EViews 4.1 Update

Quantitative Micro Software

EViews 4.1 Update

Disclaimer

The authors and Quantitative Micro Software 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

Quantitative Micro Software, LLC
4521 Campus Drive, #336, Irvine CA, 92612-2699
Telephone: (949) 856-3368
Fax: (949) 856-2044
e-mail: sales@eviews.com
web: www.eviews.com
March 11, 2002

Table of Contents

Chapter 1. Introduction

We are very pleased to bring you this free upgrade from EViews 4.0 to EViews 4.1. In response to user requests, we have added several new features to EViews. Some of these features, like added function support, an expanded Wald test output, and extensions to the modelling program language, are minor improvements to existing routines. Others, like the new unit root testing features, and extensions to both the state space error, and system instrumental variables specification languages, represent significant improvements in the set of tools for working with, and analyzing your data.

You are now looking at a self-contained .PDF document which describes the EViews 4.1 upgrade. Bear in mind that this document is completely isolated from your original .PDF documents, and that the latter documents, which may still be accessed through EViews 4.1, will contain information that is now out-of-date. Updated .PDF files will be available for optional download when EViews 4.1 is officially released.

While we believe that you will find the new features to be problem free, you should bear in mind that this version of EViews 4.1 is a Beta test version. To report problems, or to make comments about this product, please send email to: support@eviews.com. Please include in your message a comment noting that you are working with the EViews 4.1 beta along with the date of your copy of EViews 4.1. The date may be obtained by selecting **Help/About EViews** from the main menu.

New Features in 4.1

Unit Root Testing

EViews 4.1 includes support for the newest generation of unit root tests. In addition to the existing Augmented Dickey-Fuller and Phillips-Perron tests, EViews now allows you to compute the GLS-detrended Dickey-Fuller (Elliot, Rothenberg, and Stock, 1996), Kwiatkowski, Phillips, Schmidt, and Shin (KPSS, 1992), Elliott, Rothenberg, and Stock Point Optimal (ERS, 1996), and Ng and Perron (NP, 2001) unit root tests.

EViews will also perform Newey-West (1992) and Andrews (1991) automatic bandwidth selection for kernel based estimators, or automatic information criteria based selection of lag length for Dickey-Fuller tests and AR spectral density estimators. See "Unit Root Tests" on page 5.

The command support for these new unit root features is documented in the command reference for uroot (p. 43).

System Extensions

We have expanded the flexibility of instrumental variables specifications estimated by 2SLS and 3SLS.

Previously, all instrumental variables projections in systems were performed on an equation-by-equation basis. In EViews 4.1, the new `@stackinst` statement provides a new way of specifying instruments for systems of equations that allows for cross-equations on the projections of variables on instruments. You can now stack your equations and instruments prior to performing the projection. The easy-to-use syntax provides you with full control over the instrument stacking so that you may combine the earlier ordinary instrument and new stacked instrument specifications. For details, see "System Instrumental Variables" on page 19.

Sspace Improvements

EViews 4.1 extends the features of the sspace object in two distinct ways.

First, extensions to the state space syntax in EViews 4.1 allow you to write the error term for any equation as a linear combination of named errors. This syntax allows users to specify, more naturally, a wide range of important models. For discussion, see "Specifying Errors and Variances in a Sspace" on page 23.

Second, new sspace object data members give you access to output matrices containing intermediate calculations from the Kalman filter. See "Accessing Sspace Filter and Smoother Results" on page 25.

Enhanced Model Features

EViews 4.1 provides new modelling tools to aid you in model building and solution. New command syntax provides you with greater control over the model solution procedure, enhanced tools for working with scenarios, and additional commands to maintain the links in your model. See "Extended Model Commands" on page 27.

Miscellaneous Statistical Features

The output of the Wald test has been expanded to provide additional information about the restrictions and the restriction variances. As a result, you can use the expanded output to find the standard errors of functions of your coefficients. For details, see "Enhanced Wald Tests" on page 27.

The "Statistics by Classification" view of a series now allows you to compute arbitrary quantiles of your data by group. See "Quantiles by Classification" on page 32. See `statby` (p. 40) for documentation of the expanded command options.

New Functions

EViews 4.1 now supports a family of percentage change functions that complement the existing functions. These functions eliminate the need to rescale function values when working in percentage terms. See "Time Series Functions" on page 35.

In addition, we have added functions for computing base-10 and arbitrary base logarithms. See "Basic Mathematical Functions" on page 35.

Chapter 1. Unit Root Tests

EViews 4.1 includes support for the newest generation of unit root tests. In addition to the existing Augmented Dickey-Fuller (1979) and Phillips-Perron (1998) tests, EViews now allows you to compute the GLS-detrended Dickey-Fuller (Elliot, Rothenberg, and Stock, 1996), Kwiatkowski, Phillips, Schmidt, and Shin (KPSS, 1992), Elliott, Rothenberg, and Stock Point Optimal (ERS, 1996), and Ng and Perron (NP, 2001) unit root tests.

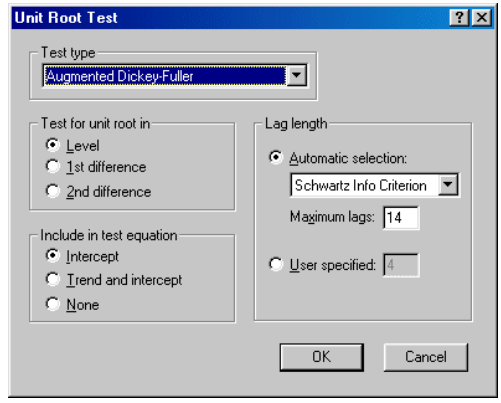
In addition, EViews now allows you to perform Newey-West (1994) and Andrews (1991) automatic bandwidth selection for kernel based estimators, or automatic information criteria based selection of lag length for Dickey-Fuller tests and AR spectral density estimators.

