaran (2001) proposes a test for cointegration that is robust to whether variables of interest are $I(0), I(1)$, or mutually cointegrated. These bounds tests are formulated as standard $F$-test or Wald tests of parameter significance in the cointegrating relationship of the CEC model Equation (0.9),

$$
\begin{align*}
& H_{0}: \quad\left\{\phi, \lambda_{1} \ldots, \lambda_{k}\right\}=0 \\
& H_{1}:\left\{\phi, \lambda_{1} \ldots, \lambda_{k}\right\} \neq 0 \tag{0.22}
\end{align*}
$$

Once the bounds test statistic is computed, the value is compared to two asymptotic critical values corresponding to the polar cases of all variables being $I(0)$, or all variables being $I(1)$. When the test statistic is below the lower critical value, we fail to reject the null and conclude that cointegration is not possible. Alternately, when the test statistic is above the upper critical value, we reject the null and conclude that cointegration is possible. In either case, knowledge of the cointegrating rank is not necessary. If the statistic falls between the lower and upper critical values, the test is inconclusive.

When the hypothesis in Equation (0.22) is rejected so that cointegration is possible, we proceed to perform a $t$-test of significance of the error correction parameter $\phi$ in Equation (0.11). As in the case of Augmented Dickey-Fuller unit-root tests, critical values for the test statistic are non-standard. If the null hypothesis of $\phi=0$ is not rejected, there is no long run relationship. Alternatively, should we reject $\phi=0$ but be unable to reject the sub-hypothesis $\left\{\lambda_{1} \ldots, \lambda_{k}\right\}=0$, the cointegrating relationship is degenerate. Otherwise, cointegration exists.

When deterministics contribute to the error correction term, they are implicitly projected onto the span of the cointegrating vector. If the ARDL model in Equation (0.2) includes a constant and a trend, say $d_{0}=c$ and $d_{1}=t$, the constants and trend coefficients must respect the restrictions implied by the expressions for $\alpha_{0}$ and $\alpha_{1}$. These restrictions translate into slight modifications of the null and alternative hypotheses in Equation (0.22).

We may outline five alternate specifications of the CEC model that are distinguished by whether deterministic terms are included into the error correction term (Pesaran, 2001). The five cases, which closely follow VEC literature (Johansen, 1995), are summarized as follows:

- Case 1 - No constant, no trend: and $\mu=\xi=0$ which implies that $\alpha_{0}=\alpha_{1}=0$. There is no change to the Equation ( 0.22 ) bounds test hypothesis.
- Case 2 - Restricted constant and no trend: $\mu \neq 0$ and $\xi=0$, so the restrictions $\alpha_{0}=\Phi(1) \mu$ and $\alpha_{1}=0$ are assumed to hold:

$$
\begin{align*}
H_{0}: & \left\{\phi, \lambda_{1} \ldots, \lambda_{k}, \alpha_{0}\right\}=0  \tag{0.23}\\
H_{1}: & \left\{\phi, \lambda_{1} \ldots, \lambda_{k}, \alpha_{0}\right\} \neq 0
\end{align*}
$$

- Case 3 - Unrestricted constant and no trend: $\mu \neq 0$ and $\xi=0$, and the restrictions $\alpha_{0} \neq 0$ and $\alpha_{1}=0$ are assumed to hold. There is no change to the Equation (0.22) bounds test hypothesis.
- Case 4 - Unrestricted constant and restricted trend: $\mu \neq 0$ and $\xi \neq 0$, and the restrictions $\alpha_{0} \neq 0$ and $\alpha_{1}=\Phi(1) \xi$ are assumed hold:

$$
\begin{align*}
& H_{0}:\left\{\phi, \lambda_{1} \ldots, \lambda_{k}, \alpha_{1}\right\}=0 \\
& H_{1}:\left\{\phi, \lambda_{1} \ldots, \lambda_{k}, \alpha_{1}\right\} \neq 0 \tag{0.24}
\end{align*}
$$

- Case 5 - Unrestricted constant and unrestricted trend: $\mu \neq 0$ and $\xi \neq 0$ and the restrictions $\alpha_{0} \neq 0$ and $\alpha_{1} \neq 0$ are assumed hold. There is no change to the Equation (0.22) bounds test hypothesis.

To perform the bounds test, click on View/ARDL Diagnostics/Bounds Test. The results are presented in a spool. Below the table of long run coefficient estimates are two additional tables, respectively titled as the $F$-Bounds Test and the $t$-Bounds Test.


These tables respectively display the $F$ - and $t$-statistics along with their associated $\mathrm{I}(0)$ (lower) and I(1) (upper) critical value bounds for the null hypotheses of no levels relationship between the dependent variable and the regressors in the CEC model. The critical values are provided for significance levels $10 \%, 5 \%, 2.5 \%$, and $1 \%$, respectively. The $t$ Bounds test in particular is a parameter significance test on the lagged value of the dependent variable. Since the distribution of this test is non-standard, the $p$-value provided in the regression output of the CEC regression is not compatible with this distribution, although the $t$-statistic is valid. Accordingly, any inference must be conducted using the $t$-Bounds test critical values provided.

We also mention here that the $F$ - critical value tables now present the critical values computed under an asymptotic regime (sample size equal to 1000) and referenced from PSS(2001), in addition to providing critical values for finite sample regimes (sample sizes running from 30 to 80 in increments of 5) and referenced from Narayan (2005).

## Symmetry Test View

Recall that the NARDL CEC representation in Equation (0.21) is quite general and can accommodate asymmetries in different combinations of short and long-run dynamics. In particular, consider the following two sets of symmetry restrictions:

1. Long-run Symmetry: Restricts $\lambda_{r}^{+}=\lambda_{r}^{-}=\lambda_{r}$ for all $r=1, \ldots, k$ so that the CEC reduces to

$$
\begin{align*}
& \Delta y_{t}=-\phi\left(y_{t-1}-\sum_{r=1}^{k} \frac{\lambda_{r}}{\phi}\left(x_{r, t-1}^{+}+x_{r, t-1}^{-}\right)\right)+  \tag{0.25}\\
& \sum_{j=1}^{p-1} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k}\left(\eta_{r}^{+} \Delta x_{r, t}^{+}+\eta_{r}^{-} \Delta x_{r, t}^{-}\right)+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}-1}\left(\delta_{r, j}^{+} \Delta x_{r, t-j}^{+}+\delta_{r, j}^{-} \Delta x_{r, t-j}^{-}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

2. Short-run Symmetry: Restricts $\eta_{r}^{+}=\eta_{r}^{-}=\eta_{r}$ and $\delta_{r, j}^{+}=\delta_{r, j}^{-}=\delta_{r, j}$ for $r=1, \ldots, k$ and $j=1, \ldots, p_{r}-1$, so the CEC reduces to

$$
\begin{align*}
& \Delta y_{t}=-\phi\left(y_{t-1}-\sum_{r=1}^{k}\left(\frac{\lambda_{r}^{+}}{\phi} x_{r, t-1}^{+}+\frac{\lambda_{r}^{-}}{\phi} x_{r, t-1}^{-}\right)\right)+  \tag{0.26}\\
& \sum_{j=1}^{p-1} \gamma_{j} \Delta y_{t-j}+\sum_{r=1}^{k} \eta_{r}\left(\Delta x_{r, t}^{+}+\Delta x_{r, t}^{-}\right)+ \\
& \sum_{r=1}^{k} \sum_{j=1}^{p_{r}-1} \delta_{r, j}\left(\Delta x_{r, t-j}^{+}+\Delta x_{r, t-j}^{-}\right)+\sum_{s=1}^{m} \alpha_{s} d_{s, t}+\epsilon_{t}
\end{align*}
$$

Imposing either set of restrictions leads to one of the previous two representations. Imposing both restrictions reduces the NARDL CEC representation to the classical ARDL CEC representation in Equation (0.9)

And of course, it is possible to generate even more complex dynamics by imposing symmetry on other subsets of the long-run and short run regressors.

Naturally, one can test for symmetry formally by performing the usual $t$-test or $F$-test of parameter equality. For example, testing for symmetry for a specific long-run (LR) variable, say $x_{j}$, is equivalent to the following hypothesis:

$$
\begin{align*}
H_{0}: & \lambda_{j}^{+}=\lambda_{j}^{-}  \tag{0.27}\\
\text {To } & \\
H_{1}: & \lambda_{j}^{+} \neq \lambda_{j}^{-}
\end{align*}
$$

To perform the symmetry test, select View/ARDL Diagnostics/Symmetry Test from the menu of a nonlinear asymmetric NARDL equation:

| E Equation: EX4 Workfile: NARD... $\square$ |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc | Object | Print | Name | Freez |  | Estimate | Fore |
| Coefficient symmetry tests <br> Null hypothesis: Coefficient is symmetric <br> Degrees of freedom (simple tests): $\mathbf{F}(1,194)$. Chi-square(1) <br> Degrees of freedom (joint tests): $F(2,194)$, Chi-square(2) Equation: EX4 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Variable |  |  | Statistic | Value |  | Probability |  |  |
| Long-run |  |  |  |  |  |  |  |  |
| LOG(REALGOVT) |  |  | F-statistic Chi-square | $\begin{array}{ll} \text { ic } & 0.525997 \\ \text { re } & 0.525997 \end{array}$ |  | $\begin{aligned} & 0.4692 \\ & 0.4683 \end{aligned}$ |  |  |
|  |  |  |  |  |  |  |  |
| Short-run |  |  |  |  |  |  |  |  |
| LOG(REALGOVT) |  |  |  | F-statistic Chi-square | $\begin{array}{ll} \text { tic } & 0.077094 \\ \text { are } & 0.077094 \end{array}$ |  | $\begin{aligned} & 0.7816 \\ & 0.7813 \end{aligned}$ |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Joint (Long-Run and Short-Run) |  |  |  |  |  |  |  |  |
| LOG(REALGOVT) |  |  | F-statistic Chi-square | $\begin{array}{ll} \text { tic } & 0.493551 \\ \text { are } & 0.987102 \end{array}$ |  | $\begin{aligned} & 0.6112 \\ & 0.6105 \end{aligned}$ |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

## Dynamic Multipliers View

Dynamic multipliers (DM) are a familiar concept which measures the marginal contribution of an explanatory variable to the dependent variable. A natural extension of the concept for time series analysis is the idea of cumulative dynamic multipliers (CDM). This is the cumulative sum of dynamic multipliers at each point in time, starting with a point in time $t$ and running through $t+h$ for horizon length $h>0$.

For standard ARDLS, CDMs are defined for each long-run distributed-lag variable as the partial derivatives:

$$
\begin{equation*}
m_{r, t}(h)=\sum_{s=0}^{h} \frac{\partial y_{t+s}}{\partial x_{r, t+h}} \tag{0.28}
\end{equation*}
$$

for $r=1, \ldots, k$.
For NARDL models, CDMs are derived for each long-run asymmetric distributed-lag variables:

$$
\begin{align*}
& m_{r, t}^{+}(h)=\sum_{s=0}^{h} \frac{\partial y_{t+s}}{\partial x_{r, t+s}^{+}}  \tag{0.29}\\
& m_{r, t}^{-}(h)=\sum_{s=0}^{h} \frac{\partial y_{t+s}}{\partial x_{r, t+s}^{-}}
\end{align*}
$$

By construction, as $h \rightarrow \infty, m_{r, t}(h) \rightarrow \lambda_{r} / \phi$ in traditional models, while $m_{r, t}^{+}(h) \rightarrow \lambda_{r}^{+} / \phi$ and $m_{r, t}^{-}(h) \rightarrow \lambda_{r}^{-} / \phi$ in asymmetric models. In the latter setting, we can employ the absolute difference between the two different CDMs, $\left|m_{r, t}^{+}(h)-m_{r, t}^{-}(h)\right|$, as a measure of asymmetric or nonlinearity.

To display cumulative dynamic multiplier graphs for each of the explanatory variables, click on View/ARDL Diagnostics/Dynamic Multiplier Graph... EViews will open a dialog containing display and computation settings:

| $\square$ Dynamic Multipliers |  |
| :--- | :---: |
| Horizon: | 301 |
| Confidence intervals |  |
| $\square$ Show CI |  |
| SShade CI band |  |
| Level: | 0.95 |
| Replications: | 999 |
| OK |  |

- You may enter the horizon length $h$ (number of periods to compute the multipliers) in the Horizon edit field.
- For NARDL models, you will be offered the opportunity to display confidence intervals for the computed absolute difference between the positive and negative components for a given regressor. CIs are not available for linear ARDL specifications.

You may check the Show CI to display the CIs, and Shade CI band to display the CIs as bands instead of lines. The Level edit field controls the size of the CI, and the Replications governs how many replications to use in resampling for computing the CI.

Click on $\mathbf{O K}$ to continue. EViews will open a spool view, with each node in the spool containing the CDM graph corresponding to one of the explanatory variables.

For symmetric linear models, each graph contains a CDM along with a dashed horizontal line denoting the long-run value.


For asymmetric nonlinear models, each graph will show the positive and negative responses and limit values, along with a line showing the absolute value of the difference between the two, and if requested, a CI for the absolute difference:


## ARDL Equation Procs

Make Regressor Group Proc
Make Cointegrating Relationship Proc

## Examples

We demonstrate ARDL and NARDL estimation using a dataset from Greene (2008, page 685). This dataset consists of a number of quarterly US macroeconomic variables between 1950 and 2000. The data are in the workfile "Ardl13.WF1"

## Example 1: Symmetric ARDL (Automatic Lag Selection)

We start with a classical (symmetric) ARDL model using the log of real consumption as the dependent variable, and the log of real GDP as a single regressor (along with a constant). Bring up the estimation dialog and enter

```
log(realcons) log(realgdp)
```

into the Linear dynamic specification edit field and select Automatic selection with a maximum of $\mathbf{8}$ lags (two years) for both the dependent variable and dynamic regressors.

We include a full set of quarterly dummies as fixed regressors. In particular, add a restricted constant to the cointegrating relationship, and the remaining quarterly dummies to the short-run regressors. Include the long-run constant by choosing Rest. Constant in the Trend specification dropdown menu, and enter

```
@expand(@quarter, @droplast)
```

into the Fixed regressors edit field to add the remaining dummy variables.


We do not make changes in the Options tab, leaving all settings at their default value. The estimation results for this specification are shown below:

Dependent Variable: LOG(REALCONS)
Method: ARDL
Date: 05/03/22 Time: 09:43
Sample (adjusted): 1951Q2 2000Q4
Included observations: 199 after adjustments
Dependent lags: 8 (Automatic)
Automatic-lag linear regressors (8 max. lags): LOG(REALGDP)
Static regressors: @EXPAND(@QUARTER, @DROPLAST)
Deterministics: Restricted constant and no trend (Case 2)
Model selection method: Akaike info criterion (AIC)
Number of models evaluated: 72
Selected model: ARDL(5,1)

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |  |  |
| :---: | ---: | ---: | ---: | ---: | :---: | :---: |
| LOG(REALCONS(-1)) | 0.854510 | 0.064428 | 13.26300 | 0.0000 |  |  |
| LOG(REALCONS(-2)) | 0.258776 | 0.082121 | 3.151153 | 0.0019 |  |  |
| LOG(REALCONS(-3) | -0.156598 | 0.071521 | -2.189542 | 0.0298 |  |  |
| LOG(REALCONS(-4)) | -0.194069 | 0.070465 | -2.754106 | 0.0065 |  |  |
| LOG(REALCONS(-5)) | 0.169457 | 0.048486 | 3.494951 | 0.0006 |  |  |
| LOG(REALGDP) | 0.547615 | 0.048246 | 11.35042 | 0.0000 |  |  |
| LOG(REALGDP(-1)) | -0.475684 | 0.051091 | -9.310547 | 0.0000 |  |  |
| @QUARTER=1 | -0.000348 | 0.001176 | -0.295813 | 0.7677 |  |  |
| @QUARTER=2 | -0.000451 | 0.001165 | -0.386775 | 0.6994 |  |  |
| @QUARTER=3 | 0.000854 | 0.001171 | 0.729123 | 0.4668 |  |  |
| C | -0.058209 | 0.027842 | -2.090705 | 0.0379 |  |  |
| R-squared | 0.999873 | Mean dependent var |  |  |  | 7.902158 |
| Adjusted R-squared | 0.999867 | S.D. dependent var | 0.502623 |  |  |  |
| S.E. of regression | 0.005805 | Akaike info criterion | -7.406420 |  |  |  |
| Sum squared resid | 0.006336 | Schwarz criterion | -7.224378 |  |  |  |
| Log likelihood | 747.9388 | Hannan-Quinn criter. | -7.332743 |  |  |  |
| F-statistic | 148407.0 | Durbin-Watson stat | 1.865392 |  |  |  |
| Prob(F-statistic) | 0.000000 |  |  |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

The first part of the output gives a summary of the settings used during estimation. Here we see that lags were determined using automatic selection (Akaike Information Criterion) with a maximum of 8 lags of both the dependent variable and the regressor. Out of the 72 models evaluated, the automatic selection procedure yielded an $\operatorname{ARDL}(5,1)$ model-5 lags of the dependent variable LOG(REALCONS), and a single lag (along with the level value) of the regressor LOG(REALGDP).

The rest of the output is standard least squares output for the selected model. Note that each of the regressors (with the exception of the quarterly dummies) is statistically significant, and that the coefficient on the one period lag of the dependent variable, LOG(REALCONS), is quite high, at 0.85 .

To view the performance of the selected model against the alternatives, we click on \View/ Model Selection Summary/Criteria Graph to view a graph of the AIC of the top twenty models:


The selected $\operatorname{ARDL}(5,1)$ model was only slightly better than an $\operatorname{ARDL}(5,2)$ model, which was in turn only slightly better than an $\operatorname{ARDL}(5,3)$. Note that the top three models all employ five lags of the dependent variable.

Click on the Name button on the equation toolbar and name the equation EX1.

## Example 2: Symmetric ARDL $(3,3)$

Instead of using automatic selection to choose the best model, Greene (Example 20.4) analyzes these data with a fixed $\operatorname{ARDL}(3,3)$ model. We can estimate this specification by clicking on Object/Copy on the EX1 toolbar to make a copy of the existing equation, name the equation EX2, then pressing the Estimate button to bring up the estimation dialog. Next, change the number of lags on both dependent and regressors to " 3 ", and then select the Fixed radio button to switch off automatic lag selection:


Click on OK to estimate the equation using these settings.
The results of this estimation are given by:

```
Dependent Variable: LOG(REALCONS)
Method: ARDL
Date: 05/03/22 Time: 09:43
Sample (adjusted): 1950Q4 2000Q4
Included observations: 201 after adjustments
Max. dependent lags: 3 (Fixed)
Fixed-lag linear regressors: LOG(REALGDP)
Static regressors: @EXPAND(@QUARTER, @DROPLAST)
Deterministics: Restricted constant and no trend (Case 2)
Selected model: ARDL(3,3)
```

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| :---: | ---: | :--- | ---: | ---: |
| LOG(REALCONS(-1)) | 0.723341 | 0.069767 | 10.36794 | 0.0000 |
| LOG(REALCONS(-2)) | 0.391367 | 0.079618 | 4.915576 | 0.0000 |
| LOG(REALCONS(-3)) | -0.233653 | 0.068672 | -3.402444 | 0.0008 |
| LOG(REALGDP) | 0.565088 | 0.051953 | 10.87699 | 0.0000 |
| LOG(REALGDP(-1)) | -0.390884 | 0.083934 | -4.657023 | 0.0000 |
| LOG(REALGDP(-2)) | -0.237950 | 0.086882 | -2.738778 | 0.0068 |
| LOG(REALGDP(-3)) | 0.190243 | 0.058922 | 3.228753 | 0.0015 |
| @QUARTER=1 | -0.000259 | 0.001266 | -0.204677 | 0.8380 |
| @QUARTER=2 | -0.000259 | 0.001259 | -0.205412 | 0.8375 |
| @QUARTER=3 | 0.000915 | 0.001256 | 0.728608 | 0.4671 |
| C | -0.109962 | 0.029236 | -3.761208 | 0.0002 |
|  | 0.999855 | Mean dependent var | 7.893303 |  |
| R-squared | 0.999847 | S.D. dependent var | 0.507884 |  |
| Adjusted R-squared | 0.006274 | Akaike info criterion | -7.251681 |  |
| S.E. of regression | 0.007479 | Schwarz criterion | -7.070903 |  |
| Sum squared resid | 739.7939 | Hannan-Quinn criter. | -7.178530 |  |
| Log likelihood | 131047.1 | Durbin-Watson stat | 1.785975 |  |
| F-statistic | 0.000000 |  |  |  |
| Prob(F-statistic) |  |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

For this specification, the one-period lag on the dependent variable remains high, at 0.72 , and again all coefficients are statistically significant (with the exception of the period dummies).

We may examine the CEC and EC forms of the estimates by selecting View/ARDL Diagnostics/Error Correction Results. EViews will display long-run output in the form of a spool with two tables showing the Conditional Error Correction regression results, and the Error Correction results. The first table displays the estimation results in the CEC form:

| E Equation: EX2 Workfile: NARDL_EV13::Untitled |  |  |  |  |  |  |  |  |  |  |  |  | - 回 | x |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc | Object | Print | Name | Freeze |  | Estimate | Forecast | Stats | Resi |  |  |  |  |
| Conditional Error Correction Error Correction |  |  |  |  |  | Conditional Error Correction <br> Dependent Variable: DLOG(REALCONS) <br> Method: ARDL <br> Date: 05/04/22 Time: 15:37 <br> Sample (adjusted): 195004200004 <br> Included observations: 201 after adjustments Max. dependent lags: 3 (Fixed) <br> Fixed-lag linear regressors: LOG(REALGDP) <br> Static regressors: @EXPAND(@QUARTER, @DROPLAST) <br> Deterministics: Restricted constant and no trend (Case 2) Selected model: ARDL $(3,3)$ |  |  |  |  |  |  |  | $\checkmark$ |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | riable | Coeffic | cient | Std. Error | t-Statistic | Prob. |  |
|  |  |  |  |  |  |  | LOG(REA | LCONS (-1))* | -0.118 | 18944 | 0.030474 | -3.903191 | 0.0001 |  |
|  |  |  |  |  |  |  | LOG(RE | ALGDP(-1)) | 0.128 | 8497 | 0.032281 | 3.918824 | 0.0001 |  |
|  |  |  |  |  |  |  |  |  | -0.105 | 9982 | 0.029238 | -3.761208 | 0.0002 |  |
|  |  |  |  |  |  |  | DLOG(RE | ALCONS(-1)) | $-0.157$ | 7714 | 0.089795 | $-2.259885$ | 0.0250 |  |
|  |  |  |  |  |  |  | DLOG(RE | ALCONS(-2)) | 0.233 | 3853 | 0.088872 | 3.402444 | 0.0008 |  |
|  |  |  |  |  |  |  | DLOG(R | REALGDP) | 0.585 | 5088 | 0.051953 | 10.87899 | 0.0000 |  |
|  |  |  |  |  |  |  | DLOG(RE | ALGDP(-1)) | 0.047 | 7706 | 0.083725 | 0.748831 | 0.4550 |  |
|  |  |  |  |  |  |  | DLOG(RE | ALGDP(-2)) | -0.19020 | 0243 | 0.058922 | -3.228753 | 0.0015 |  |
|  |  |  |  |  |  |  | @QUA | RTER=1 | $-0.000$ | 0259 | 0.001286 | $-0.204877$ | 0.8380 |  |
|  |  |  |  |  |  |  | @QUA | RTER=2 | $-0.000$ | 0259 | 0.001259 | $-0.205412$ | 0.8375 |  |
|  |  |  |  |  |  |  | @QUA | RTER=3 | 0.000 | 0915 | 0.001256 | 0.728808 | 0.4871 |  |
|  |  |  |  |  |  |  | R-squared |  | 0.469 | 9211 | Mean deper | dent var | 0.008577 |  |
|  |  |  |  |  |  |  | Adjusted R | -squared | 0.44 | 1275 | S.D. depen | dent var | 0.008393 |  |
|  |  |  |  |  |  |  | S.E. of reg | gression | 0.006 | 8274 | Akaike info | criterion | -7.251881 |  |
|  |  |  |  |  |  |  | Sum squa | ed resid | 0.007 | 7479 | Schwarz crim | erion | -7.070903 |  |
|  |  |  |  |  |  |  | Log likeliho |  | 739.7 | 7939 | Hannan-Qu | in criter. | -7.178530 |  |
|  |  |  |  |  |  |  | F-statistic |  | 16.79 | 9577 | Durbin-Wat | on stat | 1.785975 |  |
|  |  |  |  |  |  |  | Prob(F-sta | tistic) |  | 0000 |  |  |  |  |
|  |  |  |  |  |  |  | - p-value | $s$ are incomp | patible wit | with t -b | unds distribu | tion. |  |  |
|  |  |  |  |  |  | $\square$ Error Correction |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Dependent Variable: DLOG(REALCONS) Method: ARDL |  |  |  |  |  |  |  |

The EC results, which are displayed click on the Error Correction node, show that the speed of adjustment coefficient is negative ( -11.89 ) and statistically significant.