The command support for these new unit root features is documented in uroot (p. 43).

Performing Unit Root Tests in EViews

The following discussion assumes that you are familiar with the basic forms of the unit root tests, and the associated options. We provide theoretical background for these tests in "Basic Unit Root Theory" beginning on page 9, and document the settings used when performing these tests.

To begin, double click on the series to open the series window, and choose **View/Unit Root Test…**



You must specify four sets of options to carry out a unit root test. The first three settings (on the left-hand side of the dialog) determine the basic form of the unit root test. The fourth set of options (on the right-hand side of the dialog) consist of test specific advanced settings. You only need concern yourself with these latter settings if you wish to customize the calculation of your unit root test.

First, use the topmost combo box to select the type of unit root test that you wish to perform. You may choose one of six tests: ADF, DFGLS, PP, KPSS, ERS, and NP.

Next, specify whether you wish to test for a unit root in the level, first difference, or second difference of the series.

Lastly, choose your exogenous regressors. You can choose to include a constant, a constant and linear trend, or neither (there are limitations on these choices for some of the tests).

You can click on **OK** to compute the test using the specified settings, or you can customize your test using the advanced settings portion of the dialog.

The advanced settings for both the ADF and DFGLS tests allow you to specify how lagged difference terms p are to be included in the ADF test equation. You may choose to let EViews automatically select p, or you may specify a fixed positive integer value. If you choose automatic selection, you are given the additional option of selecting both the information criterion and maximum number of lags to be used in the selection procedure.

In this case, we have chosen to estimate an ADF test that includes a constant in the test regression and employs automatic lag length selection using a Schwarz Information Criterion (BIC) and a maximum lag length of 14. Applying these settings to data on the U. S. one-month Treasury bill rate for the period from March 1953 to July 1971, we can replicate Example 9.2 of Hayashi (2000, p. 596). The results are described below.

The first part of the unit root output provides information about the form of the test (the type of test, the exogenous variables, and lag length used), and contains the test output, associated critical values, and in this case, the p-value:

Null Hypothesis: TBILL has a unit root
Exogenous: Constant
Lag Length: 1 (Automatic based on SIC, MAXLAG=14)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-1.417410	0.5734
Test critical values:	1% level	-3.459898	
	5% level	-2.874435	
	10% level	-2.573719	

*MacKinnon (1996) one-sided p-values.

The ADF statistic value is -1.417 and the associated one-sided p-value (for a test with 221 observations) is .573. In addition, EViews reports the critical values at the 1%, 5% and 10% levels. Notice here that the statistic t_α value is greater than the critical values so that we do not reject the null at conventional test sizes.

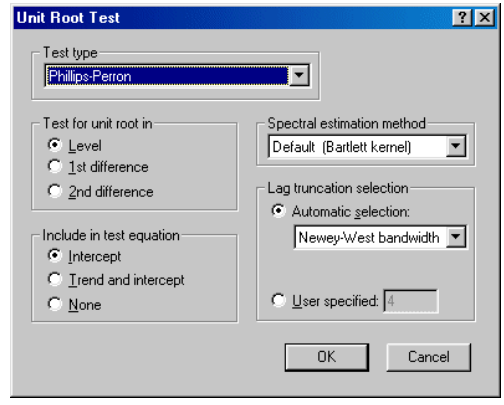
The second part of the output shows the intermediate test equation that EViews used to calculate the ADF statistic:

Augmented Dickey-Fuller Test Equation
Dependent Variable: D(TBILL)
Method: Least Squares
Date: 02/07/02 Time: 12:29
Sample: 1953:03 1971:07
Included observations: 221

Variable	Coefficient	Std. Error	t-Statistic	Prob.
TBILL(-1)	-0.022951	0.016192	-1.417410	0.1578
D(TBILL(-1))	-0.203330	0.067007	-3.034470	0.0027
C	0.088398	0.056934	1.552626	0.1220

R-squared	0.053856	Mean dependent var	0.013826
Adjusted R-squared	0.045175	S.D. dependent var	0.379758
S.E. of regression	0.371081	Akaike info criterion	0.868688
Sum squared resid	30.01882	Schwarz criterion	0.914817
Log likelihood	-92.99005	F-statistic	6.204410
Durbin-Watson stat	1.976361	Prob(F-statistic)	0.002395

If you had chosen to perform any of the other unit root tests (PP, KPSS, ERS, NP), the right side of the dialog would show the different options for the specified test. The options are associated with the method used to estimate the zero frequency spectrum term, f_0, that is used in constructing the particular test statistic. As before, you only need pay attention to these settings if you wish to change from the EViews defaults.



Here we have selected the PP test in the combo box. Note that the right-hand side of the dialog has changed, and now features a combo box for selecting the spectral estimation method. You may use this combo box to choose between various kernel or AR regression based estimators for f_0. The entry labeled "Default" will show you the default estimator for the specific unit root test—here we see that the PP default uses a kernel sum-of-covariances estimator with Bartlett weights. If, instead, you had selected a NP test, the default entry would be "AR spectral-GLS".

Lastly, you can control the lag length or bandwidth used for your spectral estimator. If you select one of the kernel estimation methods (Bartlett, Parzen, Quadratic Spectral), the dialog will give you a choice between using Newey-West or Andrews automatic bandwidth selection methods, or providing a user specified bandwidth. If, instead, you choose one of the AR spectral density estimation methods (AR Spectral - OLS, AR Spectral - OLS detrended, AR Spectral - GLS detrended), the dialog will prompt you to choose from various automatic lag length selection methods (using information criteria) or to provide a

user specified lag length. See "Automatic Bandwidth and Lag Length Selection" on page 16 of the *User's Guide*.

Once you have chosen the appropriate settings for your test, click on the **OK** button. EViews reports the test statistic along with output from the corresponding test regression. In addition, EViews reports the estimate of the frequency zero spectrum f_0 (labeled as the "HAC corrected variance") as well as the uncorrected estimate of the residual variance (where applicable). Running a PP test using the TBILL series yields:

Null Hypothesis: TBILL has a unit root
Exogenous: Constant
Bandwidth: 3.82 (Andrews using Bartlett kernel)

		Adj. t-Stat	Prob.*
Phillips-Perron test statistic		-1.519035	0.5223
Test critical values:	1% level	-3.459898	
	5% level	-2.874435	
	10% level	-2.573719	

*MacKinnon (1996) one-sided p-values.

Residual variance (no correction)	0.141569
HAC corrected variance (Bartlett kernel)	0.107615

As with the ADF test, we fail to reject the null hypothesis of a unit root in the TBILL series at conventional significance levels.

Note that your test output will differ somewhat for alternative test specifications. For example, the KPSS output only provides the asymptotic critical values tabulated by KPSS:

Null Hypothesis: TBILL is stationary
Exogenous: Constant
Bandwidth: 11 (Newey-West Fixed using Bartlett kernel)

		LM-Stat.
Kwiatkowski-Phillips-Schmidt-Shin test statistic		1.537310
Asymptotic critical values*:	1% level	0.739000
	5% level	0.463000
	10% level	0.347000

*Kwiatkowski-Phillips-Schmidt-Shin (1992, Table 1)

Residual variance (no correction)	2.415060
HAC corrected variance (Bartlett kernel)	26.11028

Similarly, the NP test output will contain results for all four test statistics, along with the NP tabulated critical values.

A word of caution. You should note that the critical values reported by EViews are valid only for unit root tests of a data series, and will be invalid if the series is based on estimated values. For example, Engle and Granger (1987) proposed a two-step method to test

for cointegration. The test amounts to testing for a unit root in the residuals of a first stage regression. Since these residuals are estimates of the disturbance term, the asymptotic distribution of the test statistic differs from the one for ordinary series. The correct critical values for a subset of the tests may be found in Davidson and MacKinnon (1993, Table 20.2).

Basic Unit Root Theory

The following discussion outlines the basic features of unit root tests. By necessity, the discussion will be brief. Users who require detail should consult the original sources and standard references (see, for example, Davidson and MacKinnon, 1993, Chapter 20, Hamilton, 1994, Chapter 17, and Hayashi, 2000, Chapter 9).

Consider a simple AR(1) process:

$$y_t = \rho y_{t-1} + x_t'\delta + \epsilon_t, \tag{1.1}$$

where x_t are optional exogenous regressors which may consist of constant, or a constant and trend, ρ and δ are parameters to be estimated, and the ϵ_t are assumed to be white noise. If $|\rho| \geq 1$, y is a nonstationary series and the variance of y increases with time and approaches infinity. If $|\rho| < 1$, y is a (trend-)stationary series. Thus, the hypothesis of (trend-)stationarity can be evaluated by testing whether the absolute value of ρ is strictly less than one.

The unit root tests that EViews provides generally test the null hypothesis H_0: $\rho = 1$ against the one-sided alternative H_1: $\rho < 1$. In some cases, the null is tested against a point alternative. In contrast, the KPSS Lagrange Multiplier test evaluates the null of H_0: $\rho < 1$ against the alternative H_1: $\rho = 1$.

The Augmented Dickey-Fuller (ADF) Test

The standard DF test is carried out by estimating Equation (1.1) after subtracting y_{t-1} from both sides of the equation:

$$\Delta y_t = \alpha y_{t-1} + x_t'\delta + \epsilon_t, \tag{1.2}$$

where $\alpha = \rho - 1$. The null and alternative hypotheses may be written as

$$\begin{aligned} H_0&: \alpha = 0 \\ H_1&: \alpha < 0 \end{aligned} \tag{1.3}$$

and evaluated using the conventional t-ratio for α:

$$t_\alpha = \hat{\alpha}/(se(\hat{\alpha})) \tag{1.4}$$

where $\hat{\alpha}$ is the estimate of α, and $se(\hat{\alpha})$ is the coefficient standard error.

Dickey and Fuller (1979) show that under the null hypothesis of a unit root, this statistic does not follow the conventional Student's t-distribution, and they derive asymptotic results and simulate critical values for various test and sample sizes. More recently, MacKinnon (1991, 1996) implements a much larger set of simulations than those tabulated by Dickey and Fuller. In addition, MacKinnon estimates response surfaces for the simulation results, permitting the calculation of Dickey-Fuller critical values and p-values for arbitrary sample sizes. The more recent MacKinnon critical value calculations are used by EViews in constructing test output.