Clicking on View/ARDL Diagnostics/Cointegrating Relations shows the specification and coefficient results for the cointegrating relationship:


The equilibrium coefficients show that the impact of a change in LOG(REALGDP) on LOG(REALCONS) is close to unity. Moreover, since the specification restricts the constant to the long-run, it shows up as part of the cointegrating equation.

## Example 3: Symmetric ARDL $(1,1)$

As a final symmetric ARDL example, we will consider Greene's Example 20.5 which estimates an $\operatorname{ARDL}(1,1)$ model. Copy the EX2 ARDL equation object, name the copy EX3, bring up the estimation dialog by clicking on the Estimate button and change the number of lags to " 1 " for both dependent and regressors, remove the quarterly dummies, and then click on OK to estimate with the new specification:


The results obtained from estimating this specification are given by:

Dependent Variable: LOG(REALCONS)
Method: ARDL
Date: 05/03/22 Time: 09:47
Sample (adjusted): 1950Q2 2000Q4
Included observations: 203 after adjustments
Max. dependent lags: 1 (Fixed)
Fixed-lag linear regressors: LOG(REALGDP)
Deterministics: Restricted constant and no trend (Case 2)
Selected model: ARDL(1,1)

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| :--- | ---: | :--- | ---: | ---: |
| LOG(REALCONS(-1)) | 0.904584 | 0.030589 | 29.57234 | 0.0000 |
| LOG(REALGDP) | 0.584210 | 0.051411 | 11.36351 | 0.0000 |
| LOG(REALGDP(-1)) | -0.483037 | 0.052177 | -9.257657 | 0.0000 |
| C | -0.085331 | 0.029285 | -2.913823 | 0.0040 |
| R-squared | 0.999820 | Mean dependent var | 7.884560 |  |
| Adjusted R-squared | 0.999817 | S.D. dependent var | 0.512951 |  |
| S.E. of regression | 0.006940 | Akaike info criterion | -7.083459 |  |
| Sum squared resid | 0.009585 | Schwarz criterion | -7.018175 |  |
| Log likelihood | 722.9711 | Hannan-Quinn criter. | -7.057048 |  |
| F-statistic | 367753.8 | Durbin-Watson stat | 2.493836 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

Following estimation, we may perform bounds test for cointegration by clicking on View/
ARDL Diagnostics/Bounds Test to bring up the cointegrating relationship view


The bounds statistic value is 17.25 . We compare this result to the critical values listed in the second table. Clearly the statistic is larger than the $I(1)$ critical value at all significance levels so that we may reject the null hypothesis of no levels relationship and conclude that LOG(REALGDP) and LOG(REALCONS) are cointegrated.

We may also evaluate the cumulative dynamic multiplier of the explanatory variable LOG(REALGDP) on LOG(REALCONS). Click on View/ARDL Diagnostics/Dynamic Multiplier Graph... to bring up the dynamic multiplier dialog:

| $\square$ Dynamic Multipliers |  |
| :--- | :--- |
| Horizon: | 301 |
| Confidence intervals |  |
| $\square$ Show CI |  |
| $\square$ Shade CI band |  |
| Level: | 0.95 |
| Replications: | 999 |
| OK |  |

Note that confidence interval settings are not available since we have estimated a purely symmetric ARDL models. Enter " 30 " as the horizon length and click on OK.

EViews will display a spool object with a dynamic multiplier graph for each distributed lag variable:


Here we see that LOG(REALGDP) approaches its long-run value of 1.06 as the horizon length increases. Moreover, it does so at a diminishing pace.

## Example 4: Asymmetric ARDL(1,1,1)

Here we will continue from the previous example, but consider the NARDL( $1,1,1$ ) model of LOG(REALCONS) on LOG(REALGDP) and LOG(REALGOVT). We will treat LOG(REALGOVT) as an asymmetric variable which is asymmetric in both the short-run and the longrun.

To estimate this model, copy the EX3 equation, name the copy EX4, then bring up the estimation dialog by clicking on the Estimate button. Next, add an Asymmetric dynamic specification group by entering "LOG(REAGOVT)" in the Long-run and short-run asymmetry edit field,

and click on OK to estimate the new specification.

```
Dependent Variable: DLOG(REALCONS)
Method: ARDL
Date: 05/04/22 Time: 15:37
Sample (adjusted): 1950Q3 2000Q4
Included observations: 202 after adjustments
Max. dependent lags: 1 (Fixed)
Fixed-lag linear regressors: LOG(REALGDP)
Fixed-lag dual non-linear regressors: LOG(REALGOVT)
Deterministics: Restricted constant and no trend (Case 2)
Selected model: ARDL(1,1,1)
```

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| :--- | ---: | :--- | ---: | ---: |
| LOG(REALCONS(-1)) | -0.126939 | 0.037213 | -3.411168 | 0.0008 |
| LOG(REALGDP(-1)) | 0.133061 | 0.040618 | 3.275890 | 0.0012 |
| @CUMDP(LOG(REALGOVT(-1))) | -0.004936 | 0.008563 | -0.576444 | 0.5650 |
| @CUMDN(LOG(REALGOVT(-1))) | -0.018402 | 0.020006 | -0.919856 | 0.3588 |
| C | -0.100803 | 0.063927 | -1.576846 | 0.1165 |
| DLOG(REALGDP) | 0.643050 | 0.050018 | 12.85647 | 0.0000 |
| @DCUMDP(LOG(REALGOVT)) | -0.149865 | 0.042682 | -3.511242 | 0.0006 |
| @DCUMDN(LOG(REALGOVT)) | -0.114135 | 0.103098 | -1.107051 | 0.2696 |
| R-squared | 0.473136 | Mean dependent var | 0.008782 |  |
| Adjusted R-squared | 0.454125 | S.D. dependent var | 0.008864 |  |
| S.E. of regression | 0.006549 | Akaike info criterion | -7.180217 |  |
| Sum squared resid | 0.008320 | Schwarz criterion | -7.049196 |  |
| Log likelihood | 733.2019 | Hannan-Quinn criter. | -7.127206 |  |
| F-statistic | 24.88805 | Durbin-Watson stat | 2.584413 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

Notice that the LOG(REALGOVT) is now split into four variables corresponding to the positive and negative cumulative sums and cumulative difference sums using the labels "@CUMDP(LOG(REALGOVT(-1))" and "@CUMDN(LOG(REALGOVT(-1))" for the longterm effects, and the labels "@DCUMDP(LOG(REALGOVT))" and "@DCUMDN(LOG(REALGOVT))" for the short-term effects.

It is extremely important to note that in contrast to purely symmetric ARDL models which display the ITD representation Equation (0.2) results as the default output (with an option to display the results for the CEC equation Equation (0.9) using the Conditional Error Correction (Long Run) Form view), the default output for asymmetric NARDL models shows the CEC equation Equation (0.19) representation. This difference in the default display is due to the need to support NARDL partial asymmetry models which may only be specified in the context of a CEC model.

We may also test whether the asymmetric assumptions for LOG(REALGOVT) are valid by testing for symmetry. Click on View/ARDL Diagnostics/Symmetry Test to conduct the test:

```
Coefficient symmetry tests Null hypothesis: Coefficient is symmetric Degrees of freedom: \(F(1,194)\), Chi-square(1)
Equation: EX4
```

| Variable | Statistic | Value | Probability |
| :---: | :---: | :---: | :---: |
|  | Long-run |  |  |
| LOG(REALGOVT) | F-statistic | 0.525997 | 0.4692 |
|  | Chi-square | 0.525997 | 0.4683 |
|  | Short-run |  |  |
| LOG(REALGOVT) | F-statistic | 0.077094 | 0.7816 |
|  | Chi-square | 0.077094 | 0.7813 |

EViews displays a table of test results for long-run and short-run restrictions. In this case, we do not reject the null hypothesis of dual asymmetry against the long-run or short-run partial asymmetric alternative.

We may also display the cumulative dynamic multiplier curves. Click on View/ARDL Diagnostics/Dynamic Multiplier Graph...\} to bring up the dynamic multiplier dialog:


Note that confidence interval settings are now available in the dialog for our asymmetric NARDL model since dynamic multiplier CIs are derived for the absolute difference in paths resulting from the positive and negative asymmetric components of a given regressor. You may check the Show CI to display the CIs, and Shade CI band to display the CIs as bands instead of lines. The Level edit field controls the size of the CI, and the Replications governs how many replications to use in resampling for computing the CI.

To proceed with our example, enter " 30 " as the horizon length and leave everything at else at the default values. Click on $\mathbf{O K}$ to continue.

The results are displayed in a spool, with one graph for each of the distributed lag explanatory variables. The first node shows the CDM for LOG(REALGDP) on the dependent variable LOG(REALCONS).


Notice that the path approaches the long-run value, which is indicated by dashed horizontal line. Since LOG(REALGDP) is symmetric, there is no CI around the CDM.

The second node shows the CDM graph for the asymmetric variable LOG(REALGOVT):

| E Equation: EX4 Workfile: NARDL_EV13::Untitled |  |  |  |  |  |  |  |  |  |  | $\square$ | 回 | $x$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc Object | Print | Name | Freeze | Estimate | Forecast | Stats | Resids |  |  |  |  |  |
| LOG(REALGDP) |  |  | $\square$ LOG(REALGOVT) |  |  |  |  |  |  |  |  |  | $\wedge$ |
|  |  |  |  | Cumulative Dynamic Multiplier: LOG(REALGOVT) on LOG(REALCONS) |  |  |  |  |  |  |  |  | , |

Since LOG(REALGOVT) is asymmetric in both the long-run and short-run, we expect the dynamic multiplier curves to differ in both the long-run and short-run. This relationship is seen in the fact that the absolute difference between these paths (the top line with shaded CI interval) never approaches zero. The remaining lines display the CDMs for the positive and negative values, which as expected, approach their long-run values.

## Example 5: Asymmetric ARDL(1,1,1,1)

Continuing with the previous example, we add a partially symmetric variable to the list of distributed-lag regressors. We treat real investments (REALINVS) as a variable which is asymmetric in the long-run, but symmetric in the short run. Make a copy of EX4, name the copy EX5, then bring up the estimation dialog by clicking on the Estimate button. Enter "LOG(REALINVS)" in the Long-run asymmetry only edit field, and click on OK to estimate the updated specification.

The results of this estimation are:

Dependent Variable: DLOG(REALCONS) Method: ARDL
Date: 05/04/22 Time: 13:23
Sample (adjusted): 1950Q3 2000Q4
Included observations: 202 after adjustments
Max. dependent lags: 1 (Fixed)
Fixed-lag linear regressors: LOG(REALGDP)
Fixed-lag dual non-linear regressors: LOG(REALGOVT)
Fixed-lag long-run non-linear regressors: LOG(REALINVS)
Deterministics: Restricted constant and no trend (Case 2)
Selected model: ARDL $(1,1,1,1)$

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| :---: | ---: | ---: | ---: | ---: |
| LOG(REALCONS(-1)) | -0.067376 | 0.029370 | -2.294022 | 0.0229 |
| LOG(REALGDP(-1)) | 0.064146 | 0.038956 | 1.646639 | 0.1013 |
| @CUMDP(LOG(REALGOVT(-1))) | -0.010560 | 0.007600 | -1.389487 | 0.1663 |
| @CUMDN(LOG(REALGOVT(-1))) | 0.007045 | 0.013878 | 0.507646 | 0.6123 |
| @CUMDP(LOG(REALINVS(-1))) | 0.002484 | 0.006468 | 0.384024 | 0.7014 |
| @CUMDN(LOG(REALINVS(-1))) | -0.006293 | 0.005391 | -1.167234 | 0.2446 |
| C | -0.002373 | 0.119394 | -0.019879 | 0.9842 |
| DLOG(REALGDP) | 1.399515 | 0.056668 | 24.69679 | 0.0000 |
| DLOG(REALINVS) | -0.187990 | 0.011299 | -16.63751 | 0.0000 |
| @DCUMDP(LOG(REALGOVT)) | -0.340204 | 0.029382 | -11.57868 | 0.0000 |
| @DCUMDN(LOG(REALGOVT)) | -0.202562 | 0.065446 | -3.095100 | 0.0023 |
| R-squared | 0.794035 | Mean dependent var | 0.008782 |  |
| Adjusted R-squared | 0.783252 | S.D. dependent var | 0.008864 |  |
| S.E. of regression | 0.004127 | Akaike info criterion | -8.089752 |  |
| Sum squared resid | 0.003253 | Schwarz criterion | -7.909599 |  |
| Log likelihood | 828.0650 | Hannan-Quinn criter. | -8.016862 |  |
| F-statistic | 73.63435 | Durbin-Watson stat | 1.907457 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

Observe now that LOG(REALINVS) is split into its positive and negative cumulative sums in the long-run labeled "@CUMDP(LOG(REALINVS(-1))" and "@CUMDN(LOG(REALINVS($1)$ )", and the short-run symmetric effect labeled "@DLOG(REALINVS)". In contrast, LOG(REALGOVT) is asymmetric in both the long-run and short-run,

We can further study the long-run relationship by clicking on Views/ARDL Diagnostics/ Error Correction Results to display the CEC and EC representations,

and by clicking on Views/ARDL Diagnostics/Cointegrating Relation to display the equilibrium relationship,


We see here that both LOG(REALGOVT) and LOG(REALINVS) enter the cointegrating relationship asymmetrically, while LOG(REALGDP) enters symmetrically.

Notice also the full expression for the error correction relationship in the Cointegrating Specification table. It is worth pointing out that the expression uses "@CUMDP" and "@CUMDN" functions that contain the series of interest and an initialization date. This is the actual, working specification for these expressions required for evaluation. For brevity, the initialization argument does not appear in the labels used in the Cointegrating Coefficients table results.

We can also conduct a symmetry test by selecting the ARDL Diagnostics/Symmetry Test menu item:

Coefficient symmetry tests
Null hypothesis: Coefficient is symmetric
Degrees of freedom (simple tests): $F(1,191)$, Chi-square(1)
Degrees of freedom (joint tests): $F(2,191)$, Chi-square(2) Equation: EX5

| Variable | Statistic | Value | Probability |
| :--- | :--- | :--- | :--- |
|  | Long-run |  |  |
| LOG(REALGOVT) | F-statistic | 2.061889 | 0.1527 |
|  | Chi-square | 2.061889 | 0.1510 |
| LOG(REALINVS) | F-statistic | 7.855393 | 0.0056 |
|  | Chi-square | 7.855393 | 0.0051 |
|  | Short-run |  |  |
| LOG(REALGOVT) | F-statistic | 2.821345 | 0.0947 |
|  | Chi-square | 2.821345 | 0.0930 |
| Joint (Long-Run and Short-Run) |  |  |  |
| LOG(REALGOVT) | F-statistic | 1.682327 | 0.1887 |
|  | Chi-square | 3.364654 | 0.1859 |

Since LOG(REALGOVT) is a fully asymmetric variable, it is tested for symmetry along both the long-run and short-run dimensions, while LOG(REALINVS) is estimated and tested only in the long-run dimension. We fail to reject symmetry for LOG(REALGOVT) at conventional significance levels in both the long-run and short-run but reject the null of LOG(REALINVS) long-run symmetry at conventional sizes.

To display the dynamic multiplier curves we again click on View/ARDL Diagnostics/
Dynamic Multiplier Graph... to bring up the dynamic multiplier dialog:

| Dynamic Multipliers $\times$ |  |
| :--- | :--- |
| Horizon: | 30 |
| Confidence intervals |  |
| $\square$ Show CI |  |
| $\square$ Shade CI band |  |
| Level: | 0.95 |
| Replications: | 999 |
| OK |  |

Here we will generate the curves for 30 periods without displaying confidence intervals. As before, enter " 30 " as the horizon length, deselect the Show CI checkbox and click on OK to proceed. EViews will display a spool containing CDM graphs for each of the explanatory variables. Of particular interest here is the final curve associated with LOG(REALINVS):


Recall that the LOG(REALINVS) is asymmetric in the long-run, but is symmetric in the short-run. This dynamic behavior is clearly seen by noting the topmost absolute asymmetry curve starts off at zero, then diverges in the long-run.

## Example 6: Asymmetric ARDL $(1,1,1,1,1)$

Next, we add a partially symmetric variable to the list of distributed-lag regressors. In particular, we will treat the log of the $t$-bill rate, LOG(TBILRATE), as a variable which is asymmetric in the short-run, but symmetric in the long-run. Copy EX5, name the copy EX6, then bring up the estimation dialog by clicking on the Estimate button. Enter "LOG(TBILRATE)" in the Short-run asymmetry only edit field, then click on $\mathbf{O K}$ to estimate the updated equation. EViews displays the CEC results:

Dependent Variable: DLOG(REALCONS)
Method: ARDL
Date: 05/04/22 Time: 14:45
Sample (adjusted): 1950Q3 2000Q4
Included observations: 202 after adjustments
Max. dependent lags: 1 (Fixed)
Fixed-lag linear regressors: LOG(REALGDP)
Fixed-lag dual non-linear regressors: LOG(REALGOVT)
Fixed-lag long-run non-linear regressors: LOG(REALINVS)
Fixed-lag short-run non-linear regressors: LOG(TBILRATE)
Deterministics: Restricted constant and no trend (Case 2)
Selected model: ARDL(1,1,1,1,1)

| Variable | Coefficient | Std. Error | t-Statistic | Prob.* |
| :---: | ---: | ---: | ---: | ---: |
| LOG(REALCONS(-1)) | -0.078875 | 0.030258 | -2.606767 | 0.0099 |
| LOG(REALGDP(-1)) | 0.077333 | 0.039794 | 1.943360 | 0.0535 |
| LOG(TBILRATE(-1)) | -0.001179 | 0.000782 | -1.506623 | 0.1336 |
| @CUMDP(LOG(REALGOVT(-1))) | -0.011931 | 0.007672 | -1.555101 | 0.1216 |
| @CUMDN(LOG(REALGOVT(-1)) | -0.001311 | 0.015198 | -0.086232 | 0.9314 |
| @CUMDP(LOG(REALINVS(-1))) | 0.002184 | 0.006571 | 0.332425 | 0.7399 |
| @CUMDN(LOG(REALINVS(-1))) | -0.006239 | 0.005425 | -1.149894 | 0.2516 |
| C | -0.019679 | 0.120458 | -0.163369 | 0.8704 |
| DLOG(REALGDP) | 1.394803 | 0.058198 | 23.96647 | 0.0000 |
| DLOG(REALINVS) | -0.187324 | 0.011409 | -16.41900 | 0.0000 |
| @DCUMDP(LOG(REALGOVT)) | -0.341241 | 0.029695 | -11.49168 | 0.0000 |
| @DCUMDN(LOG(REALGOVT)) | -0.183822 | 0.067059 | -2.741196 | 0.0067 |
| @DCUMDP(LOG(TBILRATE)) | 0.001147 | 0.004388 | 0.261357 | 0.7941 |
| @DCUMDN(LOG(TBILRATE)) | -0.003174 | 0.003929 | -0.807890 | 0.4202 |
| R-squared | 0.797226 | Mean dependent var | 0.008782 |  |
| Adjusted R-squared | 0.783204 | S.D. dependent var | 0.008864 |  |
| S.E. of regression | 0.004127 | Akaike info criterion | -8.075660 |  |
| Sum squared resid | 0.003202 | Schwarz criterion | -7.846374 |  |
| Log likelihood | 829.6417 | Hannan-Quinn criter. | -7.982891 |  |
| F-statistic | 56.85686 | Durbin-Watson stat | 1.913588 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

*Note: p-values and any subsequent tests do not account for model selection.

LOG(TBILRATE) appears as a symmetric long-run variable through the variable labeled "LOG(TBILRATE (-1))", and asymmetrically in the short-run through the variables labeled "@DCUMDP(LOG(TBILRATE))" and @DCUMDN(LOG(TBILARATE))".

Lastly, we display the dynamic multipliers for this specification. Click on View/ARDL Diagnostics/Dynamic Multiplier Graph... to bring up the dynamic multiplier dialog. As before, enter "30" as the horizon length and deselect the Show CI checkbox. Click on OK to continue:


As in the previous example, our focus is on the final graph. LOG(TBILRATE) is asymmetric in the short-run, but symmetric in the long-run. Accordingly, the absolute asymmetry graph on the top is above zero at the start of the evolution, but then settles to zero as we approach the long-run.

## References

Banerjee, A., Dolado, J. J., Galbraith, J. W., Hendry, D., et al. (1993). "Co-integration, error correction, and the econometric analysis of non-stationary data." OUP Catalogue.
Engle, R. F. and Granger, C. W. (1987). "Co-integration and error correction: representation, estimation, and testing." Econometrica: Journal of the Econometric Society, pages 251-276.
Johansen, S. (1991). "Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models." Econometrica: Journal of the Econometric Society, pages 1551-1580.
Johansen, S. (1995). "Likelihood-based inference in cointegrated vector autoregressive models." Oxford University Press on Demand.
Johansen, S. and Juselius, K. (1990). "Maximum likelihood estimation and inference on cointegration with applications to the demand for money." Oxford Bulletin of Economics and Statistics, 52(2):169210.

Narayan, P. K. (2005). "The saving and investment nexus for china: evidence from cointegration tests." Applied Economics, 37(17):1979-1990.

Park, J. (1990). "Testing for unit roots and cointegration by variable addition." Advances in Econometrics, 8(2):107-133.
Perron, P. and Ng, S. (1996). "Useful modifications to some unit root tests with dependent errors and their local asymptotic properties." The Review of Economic Studies, 63(3):435-463.
Pesaran, M. H. and Shin, Y. (1998). "An autoregressive distributed-lag modelling approach to cointegration analysis." Econometric Society Monographs, 31:371-413.
Pesaran, M. H., Shin, Y., and Smith, R. J. (2001). "Bounds testing approaches to the analysis of level relationships." Journal of Applied Econometrics, 16(3):289-326.
Phillips, P. C. and Ouliaris, S. (1990). "Asymptotic properties of residual based tests for cointegration." Econometrica: Journal of the Econometric Society, pages 165-193.
Shin, Yongcheol, Byungchul Yu, and Matthew Greenwood-Nimmo (2014). "Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework." In Festschrift in Honor of Peter Schmidt. 281-314. New York: Springer Science + Business Media.

Stock, J. H. and Watson, M. W. (1988). "Testing for common trends." Journal of the American statistical Association, 83(404):1097-1107.

## Vector Error Correction Models (VECMs)

An important aspect of analyzing the interactions among a group of endogenous variables is the identification of joint long-run and short-run dynamics. While the long-run dynamics are associated with persistent forces and the notion of economic equilibrium, the short-run dynamics are associated with transitory adjustments to long-run states.

While vector autoregressive (VAR) models are ideally suited to the study of contemporaneous dynamics among endogenous variables, the vector error correction model (VECM) is a reparameterization of the VAR process that is specifically designed for analyzing both the long-run and short-run dynamics driving the underlying variables.

## Cointegration

An understanding of VECMs requires a discussion of the notion of system-wide integration and equilibrium. Some important definitions will set the stage for our discussion:

- An individual time series $y_{k t}$ is said to be integrated of order $d, y_{k t} \sim I(d)$, if $\Delta^{d} y_{k t}$ is stationary, or $I(0)$, while $\Delta^{d-1} y_{k t}$ is non-stationary.
- A system of $K$ time series $y_{t}=\left(y_{1}, y_{2 t}, \ldots, y_{K t}\right)^{\prime}$ is said to be integrated of order $d, y_{t} \sim I(d)$, if at least one of its constituent series $y_{k t}$ is $I(d)$, and no series is $I(h)$ for $h>d$. Note that this definition does not preclude a subset of the system series from being of lower order (or even stationary).
- An $I(d)$ system is said to be cointegrated if a linear combination of the constituent series is integrated of (lower) order, $b$ where $b<d$. Further, a system $y_{t}$ that is integrated of order $I(d)$ is said to be cointegrated of order $b$ if there exists a cointegrating $K$-vector $\beta \neq 0$ such that $\beta^{\prime} y_{t} \sim I(b)$. Notice that $\beta$ is not unique since multiplication by any nonzero constant yields a different cointegrating vector.

To simplify the following discussion, we will, without loss of generality, restrict $d$ to 1 and $b$ to 0 so that $y_{t} \sim I(1)$ and $\Delta y_{t} \sim I(0)$.

The concept of cointegration introduced above is closely related to the notion of economic equilibrium. While individual economic processes may have volatile paths of evolution, there may be global forces which eventually produce stable paths of evolution. In particular, a group of economic variables may individually be $I(1)$, or non-stationary, but there may exist cointegrated processes (linear combinations) which are $I(0)$, or stationary. In this case, the cointegrated process is mean-reverting so that it while it may deviate from its expected value in the short-run, it eventually settles at its long-run (asymptotic) expected value.

## The VECM Specification

When $y_{t} \sim I(1)$, the traditional levels-form VAR process is not the most useful representation since both the number and explicit form of any cointegrating relations are not easily obtained from this specification. Consequently, when analyzing cointegrating relationships we typically work with the VECM representation of the process.

## The Basic VECM

Consider a VAR process of order $p$ :

$$
\begin{equation*}
y_{t}=A_{1} y_{t-1}+\ldots+A_{p} y_{t-p}+u_{t} \tag{0.30}
\end{equation*}
$$

where $y_{t}=\left(y_{1 t}, y_{2 t}, \ldots, y_{K t}\right)^{\prime}$ is a $K$-vector of endogenous variables, $A_{1}, \ldots, A_{K}$ are $K \times K$ matrices of coefficients, and the residual vector $u_{t}=\left(u_{1 t}, u_{2 t}, \ldots, u_{K t}\right)^{\prime}$ is distributed with mean 0 and variance matrix $\Sigma$. Note that for simplicity, we assume that there are no deterministic terms in the VAR. This restriction is relaxed in the discussion of "VECMs with Deterministics" on page 366.

The stability of the system is determined by the solutions to the determinant of the characteristic polynomial,

$$
\begin{equation*}
\operatorname{det}\left(I_{K}-A_{1} z-A_{2} z^{2}-\ldots-A_{p} z^{p}\right)=0 \tag{0.31}
\end{equation*}
$$

The process is said to be stable if the roots of the polynomial lie outside the complex unit circle, or have modulus greater than 1 .

Note that when at least one constituent series $y_{k t}$ is $I(1)$, the VAR process is unstable since we may show that

$$
\begin{equation*}
\Pi=-\left(I_{K}-A_{1}-A_{2}-\ldots-A_{p}\right) \tag{0.32}
\end{equation*}
$$

is singular, $\operatorname{det}(\Pi)=0$, and Equation (0.31) is satisfied for roots lying on the unit circle.
In general, $\Pi$ plays a key role in identifying both the number and nature of any cointegrating relationships. To better understand this role, we subtract $y_{t-1}$ from both sides of the VAR representation Equation (0.30) and rearrange terms to obtain the VECM representation:

$$
\begin{align*}
\Delta y_{t} & =\Pi y_{t-1}+\Gamma_{1} \Delta y_{t-1} \ldots+\Gamma_{p-1} \Delta y_{t-(p-1)}+u_{t} \\
& =\Pi y_{t-1}+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t} \tag{0.33}
\end{align*}
$$

where

$$
\begin{equation*}
\Gamma_{j}=-\left(A_{j+1}+\ldots+A_{p}\right) \tag{0.34}
\end{equation*}
$$

for $j=1, \ldots, p-1$.

To obtain this representation, we first take the VAR representation and subtract off the lag of the endogenous variables from both sides:

$$
\begin{equation*}
\Delta y_{t}=\left(A_{1}-I_{K}\right) y_{t-1}+A_{2} y_{t-2}+\ldots+A_{p-1} y_{t-(p-1)}+A_{p} y_{t-p}+u_{t} \tag{0.35}
\end{equation*}
$$

Next, we reparameterize the model by rewriting the remaining elements of the right-hand side as differences. Rewriting with the last two elements of the expression, we have

$$
\Delta y_{t}=\left(A_{1}-I_{K}\right) y_{t-1}+\ldots+\left(A_{p-1}-\Gamma_{p-1}\right) y_{t-(p-1)}+\Gamma_{p-1} \Delta y_{t-(p-1)}+u_{t}
$$

where

$$
\begin{equation*}
\Gamma_{p-1}=-A_{p} \tag{0.36}
\end{equation*}
$$

Similarly, we may transform the last two non-difference terms in this new expression. Focusing on just those two terms, we have

$$
\begin{equation*}
A_{p-2} y_{t-(p-2)}+\left(A_{p-1}-\Gamma_{p-1}\right) y_{t-(p-1)} \tag{0.37}
\end{equation*}
$$

Define

$$
\begin{equation*}
\Gamma_{p-2}=\Gamma_{p-1}-A_{p-1} \tag{0.38}
\end{equation*}
$$

Then we may rewrite the $y_{t-(p-1)}$ term as a difference using

$$
\begin{align*}
& A_{p-2} y_{t-(p-2)}+\left(A_{p-1}-\Gamma_{p-1}\right) y_{t-(p-1)} \\
& =A_{p-2} y_{t-(p-2)}-\Gamma_{p-2} y_{t-(p-1)}  \tag{0.39}\\
& =\left(A_{p-2}-\Gamma_{p-2}\right) y_{t-(p-2)}+\Gamma_{p-2}\left(y_{t-(p-2)}-y_{t-(p-1)}\right) \\
& =\left(A_{p-2}-\Gamma_{p-2}\right) y_{t-(p-2)}+\Gamma_{p-2} \Delta y_{t-(p-2)}
\end{align*}
$$

Notice that this process of rewriting the last two non-difference terms forms a recursion. For the remaining non-difference pairs, we may write,

$$
\begin{gather*}
A_{j} y_{t-j}+\left(A_{j+1}-\Gamma_{j+1}\right) y_{t-(j+1)}=\left(A_{j}-\Gamma_{j}\right) y_{t-j}+\Gamma_{j} \Delta y_{t-j}  \tag{0.40}\\
\Gamma_{j}=\Gamma_{j+1}-A_{j+1}
\end{gather*}
$$

for $j=p-3, \ldots, 1$. Substituting recursion Equation (0.40) into Equation (0.35), we have:

$$
\begin{equation*}
\Delta y_{t}=\left(A_{1}-I_{K}-\Gamma_{1}\right) y_{t-1}+\Gamma_{1} \Delta y_{t-1} \ldots+\Gamma_{p} \Delta y_{t-p+1}+u_{t} \tag{0.41}
\end{equation*}
$$

Then, using the initial value $\Gamma_{p-1}$ from Equation (0.36) and the recursion Equation (0.38), we have

$$
\begin{equation*}
\Gamma_{j}=-\left(A_{j+1}+\ldots+A_{p}\right) \tag{0.42}
\end{equation*}
$$

Note that we may recover the parameters of the VAR from the parameters of the VECM using the relations

$$
\begin{align*}
& A_{1}=\Pi+I_{K}+\Gamma_{1} \\
& A_{j}=\Gamma_{j}-\Gamma_{j-1}  \tag{0.43}\\
& A_{p}=-\Gamma_{p-1}
\end{align*}
$$

for $2 \leq j \leq p-1$.
To see the central role of $\Pi$ in cointegration analysis, we focus on $r$, the matrix rank of $\Pi$, where $0 \leq r \leq K$.

Since our discussion assumes that $y_{t}$ is $I(1)$, it follows that $\operatorname{det}(\Pi)=0$, and the $\Delta y_{t-j}$ are $I(0)$ for all $j \geq 0$. There are two important implications of these conditions. First, since $\operatorname{det}(\Pi)=0$, it follows that $\Pi$ has reduced rank $(r<K)$. Second, since the $\Delta y_{t-j}$ are all $I(0)$, to balance the order of both sides of Equation (0.33), $\Pi y_{t-1}$ must also be $I(0)$.

For any $r>0$, there exist $K \times r$ matrices $\alpha$ and $\beta$ each of rank $r$, such that

$$
\begin{equation*}
\Pi=\alpha \beta^{\prime} \tag{0.44}
\end{equation*}
$$

where ' is the transpose operator. Then we may write

$$
\begin{equation*}
\Pi y_{t-1}=\alpha\left(\beta^{\prime} y_{t-1}\right) \tag{0.45}
\end{equation*}
$$

and given our assumptions, $\beta^{\prime} y_{t-1}$ must be an $I(0)$ linear combination of the series in the system, with $r$ representing the cointegrating rank, and $\beta$ the $K \times r$ cointegrating matrix. $\alpha$ is typically referred to as the loading matrix.

Note that although $\beta$ is not unique, a suitable normalization is possible by rearranging the variables so that the first $r$ rows of the matrix are linearly independent:

$$
\beta=\left[\begin{array}{c}
I_{r}  \tag{0.46}\\
\beta_{(K-r)}
\end{array}\right]
$$

where $\beta_{(K-r)}$ is a $(K-r) \times r$ matrix. See Lütkepohl (2005) for details.
Lastly if $r=0$, balancing both sides of Equation (0.35) requires $\Pi=0$. In this case we say that there are no cointegrating relations since no linear combinations of $y_{t}$ are $I(0)$.

## Basic Estimation

While there are several methods for estimation of VECMs, we focus on the maximum likelihood (ML) variant, also known as reduced rank regression (RRR) (see Johansen (1995) and Lütkepohl (2005) for a detailed exposition).

Formally, RRR assumes a known cointegration rank $r$, Gaussian innovation vectors the innovation vectors $u_{t}$, a time dimension of length $T$, and is best described using the VECM matrix representation (Equation (0.33)):

$$
\begin{equation*}
\Delta Y=\Pi Y_{-1}+\Gamma \Delta X+U \tag{0.47}
\end{equation*}
$$

where

$$
\begin{align*}
\Delta Y & =\left[\Delta y_{1}, \ldots, \Delta y_{T}\right] \\
Y_{-1} & =\left[y_{0}, \ldots, y_{T-1}\right] \\
\Gamma & =\left[\Gamma_{1}, \ldots, \Gamma_{p-1}\right]  \tag{0.48}\\
\Delta X & =\left[\Delta X_{1}, \ldots, \Delta X_{T}\right] \\
\Delta X_{t} & =\left[\Delta y_{t-1}, \ldots, \Delta y_{t-(p-1)^{\prime}}\right]^{\prime} \\
U & =\left[u_{1}, \ldots, u_{T}\right]
\end{align*}
$$

The RRR estimator is then the maximizer of the log-likelihood objective function:

$$
\begin{align*}
& \log L(\alpha, \beta, \Gamma)=-\frac{K T}{2} \log (2 \pi)-\frac{T}{2} \log \left|\Sigma_{u}\right|  \tag{0.49}\\
& -\frac{1}{2} \operatorname{tr}\left(\left[\Delta Y-\alpha \beta^{\prime} Y_{-1}-\Gamma \Delta X\right]^{\prime} \Sigma_{u}^{-1}\left[\Delta Y-\alpha \beta^{\prime} Y_{-1}-\Gamma \Delta X\right]\right)
\end{align*}
$$

Johanson (1995) shows that optimizing the likelihood is equivalent to solving the eigenvalue problem

$$
\begin{equation*}
\left|\lambda S_{11}-S_{10} S_{00}^{-1} S_{10}{ }^{\prime}\right|=0 \tag{0.50}
\end{equation*}
$$

under the constraint $V^{\prime} S_{11} V=I_{K}$, where $\lambda=\left(\lambda_{1}, \ldots, \lambda_{K}\right)$ are the $K$ eigenvalues associated with eigenvector matrix $V=\left[v_{1}, \ldots, v_{K}\right]$, and

$$
\begin{align*}
S_{i j} & =\frac{1}{T} R_{i} R_{j}^{\prime} \\
R_{0} & =\Delta Y M  \tag{0.51}\\
R_{1} & =Y_{-1} M \\
M & =I_{T}-\Delta X\left(\Delta X^{\prime} \Delta X\right)^{-1} \Delta X^{\prime}
\end{align*}
$$

Solving the constrained eigenvalue problem yields $\lambda$ that are the eigenvalues of the symmetric matrix

$$
\begin{equation*}
\left(S_{11}^{-1 / 2} S_{10}\right) S_{00}^{-1}\left(S_{11}^{-1 / 2} S_{10}\right)^{\prime} \tag{0.52}
\end{equation*}
$$

In terms of computation, note that $R_{0}$ and $R_{1}$ are the residuals from regression of the $\Delta Y$ and $Y_{-1}$ on the $\Delta X$. Further, $S_{11}^{-1 / 2}$ may be obtained by first diagonalizing $S_{11}$ using the solution to the auxiliary eigenvalue problem

$$
\begin{equation*}
\left|\rho I-S_{11}\right|=0 \tag{0.53}
\end{equation*}
$$

to obtain eigenvalues $\rho=\left(\rho_{1}, \ldots, \rho_{K}\right)$ and associated eigenvector matrix $W=\left[w_{1}, \ldots, w_{K}\right]$. The square root of the inverse of $S_{11}$ can then be estimated as

$$
\begin{equation*}
S_{11}^{-1 / 2}=W \operatorname{diag}\left(\rho_{1}^{2}, \ldots, \rho_{K}^{2}\right) W^{\prime} \tag{0.54}
\end{equation*}
$$

and the log-likelihood Equation (0.49) is maximized at

$$
\begin{align*}
& \hat{\beta}=V^{\prime} S_{11}^{-1 / 2} \\
& \hat{\alpha}=S_{01} \hat{\beta}\left(\beta^{\prime} \hat{S_{11}} S_{11}\right)^{-1} \\
& \hat{\Gamma}=\left(\Delta Y-\hat{\alpha} \hat{\beta}^{\prime} Y_{-1}\right) \Delta X^{\prime}\left(\Delta X^{\prime} \Delta X\right)^{-1}  \tag{0.55}\\
& \hat{\Sigma}=\frac{1}{T}\left(\Delta Y-\hat{\alpha} \hat{\beta}^{\prime} Y_{-1}-\hat{\Gamma} \Delta X\right)\left(\Delta Y-\hat{\alpha} \hat{\beta}^{\prime} Y_{-1}-\hat{\Gamma} \Delta X\right)^{\prime}
\end{align*}
$$

## VECMs with Deterministics

The discussion thus far has ignored the presence of deterministic terms in the VAR specification. The inclusion of deterministics has important implications for the estimation and interpretation of VECMs, and there are different approaches to incorporating these terms.

## The Classical Approach

Following Lütkepohl (2005), the classical approach to incorporating deterministic terms in VECMs is to let $Y_{t}$ follow a basic $\operatorname{VAR}(p)$ as in Equation (0.30), and to work with the augmented process,

$$
\begin{equation*}
y_{t}{ }^{*}=f(t)+y_{t} \tag{0.56}
\end{equation*}
$$

where $f(t)$ denotes any $K$-dimensional deterministic function of time, often a low-order polynomial in $t$.

Substituting $y_{t}=y_{t}{ }^{*}-f(t)$ into the VECM Equation (0.33), we obtain:

$$
\begin{align*}
\Delta y_{t}^{*} & =\Delta f_{t}+\Pi\left(y_{t-1}^{*}-f(t)\right)+\sum_{j=1}^{p-1} \Gamma_{j}\left(\Delta y_{t-j}^{*}-\Delta f(t-j)\right)+u_{t}  \tag{0.57}\\
& =\left(\Delta f(t)-\sum_{j=1}^{p-1} \Gamma_{j} \Delta f(t-j)\right)+\Pi\left(y_{t-1}^{*}-f(t)\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}^{*}+u_{t}
\end{align*}
$$

For example, if $f(t)$ is a constant function, $f(t)=\phi_{0}$, then $\Delta f(t)=0$ for all $t$, and we have

$$
\begin{align*}
\Delta y_{t}^{*} & =\Pi\left(y_{t-1}{ }^{*}-\phi_{0}\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}^{*}+u_{t} \\
& =v_{0}+\Pi y_{t-1} *+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}{ }^{*}+u_{t}  \tag{0.58}\\
v_{0} & =-\Pi \phi_{0}
\end{align*}
$$

In this case, the constant function $f(t)$ may either be viewed as an intercept inside the cointegrating relation, $\Pi\left(y_{t-1}{ }^{*}-\phi_{0}\right)$, or simply as an overall intercept $v_{0}$ in the VECM. Importantly, in the latter case, the overall $v_{0}$ is said to be restricted since it must satisfy the restriction $v_{0}=-\Pi \phi_{0}$ imposed by the cointegrating relationship.

Likewise, if $f(t)$ is a linear trend, $f(t)=\phi_{0}+\phi_{1} t$, then $\Delta f(t)=\phi_{1}$ for all $t$, we have

$$
\begin{align*}
\Delta y_{t}^{*} & =\left(I_{K}-\sum_{j=1}^{p-1} \Gamma_{j}\right) \phi_{1}+\Pi\left(y_{t-1}^{*}-\phi_{0}-\phi_{1} t\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}^{*}+u_{t} \\
& =v_{0}+\Pi\left(y_{t-1}^{*}-\phi_{1} t\right)+\sum_{j=1}^{p-1} \Gamma_{j}\left(\Delta y_{t-j}^{*}\right)+u_{t} \\
& =v_{0}+v_{1} t+\Pi y_{t-1}^{*}+\sum_{j=1}^{p-1} \Gamma_{j}\left(\Delta y_{t-j}^{*}\right)+u_{t}  \tag{0.59}\\
v_{0} & =-\Pi \phi_{0}+\left(I_{K}-\sum_{j=1}^{p-1} \Gamma_{j}\right) \phi_{1} \\
v_{1} & =-\Pi \phi_{1}
\end{align*}
$$

In this case, the trend function $f(t)$ may be included as a term in the cointegrating relation, $\Pi\left(y_{t-1} *-\phi_{0}-\phi_{1} t\right)$ along with the $\left(I_{K}-\sum \Gamma_{j}\right) \phi_{1}$ term appearing in the short-run dynamics, or as an overall intercept and trend in VECM ( $v_{0}+v_{1} t$ ). Notably, while the overall trend coefficient $v_{1}$ is restricted by the cointegrating relationship, the constant $v_{0}$ is unrestricted as it contains free parameters unrelated to $\Pi$ from the short-run dynamics.

Lütkepohl (2005) emphasizes the importance of the cointegrating restrictions in governing the dynamic behavior of the levels of $y_{t}{ }^{*}$, noting that their removal induces additional deterministics in the integrated VAR representation of the VECM. For example, if the restriction on $v_{0}$ in Equation ( 0.58 ) is removed, the corresponding integrated VAR specification will have a deterministic trend in the mean. Similarly, removing the restriction on $v_{1}$ in Equation (0.59) will generate a quadratic trend in the VAR.

We can make the division between restricted and unrestricted deterministics concrete by reparameterizing Equation ( 0.57 ) to provide a general framework for a VECM with deterministics:

$$
\begin{equation*}
\Delta y_{t}^{*}=\gamma \tau(t)+\Pi\left(y_{t-1}^{*}-\eta \tau_{\perp}(t-1)\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}^{*}+u_{t} \tag{0.60}
\end{equation*}
$$

where $\tau(t)$ and $\tau_{\perp}(t)$ are vector-valued functions denoting unrestricted and restricted deterministics, respectively, with corresponding coefficients $\gamma$ and $\eta . \tau(t)$ and $\tau_{\perp}(t)$ are assumed to be exclusive so that any deterministic function in $\tau(t)$ is not included in $\tau_{\perp}(t)$,
and vice versa. For the constant function $f(t)=\phi_{0}$ above, $\tau(t)$ is empty and $\tau_{\perp}(t)=[1]$. For the linear trend function $f(t)=\phi_{0}+\phi_{1} t, \tau(t)=$ [1], and $\tau_{\perp}(t)=[t]$.

Lastly, note that while this discussion has focused on deterministic functions of time, the framework allows for the consideration of other types of exogenous variables which enter either the restricted cointegrating or the unrestricted transitory space.