The simple Dickey-Fuller unit root test described above is valid only if the series is an AR(1) process. If the series is correlated at higher order lags, the assumption of white noise disturbances ϵ_t is violated. The Augmented Dickey-Fuller (ADF) test constructs a parametric correction for higher-order correlation by assuming that the y series follows an AR(p) process and adding p lagged difference terms of the dependent variable y to the right-hand side of the test regression:

$$\Delta y_t = \alpha y_{t-1} + x_t'\delta + \beta_1 \Delta y_{t-1} + \beta_2 \Delta y_{t-2} + \ldots + \beta_p \Delta y_{t-p} + v_t \,. \tag{1.5}$$

This augmented specification is then used to test (1.3) using the t-ratio (1.4). An important result obtained by Fuller is that the asymptotic distribution of the t-ratio for α is independent of the number of lagged first differences included in the ADF regression. Moreover, while the assumption that y follows an autoregressive (AR) process may seem restrictive, Said and Dickey (1984) demonstrate that the ADF test is asymptotically valid in the presence of a moving average (MA) component, provided that sufficient lagged difference terms are included in the test regression.

You will face two practical issues in performing an ADF test. First, you must choose whether to include exogenous variables in the test regression. You have the choice of including a constant, a constant and a linear time trend, or neither, in the test regression. One approach would be to run the test with both a constant and a linear trend since the other two cases are just special cases of this more general specification. However, including irrelevant regressors in the regression will reduce the power of the test to reject the null of a unit root. The standard recommendation is to choose a specification that is a plausible description of the data under both the null and alternative hypotheses. See, Hamilton (1994a, p. 501) for discussion.

Second, you will have to specify the number of lagged difference terms (which we will term the "lag length") to be added to the test regression (0 yields the standard DF test; integers greater than 0 correspond to ADF tests). The usual (though not particularly useful) advice is to include a number of lags sufficient to remove serial correlation in the residuals. EViews provides both automatic and manual lag length selection options. For details, see "Automatic Bandwidth and Lag Length Selection" beginning on page 16.

Dickey-Fuller Test with GLS Detrending (DFGLS)

As noted above, you may elect to include a constant, or a constant and a linear time trend, in your ADF test regression. For these two cases, ERS (1996) propose a simple modification of the ADF tests in which the data are detrended so that explanatory variables are "taken out" of the data prior to running the test regression.

ERS define a quasi-difference of y_t that depends on the value a representing the specific point alternative against which we wish to test the null:

$$d(y_t|a) = \begin{cases} y_t & \text{if } t = 1 \\ y_t - a y_{t-1} & \text{if } t > 1 \end{cases} \tag{1.6}$$

Next, consider an OLS regression of the quasi-differenced data $d(y_t|a)$ on the quasi-differenced $d(x_t|a)$:

$$d(y_t|a) = d(x_t|a)'\delta(a) + \eta_t \tag{1.7}$$

where x_t contains either a constant, or a constant and trend, and let $\hat{\delta}(a)$ be the OLS estimates from this regression.

All that we need now is a value for a. ERS recommend the use of $a = \bar{a}$, where

$$\bar{a} = \begin{cases} 1 - 7/T & \text{if } x_t = \{1\} \\ 1 - 13.5/T & \text{if } x_t = \{1, t\} \end{cases} \tag{1.8}$$

We now define the *GLS detrended data,* y_t^d using the estimates associated with the $\bar{a}$:

$$y_t^d \equiv y_t - x_t'\hat{\delta}(\bar{a}) \tag{1.9}$$

Then the DFGLS test involves estimating the standard ADF test equation, (1.5), after substituting the GLS detrended y_t^d for the original y_t:

$$\Delta y_t^d = \alpha y_{t-1}^d + \beta_1 \Delta y_{t-1}^d + \ldots + \beta_p y_{t-p}^d + v_t \tag{1.10}$$

Note that since the y_t^d are detrended, we do not include the x_t in the DFGLS test equation. As with the ADF test, we consider the t-ratio for $\hat{\alpha}$ from this test equation.

While the DFGLS t-ratio follows a Dickey-Fuller (no constant) distribution in the constant only case, the asymptotic distribution differs when you include both a constant and trend. ERS (1996, Table 1, p. 825) simulate the critical values of the test statistic in this latter setting for $T = \{50, 100, 200, \infty\}$. Thus, the EViews lower tail critical values use the MacKinnon simulations for the constant only case, but are interpolated from the ERS simulated values for the constant and trend case. The null hypothesis is rejected for values that fall below these critical values.

The Phillips-Perron (PP) Test

Phillips and Perron (1988) propose an alternative (nonparametric) method of controlling for serial correlation when testing for a unit root. The PP method estimates the non-augmented DF test equation (1.2), and modifies the t-ratio of the α coefficient so that serial correlation does not affect the asymptotic distribution of the test statistic. The PP test is based on the statistic:

$$\tilde{t}_\alpha = t_\alpha\left(\frac{\gamma_0}{f_0}\right)^{1/2} - \frac{T(f_0-\gamma_0)(se(\hat{\alpha}))}{2f_0^{1/2}s} \tag{1.11}$$

where $\hat{\alpha}$ is the estimate, and t_α the t-ratio of α, $se(\hat{\alpha})$ is coefficient standard error, and s is the standard error of the test regression. In addition, γ_0 is a consistent estimate of the error variance in (1.2) (calculated as $(T-k)s^2/T$, where k is the number of regressors). The remaining term, f_0, is an estimator of the residual spectrum at frequency zero.

There are two choices you will have make when performing the PP test. First, you must choose whether to include a constant, a constant and a linear time trend, or neither, in the test regression. Second, you will have to choose a method for estimating f_0. EViews supports estimators for f_0 based on kernel-based sum-of-covariances, or on autoregressive spectral density estimation. See "Frequency Zero Spectrum Estimation" beginning on page 14 for details.

The asymptotic distribution of the PP modified t-ratio is the same as that of the ADF statistic. EViews reports MacKinnon lower-tail critical and p-values for this test.

The Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) Test

The KPSS (1992) test differs from the other unit root tests described here in that the series y_t is assumed to be (trend-) stationary under the null. The KPSS statistic is based on the the residuals from the OLS regression of y_t on the exogenous variables x_t:

$$y_t = x_t'\delta + u_t \tag{1.12}$$

The LM statistic is be defined as:

$$LM = \sum_t S(t)^2/(T^2 f_0) \tag{1.13}$$

where f_0, is an estimator of the residual spectrum at frequency zero and where $S(t)$ is a cumulative residual function:

$$S(t) = \sum_{r=1}^{t} \hat{u}_r \tag{1.14}$$

based on the residuals $\hat{u}_t = y_t - x_t'\hat{\delta}(0)$. We point out that the estimator of δ used in this calculation differs from the estimator for δ used by GLS detrending since it is based on a regression involving the original data, and not on the quasi-differenced data.

To specify the KPSS test, you must specify the set of exogenous regressors x_t and a method for estimating f_0 .

The reported critical values for the LM test statistic are based upon the asymptotic results presented in KPSS (Table 1, p. 166).

Elliot, Rothenberg, and Stock Point Optimal (ERS) Test

The ERS Point Optimal test is based on the quasi-differencing regression defined in Equations (1.7). Define the residuals from (1.7) as $\hat{\eta}_t(a) = d(y_t|a) - d(x_t|a)'\hat{\delta}(a)$, and let $\text{SSR}(a) = \sum \hat{\eta}_t^2(a)$ be the sum-of-squared residuals function. The ERS (feasible) point optimal test statistic of the null that $\alpha = 1$ against the alternative that $\alpha = \bar{a}$, is then defined as

$$P_T = (\text{SSR}(\bar{a}) - \bar{a}\text{SSR}(1))/f_0 \tag{1.15}$$

where f_0 , is an estimator of the residual spectrum at frequency zero.

To compute the ERS test you must specify the set of exogenous regressors x_t and a method for estimating f_0 .

Critical values for the ERS test statistic are computed by interpolating the simulation results provided by ERS (1996, Table 1, p. 825) for $T = \{50, 100, 200, \infty\}$.

Ng and Perron (NP) Tests

Ng and Perron (2001) construct four test statistics that are based upon the GLS detrended data y_t^d . These test statistics are modified forms of Phillips and Perron Z_α and Z_t statistics, the Bhargava (1986) R_1 statistic, and the ERS Point Optimal statistic. First, define the term:

$$\kappa = \sum_{t=2}^{T} (y_{t-1}^d)^2/T^2 \tag{1.16}$$

The GLS-detrended modified statistics may then be written as

$$\begin{aligned} MZ_\alpha^d &= (T^{-1}(y_T^d)^2 - f_0)/(2\kappa) \\ MZ_t^d &= MZ_\alpha \times MSB \\ MSB^d &= (\kappa/f_0)^{1/2} \\ MP_T^d &= \begin{cases} (\bar{c}^2\kappa - \bar{c}T^{-1}(y_T^d)^2)/f_0 & \text{if } x_t = \{1\} \\ (\bar{c}^2\kappa + (1-\bar{c})T^{-1}(y_T^d)^2)/f_0 & \text{if } x_t = \{1, t\} \end{cases} \end{aligned} \tag{1.17}$$

where

$$\bar{c} = \begin{cases} -7 & \text{if } x_t = \{1\} \\ -13.5 & \text{if } x_t = \{1, t\} \end{cases} \tag{1.18}$$

The NP tests require a specification for x_t and a choice of method for estimating f_0

Frequency Zero Spectrum Estimation

Many of the unit root tests described above require a consistent estimate of the residual spectrum at frequency zero. EViews supports two classes of estimators for f_0: kernel-based sum-of-covariances estimators, and autoregressive spectral density estimators.

Kernel Sum-of-Covariances Estimation

The kernel-based estimator of the frequency zero spectrum is based on a weighted sum of the autocovariances, with the weights are defined by a kernel function. The estimator takes the form

$$\hat{f}_0 = \sum_{j=-(T-1)}^{T-1} \hat{\gamma}(j) \cdot K(j/l) \tag{1.19}$$

where l is a bandwidth parameter, K is a kernel function, and where $\hat{\gamma}(j)$, the j-th sample autocovariance of the residuals $\tilde{u}_t$, is defined as

$$\hat{\gamma}(j) = \sum_{t=j+1}^{T} (\tilde{u}_t \tilde{u}_{t-j})/T \tag{1.20}$$

Note that the residuals $\tilde{u}_t$ that EViews uses in estimating the autocovariance functions in (1.20) will differ depending on the specified unit root test:

Unit root test	Source of $\tilde{u}_t$ residuals for kernel estimator
ADF, DFGLS	*not applicable.*
PP, ERS Point Optimal, NP	residuals from the Dickey-Fuller test equation, (1.2).
KPSS	residuals from the OLS test equation, (1.12).

EViews supports the following kernel functions:

Bartlett:	$K(x) = \begin{cases} 1-\lvert x\rvert & \text{if } \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Parzen:	$K(x) = \begin{cases} 1-6x^2+6\lvert x\rvert^3 & \text{if } 0 \le \lvert x\rvert \le (1/2) \\ 2(1-\lvert x\rvert)^3 & \text{if } (1/2) < \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Quadratic Spectral	$K(x) = \frac{25}{12\pi^2 x^2}\left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5)\right)$

The properties of these kernels are described in Andrews (1991).