## The Johansen, Hendry, and Juselius Approach

An alternative treatment of deterministics follows the conventions outlined in Johansen (1995), Hendry and Juselius (2001), and Juselius (2006), which we will term the JHJ approach. The approach begins with the VECM specification:

$$
\begin{equation*}
\Delta y_{t}=f(t)+\Pi y_{t-1}+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t} \tag{0.61}
\end{equation*}
$$

By virtue of cointegration, both $\Delta X_{t}$ and $\beta^{\prime} y_{t-1}$ are stationary around their expected values. Taking expectations of Equation (0.61) yields:

$$
\begin{align*}
\left(I_{K}-\Gamma(L)\right) E\left(\Delta y_{t}\right) & =f(t)+\alpha E\left(\beta^{\prime} y_{t-1}\right) \\
\Gamma(L) & =\sum_{j=1}^{p-1} \Gamma_{j} L^{j} \tag{0.62}
\end{align*}
$$

where $L$ is the lag operator.
Let $g(t)=E\left(\Delta y_{t}\right)$ and $h(t)=E\left(\beta^{\prime} y_{t-1}\right)$ be the expected value paths of $\Delta y_{t}$ and $\beta^{\prime} y_{t-1}$. Then rewriting Equation (0.62) in terms of the deterministic component $f(t)$ yields,

$$
\begin{equation*}
f(t)=\left(I_{K}-\Gamma(L)\right) g(t)-\alpha h(t) \tag{0.63}
\end{equation*}
$$

Johansen (1995) and Juselius (2006) show that $g(t)$ and $h(t)$ may be thought of as transformations of $f(t)$ in the direction $\beta_{\perp}$ and $\beta$, where $\beta_{\perp}$ is the orthogonal complement of the latter. In other words, the deterministic function $f(t)$ may be additively decomposed into the space spanned by the transitory variables $\Delta y_{t}$, and the space spanned by the cointegrating relations $\beta^{\prime} y_{t-1}$. Note that this approach differs from the classical approach in that deterministic terms can appear simultaneously in both the unrestricted (transitory) and in the restricted (cointegrated) spaces.

When $f(t)$ is a constant function so that $g(t)=\gamma_{0}$ and $h(t)=\eta_{0}$ are also constant functions, the VECM in Equation (0.61) may be written as:

$$
\begin{align*}
\Delta y_{t} & =\left(I_{K}-\Gamma(L)\right) \gamma_{0}-\alpha \eta_{0}+\Pi y_{t-1}+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t}  \tag{0.64}\\
& =\gamma_{0}+\alpha\left(\beta^{\prime} y_{t-1}-\eta_{0}\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t}
\end{align*}
$$

Similarly, when $f(t)$ is a linear trend function so that $g(t)=\gamma_{0}+\gamma_{1} t$ and $h(t)=\eta_{0}+\eta_{1} t$, the VECM in Equation (0.61) is given by

$$
\begin{align*}
\Delta y_{t} & =\left(I_{K}-\Gamma(L)\right)\left(\gamma_{0}+\gamma_{1} t\right)-\alpha\left(\eta_{0}+\eta_{1} t\right)+\Pi y_{t-1}+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t}  \tag{0.65}\\
& =\gamma_{0}+\gamma_{1} t+\alpha\left(\beta^{\prime} y_{t-1}-\eta_{0}-\eta_{1} t\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t}
\end{align*}
$$

## Estimating Models with Deterministics

Estimation of VECMs with deterministics requires modification of the approach outlined in "Basic Estimation" on page 364.

Deterministic specifications which derive from the classical approach in Equation (0.57) are accommodated by including the restricted deterministic regressors in the cointegrating space, and the unrestricted deterministics in the overall VECM. We modify Equation (0.47) to provide:

$$
\begin{equation*}
\Delta Y=\Pi^{+} Y_{-1}^{+}+\Gamma^{+} \Delta X^{+}+U \tag{0.66}
\end{equation*}
$$

where

$$
\begin{align*}
\Delta Y & =\left[\Delta y_{1}, \ldots, \Delta y_{T}\right] \\
Y_{-1}^{+} & =\left[y_{0}^{+}, \ldots, y_{T-1}^{+}\right] \\
y_{t}^{+} & =\left[y_{t}^{\prime}, \tau_{\perp}(t)^{\prime}\right]^{\prime} \\
\Pi^{+} & =\alpha\left[\beta^{\prime}, \eta^{\prime}\right]  \tag{0.67}\\
\Gamma^{+} & =\left[\Gamma_{1}, \ldots, \Gamma_{p-1}, \gamma\right] \\
\Delta X^{+} & =\left[\Delta X_{1}^{+}, \ldots, \Delta X_{T}^{+}\right] \\
\Delta X_{t}^{+} & =\left[\Delta y_{t-1}^{\prime}, \ldots, \Delta y_{t-(p-1)^{\prime}}, \tau(t)^{\prime}\right]^{\prime} \\
U & =\left[u_{1}, \ldots, u_{T}\right]
\end{align*}
$$

where $\gamma$ and $\tau(t), \eta$ and $\tau_{\perp}(t)$ are the unrestricted and restricted deterministic components and coefficients, respectively.

When deterministics are incorporated using the JHJ convention, the estimator must allow for the possibility that the same deterministic term can appear both inside the cointegrating equation and outside it.

It is useful to divide the cointegrating regressors and coefficients into those present only inside the cointegrating equation ( $\tau_{1 \perp}$ and $\eta_{1}$ ), those present only outside the equation ( $\tau_{1}$ and $\left.\gamma_{1}\right)$, and those that are both inside and outside the cointegrating relation $\left(\tau_{2}{ }^{*}, \eta_{2}\right.$, and $\gamma_{2}$ ) so that we have $\tau_{\perp}=\left[\tau_{1 \perp}{ }^{\prime}, \tau_{2}{ }^{* \prime}\right]^{\prime}$ and $\tau=\left[\tau_{1}{ }^{\prime}, \tau_{2}{ }^{* \prime}\right]^{\prime}$, with coefficients $\gamma=\left[\gamma_{1}, \gamma_{2}\right]$, and $\eta=\left[\eta_{1}, n_{2}\right]$.

One estimation approach hinges on the ideas that the cointegrating (equilibrium) equation is stable around its mean of zero:

$$
\begin{equation*}
E\left(\beta^{\prime} y_{t}-\eta_{1}{ }^{\prime} \tau_{1 \perp}(t-1)-\eta_{2}{ }^{\prime} \tau_{2}{ }^{*}(t-1)\right)=0 \tag{0.68}
\end{equation*}
$$

Given this requirement, estimation may be conducted in three steps:

- Step 1: All dual deterministic regressors $\tau_{2}{ }^{*}$ are first removed from inside the cointegrating relationship, but retained outside. Then $\beta, \alpha$ and $\eta_{1}$ are estimated using the classical approach outlined in Equation (0.55) and Equation (0.67) using deterministics $\tau_{1 \perp}$ and coefficients $\eta_{1}$.
- Step 2: Given the estimates $\hat{\beta}$ and $\hat{\eta}_{1}$ from Step $1, \eta_{2}$ is estimated by choosing values of the coefficients so that the cointegrating equation has conditional mean zero:

$$
\begin{equation*}
E\left(\hat{\beta}^{\prime} y_{t}-\hat{\eta}_{1}{ }^{\prime} \tau_{1 \perp}(t-1)-\eta_{2}{ }^{\prime} \tau_{2}{ }^{*}(t-1)\right)=0 \tag{0.69}
\end{equation*}
$$

- Step 3: Using the estimates from Steps 1 and 2 , the short-run coefficients $\Gamma^{+}$are estimated using appropriately modified versions of the expressions in Equation (0.55) and Equation (0.67) with deterministics $\tau$ and coefficients $\gamma$.

While there is no single approach for estimating the coefficients in Step 2 above, a simple linear regression of $\hat{\beta}^{\prime}\left(y_{t}-\hat{\eta}_{1}{ }^{\prime} \tau_{1 \perp}(t-1)\right)$ on $\tau_{2}{ }^{*}(t-1)$ ) satisfies the desired condition. When the deterministic regressors are the usual constant and trend, this regression reduces to a familiar least squares detrending of the cointegrating relation. See also Proposition 7.5 in Lütkepohl (2005). This is the method employed by EViews.

For example, consider a model where the constant and trend terms appear both inside and outside the cointegrating equation:

$$
\begin{equation*}
\Delta y_{t}==\gamma_{0}+\gamma_{1} t+\alpha\left(\beta^{\prime} y_{t-1}-\eta_{0}^{\prime}-\eta_{1}^{\prime} t\right)+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t} \tag{0.70}
\end{equation*}
$$

- Step 1 estimates the classical model,

$$
\begin{equation*}
\Delta y_{t}==\gamma_{0}+\gamma_{1} t+\alpha \beta^{\prime} y_{t-1}+\sum_{j=1}^{p-1} \Gamma_{j} \Delta y_{t-j}+u_{t} \tag{0.71}
\end{equation*}
$$

to obtain estimates $\hat{\beta}$ and $\hat{\alpha}$, along with $\tilde{\gamma}_{0}, \tilde{\gamma}_{1}$, and $\tilde{\Gamma}_{j}$ for $j=1, \ldots, p-1$.

- Step 2 uses the least squares regression,

$$
\begin{equation*}
\hat{\beta}^{\prime} y_{t-1}=\eta_{0}{ }^{\prime}+\eta_{1}{ }^{\prime} t+\epsilon_{t} \tag{0.72}
\end{equation*}
$$

to obtain estimates $\hat{\eta}_{0}$ and $\hat{\eta}_{1}$.

- Step 3 updates the estimates of $\gamma_{0}, \gamma_{1}$, and the $\Gamma_{j}$ using standard regression,

$$
\begin{equation*}
\hat{\Gamma}^{+}=\left[\hat{\Gamma}_{1}, \ldots, \hat{\Gamma}_{p-1}, \hat{\gamma}\right]=\left(Y-\hat{\alpha}\left[\hat{\beta}^{\prime}, \hat{\eta}_{0}{ }^{\prime}, \hat{\eta}_{1}^{\prime}\right] Y_{-1}^{+}\right) \Delta X^{\prime}\left(\Delta X^{\prime} \Delta X\right)^{-1} \tag{0.73}
\end{equation*}
$$

The final coefficient estimates are given by $\hat{\beta}$ and $\hat{\alpha}$ from Step 1, $\hat{\eta}_{0}$ and $\hat{\eta}_{1}$ from Step 2, and $\hat{\Gamma}^{+}$from Step 3.

## Popular Deterministic Models

The empirical literature has centered around five scenarios involving deterministic terms:

- Case 1: No deterministics, so that $f(t)=0, \tau(t)=\tau_{\perp}(t)=$ [0] in the classical framework, and $g(t)=h(t)=0$ in the JHJ approach.
- Case 2: Restricted constant, so that $\tau(t)=[0]$ and $\tau_{\perp}(t)=[1]$ and $\eta_{0} \neq 0$ in the classical formulation, and $g(t)=0$ and $h(t)=\eta_{0} \neq 0$ under JHJ.
Here, the constant is restricted to the cointegrating space and does not appear in the transitory space. The cointegrating mean is non-zero and the cointegrating equation restriction ensures that no linear trends exist in the corresponding VAR.
- Case 3: Unrestricted constant, so that $\tau(t)=[1], \gamma_{0} \neq 0$ and $\tau_{\perp}(t)=[0]$ in the classical approach, and $g(t)=\gamma_{0} \neq 0$ and $h(t)=0$ for JHJ.

In this case, the constant appears in the transitory space but does not appear in the cointegrating space. The model has no deterministics in the cointegrating space, and exhibits a linear trend in the VAR representation.

- Case 3 (JHJ): Unrestricted constant in both the transitory and the cointegrating space so that $g(t)=\gamma_{0} \neq 0$ and $h(t)=\eta_{0} \neq 0$.

This JHJ model has a non-zero mean in the cointegrating relations, and has a linear trend in the corresponding VAR. Note that this case reduces to classical Case 3 when the restriction $\eta_{0}=0$ is imposed.

- Case 4: Unrestricted constant and restricted trend so that $\tau(t)=[1], \gamma_{0} \neq 0$ and $\tau_{\perp}(t)=[t], \gamma_{1} \neq 0$ in the classical approach; $g(t)=\gamma_{0} \neq 0$ and $h(t)=\eta_{1} t$ for $\eta_{1} \neq 0$ under JHJ.
Here, a constant appears in the transitory space but does not appear in the cointegrating space, and the trend term appears only in the cointegrating space. This model has
a linear trend in the cointegrating relations, and has a linear trend in the corresponding VAR representation.
- Case 4 (JHJ): Unrestricted constant and restricted trend so that $g(t)=\gamma_{0} \neq 0$ and $h(t)=\eta_{0}+\eta_{1} t$ in the JHJ framework.
The constant appears in both the transitory space and the cointegrating space, while the trend term appears only in the cointegrating space. The model has a non-zero mean and non-zero trend in the cointegrating relations, and has a linear trend in the corresponding VAR formulation. Note that this case reduces to classical Case 4 when the restriction $\eta_{0}=0$ is imposed.
- Case 5: Unrestricted constant and trend, so that $\tau(t)=[1, t], \gamma_{0}, \gamma_{1} \neq 0$ and $\tau_{\perp}(t)=[0]$ in the classical approach; $g(t)=\gamma_{0}+\gamma_{1} t, \gamma_{0}, \gamma_{1} \neq 0$ and $h(t)=0$ under JHJ.
Here, the constant and trend appears in the transitory space. The model has a nonzero mean and non-zero trend in the cointegrating relations, and has a quadratic trend in the corresponding VAR.
- Case 5 (JHJ): Unrestricted constant and trend, so that $g(t)=\gamma_{0}+\gamma_{1} t, \gamma_{0}, \gamma_{1} \neq 0$ and $h(t)=\eta_{0}+\eta_{1} t, \eta_{0}, \eta_{1} \neq 0$.
In this case, the constant and trend appear in both the transitory and cointegrating spaces. The model has a non-zero mean and non-zero trend in the cointegrating relations, and has a quadratic trend in the corresponding VAR formulation. Note that this case reduces to classical Case 5 when the restrictions $\eta_{0}, \eta_{1}=0$ are imposed.


## Determination of Cointegrating Rank

A fundamental prerequisite of VECM estimation is a priori knowledge of the number of cointegrating relations $r$. Accordingly, it is important to discuss formal methods for determining this order.

As in Equation (0.33), we begin with a VECM with $K$ endogenous variables and no deterministic terms, and suppose that $\Pi$ has rank $\operatorname{rk}(\Pi)=r$ with $0 \leq r \leq K$.

Then, two different sets of hypotheses are of interest:

1. $H_{0}: \operatorname{rk}(\Pi)=r_{0}$ versus $H_{1}: r_{0}<\operatorname{rk}(\Pi) \leq r_{1}=K$
2. $H_{0}: \operatorname{rk}(\Pi)=r_{0}$ versus $H_{1}: r_{0}<\operatorname{rk}(\Pi)=r_{1}=r_{0}+1$

Both sets of hypotheses may be tested with the likelihood ratio statistic:

$$
\begin{align*}
\lambda_{L R}\left(r_{0}, r_{1}\right) & =2\left(\ln \left(L\left(r_{1}\right)\right)-\ln \left(L\left(r_{0}\right)\right)\right) \\
& =T\left(-\sum_{i=1}^{r_{1}} \ln \left(1-\lambda_{i}\right)+\sum_{i=1}^{r_{0}} \ln \left(1-\lambda_{i}\right)\right)  \tag{0.74}\\
& =-T\left(\sum_{i=r_{0}+1}^{r_{1}} \ln \left(1-\lambda_{i}\right)\right)
\end{align*}
$$

where $L(s)$ denotes the maximized value of the Gaussian likelihood function for cointegration rank $s$, and $\lambda_{i}$ are the eigenvalues associated with the symmetric matrix in Equation (0.52).

The asymptotic distributions of $\lambda_{L R}\left(r_{0}, r_{1}\right)$ are nonstandard and are given by:

$$
\begin{align*}
\lambda_{L R}\left(r_{0}, K\right) & \rightarrow \operatorname{tr}(D) \\
\lambda_{L R}\left(r_{0}, r_{0}+1\right) & \rightarrow \lambda_{\max }(D) \tag{0.75}
\end{align*}
$$

where $\operatorname{tr}(D)$ and $\lambda_{\max }(D)$ are respectively the trace and maximum eigenvalue of the matrix

$$
\begin{equation*}
D=\left(\int_{0}^{1} W d W^{\prime}\right),\left(\int_{0}^{1} W W^{\prime} d s\right)\left(\int_{0}^{1} W d W^{\prime}\right) \tag{0.76}
\end{equation*}
$$

where $W=W_{K-r_{0}}(s)$ is the $K-r_{0}$-dimensional standard Wiener process. $\lambda_{L R}\left(r_{0}, K\right)$ and $\lambda_{L R}\left(r_{0}, r_{0}+1\right)$ are commonly referred to as the trace eigenvalue test and maximum eigenvalue test, respectively.

The basic approach to cointegration rank determination is to run a battery of these hypothesis tests starting with the null hypothesis $r_{0}=0$. If a test rejects, increase the null $r_{0}$ by one and repeat the test. The estimated cointegration rank is then the first (minimum) $r_{0}$ for which the test fails to reject the null.

When deterministic terms are present, the test statistics and the test strategy are unchanged, although the eigenvalues of the matrix in Equation (0.52) will differ since they are based on the extended regressors $Y_{-1}^{0}$ and $\Delta X^{0}$ instead of $Y_{-1}$ and $\Delta X$. This difference is naturally reflected in the limiting distribution of the statistics. Critical values for statistics with and without deterministic terms are tabulated in MacKinnon, Haug and Michelis (1999).

## Estimating VEC Models in EViews

Estimation of VEC model in EViews is a special case of estimation in a var object. From the main application menu of an existing var object, click on the Estimate button to open the VAR Specification estimation dialog. Alternately, you may create a new VAR object by
selecting Object/New Object... group, then selecting VAR. Once the dialog appears, select Vector Error Correction in the Method dropdown menu to display the VEC estimation dialog:


Once you have filled the in the dialog, simply click $\mathbf{O K}$ to estimate the VEC. Estimation of a VEC model is carried out in two steps. In the first step, we estimate the cointegrating relations from the Johansen procedure as used in the cointegration test. We then construct the error correction terms from the estimated cointegrating relations and estimate a VAR in first differences including the error correction terms as regressors.

There are three tabs in the dialog: Basics, Cointegration, and VEC Restrictions. We discuss each of these tabs in turn.

## Basics

In the Basics tab, you will provide the usual information about the Lag intervals, Estimation sample, Endogenous variables, and lists of different types of Exogenous variables:

- Importantly, in contrast to the standard VAR case, the Lag intervals specification refers to lags of the first difference terms in the conditional EC representation of the VEC. For example, the lag specification " 11 " will include lagged first difference terms
on the right-hand side of the VEC. Rewritten in levels, this VEC is a restricted VAR with two lags. To estimate a VEC with no lagged first difference terms, specify the lag as " 0 " "
- The Exogenous variables section allows you to specify exogenous variables that are not included in the standard in-built deterministic trend cases. This convention means that the constant and linear trend term should not be included in the Exogenous variables edit boxes. The constant and trend specification for VECs should be specified using the dropdown menu in the Cointegration tab.

You should enter any other variables in the edit field corresponding to whether they appear in the Short-run, Long-run, or Both log-run and short-run lists of variables.

## Cointegration

Important options related to cointegration can be accessed by clicking on the Cointegration tab:

| VAR Specification |
| :--- | :--- | :--- |
| BasicsCointegration <br> Cointegrating rank <br> Number of cointegrating relations: <br> Deterministic trend specifications <br> 3 (JHJ): Unrest. constant <br> Case 3 (Johansen-Hendry-Juselius): <br> Constant belongs to short-run regressors and <br> artificially to long-run regressors. Associated VAR has <br> both a constant and trend. |

- As VEC estimation requires a priori specification of the Number of cointegrating relations, you should use the dropdown menu to select the appropriate number.
- Furthermore the Deterministic trend specification governs the use of trend terms in the CEC specification as described in "Popular Deterministic Models" on page 371. Note that as you make a selection in the dropdown, the text below will change to give you a more detailed description of the assumptions underlying the choice.


## VEC Restrictions

Since the cointegrating vector $\beta$ is not fully identified, EViews applies standard normalizations to identify the remaining coefficients. Alternately, you may wish to impose your own identifying restrictions when performing estimation. Restrictions may be imposed on the cointegrating vector (elements of the $\beta$ matrix) and/or on the adjustment coefficients (elements of the $\alpha$ matrix).

To impose restrictions in estimation, click on the VEC Restrictions tab to display the restrictions dialog. You will enter your restrictions in the edit box that appears when you check the Impose Restrictions box:

"Specifying VEC Restrictions" on page 377 describes the syntax for specifying these restrictions in greater detail.

## Specifying VEC Restrictions

Restrictions can be imposed on the cointegrating vector (elements of the $\beta$ matrix) and/or on the adjustment coefficients (elements of the $\alpha$ matrix).

## Restrictions on the Cointegrating Vector

To impose restrictions on the cointegrating vector $\beta$, you must refer to the ( $i, j$ )-th element of the transpose of the $\beta$ matrix by $\mathrm{B}(i, j)$. The $i$-th cointegrating relation has the representation:

```
B(i,1)*y1 + B(i,2)*y2 + ... + B(i,k)*yk
```

where $y 1, y 2, \ldots$ are the (lagged) endogenous variable. Then, if you want to impose the restriction that the coefficient on y 1 for the second cointegrating equation is 1 , you would type the following in the edit box:

$$
B(2,1)=1
$$

You can impose multiple restrictions by separating each restriction with a comma on the same line or typing each restriction on a separate line. For example, if you want to impose the restriction that the coefficients on $y 1$ for the first and second cointegrating equations are 1 , you would type:

$$
\begin{aligned}
& B(1,1)=1 \\
& B(2,1)=1
\end{aligned}
$$

Currently all restrictions must be linear (or more precisely affine) in the elements of the $\beta$ matrix. So for example

$$
B(1,1) * B(2,1)=1
$$

will return a syntax error.

## Restrictions on the Adjustment Coefficients

To impose restrictions on the adjustment coefficients, you must refer to the ( $i, j$ )-th elements of the $\alpha$ matrix by A $(i, j)$. The error correction terms in the $i$-th VEC equation will have the representation:

```
A(i,1)*CointEq1 + A(i,2)*CointEq2 + ... + A(i,r)*CointEqr
```

Restrictions on the adjustment coefficients are currently limited to linear homogeneous restrictions so that you must be able to write your restriction as $R \cdot \operatorname{vec}(\alpha)=0$, where $R$ is a known $q k \times r$ matrix. This condition implies, for example, that the restriction,

$$
A(1,1)=A(2,1)
$$

is valid but:

$$
A(1,1)=1
$$

will return a restriction syntax error.

One restriction of particular interest is whether the $i$-th row of the $\alpha$ matrix is all zero. If this is the case, then the $i$-th endogenous variable is said to be weakly exogenous with respect to the $\beta$ parameters. See Johansen (1995) for the definition and implications of weak exogeneity. For example, if we assume that there is only one cointegrating relation in the VEC, to test whether the second endogenous variable is weakly exogenous with respect to $\beta$ you would enter:

$$
A(2,1)=0
$$

To impose multiple restrictions, you may either separate each restriction with a comma on the same line or type each restriction on a separate line. For example, to test whether the second endogenous variable is weakly exogenous with respect to $\beta$ in a VEC with two cointegrating relations, you can type:

$$
\begin{aligned}
& A(2,1)=0 \\
& A(2,2)=0
\end{aligned}
$$

You may also impose restrictions on both $\beta$ and $\alpha$. However, the restrictions on $\beta$ and $\alpha$ must be independent. So for example,

$$
\begin{aligned}
& A(1,1)=0 \\
& B(1,1)=1
\end{aligned}
$$

is a valid restriction but:

$$
A(1,1)=B(1,1)
$$

will return a restriction syntax error.

## Identifying Restrictions and Binding Restrictions

EViews will check to see whether the restrictions you provided identify all cointegrating vectors for each possible rank. The identification condition is checked numerically by the rank of the appropriate Jacobian matrix; see Boswijk (1995) for the technical details. Asymptotic standard errors for the estimated cointegrating parameters will be reported only if the restrictions identify the cointegrating vectors.