As with most kernel estimators, the choice of the bandwidth parameter l is of considerable importance. EViews allows you to specify a fixed parameter, or to have EViews select one using a data-dependent method. Automatic bandwidth parameter selection is discussed in "Automatic Bandwidth and Lag Length Selection" beginning on page 16.

Autoregressive Spectral Density Estimator

The autoregressive spectral density estimator at frequency zero is based upon the residual variance and estimated coefficients from the auxiliary regression:

$$\Delta \tilde{y}_t = \alpha \tilde{y}_{t-1} + \varphi \cdot \tilde{x}_t'\delta + \beta_1 \Delta \tilde{y}_{t-1} + \ldots + \beta_p \Delta \tilde{y}_{t-p} + u_t \tag{1.21}$$

EViews provides three autoregressive spectral methods: OLS, OLS detrending, and GLS detrending, corresponding to difference choices for the data $\tilde{y}_t$. The following table summarizes the auxiliary equation estimated by the various AR spectral density estimators:

AR spectral method	Auxiliary AR regression specification
OLS	$\tilde{y}_t = y_t$, and $\varphi = 1$, $\tilde{x}_t = x_t$.
OLS detrended	$\tilde{y}_t = y_t - x_t'\hat{\delta}(0)$, and $\varphi = 0$.
GLS detrended	$\tilde{y}_t = y_t - x_t'\hat{\delta}(\bar{a}) = y_t^d$. and $\varphi = 0$.

where $\hat{\delta}(a)$ are the coefficient estimates from the regression defined in (1.7).

The AR spectral estimator of the frequency zero spectrum is defined as:

$$\hat{f}_0 = \hat{\sigma}_u^2/(1-\hat{\beta}_1-\hat{\beta}_2-\ldots-\hat{\beta}_p) \tag{1.22}$$

where $\hat{\sigma}_u^2 = \sum \tilde{u}_t^2 / T$ is the residual variance, and $\hat{\beta}$ are the estimates from (1.21). We note here that EViews uses the non-degree-of-freedom estimator of the residual variance. As a result, spectral estimates computed in EViews may differ slightly from those obtained from other sources. Moreover, the estimator of the variance differs conceptually from the estimator s used in computing the t-ratios used in ADF and PP test statistics.

Not surprisingly, the spectrum estimator is sensitive to the number of lagged difference terms in the auxiliary equation. You may either specify a fixed parameter, or have EViews automatically select one based on an information criterion. Automatic lag length selection is examined in "Automatic Bandwidth and Lag Length Selection" on page 16.

Default f_0 Estimation Settings

By default, EViews will choose the estimator of f_0 used by the authors of a given test specification. You may, of course, override the default settings and choose from either family of estimation methods. The default settings are listed below:

Unit root test	Frequency zero spectrum default method
ADF, DFGLS	*not applicable*
PP, KPSS	Kernel (Bartlett) sum-of-covariances
ERS Point Optimal	AR spectral regression (OLS)
NP	AR spectral regression (GLS-detrended)

Automatic Bandwidth and Lag Length Selection

There are three distinct situations in which EViews can automatically compute a bandwidth or a lag length parameter.

The first situation occurs when you are selecting the bandwidth parameter l for the kernel-based estimators of f_0. For the kernel estimators, EViews provides you with the option of using the Newey-West (1994) or the Andrews (1991) data-based automatic bandwidth parameter methods. For those familiar with the Newey-West procedure, we note that EViews uses the lag selection parameter formulae given in the corresponding first lines of Table II-C. The Andrews method is based on an AR(1) specification. See the original sources for details.

The latter two situations occur when the unit root test requires estimation of a regression with a parametric correction for serial correlation as in the ADF and DFGLS test equation regressions, and in the AR spectral estimator for f_0. In both of these settings, p lagged difference terms are added to a regression equation. The automatic selection methods choose p (less than the specified maximum) to minimize one of the following criteria:

Information criterion	Definition
Akaike (AIC)	$-2(l/T) + 2k/T$
Schwarz (SIC)	$-2(l/T) + k\log(T)/T$
Hannan-Quinn (HQ)	$-2(l/T) + 2k\log(\log(T))/T$
Modified AIC (MAIC)	$-2(l/T) + 2(k+\tau)/T$
Modified SIC (MSIC)	$-2(l/T) + (k+\tau)\log(T)/T$
Modified Hannan-Quinn (MHQ)	$-2(l/T) + 2(k+\tau)\log(\log(T))/T$

where the modification factor τ is computed as

$$\tau = \alpha^2 \sum_t \tilde{y}_{t-1}^2 / \hat{\sigma}_u^2 \tag{1.23}$$

for $\tilde{y}_t = y_t$, when computing the ADF test equation, and for $\tilde{y}_t$ as defined in "Autoregressive Spectral Density Estimator" on page 15, when estimating f_0. NP (2001) propose and examine the modified criteria, concluding with a recommendation of the MAIC.

For the information criterion selection methods, you must also specify an upper bound to the lag length. By default, EViews chooses a maximum lag of

$$k_{\max} = \text{int}(12(T/100)^{1/4}) \tag{1.24}$$

but you may substitute any positive integer value.

See Hayashi (2000, p. 594) for a discussion of the selection of this upper bound.

Chapter 2. System Instrumental Variables

In EViews 4.1, the new `@stackinst` statement provides a new way of specifying instruments for systems of equations that allows for cross-equations on the projections of variables on instruments. Previously, all instrumental variables projections in systems were performed on an equation-by-equation basis. You can now stack your equations and instruments prior to performing the projection. The easy-to-use syntax provides you with full control over the instrument stacking so that you may combine the earlier ordinary instrument and new stacked instrument specifications.