If the restrictions are binding, EViews will report the LR statistic to test the binding restrictions. The LR statistic is reported if the degrees of freedom of the asymptotic $\chi^{2}$-distribution is positive. Note that the restrictions can be binding even if they are not identifying, (e.g. when you impose restrictions on the adjustment coefficients but not on the cointegrating vector).

## Options for Restricted Estimation

Estimation of the restricted cointegrating vectors $\beta$ and adjustment coefficients $\alpha$ generally involves an iterative process. The VEC Restrictions tab provides iteration control for the maximum number of iterations and the convergence criterion. EViews estimates the restricted $\beta$ and $\alpha$ using the switching algorithm as described in Boswijk (1995). Each step
of the algorithm is guaranteed to increase the likelihood and the algorithm should eventually converge (though convergence may be to a local rather than a global optimum). You may need to increase the number of iterations in case you are having difficulty achieving convergence at the default settings.

Once you have filled the dialog, simply click $\mathbf{O K}$ to estimate the VEC. Estimation of a VEC model is carried out in two steps. In the first step, we estimate the cointegrating relations from the Johansen procedure as used in the cointegration test. We then construct the error correction terms from the estimated cointegrating relations and estimate a VAR in first differences including the error correction terms as regressors.

## Examples

Below, we demonstrate VEC estimation using the EViews example workfile "var1.WF1", located under the Chapter 44 - Vector Autoregression and Error Correction Models folder. This is a workfile with a number of classic macroeconomic variables including gross domestic product, various measure of money supply, treasury bills of different maturations, industrial production, producer price index, the unemployment.

## Example 1: Unrestricted Constant (JHJ)

We begin with the classical problem of studying the relationship between money supply (M1), gross domestic product (GDP), and 3-month Treasury bills (TB3).

These three endogenous variables will enter the VEC system with lags 1 through 4, and we assume that there exists a single cointegrating relationship. Furthermore, we will estimate the VEC using the default deterministic specification - Case 3 (JHJ): Unrest. constant. In this case, the constant is not restricted to the cointegrating relations, but is artificially inserted into the cointegrating vector using orthogonalization ("The Johansen, Hendry, and Juselius Approach" on page 368).

To estimate this model, select Vector Error Correction in the Method dropdown menu to display the VEC estimation dialog then enter "

```
m1 gdp tb3
```

in the Endogenous variables field on the Basics tab.
Furthermore, specify
14
in the Lag intervals for diff. endog field
and
in the Estimation sample edit field. We emphasize again that the lag interval specification refers to the differences of the dependent variable in the conditional error correction equation, and not the dependent variable itself in the levels equation.

You may leave the remaining fields and options at their default values. Hit OK to estimate the VEC with this specification. EViews will estimate the VEC and display the output in a table which contains four sections. Click on the Name button and enter VEC1.

At the top of the output, EViews shows a summary of the estimation procedure, including the sample, lag specification, variables, and deterministic assumptions used on constructing the estimates:

```
Vector Error Correction Estimates
Date: 05/30/22 Time: 11:25
Sample (adjusted): 1959M06 1982M03
Included observations: }274\mathrm{ after adjustments
Standard errors in () & t-statistics in []
Lags interval (in first differences): 1 to 4
Endogenous variables: M1 GDP TB3
Deterministic assumptions: Case 3 (Johansen-Hendry-Juselius):
    Constant belongs to short-run regressors and artificially to
    long-run regressors.
```

Next is a table of coefficient estimates for the cointegrating relation. In this case which is estimated assuming the default of one cointegrating vector, there is a single column of coefficients representing the only column of the cointegrating matrix. Since the deterministics are assumed to follow Johansen-Hendry-Juselius variant Case 3, the cointegrating relation includes an orthogonalized intercept estimate of -170.6729

| Cointegrating Eq: | CointEq1 |
| :---: | :---: |
| M1(-1) | 1.000000 |
|  |  |
| GDP(-1) | -0.007056 |
|  | $(0.01726)$ |
|  | $[-0.40893]$ |
| TB3(-1) | -10.01482 |
|  | $(3.03592)$ |
|  | $[-3.29877]$ |
| C | -170.6729 |

Notably, there is no standard estimate for the orthogonalized intercept estimate.
Next, EViews displays a table containing the coefficient estimates for the error correction regressions, with the results for each dependent variable appearing in columns.

| (var) Var: VEC1 Workfile: VAR1::Var1\} |  |  |  |  |  | $\square$ | 回 | $\times$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc | Object | Print | Name | Freeze | Estimate F | Forecast | Stat: |
| Vector Error Correction Estimates |  |  |  |  |  |  |  |  |
| Error Correction: |  |  | D(M1) |  | D(GDP) | D(TB3) |  | $\wedge$ |
| COINTEQ1 |  |  | $\begin{aligned} & \hline 0.012566 \\ & (0.00238) \\ & {[5.32547]} \end{aligned}$ |  |  | $\begin{array}{r} -0.003885 \\ (0.00101) \\ {[-3.82883]} \end{array}$ |  |  |
| D(M1(-1)) |  |  | $\begin{aligned} & 0.025351 \\ & (0.05988) \\ & {[0.42351]} \end{aligned}$ |  | $\begin{aligned} & 1.708220 \\ & (0.85934) \\ & {[2.58778]} \end{aligned}$ | $\begin{aligned} & 0.299034 \\ & (0.02574) \\ & {[11.6184]} \end{aligned}$ |  |  |
| D(M1(-2)) |  |  | $\begin{aligned} & 0.055289 \\ & (0.07255) \\ & {[0.76209]} \end{aligned}$ |  | $\begin{gathered} -1.489429 \\ (0.79912) \\ {[-1.88383]} \end{gathered}$ | $\begin{aligned} & 0.002297 \\ & (0.03119) \\ & {[0.07383]} \end{aligned}$ |  |  |
| D(M1(-3)) |  |  | $\begin{aligned} & 0.088959 \\ & (0.07436) \\ & {[1.16942]} \end{aligned}$ |  | $\begin{gathered} 0.837191 \\ (0.81907) \\ {[0.77794]} \end{gathered}$ | $\begin{aligned} & 0.074398 \\ & (0.03197) \\ & {[2.32888]} \end{aligned}$ |  |  |
| D(M1 (-4)) |  |  | $\begin{gathered} -0.315907 \\ (0.07314) \\ {[-4.31939]} \end{gathered}$ |  | 2.414398 <br> (0.80560) <br> [2.99703] | $\begin{gathered} 0.027708 \\ (0.03145) \\ {[0.88109]} \end{gathered}$ |  |  |
| $\mathrm{D}(\mathrm{GDP}(-1))$ |  |  | $\begin{aligned} & 0.025946 \\ & (0.00580) \\ & {[4.83585]} \end{aligned}$ |  | $-0.192725$ <br> $(0.08165)$ $[-3.12822]$ <br> [-3.12622] | $\begin{aligned} & 0.008172 \\ & (0.00241) \\ & {[3.39582]} \end{aligned}$ |  |  |
| $\mathrm{D}(\operatorname{GDP}(-2))$ |  |  | $\begin{aligned} & 0.002083 \\ & (0.00527) \\ & {[0.39533]} \end{aligned}$ |  | $\begin{array}{r} -0.321851 \\ (0.05804) \\ {[-5.54532]} \end{array}$ | $\begin{aligned} & 0.000396 \\ & (0.00227) \\ & {[0.17488]} \end{aligned}$ |  |  |
| $\mathrm{D}(\mathrm{GDP}(-3))$ |  |  | $\begin{gathered} 0.009181 \\ (0.00520) \\ {[1.76572]} \end{gathered}$ |  | $\begin{gathered} 0.480093 \\ (0.05727) \\ {[8.39338]} \end{gathered}$ | $\begin{array}{r} -0.000556 \\ (0.00224) \\ {[-0.24858]} \end{array}$ |  |  |
| $D(\operatorname{GDP}(-4))$ |  |  | $\begin{gathered} -0.023804 \\ (0.00558) \\ {[-4.28159]} \end{gathered}$ |  | $\begin{array}{r} -0.215175 \\ (0.06124) \\ {[-3.51364]} \end{array}$ | $\begin{array}{r} -0.008409 \\ (0.00239) \\ {[-3.51767]} \end{array}$ |  |  |
| D(TB3 (-1)) |  |  | $\begin{array}{r} -0.793881 \\ (0.15071) \\ {[-5.28762]} \end{array}$ |  | $\begin{aligned} & 2.988772 \\ & (1.68005) \\ & {[1.78716]} \end{aligned}$ | $\begin{aligned} & 0.208073 \\ & (0.08480) \\ & {[3.18004]} \end{aligned}$ |  |  |
|  | D(TB3 |  | $\begin{array}{r} -0.272277 \\ (0.16319) \\ {[-1.68842]} \end{array}$ |  | $\begin{array}{r} -0.131306 \\ (1.79757) \\ {[-0.07305]} \end{array}$ | $\begin{array}{r} -0.139815 \\ (0.07017) \\ {[-1.99251]} \end{array}$ |  |  |
|  | D(TB3 |  | $\begin{aligned} & 0.341199 \\ & (0.18438) \\ & {[2.07564]} \end{aligned}$ |  | $\begin{aligned} & 2.227258 \\ & (1.81088) \\ & {[1.23008]} \end{aligned}$ | $\begin{aligned} & 0.050358 \\ & (0.07088) \\ & {[0.71247]} \end{aligned}$ |  |  |
|  | D(TB3 |  | $\begin{gathered} -0.188183 \\ (0.12701) \\ {[-1.32419]} \end{gathered}$ |  | $\begin{aligned} & 1.816820 \\ & (1.39898) \\ & {[1.29853]} \end{aligned}$ | $\begin{gathered} 0.022086 \\ (0.05461) \\ {[0.40406]} \end{gathered}$ |  |  |
|  | c |  | $\begin{aligned} & 1.185120 \\ & (0.19534) \\ & {[5.98470]} \end{aligned}$ |  | $\begin{aligned} & 8.100411 \\ & (2.15161) \\ & {[3.76482]} \end{aligned}$ | $\begin{array}{r} -0.408057 \\ (0.08399) \\ {[-4.85838]} \end{array}$ | ) |  |

The long-run portion of these results, the adjustment coefficients $\alpha$, appear at the top of the table, as the estimated coefficient on COINTEQ1.

The remaining coefficients are estimates of the short-run-dynamics coefficients $\Gamma$. Note that for JHJ Case 3, the short-run results include estimates of the intercept, C. Since C is both inside and outside the cointegrating equation, keep in mind that the short-run estimate is obtained conditionally on the orthogonalized estimate of C in the cointegrating equation.

Just below the remainder of the short-run estimates including the estimate for C is the last part of the output showing summary statistics associated with the overall fit.


## Example 2: Unrestricted Constant, Restricted Trend

We modify the previous example to use only the first 2 lags, to have cointegration rank 2 , and assume that the constant is entirely unrestricted, but restricting the trend to the cointegrating relation and the intercept to the short-run equation. To proceed, copy the existing var object, click on the Estimate button to bring up the VAR estimation dialog again, and then change the Lag intervals for diff. endog to " 12 ":

then click on the Cointegration tab. select $\mathbf{2}$ as the Number of cointegrating relations, and change the Deterministic trend specification dropdown to 4 : Unrestricted constant and restricted trend.

| VAR Specification |
| :--- | :--- | :--- |
| BasicsCointegrationCointegrating rank <br> Number of cointegrating relations: <br> Deterministic trend specifications <br> 4: Unrestricted constant and restricted trend <br> Case 4: <br> Constant belongs to short-run regressors. Trend <br> belongs to long-run regressors. Associated VAR has a <br> trend. |

Click on OK to estimate the revised model, then press Name and enter VEC2. the top portion of the output is given by:

```
Vector Error Correction Estimates
Date: 05/30/22 Time: 11:58
Sample (adjusted): 1959M04 1982M03
Included observations: }276\mathrm{ after adjustments
Standard errors in ( ) & t-statistics in [ ]
Lags interval (in first differences): 1 to 2
Endogenous variables: M1 GDP TB3
Deterministic assumptions: Case 4: Constant belongs to short-run
    regressors. Trend belongs to long-run regressors.
```

| Cointegrating Eq: | CointEq1 | CointEq2 |
| :---: | :---: | :---: |
| M1(-1) | 1.000000 | 0.000000 |
|  |  |  |
| GDP(-1) | 0.000000 | 1.000000 |
|  |  |  |
| TB3(-1) | -16.48667 | -141.5224 |
|  | $(3.23023)$ | $(24.0537)$ |
|  | $[-5.10386]$ | $[-5.88360]$ |
|  | -0.112890 | 4.612925 |
| @TREND | $(0.14657)$ | $(1.09141)$ |
|  | $[-0.77022]$ | $[4.22657]$ |

Notice that there are now two cointegrating vectors, CointEq1 and CointEq2, which include a trend, with coefficient estimates -0.1129 and 4.6129 , respectively, and standard errors, but not a constant since the latter is in the short-run regressors.

The error correction results, which now include the two cointegrated series COINTEQ1 and COINTEQ2, and an intercept, and the summary statistics results are presented below:

| Var) Var: VEC1 Workfile: VAR1::Var1\} |  |  |  |  |  |  | 回 | x |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| View | Proc | Object | Print | Name | Freeze | Estimate | Forecast | Stats |
| Vector Error Correction Estimates |  |  |  |  |  |  |  |  |
| Error Correction: |  |  | D(M |  | D(GDP) | D(TB3) |  | $\wedge$ |
| COINTEQ1 |  |  | 0.004 $(0.004$ $[0.840$ | (02029 | $\begin{aligned} & 0.029883 \\ & (0.06242) \\ & {[0.47872]} \end{aligned}$ | 0.006143 (0.00192) [3.20710] |  |  |
| COINTEQ2 |  |  | 0.000 (0.000 $[2.230$ | 0839 | $\begin{aligned} & 0.016817 \\ & (0.00373) \\ & {[4.50745]} \end{aligned}$ | $\begin{array}{r} -0.000579 \\ (0.00011) \\ {[-5.08035]} \end{array}$ |  |  |
| D(M1(-1)) |  |  | $\begin{aligned} & 0.048 \\ & (0.082 \\ & {[0.776} \end{aligned}$ | (8532 | $\begin{aligned} & 1.890838 \\ & (0.81395) \\ & {[2.32278]} \end{aligned}$ | $\begin{gathered} 0.287028 \\ (0.02497) \\ {[11.4931]} \end{gathered}$ |  |  |
| D(M1(-2)) |  |  | 0.082 (0.07 $[1.10071]$ | 2385 | $\begin{gathered} -2.713300 \\ (0.97419) \\ {[-2.78520]} \end{gathered}$ | $\begin{gathered} -0.002511 \\ (0.02989) \\ {[-0.08400]} \end{gathered}$ |  |  |
| $\mathrm{D}(\operatorname{GDP}(-1))$ |  |  | $\begin{aligned} & 0.000 \\ & (0.00 \\ & {[0.044} \end{aligned}$ | $\begin{aligned} & 0177 \\ & 0398) \\ & \hline 481] \end{aligned}$ | $\begin{array}{r} -0.588879 \\ (0.05150) \\ {[-11.4304]} \end{array}$ | 0.003139 <br> (0.00158) <br> [1.98841] |  |  |
| $\mathrm{D}(\operatorname{GDP}(-2))$ |  |  | $\begin{aligned} & -0.001 \\ & (0.003 \\ & {[-0.44} \end{aligned}$ | [1745 | $\begin{aligned} & -0.585134 \\ & (0.05117) \\ & {[-11.0437]} \end{aligned}$ | 0.001836 <br> (0.00157) <br> [1.18953] |  |  |
| D(TB3(-1)) |  |  | $\begin{array}{r} -0.743 \\ (0.15] \\ {[-4.712} \end{array}$ | $\begin{aligned} & 3638 \\ & 5780) \\ & 1261] \end{aligned}$ | $\begin{aligned} & 7.137713 \\ & (2.05436) \\ & {[3.47442]} \end{aligned}$ | $\begin{aligned} & 0.240181 \\ & (0.08303) \\ & {[3.81047]} \end{aligned}$ |  |  |
| $\mathrm{D}(\mathrm{TB3}(-2))$ |  |  | $\begin{array}{r} -0.181 \\ (0.12 \\ {[-1.41} \end{array}$ | $\begin{aligned} & 1088 \\ & 2815) \\ & 13121 \end{aligned}$ | $\begin{aligned} & 5.628032 \\ & (1.68835) \\ & {[3.37342]} \end{aligned}$ | $\begin{array}{r} -0.025843 \\ (0.05119) \\ {[-0.50096]} \end{array}$ |  |  |
| c |  |  | $\begin{aligned} & -0.220 \\ & (0.33 \\ & (-0.658 \end{aligned}$ | $\begin{aligned} & 0388 \\ & 3467) \\ & 5851] \end{aligned}$ | $\begin{gathered} -1.817315 \\ (4.35711) \\ {[-0.41709]} \end{gathered}$ | $\begin{array}{r} -0.455287 \\ (0.13389) \\ {[-3.40552]} \end{array}$ |  |  |
| R-squared |  |  | 0.383 | 3438 | 0.478575 | 0.477784 |  |  |
| Adj. R-squared |  |  | 0.384 | 4982 | 0.462952 | 0.462137 |  |  |
| Sum sq. resids |  |  | 359.0 | 0179 | 60851.05 | 57.28440 |  |  |
| S.E. equation |  |  | 1.159 | 9584 | 15.09857 | 0.483194 |  |  |
| F-statistic |  |  | 20.75 | 6560 | 30.63230 | 30.53535 |  |  |
| Log likelihood |  |  | -427.917 | 9171 | -1138.245 | -174.6396 |  |  |
| Akaike AIC |  |  | 3.160 | 8086 | 8.298878 | 1.330722 |  |  |
| Schwarz SC |  |  | 3.284 | 4122 | 8.416935 | 1.448779 |  |  |
| Mean dependent |  |  | 1.096 | 8739 | 9.454347 | 0.034931 |  |  |
| S. D. dependent |  |  | 1.455 | 5132 | 20.60022 | 0.631578 |  |  |
| Determinant resid covariance (dof adj.) |  |  |  |  | 59.71231 |  |  |  |
| Determinant resid covariance |  |  |  |  | 54.05930 |  |  |  |
| Log likelihood |  |  |  |  | -1725.512 |  |  |  |
| Akaike | inform | ation criter |  |  | 12.75734 |  |  |  |
| Schwarz criterion |  |  |  |  | 13.21644 |  |  |  |
| Number of coefficients |  |  |  |  | 35 |  |  |  |

## Example 3: Unrestricted Constant, Restricted Trend, and Exogenous

Extending the previous model, let us augment the cointegrating relation and the short-run dynamics by including exogenous variables. These exogenous variables can enter the cointegrating relation, so that they affect the long-run relationship, and they can be in the short-run relationship where they affect the dynamics of convergence to equilibrium.

Let us assume that the 10-year Treasury bill rate (TB10Y) is an exogenous variable inside the cointegrating relation but not a part of the short-run dynamics, that the Producer Price Index (PPI), a measure of inflation, impacts only the short-run dynamics to convergence, and that the unemployment rate (UNRATE) is in both the short and long-run relationships.

Copy the existing var then click on Estimate to modify the specification. We enter "PPI" in the Short-run (outside cointegrating equation) field, "TB10Y" in the Long-run (inside cointegrating equation) field, and "URATE" in the Both long-run and short-run field:


Furthermore, we'll assume there's a single cointegrating relation, and that the deterministic case specifies a constant and trend only affect the adjustment to equilibrium dynamics (short-run).

Click on the Cointegration tab and change the Number of cointegrating relations dropdown to 1, and set the Deterministic trend specification dropdown to 5: (JHJ) Unrest. constant and trend:


Click on OK to estimate the updated specification.


Notice that in addition to a description of the deterministic trend assumption, the output header now lists the exogenous variables included in the specification, by type.

Below the header, the results for the cointegrating vector show the three endogenous variables, followed by the coefficient for the long-run only variable TB10Y, and the both long and short-run variable URATE. Since the latter is included in the cointegrating equation via orthogonalization, is no standard error associated with the estimated coefficient.

The error correction results include estimates for the two short-run only deterministic trend variables, C and @TREND, along with the short-run only PPI, and the both long and shortrun URATE. As with other both long and short-run variables, the coefficient of URATE is estimated conditionally on the orthogonalization.


## Example 4: VEC Restrictions

We may continue with the previous example after imposing restrictions on elements of the $\beta$ matrix.

Once again, copy the existing var object, click on the Estimate button to bring up the VAR estimation dialog. Leave the existing specification in place, including the exogenous variables, but click on the VEC Restrictions tab to display the restrictions settings dialog. Click
on the Impose Restrictions to enable the restrictions and enter " $\mathrm{B}(1,1)=1, \mathrm{~B}(1,2)=0.25$, $B(1,3)=0.5$ " in the edit field:


This specification restricts the first three elements of the cointegrating vector to the specified values. Click on OK to estimate the restricted VEC.

Vector Error Correction Estimates
Date: 05/30/22 Time: 13:22
Sample (adjusted): 1959M04 1982M03
Included observations: 276 after adjustments
Standard errors in ( ) \& t-statistics in [ ]
Lags interval (in first differences): 1 to 2
Endogenous variables: M1 GDP TB3
Exogenous variables (short-run only): PPI
Exogenous variables (long-run only): TB10Y
Exogenous variables (short-run and long-run): URATE
Deterministic assumptions: Case 5: Constant and trend both belong to short-run regressors.

Cointegrating restrictions:

$$
b(1,1)=1
$$

$b(1,2)=0.25$
$b(1,3)=0.5$
Convergence achieved after 423 iterations.
Restrictions identify all cointegrating vectors.
LR test for binding restrictions (rank $=1$ ):

| Chi-square(7) | 68.76786 |
| :--- | :--- |
| Probability | 0.000000 |

The familiar heading information is augmented to show the cointegrating restrictions, information about estimation and convergence, an analysis of whether the restrictions are identifying, and the results for a LR test for those restrictions that are binding.

The reported estimates of the cointegrating relation show both the restricted and unrestricted coefficient values:

| Cointegrating Eq: | CointEq1 |
| :---: | :--- |
| M1(-1) | 1.000000 |
| GDP(-1) | 0.250000 |
| TB3(-1) | 0.500000 |
| TB10Y | 94.02222 |
|  | $(18.4081)$ |
|  | $[5.10766]$ |
| URATE | -209.8535 |

Note that the elements of the cointegrating value reflect the restrictions imposed in estimation, and that there are no standard errors for the restricted values.

The form of the remaining output (not shown), which consists of the error correction regression results and summary statistics is unchanged from unrestricted estimation, with the exception of the number of coefficients.

## Example 5: Cointegration Testing

We may carry out a Johansen cointegration test to determine the rank that should be used in estimation of the VEC in Example 3. This test is available from a group window using View/ Cointegration Test/Johansen System Cointegration Test..., or from an estimated VAR object window using Views/Cointegration Test... In the latter case, the test dialog will be pre-filled with the cointegration specification, if applicable:


For the VEC in Example 3, we can keep everything at the default values and click on OK to perform the test. The output is a spool object with 4 tables:


The first table is a Summary of the estimation specifications under which the cointegration rank test is conducted.

Following the summary is a Rank Tests table showing Johansen cointegration tests based on the trace and maximum eigenvalue statistics. The trace statistic reported in the first block tests the null hypothesis of at most $r$ cointegrating relations against the alternative of $K$ cointegrating relations, where $K$ is the number of endogenous variables. The maximum eigenvalue statistics tests the null hypothesis of $r$ cointegrating relations against the alternative of $r+1$ cointegrating relations.

There are a few other details to keep in mind:

1. Simulated critical values are available for up to $K=10$ series. Also note that the critical values depend on the trend assumptions, and may not be appropriate for models that contain other deterministic regressors.
2. The trace statistic and the maximum eigenvalue statistic may yield conflicting results. For such cases, we recommend that you examine the estimated cointegrating vector
and base your choice on the interpretability of the cointegrating relations; see Johansen and Juselius (1990) for an example.
3. In some cases, the individual unit root tests will show that some of the series are integrated, but the cointegration test will indicate that the matrix $\Pi$ has full rank ( $r=K$ ). This apparent contradiction may be the result of low power of the cointegration tests, stemming perhaps from a small sample size, or it may serve as an indicator of specification error.

The next node contains the Unrestricted Coefficients table which provides estimates of the cointegrating vector $\beta$ and the adjustment parameters $\alpha$. As is well known, the cointegrating vector $\beta$ is not identified unless we provide an arbitrary normalization. This table reports estimates of $\beta$ and $\alpha$ under the normalization $\beta^{\prime} S_{11} \beta=I_{K}$ where $S_{11}$ is defined in Equation (0.51). Note that the transpose of $\beta$ is reported in the table so that the first row is the first cointegrating vector, the second row is the second cointegrating vector, and so on.

The final node contains the Normalized Coefficients table which reports estimates from a different normalization for each possible number of cointegrating relations. This alternative normalization expresses the first $r$ variables as functions of the remaining $K-r$ variables in the system. Asymptotic standard errors are reported in parentheses for the parameters that are identified.


## Example 6: Rank Tests

It is possible to conduct a battery of rank tests for each of the deterministic assumptions that are available for estimation. To do this, proceed from the previous example and click on Views/Cointegration Test... to once again bring up the testing dialog.


This time, click on Summarize all deterministic case assumptions and hit OK.
The top portion of the spool output shows the test settings and assumptions, along with the rank selection results broken down by test type and deterministic case:


The remainder of the output shows the information criteria broken down by rank and deterministic case:


## References

Hendry, David F, and Katarina Juselius (2001). "Explaining Cointegration Analysis: Part II." The Energy Journal, 22(1).
Johansen, Søren (1995). Likelihood-based Inference in Cointegrated Vector Autoregressive Models. Oxford University Press on Demand.
Juselius, Katarina (2006). The Cointegrated VAR Model: Methodology and Applications. Oxford University Press.
Lütkepohl, Helmut (2005). New Introduction to Multiple Time Series Analysis. Springer Science \& Business Media.
MacKinnon, James G, Alfred A. Haug, and Leo Michelis (1999). "Numerical distribution functions of likelihood ratio tests for cointegration." Journal of Applied Econometrics, 14(5): 563-577.

## Difference-in-Difference Estimation

Difference-in-difference (DiD) estimation is a popular method of causal inference that allows estimation of the average impact of a treatment on individuals.

Although DiD can be estimated on repeated cross-sectional data, most research in econometrics has concentrated on estimation of DiD models in a panel data setting, and EViews offers built in tools for estimating DiD models only within panel workfiles.

This chapter will introduce estimation of the DiD model using the common two-way fixedeffects (TWFE) method, as well as post-estimation diagnostics of the TWFE model, such as those by Goodman-Bacon (2021), Callaway and Sant'Anna (2021), and Borusyak, Jaravel, and Spiess (2021).

## Background

Difference-in-difference estimation is a method of analyzing the impact of a treatment or policy on an outcome by comparing the difference in the outcome variable before and after treatment for those who participated in the treatment, with those who did not participate in the treatment.

## Single Treatment Date

To begin, we shall suppose that there is a single treatment date, $D^{*}$, on which some of the individuals in our study are given the treatment. The remainder of the individuals never receive treatment. We denote the treated individuals as belonging to group $T$, and those not treated as belonging to group $N T$. Along with data on whether an individual is treated, we also have time series data on an outcome variable, $Y$, with periods both before and after the treatment date, for all individuals.

To estimate the average impact of the treatment on $Y$, we compare the difference in the mean of $Y$ between the treated and never treated groups before and after the treatment date:

$$
\begin{equation*}
A T E T=\left(\hat{Y}_{T, t \geq D^{*}}-\hat{Y}_{N T, t \geq D^{*}}\right)-\left(\hat{Y}_{T, t<D^{*}}-\hat{Y}_{N T, t<D^{*}}\right) \tag{0.77}
\end{equation*}
$$

which we term the Average Treatment Effect for the Treated (ATET).
The ATET may be visualized as:


The calculation of the ATET may be embedded in an OLS regression structure, using the model

$$
\begin{equation*}
Y=\beta_{0}+\beta_{1} D+\beta_{2} T+\beta_{3}(D \cdot T)+\epsilon \tag{0.78}
\end{equation*}
$$

where $D$ is a dummy variable equal to 1 if the observation lies after $D^{*}$, and $T$ is a dummy variable equal to 1 if the observation is in the treated group. For notational simplicity, we ignore additional exogenous regressors in Equation (0.78), though note that their inclusion does not change the behavior of the estimator.

It is easy to see that the ATET is equal to the value of $\beta_{3}$, which measures the marginal effect of being in the treatment group after $D^{*}$.

Alternately, the ATET may be estimated using a fixed effects model with fixed effects in both the group and time dimensions, and a dummy variable corresponding to whether the observations is treated (i.e., in the treated group and post-treatment date),

$$
\begin{equation*}
Y_{i, g, t}=a_{g}+\gamma_{t}+\delta D_{g, t}+\epsilon_{i, g, t} \tag{0.79}
\end{equation*}
$$

where $g$ identifies the treatment $(g=1)$ and the non-treatment $(g=0)$ groups, and $D_{g, t}$ takes a value of 1 whenever a treatment group observation is actually treated. In this specification, the ATET is equal to the value of $\delta$.

Further, the estimate $\delta$ in Equation ( 0.79 ) is numerically identical to the result obtained from the panel two-way fixed effects (TWFE) regression:

$$
\begin{equation*}
Y_{i, g, t}=a_{i}+\gamma_{t}+\delta D_{g, t}+\epsilon_{i, g, t} \tag{0.80}
\end{equation*}
$$

since the group identifier $g$ is constant for an individual $i$. In the latter specification, the group fixed effects have been replaced by individual cross-section effects.

Since the TWFE is estimated using ordinary least squares regression, all assumptions required for an OLS model are also required for DiD estimation by TWFE.

Crucially, for estimates of $\delta$ to be unbiased for the ATET, DiD requires an additional parallel trends assumption. The parallel trends assumption requires that in the counter-factual where individuals in the treated group are not treated, the difference between the treatment group and the control group is constant through time. Intuitively, when comparing a treatment group to a never-treated or always-treated control group, unbiased estimation requires that the change in outcomes for the control group before and after the treatment date is a good measure of the (unobserved) untreated change in the treatment group.

## Multiple Treatment Dates

The model given by Equation (0.77) assumes a single treatment date, $D^{*}$, with all individuals who will be treated, receiving treatment on the same date.

A natural extension to the single treatment model is to assume that different groups of individuals have different treatment dates:


Here we have 3 groups of individuals, some are never treated $N T$, some are treated at the earlier date $D 1(T E)$, and some are treated at a later date $D 2(T L)$.

In such cases we are still interested in the average treatment effect for the treated (ATET), where the treatment effect is calculated as the effect on the trend post-treatment, whenever that treatment occurs.

The natural estimator for the ATET is this case is to simply extend the TWFE model Equation ( 0.80 ) to incorporate more than 2 groups:

$$
\begin{equation*}
Y_{i, g, t}=a_{i}+\gamma_{t}+\delta D_{g, t}+\epsilon_{i, g, t} \tag{0.81}
\end{equation*}
$$

where $g$ now indexes the multiple treatment date groups. Note that the presence of multiple treatment dates does not affect the form of the TWFE estimator.

## Estimating DiD in EViews

The panel data TWFE model may be estimated using EViews' built-in estimator for least squares regression with fixed effects. However, EViews also offers a simple interface for estimating the panel TWFE model, featuring tools for performing post-estimation diagnostics such as parallel trends tests, the Goodman-Bacon decomposition and the CallawaySant'Anna estimation diagnostic.
(To estimate DiD models on non-panel workfiles, you will need to specify the appropriate least squares model manually using standard least squares regression.)

To estimate a DiD model in EViews, bring up the equation dialog by clicking on Object/New Object.../Equation or Quick/Estimate Equation... from the main menu bar in your panel workfile. EViews will detect the presence of your panel structure and in place of the standard equation dialog will open the panel equation estimation dialog. Select DiD - Differ-ence-in-Difference in the Method dropdown display the DiD dialog:


In the Equation specification edit field you should enter the dependent variable followed by any exogenous regressors apart from the treatment variable.

The treatment variable should be entered in the Treatment Variable edit field. The treatment series should be a binary variable indicating whether the individual has been treated (i.e., is 1 if the observation in a treatment group which is post-treatment date for that group, and 0 otherwise).

The Options tab contains a single Coefficient name edit field that allows you to change the default coefficient vector.

Click on $\mathbf{O K}$ to perform the difference-in-difference estimation and display the output:

Dependent Variable: L_HOMICIDE
Method: Difference-in-Difference
Date: 05/13/22 Time: 10:56
Periods included: 11
Cross-sections included: 50
Total panel (balanced) observations: 550

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | ---: |
| POST | 0.081812 | 0.064117 | 1.275980 | 0.2026 |
| R-squared | 0.910576 | Mean dependent var | 1.405760 |  |
| Adjusted R-squared | 0.899604 | S.D. dependent var | 0.590154 |  |
| S.E. of regression | 0.186992 | Akaike info criterion | -0.411237 |  |
| Sum squared resid | 17.09842 | Schwarz criterion | 0.066772 |  |
| Log likelihood | 174.0902 | Hannan-Quinn criter. | -0.224439 |  |
| F-statistic | 82.98904 | Durbin-Watson stat | 1.469473 |  |
| Prob(F-statistic) | 0.000000 | Parallel trend stat | 0.820393 |  |
| Prob(P. trend) | 0.411992 |  |  |  |

The basic equation output for a DiD estimation is identical to that of least squares estimation, but with the display of only the single treatment coefficient. Also, the test statistic and associated $p$-value of a test of the parallel trends assumption is displayed with the summary statistics at the bottom of the estimation. This test is a simple Wald-test run on the auxiliary regression:

$$
\begin{equation*}
Y_{i, g, t}=a_{i}+\gamma_{t}+\delta D_{i, g, t}+\theta \operatorname{Tr} \cdot D_{i, g, t}+\epsilon_{i, g, t} \tag{0.82}
\end{equation*}
$$

where $\operatorname{Tr}$ is a trend term, and the test evaluates $H_{0}: \theta=0$.
It should also be noted that when performing difference-in-difference estimation, EViews uses cross-section cluster-robust standard errors.

## Post-Estimation Views and Procs

Since the TWFE model of DiD is a simple regression model, the standard views and procs available for least squares models are also available for DiD estimation.

There are a set of DiD specific views available under the Difference-in-Difference Diagnostics view menu entry.

## Trends Summary

The Trends Summary Table and Trends Summary Graph views display the average of the outcome variable by year, categorized by treatment group (i.e., the date at which treatment occurs).

Trends Summary
Means of L_HOMICIDE
Date: 05/13/22 Time: 10:58
Sample: 20002010
Included observations: 550

| Mean |  | 2005 | 2006 | 2007 | 2008 | 2009 | Never | All |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 2000 | 1.756288 | 1.786845 | 1.307029 | 1.143948 | 0.823902 | 1.238060 | 1.384578 |
|  | 2001 | 1.697463 | 1.789791 | 1.483962 | 1.114078 | 1.352019 | 1.238571 | 1.407987 |
|  | 2002 | 1.719649 | 1.755136 | 1.353935 | 1.365008 | 0.592638 | 1.243661 | 1.386819 |
|  | 2003 | 1.715632 | 1.806743 | 1.467548 | 1.485847 | 1.212344 | 1.253548 | 1.432208 |
|  | 2004 | 1.717022 | 1.802504 | 1.443218 | 1.428943 | 1.201863 | 1.254353 | 1.427168 |
|  | 2005 | 1.625275 | 1.775398 | 1.622460 | 1.598881 | 0.681382 | 1.282884 | 1.445561 |
|  | 2006 | 1.831149 | 1.869994 | 1.447267 | 1.526418 | 1.280735 | 1.269485 | 1.461576 |
|  | 2007 | 1.897956 | 1.911203 | 1.581592 | 1.411828 | 0.887561 | 1.258404 | 1.465498 |
|  | 2008 | 1.860869 | 1.804862 | 1.363984 | 1.438825 | 1.238401 | 1.248592 | 1.422104 |
|  | 2009 | 1.719375 | 1.736801 | 1.564256 | 1.527684 | 1.207879 | 1.115439 | 1.343316 |
|  | 2010 | 1.664930 | 1.671675 | 1.427658 | 1.314476 | 0.971882 | 1.090319 | 1.286549 |
|  | All | 1.745964 | 1.791905 | 1.460264 | 1.395994 | 1.040964 | 1.226665 | 1.405760 |

This trends summary graph offers a quick visual representation of the means by treatment group to check whether the different groups have similar trends.


## Goodman-Bacon Decomposition

Recent research has noted that the TWFE model is not suitable in multiple-timing DiD models if the impact of treatment changes as time from treatment increases. In this case, the

TWFE estimator will exhibit bias since using observations from an already-treated group as the comparison violates the parallel trends assumption.

The Goodman-Bacon Decomposition view calculates the treatment effects for individual pairs of treatment groups. By focusing on results for different categories of comparisons, the Goodman-Bacon decomposition looks for the presence of parallel trends bias in the computation of the multiple-timing TWFE.

Goodman-Bacon (2021) shows that the TWFE estimator is a weighted average of DiD estimators for every combination of two groups of observations defined by two treatment dates ( $2 \times 2$ comparisons). The weights used in the average are based upon the number of observations and the variance of the estimated treatment effect in these individual DiD comparisons.

Specifically, for every treatment group with a given treatment date (group $T$ ), we compute all $2 \times 2$ comparisons with:

- observations treated at later dates ( $L$ groups)
- observations treated at earlier dates ( $E$ groups)
- never treated observations (group $N$ )
- always treated observations (group $A$ )

For the first set of cases where $T$ is an early treatment group $(E)$ compared with a later group ( $L$ ), we employ only untreated $L$ observations from periods up to the beginning of the treatment date, so that the $L$ can serve as a non-changing control.

For the second set of cases where $T$ is a later treatment group ( $L$ ) compared with an earlier group ( $E$ ), we employ only treated $E$ observations from periods following the treatment date, so that the $E$ can serve as a non-changing control.

It is important to note that both the $L$ against $E$, and the $T$ against $A$ comparisons employ control groups that consist of previously treated observations. These comparisons are particularly susceptible to parallel trends bias when the treatment effect varies from the date of treatment.

To display the decomposition, click on View/Difference-in-Difference Diagnostics/Good-man-Bacon decomposition. EViews will display a spool view of output divided into three sections.

The first section shows weighted means of the treatment effect categorized by the type of $2 \times 2$ group comparison:

Goodman-Bacon Decomposition Dependent Variable: ASMRS Treatment Variable: POST Date: 05/11/22 Time: 12:27 Sample: 19641996 Included observations: 1617

| Component | Cases | Mean Coef. | Weight |
| :--- | :---: | ---: | :---: |
| Earlier Vs Later | 66 | -0.186770 | 0.110654 |
| Later Vs Earlier | 66 | 3.511967 | 0.264644 |
| Treated Vs Never | 12 | -5.330907 | 0.240270 |
| Treated Vs Always | 12 | -7.043689 | 0.384432 |

The Cases column shows the number of year pairs used in the comparison. In this example, there are 66 treatment date pairs in which $E$ and untreated $L$ observations are compared, and 12 years in which both the $E$ and $L$ observations are compared with the $N$ observations.

The Mean Coef column the estimate of the ATET, while the Weight column contains the weights used in forming the overall ATET estimate. Here the estimated ATET across $E$ vs. untreated $L$ comparisons is -0.1868 with a weight of 0.1107 . The estimated ATET for the $E$ and $L$ observations vs. $N$ observations is -7.0437, with a weight of 0.3844 .

Each of the ATET components in this table is itself a weighted average of individual treat-ment-year pair comparisons. The second section of the spool displays the individual $2 \times 2$ coefficient estimates and weights for every comparison. For example, the first few lines of a Goodman-Bacon decomposition are given by:

| Component | Coef. | Weight |
| :--- | ---: | :---: |
| 1969 Vs 1970 | 3.089865 | 0.000126 |
| 1970 Vs 1969 | -23.83421 | 0.000680 |
| 1969 Vs 1971 | 0.728642 | 0.000882 |
| 1971 Vs 1969 | -8.258332 | 0.004586 |
| 1969 Vs 1972 | 1.676302 | 0.000567 |
| 1972 Vs 1969 | -7.148349 | 0.002835 |
| 1969 Vs 1973 | 0.672705 | 0.002520 |
| 1973 Vs 1969 | -7.433827 | 0.012095 |
| 1969 Vs 1974 | 0.148880 | 0.000945 |
| 1974 Vs 1969 | -5.158778 | 0.004347 |
| 1969 Vs 1975 | 9.615416 | 0.000756 |
| 1975 Vs 1969 | 0.668246 | 0.003326 |
| 1969 Vs 1976 | 3.482368 | 0.000441 |
| 1976 Vs 1969 | 4.239865 | 0.001852 |
| 1969 Vs 1977 | -3.567539 | 0.001512 |
| 1977 Vs 1969 | -13.05293 | 0.006048 |

The labeling of the individual components indicates the nature of the group comparison. For example, the " 1969 vs. 1970 " comparison is an $E$ against $L$ comparison, while the " 1970 vs. 1969 " is the $L$ against $E$ comparison.

The final section in the spool contains a graph of the individual $2 \times 2$ component results, with different symbols and colors indicating the type of comparison. The graphical representation of the individual effects shown in the third section allows a quick visualization of the effects:


## Group-Time Effects (Callaway-Sant'Anna)

The Group-Time Average Treatment Effects (Callaway-Sant’Anna) view computes the Callaway and Sant'Anna (2021) estimator of the average treatment effects.

A study by Callaway and Sant'Anna (2020, CS) derives a new estimator for the DiD model with multiple treatment time periods, breaking away from the TWFE estimator by using weighted averages of the differences model Equation (0.77) with appropriate comparison pairings.

Their estimator is robust to the treatment effect changing as time from treatment increases so that the CS estimator can be useful as a diagnostic comparison to the TWFE estimator to judge the reliability of the TWFE estimates.

To display the view select View/Difference-in-Difference Diagnostics/CS Group-Time
Effects... from the equation menu. EViews displays a dialog:

```
Callaway & Sant'Anna Group-Time Effects X
    Additional regressors
    (leave blank to use original equation regressors only)
    |
    Comparison group
    O}\mathrm{ Never treated
    ONot yet treated
    Both
```



- The Additional regressors edit field allows you to estimate the Callaway and Sant'Anna model on the underlying estimated TWFE model, but with additional regressors. This allows you to estimate models via Callaway and Sant'Anna that may be impossible to estimate via TWFE due to perfect collinearity. Simply enter the name of series, or series expressions, you wish to add to the estimation in the edit field.

By default the CS estimator compares treated individuals grouped by treatment date against a control group of individuals who never receive treatment. However these comparisons could be modified to include individuals who have not yet been treated as the control group.

Each pairing compares groups the treated into year-of-treatment groups, and then compares the difference between the output variable in each year after treatment with the year prior to treatment. This difference is compared for the treatment group and the comparison group.

- The Comparison group radio buttons specify whether the comparison/control groups in the Callaway-Sant'Anna model will be only those individuals who never receive treatment (Never treated), or those who have yet to receive treatment in the current time period (Not yet treated), or either (Both).

Clicking OK will estimate the Callaway and Sant'Anna model. The output is in the form of a spool with five sections.

The first section provides a summary view of the CS results, including the overall average treatment effect of the treated (calculated as the weighted average of the individual pairing calculations), along with its standard-error, $z$-statistic and associated $p$-value.

Group-Time Average Treatment Effects
Dependent Variable: LEMP
Treatment Variable: TREATED
Date: 05/13/22 Time: 12:21
Sample: 20032007
Included observations: 2500

| Overall | Coefficient | Std. Error | z-Statistic | Prob. |
| :---: | :---: | :---: | :---: | :---: |
| TREATED | -0.041969 | 0.011445 | -3.667037 | 0.0002 |

The second section provides each of the individual pairings used in the weighted average ATET estimation, including the estimated coefficient for the pairing, and its weight.

Individual Effects

| Date | Treatment | Comparison | Coef. | Cases | Weight |
| :---: | :---: | :---: | ---: | :---: | :---: |
| 2004 | 2004 | 2003 | -0.014911 | 20 | 0.040000 |
| 2005 | 2004 | 2003 | -0.076996 | 20 | 0.040000 |
| 2006 | 2004 | 2003 | -0.141080 | 20 | 0.040000 |
| 2007 | 2004 | 2003 | -0.107544 | 20 | 0.040000 |
| 2004 | 2006 | 2003 | -0.002066 | 40 | 0.000000 |
| 2005 | 2006 | 2004 | -0.006968 | 40 | 0.000000 |
| 2006 | 2006 | 2005 | 0.000766 | 40 | 0.080000 |
| 2007 | 2006 | 2005 | -0.041536 | 40 | 0.080000 |
| 2004 | 2007 | 2003 | 0.026366 | 131 | 0.000000 |
| 2005 | 2007 | 2004 | -0.004760 | 131 | 0.000000 |
| 2006 | 2007 | 2005 | -0.028502 | 131 | 0.000000 |
| 2007 | 2007 | 2006 | -0.028789 | 131 | 0.262000 |

The remaining sections computes aggregations of the individual pairings, where the aggregations are over treatment date (group), observation date (date), and time-since treatment (duration). For each aggregation, the coefficients, standard errors, z-Statistics and associated p-values are tabulated, along with a graph of the coefficients and a $95 \%$ confidence interval.

For example, the group effects consist of the weighted averages of the ATET coefficients

Group Effects

| Group | Coefficient | Std. Error | z-Statistic | Prob. |
| :---: | :---: | :---: | :---: | :---: |
| 2004 | -0.085133 | 0.024251 | -3.510462 | 0.0004 |
| 2006 | -0.020385 | 0.017403 | -1.171386 | 0.2414 |
| 2007 | -0.028789 | 0.016168 | -1.780661 | 0.0750 |

and the corresponding plot

shows the values along with $95 \%$ confidence intervals
Borusyak, Jaravel, and Spiess Imputed Estimator
A third study by Borusyak, Jaravel and Spiess (2021, BJS) describes a number of potential issues with the TWFE estimator and derives an estimator that alleviates these issues. They term their estimator an "imputation" estimator, since it uses estimates from untreated observations to forecast the outcome variable for observations for which individuals have been treated, and then for each treated observation they impute the impact of treatment as the difference between the forecast of the outcome, and the observed outcome. This series of differences can then be averaged over to compute the average treatment effect.