Specifying Instruments in Systems

If you plan to estimate your system using two-stage least squares, three-stage least squares, or GMM, you must specify the instrumental variables to be used in estimation. There are several ways to specify your instruments, with the appropriate form depending on whether you wish to have identical instruments in each equation, and whether you wish to compute the projections on an equation-by-equation basis, or whether you wish to compute a restricted projection using the stacked system.

In the simplest (default) case, EViews will form your instrumental variable projections on an equation-by-equation basis. If you prefer to think of this process as a two-step (2SLS) procedure, the first-stage regression of the variables in your model on the instruments will be run separately for each equation.

In this setting, there are two ways to specify your instruments. If you would like to use identical instruments in every equations, you should include a line beginning with the keyword "@INST" or "INST", followed by a list of all the exogenous variables to be used as instruments. For example, the line

```
@inst gdp(-1 to -4) x gov
```

instructs EViews to use these six variables as instruments for all of the equations in the system. System estimation will involve a separate projection for each equation in your system.

You may also specify different instruments for each equation by appending an "@"-sign at the end of the equation, followed by a list of instruments for that equation. For example,

```
cs = c(1)+c(2)*gdp+c(3)*cs(-1) @ cs(-1) inv(-1) gov
inv = c(4)+c(5)*gdp+c(6)*gov @ gdp(-1) gov
```

The first equation uses CS(-1), INV(-1), GOV, and a constant as instruments, while the second equation uses GDP(-1), GOV, and a constant as instruments.

Lastly, you can mix the two methods. Any equation without individually specified instruments will use the instruments specified by the `@inst` statement. The system

```
@inst gdp(-1 to -4) x gov
cs = c(1)+c(2)*gdp+c(3)*cs(-1)
inv = c(4)+c(5)*gdp+c(6)*gov @ gdp(-1) gov
```

will use the instruments GDP(-1), GDP(-2), GDP(-3), GDP(-4), X, GOV, and C, for the CS equation, but only GDP(-1), GOV, and C, for the INV equation.

As noted above, the EViews default behavior is to perform the instrumental variables projection on an equation-by-equation basis. You may, however, wish to perform the projections on the stacked system. Notably, where the number of instruments is large, relative to the number of observations, stacking the equations and instruments prior to performing the projection may be the only feasible way to compute 2SLS estimates.

To designate instruments for a stacked projection, you should use the `@stackinst` statement (note: this statement is only available for systems estimated by 2SLS or 3SLS; it is not available for systems estimated using GMM).

In a `@stackinst` statement, the "@STACKINST" keyword should be followed by a list of stacked instrument specifications. Each specification is a comma delimited list of series enclosed in parentheses (one per equation), describing the instruments to be constrained in a stacked specification.

For example, the following `@stackinst` specification creates two instruments in a three equation model:

```
@stackinst (z1,z2,z3) (m1,m1,m1)
```

This statement instructs EViews to form two stacked instruments, one by stacking the separate series Z1, Z2, and Z3, and the other formed by stacking M1 three times. The first-stage instrumental variables projection is then of the variables in the stacked system on the stacked instruments.

When working with systems that have a large number of equations, the above syntax may be unwieldy. For these cases, EViews provides a couple of shortcuts. First, for instruments that are identical in all equations, you may us an "*" after the comma to instruct EViews to repeat the specified series. Thus, the above statement is equivalent to

```
@stackinst (z1,z2,z3) (m1,*)
```

Second, for non-identical instruments, you may specify a set of stacked instruments using an EViews group object, so long as the number of variables in the group is equal to the number of equations in the system. Thus, if you create a group Z with

```
group z z1 z2 z3
```

the above statement can be simplified to:

```
@stackinst z (m1,*)
```

You can, of course, combine ordinary instrument and stacked instrument specifications. This situation is equivalent to having common and equation specific coefficients for variables in your system. Simply think of the stacked instruments as representing common (coefficient) instruments, and ordinary instruments as representing equation specific (coefficient) instruments. For example, consider the system given by

```
@stackinst (z1,z2,z3) (m1,*)
@inst ia
y1 = c(1)*x1
y2 = c(1)*x2
y3 = c(1)*x3 @ ic
```

The stacked instruments for this specification may be represented as:

$$\begin{bmatrix} Z1 & M1 & IA & 0 & 0 & 0 \\ Z2 & M1 & 0 & IA & 0 & 0 \\ Z3 & M1 & 0 & 0 & IA & IC \end{bmatrix} \tag{2.1}$$

so it is easy to see that this specification is equivalent to the following stacked specification

```
@stackinst (z1, z2, z3) (m1, *) (ia, 0, 0) (0, ia, 0) (0, 0, ia)
    (0, 0, ic)
```

since the common instrument specification

```
@inst ia
```

is equivalent to

```
@stackinst (ia, 0, 0) (0, ia, 0) (0, 0, ia)
```

Additional Comments

- If you include a “C” in the stacked instrument list, it will not be included in the individual equations. If you do not include the “C” as a stacked instrument, it will be included as an instrument in every equation, whether specified explicitly or not.

- You should list all exogenous right-hand side variables as instruments for a given equation.
- Identification requires that there should be at least as many instruments (including the constant) in each equation as there are right-hand side variables in that equation.
- The `@stackinst` statement is only available for estimation by 2SLS and 3SLS. It is not currently supported for GMM.
- If you estimate your system using a method that does not use instruments, all instrument specification lines will be ignored.

Chapter 3. State Space Modeling

EViews 4.1 extends the features of the sspace object in two distinct ways.