To display the view select View/Difference-in-Difference Diagnostics/BJS Imputed Estimator from the equation menu.

EViews estimates the BJS estimator and displays the output in a spool. As with the GroupTime Effects view ("Group-Time Effects (Callaway-Sant'Anna)" on page 407), there are five sections in the output: the summary view with the overall ATET, its standard-error, $z$-statistic and associated $p$-value, the individual pairings used in the weighted average ATET estimation, and sections for aggregations over treatment date (group), observation date (date), and time-since treatment (duration).

## Make Underlying Equation Proc

One additional DiD-specific tool is the Make Underlying Equation proc. Clicking on Proc/ Make Underlying Equation will produce a new equation object in which the difference-indifference model as a standard least squares equation. Having an equation with the DiD specification estimated by least squares may prove useful for performing diagnostics or additional analysis.

## Examples

To demonstrate the use of DiD estimation in EViews we will replicate results from familiar papers.

## Card and Krueger (1994)

We first replicate results from one of the most famous DiD studies, the Card and Krueger (1994) paper analyzing the impact of minimum wage laws on employment in fast food workers.

Although Card and Krueger's study is involves difference-in-difference estimation, they only have two time periods: February/March 1992 and November/December 1992. They term these two periods Wave 1 and Wave 2. During the gap between these two periods, New Jersey increased the state minimum wage from $\$ 4.25$ to $\$ 5.05$. Pennsylvania, a neighboring state, maintained its minimum wage at $\$ 4.25$ during the same time period.

A subset of the data used in this paper are available in the workfile "CardKrueger.wf1". The data in the workfile contain data for 410 fast-food restaurants, 331 in New Jersey and 79 in Pennsylvania. There are three series containing employment data: EMPFT has the number of full time employees in each store, EMPTOT has the total number of employees, and EMPTOTC has the total number of employees, with stores that have closed down coded as a 0 (instead of an NA). The series HRSOPEN contains the number of hours, per day, each store is open. The NEWMIN series is a binary variable indicating whether a store is subject to an increased in the minimum wage (i.e. is based in New Jersey during wave 2).

We can replicate one of the results in the Card and Krueger paper by performing a simple difference-in-difference estimation of EMPTOT against NEWMIN. We click on Quick/Estimate Equation and then select DiD - Difference-in-Difference from the Method dropdown to display the dialog:


Enter "EMPTOT C" in the Equation specification edit field, the name of the treatment indicator "NEWMIN" in the Treatment variable edit field, and "12" in the Sample edit field. Click on OK to estimate this specification. EViews will display the estimation results:

Dependent Variable: EMPTOT
Method: Difference in Difference
Date: 05/10/22 Time: 11:10
Periods included: 2
Cross-sections included: 410
Total panel (unbalanced) observations: 794

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | :--- |
| NEWMIN | 2.750000 | 2.718648 | 1.011532 | 0.3124 |
| R-squared | 0.781841 | Mean dependent var | 21.02651 |  |
| Adjusted R-squared | 0.547121 | S.D. dependent var | 9.422746 |  |
| S.E. of regression | 6.341161 | Akaike info criterion | 6.838121 |  |
| Sum squared resid | 15360.34 | Schwarz criterion | 9.265021 |  |
| Log likelihood | -2302.734 | Hannan-Quinn criter. | 7.770758 |  |
| F-statistic | 3.330944 | Durbin-Watson stat | 4.124675 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

The estimate of the impact of the increase in the minimum wage on employment is 2.75 . This matches the result in Row 4., column (iii) of Table 3 in Card and Krueger, and forms part of the basis for their overall finding that raising the minimum wage in New Jersey actually led to an increase in employment.

As noted earlier, the standard error of the estimate and the $t$-statistic $p$-value employ crosssection cluster-robust standard error calculations.

Note the test statistic and $p$-value for the parallel trends test are not displayed at the bottom of the table, since with only two time periods, an estimate of trend values cannot be computed.

If we use EMPTOTC as the outcome variable instead of EMPTOT, we can replicate the result in Row 5, column (iii) of Table 3 in Card and Krueger, which reports that the impact of the raise in minimum wage is 2.51 .

| Dependent Variable: EMPTOTC |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Method: Difference in Difference |  |  |  |  |
| Date: 05/10/22 Time: 10:37 |  |  |  |  |
| Periods included: 2 |  |  |  |  |
| Cross-sections included: 410 |  |  |  |  |
| Total panel (unbalanced) observations: 798 |  |  |  |  |
| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| NEWMIN | 2.509212 | 2.718822 | 0.922904 | 0.3566 |
| R-squared | 0.776508 | Mean depend | nt var | 20.92112 |
| Adjusted R-squared | 0.538540 | S.D. depend | t var | 9.515792 |
| S.E. of regression | 6.464156 | Akaike info crit | erion | 6.876732 |
| Sum squared resid | 16129.13 | Schwarz crite |  | 9.294061 |
| Log likelihood | -2331.816 | Hannan-Quin | criter. | 7.805471 |
| F-statistic | 3.263085 | Durbin-Wats | stat | 4.102828 |
| Prob(F-statistic) | 0.000000 |  |  |  |

We could extend the specification to include the number of hours a store is open as a covariate, simply by entering "HRSOPEN" as a regressor in the Equation specification edit field:


Although not a specification estimated by Card and Krueger, estimation results for the extended model allow evaluation of whether the basic result is sensitive to inclusion of this covariate.

Click on OK to estimate the expanded specification. The results are given by:

Dependent Variable: EMPTOT
Method: Difference in Difference
Date: 05/11/22 Time: 08:53
Periods included: 2
Cross-sections included: 410
Total panel (unbalanced) observations: 787

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | ---: |
| NEWMIN | 2.839666 | 0.729372 | 3.893304 | 0.0001 |
| R-squared | 0.783983 | Mean dependent var | 21.17668 |  |
| Adjusted R-squared | 0.546018 | S.D. dependent var | 9.279035 |  |
| S.E. of regression | 6.252047 | Akaike info criterion | 6.809278 |  |
| Sum squared resid | 14618.95 | Schwarz criterion | 9.259060 |  |
| Log likelihood | -2266.451 | Hannan-Quinn criter. | 7.751101 |  |
| F-statistic | 3.294527 | Durbin-Watson stat | 4.164021 |  |
| Prob(F-statistic) | 0.000000 |  |  |  |

Additional covariates: HRSOPEN

We can see that the inclusion of the HRSOPEN covariate (which is indicated at the bottom of the output) did not have a large impact on the estimate of the effect of the minimum wage
increase (2.84 compared with the original 2.75), though the statistical significance of the result is enhanced considerably.

## Stevenson and Wolfers (2006)

As a second example, we will first replicate the study by Stevenson and Wolfers (2006), which analyzed the impact of the introduction of no-fault divorce reforms on female suicide rates. The dependent variable consists of annual suicide rates for US states between 1964 and 1996. Throughout this period a number of states, at different times, introduced no-fault divorce reform.

This paper and the corresponding data was also studied by Goodman-Bacon (2021) as an application of the Goodman-Bacon decomposition.

A subset of the data used in Stevenson and Wolfers is available in Stata format from Austin Nichols' website. We can easily open this file in EViews, and EViews will automatically detect the panel nature of the data and set up the workfile with that structure:
wfopen http://pped.org/bacon_example.dta
The file contains data on the female suicide mortality rate in the series ASMRS, a binary series, POST, equal to 1 if an observation is in a no-fault divorce environment, and zero if no-fault divorces are not allowed, as well as cross-section (state) and date (year) series.

To determine the impact of no-fault divorce reform on female suicide mortality rate we can perform a simple difference-in-difference, with multiple timings, estimation. We click on Quick/Estimate Equation and then change the Method dropdown to DiD - Difference-inDifference. Note this method is only available because we have a workfile structured as a panel. We enter "ASMRS" as the dependent/outcome variable, followed by our treatment dummy, "POST", and the sample pair "1964 1996":


Clicking OK produces the estimation output:

Dependent Variable: ASMRS
Method: Difference in Difference
Date: 05/10/22 Time: 11:31
Periods included: 33
Cross-sections included: 49
Total panel (balanced) observations: 1617

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | ---: |
| POST | -3.079926 | 2.506623 | -1.228715 | 0.2194 |
| R-squared | 0.700706 | Mean dependent var | 52.16641 |  |
| Adjusted R-squared | 0.684913 | S.D. dependent var | 19.62188 |  |
| S.E. of regression | 11.01428 | Akaike info criterion | 7.685642 |  |
| Sum squared resid | 186217.5 | Schwarz criterion | 7.958891 |  |
| Log likelihood | -6131.842 | Hannan-Quinn criter. | 7.787054 |  |
| F-statistic | 44.36712 | Durbin-Watson stat | 1.257677 |  |
| Prob(F-statistic) | 0.000000 | Parallel trend stat | -2.031163 |  |
| Prob(P. trend) | 0.042238 |  |  |  |

The estimate of the impact of no-fault divorce reform on suicide mortality in females is 3.08, which matches the number presented in Section III of Goodman-Bacon (2021).

To view the Goodman-Bacon decomposition ("Goodman-Bacon Decomposition" on page 404) we click on View/Difference-in-Difference Diagnostics/Goodman-Bacon Decomposition:


EViews opens a spool with output divided into three sections. The Individual components middle section showing all of the $2 \times 2$ individual coefficients and weights for each pair of treatment dates is closed by default. These individual results can be displayed by clicking the " + " symbol:


We can see that the $2 \times 2$ coefficient comparing observations that introduced divorce reform in 1969 with those that introduced reform in 1970 is 3.09 .

The weighted average of these individual year pair-comparisons, combined into groups, is displayed in the first section. We can see that each of the groupings has a negative coefficient, apart from the "Later vs. Earlier" group. Since this comparison group may violate the parallel trends assumption, it is possible that a biased positive coefficient with large weight may have upwardly biased the overall TWFE coefficient to the reported value of -3.08 .

We can see that there is a large number of the "Later vs. Earlier" group above the overall estimated coefficient, again indicating that this group may be upwardly biasing the overall estimate.

## Callaway and Sant'Anna (2021)

As a third example, we'll examine the data used in Callaway and Sant'Anna (2021). Similar to the Card and Krueger (1994_, Callaway and Sant'Anna study the impact of the minimum wage on employment levels, but this time concentrating on teen employment. The data contains county level teen employment between 2003-2007. During this time period the federal minimum wage remained constant at $\$ 5.15$ per hour. However some states increased the minimum wage within the state above the federal level during the time period. Counties within such states are the treated group (with different treatment times). Counties within states that did not change their minimum wage are in the control group.

These data are provided in the workfile "CallawaySantanna.wf1". The series TREATED is a binary variable indicating whether a county has been subject to an increased minimum wage in the time period, the series LEMP contains the log of employment for that county in that time period, and the series LPOP contains the log of the population within the county. A TWFE estimation of the impact of minimum wage on employment can be performed by clicking on Quick/Estimate Equation and then changing the Method dropdown to DiD - Difference-in-Difference. We enter "LEMP" as the dependent/outcome variable, followed by our treatment dummy, "TREATED", and sample pair "2003 2007":


Click on $\mathbf{O K}$ to estimate the specification:

Dependent Variable: LEMP
Method: Difference in Difference
Date: 05/10/22 Time: 11:59
Periods included: 5
Cross-sections included: 500
Total panel (balanced) observations: 2500

| Variable | Coefficient | Std. Error | t-Statistic | Prob. |
| :--- | ---: | :--- | ---: | ---: |
| TREATED | -0.036549 | 0.016566 | -2.206304 | 0.0275 |
| R-squared | 0.993218 | Mean dependent var | 5.772516 |  |
| Adjusted R-squared | 0.991505 | S.D. dependent var | 1.508781 |  |
| S.E. of regression | 0.139060 | Akaike info criterion | -0.929474 |  |
| Sum squared resid | 38.57851 | Schwarz criterion | 0.246984 |  |
| Log likelihood | 1666.842 | Hannan-Quinn criter. | -0.502364 |  |
| F-statistic | 579.7363 | Durbin-Watson stat | 1.686589 |  |
| Prob(F-statistic) | 0.000000 | Parallel trend stat | -0.249133 |  |
| Prob(P. trend) | 0.803258 |  |  |  |

Note the TWFE estimate of the impact of treatment matches that given in Callaway and Sant'Anna, Table 3, section A.

To view the Callaway-Sant'Anna estimator we click on View/Difference-in-Difference Diagnostics/CS Group-Time Effects... which brings up the Callaway-Sant'Anna dialog:


Note, we'll include an additional regressor, LPOP, and set the comparison group to by those states who never perform divorce reform. Since county population is constant through the time period being analyzed (population data is taken from the 10 year census), it would not be possible to include it as an explanatory variable in TWFE estimation. However, it may be used in the CS estimator.


The top portion of the output displays the overall estimate of the treatment effect, -0.042 , which compares with the TWFE estimate of -0.37 .

Below this section we can see the individual coefficients for each date pairing, as well as aggregations of them by treatment date (Group), observation date (Date) and time-sincetreatment (Duration).


## References

Callaway, P. and H. C. Sant'Anna, 2021. "Difference-in-Differences with multiple time periods," Journal of Econometrics, 225(2), 200-230.
Card, D. and A. Krueger, 1994. "Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania," American Economic Review, 84(4), 772-93.
Goodman-Bacon, A, 2021. "Difference-In-Differences With Variation In Treatment Timing," Journal of Econometrics, 225(2), 254-277.
Stevenson, B. and J. Wolfers, 2006. "Bargaining in the Shadow of the Law: Divorce Laws and Family Distress." The Quarterly Journal of Economics, 121(1), 267-288.

## Bayesian Time-varying Coefficients VAR Models

It is often difficult to justify the VAR assumption that model parameters are constant over time. For example, a basic VAR fitted to post-war macroeconomic data assumes that economic relationships have not changed since the mid-1940s. Two popular modeling approaches that do away with this assumption are the switching VAR and the time-varying coefficients VAR, or TVCVAR. The switching VAR deals with occasional discrete changes (e.g., structural breaks), whereas the TVCVAR handles constant, smooth changes. The discussion here pertains to the latter.

The EViews implementation of TVCVAR is Bayesian. The Bayesian TVCVAR, or BTVCVAR, combines the TVCVAR model with a prior distribution. The BTVCVAR is popular even among those who do not identify as Bayesian because the prior provides a convenient way to induce shrinkage in a model that needs it.

The remainder of this discussion is organized as follows: We begin with an overview of the BTVCVAR methodology. We then demonstrate how estimation and other post-updating procedures are carried out for the BTVCVAR in EViews. We provide implementation details next, and close our discussion with an example.

## Background

The TVCVAR consists of two equations, an observation equation and a process equation. The observation equation is a VAR equation with period-specific coefficients, and the process equation specifies a law of motion for the coefficients. The BTVCVAR combines the TVCVAR with a prior distribution over the initial state of the coefficients process and the model parameters.

## Observation Equation

Let $y_{t}$ denoted the $N$-vector of endogenous variables observed on date $t$ for $t=1,2, \ldots, T$. To motivate the two equations that form the TVCVAR, we start with the basic VAR model. The VAR equation under the basic VAR is

$$
y_{t}^{\prime}=x_{t}^{\prime} B+e_{t}^{\prime}
$$

where the covariate vector

$$
\begin{equation*}
x_{t}=\left(y_{t-1}{ }^{\prime}, y_{t-2}{ }^{\prime}, \ldots, y_{t-p}^{\prime}, w_{t}^{\prime}\right)^{\prime} \tag{0.83}
\end{equation*}
$$

consists of $p$ lags of $y_{t}$ and the vector of exogenous variables $w_{t}$. The coefficient matrix $B$ is constant over time. We can get rid of this assumption by adding a time subscript on $B$. Doing so yields the observation equation:

$$
\begin{equation*}
y_{t}^{\prime}=x_{t}^{\prime} B_{t}+e_{t}^{\prime} \tag{0.84}
\end{equation*}
$$

Vectorizing both sides of the observation equation gives

$$
\begin{equation*}
y_{t}=X_{t} b_{t}+e_{t} \tag{0.85}
\end{equation*}
$$

where $X_{t}=I_{N} \otimes x_{t}^{\prime}$ and $b_{t}=\operatorname{vec}\left(B_{t}\right)$. The error vector $e_{t}$ is given by $e_{t} \mid S \sim N(0, S)$ where $S$ is the observation covariance matrix.

## Process Equation

Making coefficients period-specific resolves the original problem, but introduces a new one: Taken alone, the observation equation results in a model that is over-parameterized for any sample size. We can mitigate this issue by specifying a law of motion for the coefficients. Typically this law of motion takes the form of a random walk process:

$$
b_{t}=b_{t-1}+u_{t}
$$

This is the process equation. The process error is given by $u_{t} \mid Q \sim N(0, Q)$ where $Q$ is the process covariance matrix. The initial state of this process, $b_{0}$, is specified as part of the prior.

## Prior Distribution

The process equation is certainly helpful, but may not completely eliminate the problems that are associated with over-parameterization. Indeed, over-parameterization is often an issue even for the basic VAR. The usual solution is to "shrink" the model towards a simpler or stylized version of itself. The concept of shrinkage brings us to our next topic, the BTVCVAR.

The BTVCVAR combines the TVCVAR with a prior distribution. Bayesians form priors based on information they have on the subject under study prior to seeing the data. For many nonBayesians, the prior is simply a means for achieving shrinkage. To shrink the model towards a simpler or stylized version of itself, center the prior at the simpler/stylized model. Tightening the prior will then pull estimates towards the model at the center of the prior.

One example of shrinkage was already mentioned; the TVCVAR can be made to shrink towards the basic VAR by specifying a tight prior about zero for the process error variance terms. Shrinking towards the basic VAR to some extent is desirable because it yields coefficient estimates that evolve more smoothly over time.

The prior over the initial coefficient vector $b_{0}$ and covariance matrices $S$ and $Q$ is

$$
\pi\left(b_{0}, S, Q\right)=\pi\left(b_{0}\right) \pi(S) \pi(Q)
$$

where

$$
\begin{align*}
b_{0} & \sim N\left(\underline{b_{0}}, \underline{B_{0}}\right) \\
S & \sim I W(\underline{S}, \underline{s})  \tag{0.86}\\
Q & \sim I W(\underline{Q}, \underline{q})
\end{align*}
$$

See the prior specification subsection for details on how to set the prior hyper-parameters.
The exposition here assumes an unrestricted process covariance matrix. In practice, it is more common to work with a diagonal $Q$. See the implementation details section for more information.

## Posterior Distribution

The prior distribution combines with the likelihood function to form the posterior distribution, which becomes the basis for inference, predictions, etc. For example, a Bayesian point estimate is usually just the mean or median of the posterior distribution.

Let $y$ denote the set of all observed data, and let $b$ denote the set of all coefficients including $b_{0}$. The posterior distribution over the model unknowns $b, S$, and $Q$ is given by $T$

$$
\pi(b, S, Q \mid y) \propto \pi\left(b_{0}, S, Q\right) \prod_{t=1} f\left(y_{t} \mid b_{t}, S\right) f\left(b_{t} \mid b_{t-1}, Q\right)
$$

where the first, second, and third terms on the right-hand side of the proportionality symbol, $\propto$, correspond to the prior distribution, the observation equation, and the process equation, respectively.

This distribution does not lend itself to vanilla Monte Carlo sampling. Fortunately, posterior simulation is possible using the Gibbs sampler. See the implementation details section for additional information.

## Estimating BTVCVAR in EViews

To estimate BTVCVAR in EViews, click on Quick/Estimate VAR... or run var in the command window. This will open the VAR Specification dialog. Select Bayesian TVCVAR from the VAR type dropdown menu. The dialog should now have the Basics, Prior, and Options tabs.


Endogenous variables, lags, exogenous variables, and the estimation sample are set in the Basics tab. Lags are required to be contiguous.

EViews gives users the option of setting aside a subset of the estimation sample for the purpose of specifying a prior distribution. The observations that go towards specifying the prior comprise the prior sample. The remaining observations make up the data sample, which is used to update the prior. See the prior specification subsection for details.


Prior hyper-parameters are set inside the Prior tab. To help with setting the prior, EViews maps the BTVCVAR prior hyper-parameters to a set of six scalar quantities. Users set these scalars to specify a prior sample, control the variability of the time-varying coefficients, etc. See the "Prior Specification" on page 430 for details.


There are four categories in the Options tab: Sampler, Display, Smoother, and Stability.
The Sampler options determine how the posterior sample is generated.

- The burn-in size is the number of initial draws to discard. It is specified as a count. The burn-in process gives the underlying Markov chain time to converge to the posterior distribution.
- The posterior sample size is used to determine how many posterior draws are used to carry out post-updating procedures (estimation, forecasting, impulses responses analysis, etc.).
- The thinning size is used to thin the Markov chain. A thinning size of $r$ indicates that every $r$-th draw is stored. For example, no thinning occurs when $r$ is set to 1 , and every other draw is stored when $r$ is set to 2 . By definition, thinning does not apply to burn-in draws.
- The seed field is used to set the random seed for the posterior simulator. EViews will generate a seed automatically if the user does not specify one. Click Clear to clear the seed field.
- The number of subchains field determines how many subchains are used when the posterior sample gets regenerated. Regeneration is typically much faster than initial generation since subchains can be run in parallel.

Display options determine what to report as estimation results. Users can pick either posterior median or posterior mean as their point estimate. The point estimate type selected here will be applied to the coefficients, the observation covariance matrix, and the errors. Users can also display equal-tailed credibility intervals (bands) at one or more credibility levels for the coefficients. To do so, check the box next to Show credibility intervals. Bands use shading by default. To use lines instead, check the box next to Use lines.

A simulation smoother can be selected from the dropdown menu under Smoother. EViews currently supports three simulation smoothers: the Cholesky factor algorithm (CFA), the Kalman filter and smoother (KFS), and the method of McCausland, Miller, \& Pelletier (MMP). For more information, see McCausland, Miller, \& Pelletier (2011) and the references therein.

To enforce stable VAR coefficients at each date within the data sample, select Cogley \& Sargent from the dropdown menu under Stability. The maximum number of attempts threshold ensures that the sampler does not get stuck in an infinite loop in an attempt to obtain stable draws.

Once the BTVCVAR model has been specified, click on $\mathbf{O K}$ to run the posterior simulator. Progress is displayed in the bottom left corner of the EViews window. Once posterior simulation is complete, estimates and other statistics based on the posterior sample are computed.

Estimation results are presented in a spool-like object. The nodes under Output Sections in the left pane are used for navigation. For example, clicking on the Summary node will bring the summary table into focus. The checkboxes that appear below Display Coefficients are used to show/hide coefficient series that are associated to specific endogenous variables, lags, and exogenous variables. For example, unchecking the box next to $\mathbf{C}$ hides the coefficient series associated to the constant term in all graphs.

## Working with a BTCVAR

## Procs

To get posterior draws from the BTVCVAR object (to generate trace plots, for example), click on Proc/Put Posterior Draws in New Page. Click on Proc/Make IF, RNE, and ESS to get inefficiency factors (IF), relative numerical efficiencies (RNE), and effective sample sizes (ESS) for all model parameters. NAs are given if sample autocorrelation does not fall below 0.05 in the first 100 lags.

Other procs let users make coefficient estimates, coefficient CI bands, and residuals.

## Residuals

To view an estimate of the error matrix in the form of a graph, go to View/Residuals/ Graphs or click on the Residuals button in the VAR toolbar. Go to View/Residuals/Spreadsheet to see the same information in a table. Click on View/Residuals/Covariance Matrix for an estimate of the observation covariance matrix.

## Impulse Response

To generate impulse responses, go to View/Impulse Response... or click on the Impulse button in the VAR toolbar. This will open the BTVCVAR version of the Impulse Responses dialog.


The Display tab has two sections: Display information and Display options. The former is identical to that used by the standard VAR model. The latter is specific to the BTVCVAR.

Impulse responses are generated based on VAR coefficients at the dates entered in the Impulse date field. Impulse dates must fall within the data sample period.

The remaining display options are identical to those used in estimation.

## Forecasting

To generate forecasts, go to Proc/Forecast... or click on the Forecast button in the VAR toolbar.


If Dynamic forecast is selected, a standard VAR is assumed over the forecast period. Forecasts are generated based on VAR coefficients taken from the period prior to the start of the forecast period.

Forecast display options are identical to those used in estimation.

## Implementation Details

## Prior Specification

The prior hyper-parameters are $b_{0}, B_{0}, \underline{S}, \underline{s}, \underline{Q}$, and $\underline{q}$. Users set prior hyper-parameters through the six scalar quantities $T_{0}, \bar{\tau}_{0}, \tau_{1}, \tau_{2}, v_{1}$, and $v_{2} . T_{0}$ is the prior sample size, $\tau_{0}$ is a scaling factor controlling the prior variance of $b_{0}, \tau_{1}$ and $\tau_{2}$ are prior scaling factors for $\underline{S}$ and $\underline{Q}$, respectively, and $v_{1}=\underline{s}$ and $v_{2}=\underline{q}$ are prior degrees of freedom parameters.

Users have the option to set aside the first $T_{0}$ observations of the estimation sample for the purpose of specifying the prior distribution. The way in which the scalar quantities map to the original prior hyper-parameters depends on whether a prior sample is present.
Let $\hat{b}_{0}$ and $\hat{\operatorname{var}}\left(\hat{b}_{0}\right)$ denote OLS results based on the prior sample. If $T_{0}>k$, where $k$ is the number of coefficients per equation, then the mapping

$$
\begin{aligned}
\underline{b_{0}} & =\hat{b}_{0} \\
\underline{B_{0}} & =\tau_{0} \hat{\operatorname{var}}\left(\hat{b}_{0}\right) \\
\underline{S} & =\tau_{1} I \\
\underline{Q} & =\tau_{2} \hat{\operatorname{var}}\left(\hat{b}_{0}\right)
\end{aligned}
$$

is used. When a prior sample is not present, i.e., when $T_{0}=0$, then the mapping is given by

$$
\begin{align*}
\underline{b_{0}} & =0 \\
\underline{B_{0}} & =\tau_{0} I  \tag{0.87}\\
\underline{S} & =\tau_{1} I \\
\underline{Q} & =\tau_{2} I
\end{align*}
$$

Entering invalid values for $T_{0}$ (e.g., $1 \leq T_{0} \leq k$ ) returns an error.

## Posterior Simulation

Posterior simulation is carried out using the Gibbs sampler, which is a Markov chain Monte Carlo (MCMC) method for sampling from multivariate densities. It generates a Markov chain by iteratively sampling from the full conditionals of the target density. Draws obtained after the burn-in period are used to compute estimates, generate predictions, and so on.

The chain is initialized at the mode of the prior. Posterior full conditionals are described below.

## Posterior Full Conditional for b

Let

$$
y=\left[\begin{array}{l}
y_{1}  \tag{0.88}\\
\ldots \\
y_{T}
\end{array}\right], X=\left[\begin{array}{ccccc}
O & X_{1} & O & \ldots & O \\
\ldots & O & X_{2} & O & \ldots \\
O & \ldots & O & \ldots & O \\
O & O & \ldots & O & X_{T}
\end{array}\right], b=\left[\begin{array}{c}
b_{0} \\
b_{1} \\
\ldots \\
b_{T}
\end{array}\right]
$$

where the $O$ 's are zero matrices. Let

$$
D=\left[\begin{array}{ccccc}
1 & 0 & 0 & \ldots & 0  \tag{0.89}\\
-1 & 1 & 0 & 0 & \ldots \\
0 & -1 & 1 & 0 & 0 \\
\ldots & 0 & \ldots & \ldots & 0 \\
0 & \ldots & 0 & -1 & 1
\end{array}\right]
$$

denote the matrix of first differences. It is a square matrix of order $T+1$. Then

$$
\begin{equation*}
(b \mid y, S, Q) \sim N(\bar{b}, \bar{B}) \tag{0.90}
\end{equation*}
$$

where

$$
\begin{align*}
& \bar{B}=\left\{(D \otimes I)^{\prime}\left[\begin{array}{cc}
\underline{B}_{0}^{-1} & O \\
O & I \otimes Q^{-1}
\end{array}\right](D \otimes I)+X^{\prime}\left(I \otimes S^{-1}\right) X\right\}^{-1} \\
& \bar{b}=\bar{B}\left\{(D \otimes I)^{\prime}\left[\begin{array}{c}
\underline{B}_{0}^{-1} \underline{b}_{0} \\
0
\end{array}\right]+X^{\prime}\left(I \otimes S^{-1}\right) y\right\}
\end{align*}
$$

## Posterior Full Conditional for S

Let

$$
E=\left(e_{1}, e_{2}, \ldots, e_{T}\right)^{\prime}
$$

denote the matrix of observation errors. Then

$$
(S \mid y, b, Q) \sim I W(\bar{S}, \bar{s})
$$

where

$$
\begin{gather*}
\bar{S}=\underline{S}+E^{\prime} E  \tag{0.92}\\
\bar{s}=\underline{s}+T
\end{gather*}
$$

## Posterior Full Conditional for Q

Let

$$
U=\left(u_{1}, u_{2}, \ldots, u_{T}\right)^{\prime}
$$

denote the matrix of process errors. Then

$$
\begin{equation*}
(Q \mid y, b, S) \sim I W(\bar{Q}, \bar{q}) \tag{0.93}
\end{equation*}
$$

where

$$
\begin{gather*}
\bar{Q}=\underline{Q}+U^{\prime} U  \tag{0.94}\\
\bar{q}=\underline{q}+T
\end{gather*}
$$

## Diagonal Q

EViews follows the common practice of requiring $Q$ to be diagonal. Under this restriction, the prior on $Q$ cannot be inverse Wishart. Instead, we have

$$
\pi(Q)=\prod \pi\left(Q_{j j}\right)
$$

where the prior on the $j$-th diagonal element $\dot{Q}_{j j}$ is given by

$$
Q_{j j} \sim I W\left(\underline{Q}_{j j}, \underline{q}\right)
$$

The notation $\underline{Q}$ and $\underline{q}$ are as they were defined in the unrestricted case. It can be shown that the posterior full conditional for $Q_{j j}$ is

$$
\begin{equation*}
\left(Q_{j j} \mid y, b, S\right) \sim I W\left(\bar{Q}_{j j}, \bar{q}\right) \tag{0.95}
\end{equation*}
$$

where $\bar{Q}$ and $\bar{q}$ are as they were defined in the unrestricted case. Conditioning on other diagonal elements is suppressed in the notation; notice that the diagonal elements of $Q$ are conditionally independent in any case.

## Memory Use

Following posterior simulation, a BTVCVAR object will hold onto draws until either the object is deleted or the workfile is closed. Estimation results are written to disk; therefore, a BTVCVAR object in a saved workfile can display estimation results without having to regenerate the posterior sample. Regeneration of the posterior sample is required if display options are changed or when conducting other post-sampling procedures.

## Replications

For reproducibility, set the random seed, the random number generator type, and the number of subchains. Specifying these options ensures that their default values are not used.

When a posterior sample is regenerated, it has the same set of draws as the original sample. If the posterior sample cannot be regenerated for any reason, EViews will give you the option to generate a new set of draws.

## Example

We demonstrate the EViews implementation of BTVCVAR using a data set from Chan and Jeliazkov (2009). Open the EViews workfile cj09.wf1 containing the series GDP (output growth), UNEMP (unemployment rate), INTEREST (interest rate), and INFLATION (inflation rate). Set the workfile sample to 1948Q4 2009Q4 to match the sample used by the code accompanying the paper. Next, run

```
var myvar.btvcvar(nu1=7, nu2=6, size=20000, burn=1000, usemean) 1 1
    gdp unemp interest inflation
```

in the command window. EViews will indicate the progress of the sampler in the bottom left corner of the window.

Iterations: 4678 of 21000 ( $22.2 \%$ done); VAR: MYVAR
Once complete, double click on MYVAR to view the estimation output.


Estimation output for the BTVCVAR is presented as a spool object. A summary is provided at the top, followed by graphs showing the evolution of coefficients over time for each equation.

To change display options, click on the Estimate button in the VAR toolbar and go to the Options tab. As an example, to show shaded $30 \%$ credibility bands, check the box next to Show credibility intervals and enter 0.3 for Credibility levels. Click OK. The estimation output will update to look like this:


Note that changing display options will not prompt the posterior simulator to run if posterior draws are already available.

## References

Blake, A. P., \& Mumtaz, H. (2012). Applied Bayesian Econometrics for Central Bankers. Technical Books: Centre for Central Banking Studies, Bank of England.
Canova, F. (1993). "Modelling and forecasting exchange rates with a Bayesian time-varying coefficient model," Journal of Economic Dynamics and Control, 17(1-2), 233-261.
Cogley, T., and Sargent, T. J. (2002). "Evolving post-World War II US inflation dynamics," NBER macroeconomics Annual, 16, 331-388.
Cogley, T., and Sargent, T. J. (2005). "Drifts and volatilities: monetary policies and outcomes in the post WWII US," Review of Economic Dynamics, 8(2), 262-302.
Del Negro, M., and Schorfheide, F. (2011). "Bayesian Macroeconometrics," in The Oxford Handbook of Bayesian Econometrics, Geweke, J., Koop, G., and Van Dijk H., eds., Oxford: Oxford University Press.

Frühwirth-Schnatter, S. (1994). "Data augmentation and dynamic linear models," Journal of Time Series Analysis, 15(2), 183-202.
Giordani, P., Pitt, M., and Kohn, R. (2011). "Bayesian inference for time series state space models," in The Oxford Handbook of Bayesian Econometrics, Geweke, J., Koop, G., and Van Dijk H., eds., Oxford: Oxford University Press.
Karlsson, S. (2013). "Forecasting with Bayesian vector autoregression," in Handbook of Economic Forecasting, Volume 2, Part B, Elliott, G. and A. Timmermann, eds., 791-897, Elsevier.
Koop, G., and Korobilis, D. (2010). Bayesian multivariate time series methods for empirical macroeconomics, Now Publishers Inc.
Koop, G., and Potter, S. M. (2011). "Time varying VARs with inequality restrictions," Journal of Economic Dynamics and Control, 35(7), 1126-1138.
Louzis, D. P. (2019). Steady-state modeling and macroeconomic forecasting quality. Journal of Applied Econometrics, 34(2), 285-314.
Lubik, T. A., and Mathes, C. (2015). "Time-varying parameter vector autoregressions: Specification, estimation, and an application. Estimation, and an Application," Economic Quarterly, 101 (4), 323-352.
Nakajima, J. (2011). "Time-varying parameter VAR model with stochastic volatility: An overview of methodology and empirical applications," Monetary and Economic Studies, 29, 107-142.
Primiceri, G. E. (2005). "Time varying structural vector autoregressions and monetary policy," The Review of Economic Studies, 72(3), 821-852.

## Index

## Symbols

186, 210

## A

Add-ins
delete 142
Andrews automatic bandwidth
robust standard errors 222
ARDL
Bounds test 118
Dynamic multipliers 150
panel
pooled mean group estimation
similarity test 273
Symmetry test multipliers 276
ardl 112
AREMOS 140
AREMOS-TSD 140
ARIMA models
automatic forecasting 115
automatic selection 115
ASCII file
export coef to file 158
export matrix to file 161
export sym to file 167
export vector to file 164,170
open file as workfile 278
save table to file 235
Australian Bureau of Statistics 69
ABS 140
autoarma 115
Automatic bandwidth selection
robust standard errors 222
Automatic forecast
ARIMA 115
ETS smoothing 61, 117, 155
Autoregressive distributed lag models
See ARDL.

## B

Backcast
MA terms 221
Background color 245, 250, 255

Bandwidth
GMM estimation 222
Bartlett kernel robust standard errors 222
Bayesian time-varying coefficients VAR See BTVCVAR
BEA 140
Binary file 278
Bloomberg data 140
BLS 140
Bohman kernel robust standard errors 222
Bounds test 118
boundstest 118
BTVCVAR 119
estimation 425
forecasting 429
impulse response 429
memory 433
posterior 425, 431
post-updating procedures 428
prior 424
procs 428
replication 433
residuals 429
BTVCVAR 119
Bureau of Economic Analysis 140
Bureau of Labor Statistics 140

## C

CEIC 140
Cell
background color 250, 255
text color 264, 269
clearcollabels 121,122
clearowrlabels 122
clearrowlabels 123
Cloud drive 281, 297
Coef 105
procs 105
coef
export to disk 158
Coefficient
See coef.
coint 124,133
Cointegration
Engle-Granger test 124
Johansen test 133
Phillips-Ouliaris test 124
restrictions 377
test 124
test from a VAR 133
Color
keywords for specifying $244,249,254,258$, 267, 272
@RGB specification 244, 249, 254, 258, 267, 272
Column
matrix labels 121, 238
rowvector labels 121, 238
svector label 239
svector labels 122
sym labels 239
vector label 240
vector labels 122
COM session
open 298
Convergence criterion 221, 222
Convert
scalar to string 94
string to scalar 99
CSV
export coef to file 158
export matrix to file 161
export sym to file 167
export vector to file 164,170
save table to file 235
@cumprod 91
@cumsum 91
Cumulative statistics
functions 91
product 91
sum 91

## D

Daily seasonal adjustment 146
Daniell kernel
robust standard errors 222
Data
import 278
import as matrix 192
import as rowvector 198
import as sym 204
import as vector 186, 210
Data members
geomap 106
Database
Australian Bureau of Statistics 69
Deutsche Bundesbank 69
EIA 140
Insee 69
open existing 139
SDMX 69
Datastream 140
datelabel 137
DBnomics 140
dbopen 139
deleteaddin 142
Deutsche Bundesbank 69, 140
did 143
DIfference-in-difference 143
Double exponential smoothing 274
DRI DDS 140
DRIPro link 140
Dropbox 281, 297
dsa 146
Dynamic forecasting 180
Dynamic multipliers 150
dynmult 150

## E

ec 151
ECB 140
EIA 140
Enterprise Edition 141
Equation 105
methods 105
procs 61, 106
views 105
Error correction model
See VEC and VAR.
Error-trend-seasonal smoothing
See ETS smoothing
Estimation
BTVCVAR 425
panel
Estimation methods
ARMA 220
for var 111
generalized least squares 220
least squares 220
nonlinear least squares 220
ets 154
ETS smoothing 154
forecast $61,117,155$
Eurostat database 140
EViews database 140
EViews Enterprise Edition 141
Excel
export coef to file 158
export matrix to file 161
export sym to file 167
export vector to file 164,170
Excel file 278
export data to file 293, 295
importing data into matrix 192
importing data into rowvector 198
importing data into sym 204
importing data into vector 186,210
Exponential smoothing 154, 274
See also ETS smoothing
Holt-Winters additive 275
Holt-Winters multiplicative 275
Holt-Winters no seasonal 275
Export 229
workfile data 229, 293
export 158, 161, 164, 167, 170
External application
string 102

## $F$

FAME database 140
Federal Reserve Economic Data 140
Files
opening/saving on a cloud location 281, 297 Fill
opacity 243
svector 94
fill 173
Fisher-Johansen 130
fit 174
Fitted index 174
Fitted values 174
forcavg 182
Forecast
ARIMA 115
automatic with ARIMA models 115
automatic with ETS smoothing 61, 117, 155
averaging 182
combining 182
dynamic (multi-period) 180
ETS smoothing 61, 117, 155
static (one-period ahead) 174
VAR/VEC 184
forecast 180, 184
Foreign data
open as workfile 278
save workfile data as 293
FRED 140
@-functions
cumulative statistics functions 91
fxrow 179
fxrowcol 180

## G

Gauss file 278, 293
Gauss-Newton 221
Geomap 106
fill (background) color for shapes 245
procs 106
shape labels 259, 263
Geomap (object)
data members 106
GiveWin data 140
GLS 220
Google Drive 281, 297
Graph 107
axis labeling 137
element opacity 242,243
options for individual elements 240
procs 107
Group 107
fill (background) color for cells 250
procs 107
text color for cells 264
views 107

## H

HAC
robust standard errors 221, 223
Hausman test
similarity in ARDL/PMG 273
Haver Analytics Database 140
@holiday 65
Holiday functions 65
@holidayset 65

Holt-Winters 274
HTML
export coef to file 158
export matrix to file 161
export sym to file 167
export vector to file 164,170
open page as workfile 278
save table to file 235
Huber/White standard errors 221, 223

## I

IHS 140
IHS Global Insight data 140
IHSMarkit 140
IMF 140
import 192, 198
Import data 278
as matrix 192
as rowvector 198
as sym 204
as vector 186,210
importmat 204
impulse 216
Impulse response 216
Index
fitted from binary models 174
fitted from censored models 174
fitted from truncated models 174
Insee 69
INSEE (National Institute of Statistics and Economic Studies) 141
Iteration 221
optimization method 221

## $J$

Johansen cointegration test 124
from a VAR 133

## $K$

Kao panel cointegration test 130
Kernel
robust standard errors 222

## $L$

LaTeX
export coef as 158
export matrix as 161
export sym as 167
export vector as 164,170
save table as 235
Line
opacity 242
Load
workfile 278
Lotus file 278, 293
export data to file 293 open 278
ls 220

## M

Magellan 140
@makevalidname 93
Markdown
export coef as 158
export matrix as 161
export sym as 167
export vector as 164,170
save table as 235
Matrix 107
column labels 121, 238
export to disk 161
procs 107
resize 233
row headers 122
row labels 260
Matrix commands and functions
utility 92
Model averaging 182
Moody's Economy.com 141

## $N$

Name
make a valid object name 93
National Oceanic and Atmospheric Administration 141
Newey-West automatic bandwidth robust standard errors 222
Newton-Raphson 221
NOAA 141
Nonlinear ARDL
Symmetry test 276
Nonlinear least squares
single equation estimation 220
Numbers
converting from strings 99

## 0

Object
make a valid name 93
ODBC 278
OECD 141
OLS (ordinary least squares)
single equation estimation 220
OneDrive 281, 297
Open
database 139
foreign data as import 204
foreign data as matrix $186,192,210$
foreign data as rowvector 198
foreign data as sym 204
foreign source data 278
workfile 278
Optimization
methods 221
Optimization algorithms
Gauss-Newton 221
Newton-Raphson 221

## $P$

Page
save or export 229
pageload 228
pagesave 229
Panel cointegrating regression
PMG models
Panel data
estimation See Panel estimation.
Panel estimation 105
Parzen kernel
robust standard errors 222
Parzen-Cauchy kernel
robust standard errors 222
Parzen-Geometric kernel
robust standard errors 222
Parzen-Riesz kernel
robust standard errors 222
PcGive data 140
PDF
export coef as 158
export matrix as 161
export sym as 167
export vector as 164,170
save table as 235
Pedroni panel cointegration test 130

Pesaran, Shin and Smith 118
PMG
similarity test 273
Pooled Mean Group estimation
Product
cumulative 91
Program evaluation 143

## Q

Quadratic spectral kernel robust standard errors 222

## $R$

@range 93
Range of integers 93
RATS data
4.x native format 141
open 278
portable format 141
Read 186, 192, 198, 204, 210, 278
data from foreign file as matrix 192
data from foreign file as rowvector 198
data from foreign file as sym 204
data from foreign file as vector 186,210
resize 233,234
@RGB specification of colors 244, 249, 254, 258, 267, 272
Row
matrix headers 122
matrix labels 260
rowvector headers 123
rowvector labels 261
svector labels 123, 261
sym labels 262
vector labels 123, 262
Rowector
export to disk 164
Rowvector 108
column labels 121, 238
procs 108
row headers 123
row labels 261
RTF
export coef to file 158
export matrix to file 161
export sym to file 167
export vector to file 164,170
save table to file 235

## $S$

SAS file 278
Save 293
workfile 293
workfile as foreign file 293
save 235
SDMX Database
Australian Bureau of Statistics 69
Deutsche Bundesbank 69
Insee 69
SDMX_ML format 141
Seasonal adjustment
daily 146
Series 109
fill (background) color for cells 255
smoothing 274
text color for cells 269
setcollabels 238, 239, 240
setelem 240
setfillcolor $245,250,255$
setjust 259
setrowlabels 260, 261, 262
setshapelabel 263
settextcolor 264,269
@sfill 94
Shape
background color 245
Shape labels 259, 263
smooth 274
Smoothing
ETS model 154
exponential smooth series 274
likelihood based 154
Spreadsheet
file export 293
file import 278
file import as matrix 192
file import as rowvector 198
file import as sym 204
file import as vector 186,210
SPSS file 278
Stata file 278
Static forecast 174
©str 94
String
convert from a number 94
convert to number 99
String list
replace string 102
Sum
cumulative 91
Svector 109
column labels 122, 239
fill values 173
initialize 173
resize 234
row label 123
row labels 261
sfill 94
Sym 109
column labels 239
export to disk 167
procs 109
row labels 262
Symmetric matrix
See Sym.
Symmetry test 276
symmtest 276
Symr
resize 234

## $T$

Table 111
fix rows and columns in display 180
fix rows in display 179
procs 111
save to diskExcel
save table to file 235
Test
Johansen cointegration 124
Johansen cointegration from a VAR 133
Text color 264, 269
Text file
import as matrix 192
import as rowvector 198
import as sym 204
import as vector 186,210
open as workfile 278
save workfile as 293
Trading Economics 141
TSP portable data format 141
Tukey-Hamming kernel
robust standard errors 222
Tukey-Hanning kernel
robust standard errors 222
Tukey-Parzen kernel
robust standard errors 222

## U

U.S. Energy Administration 140

UN 141
US Census 141

## V

@val 99
VAR
forecasting 184
impulse response 216
methods 111
procs 111
variance decomposition 276
Var 111
Variance decomposition 276
vdecomp 276
VEC
estimating 151
Vector 111
column labels 122,240
export to disk 170
integer range 93
procs 111
resize 234
row label 123
row labels 262
Vector error correction model See VEC and VAR.

## w

wfopen 278
wfsave 293
WHO 141
Workfile
export 293
load workfile pages into 228
open existing 278
open foreign source into 278
save 293
save individual page 229
World Bank 141
World Health Organization 141
@wreplace 102

## $x$

xopen 298


[^0]:    S\&P Global Inc.
    3030 Old Ranch Parkway
    2nd Floor
    Suite 260
    Seal Beach, CA 90740
    Telephone: (949) 856-3368
    Fax: (949) 856-2044
    e-mail: sales@eviews.com
    web: www.eviews.com
    June 3, 2022