First, extensions to the state space syntax in EViews 4.1 allow you to write the error term for any equation as a linear combination of named errors. This syntax allows users to specify, more naturally, a wide range of important models.

Second, new sspace object data members give you access to output matrices containing intermediate calculations from the Kalman filter.

Specifying Errors and Variances in a Sspace

While EViews always adds an implicit error term to each equation in an equation or system object, the handling of error terms differs in a sspace object. In a sspace object, the equation specifications in a signal or state equation do not contain error terms unless specified explicitly.

The easiest way to add an error to a state space equation is to specify an implied error term using its variance. You can simply add an error variance expression, consisting of the keyword "VAR" followed by an assignment statement (all enclosed in square brackets), to the existing equation:

```
@signal y = c(1) + sv1 + sv2 + [var = 1]
@state sv1 = sv1(-1) + [var = exp(c(2))]
@state sv2 = c(3) + c(4)*sv2(-1) + [var = exp(c(2)*x)]
```

The specified variance may be a known constant value, or it can be an expression containing unknown parameters to be estimated. You may also build time-variation into the variances using a series expression. Variance expressions may not, however, contain state or signal variables.

While straightfoward, this direct variance specification method does not admit correlation between errors in different equations (by default, EViews assumes that the covariance between error terms is 0). If you require a more flexible variance structure, you will need to use the "named error" approach to define named errors with variances and covariances, and then to use these named errors as parts of expressions in the signal and state equations.

The first step of this general approach is to define your named errors. You may declare a named error by including a line with the keyword "@ENAME" followed by the name of the error:

```
@ename e1
@ename e2
```

Once declared, a named error may enter linearly into state and signal equations. In this manner, one can build correlation between the equation errors. For example, the errors in the state and signal equations in

```
y = c(1) + sv1*x1 + e1
@state sv1 = sv1(-1) + e2 + c(2)*e1
@ename e1
@ename e2
```

are, in general, correlated since the named error E1 appears in both equations.

In the special case where a named error is the only error in a given equation, you can both declare and use the named residual by adding an error expression consisting of keyword "ENAME" followed by an assignment and a name identifier.

```
y = c(1) + sv1*x1 + [ename = e1]
@state sv1 = sv1(-1) + [ename = e2]
```

The final step in building a general error structure is to define the variances and covariances associated with your named errors. You should include a sspace line comprised of the keyword "@EVAR" followed by an assignment statement for the variance of the error or the covariance between two errors:

```
@evar cov(e1, e2) = c(2)
@evar var(e1) = exp(c(3))
@evar var(e2) = exp(c(4))*x
```

The syntax for the @EVAR assignment statements should be self-explanatory. Simply indicate whether the term is a variance or covariance, identify the error(s), and enter the specification for the variance or covariance. There should be a separate line for each named error covariance or variance that you wish to specify. If an error term is named, but there are no corresponding "VAR=" or @EVAR specifications, the missing variance or covariance specifications will remain at the default values of "NA" and "0", respectively.

As you might expect, in the special case where an equation contains a single error term, you may combine the named error and direct variance assignment statements:

```
@state sv1 = sv1(-1) + [ename = e1, var = exp(c(3))]
@state sv2 = sv2(-1) + [ename = e2, var = exp(c(4))]
@evar cov(e1, e2) = c(5)
```

Accessing Sspace Filter and Smoother Results

The following functions allow you to extract the filter and smoother results for the estimation sample and place them in matrix objects. In some cases, the results overlap those available thorough the sspace procs, while in other cases, the matrix results are the only way to obtain the results.

Note also that since the computations are only for the estimation sample, the one-step-ahead predicted state and state standard error values *will not* match the final values displayed in the estimation output. The latter are the predicted values for the first out-of-estimation sample period.

@pred_signal.....matrix or vector of one-step ahead predicted signals.

@pred_signalcov...matrix where every row is the @vech of the one-step ahead predicted signal covariance.

@pred_signalse.....matrix or vector of the standard errors of the one-step ahead predicted signals.

@pred_err .....matrix or vector of one-step ahead prediction errors.

@pred_errcov .....matrix where every row is the @vech of the one-step ahead prediction error covariance.

@pred_errcovinv ..matrix where every row is the @vech of the inverse of the one-step ahead prediction error covariance.

@pred_errse .....matrix or vector of the standard errors of the one-step ahead prediction errors.

@pred_errstd .....matrix or vector of standardized one-step ahead prediction errors.

@pred_state.....matrix or vector of one-step ahead predicted states.

@pred_statecov.....matrix where each row is the @vech of the one-step ahead predicated state covariance.

@pred_statese.....matrix or vector of the standard errors of the one-step ahead predicted states.

@pred_stateerr .....matrix or vector of one-step ahead predicted state errors.

@curr_err.....matrix or vector of filtered error estimates.

@curr_gain.....matrix or vector where each row is the @vec of the Kalman gain.

@curr_state .....matrix or vector of filtered states.

@curr_statecov .....matrix where every row is the @vech of the filtered state covariance.

@curr_statese .....matrix or vector of the standard errors of the filtered state estmates.

@sm_signal .....matrix or vector of smoothed signal estimates.

@sm_signalcov .... matrix where every row is the @vech of the smoothed signal covariance.

@sm_signalse ..... matrix or vector of the standard errors of the smoothed signals.

@sm_signalerr ..... matrix or vector of smoothed signal error estimates.

@sm_signalerrcov matrix where every row is the @vech of the smoothed signal error covariance.

@sm_signalerrse .. matrix or vector of the standard errors of the smoothed signal error.

@sm_signalerrstd. matrix or vector of the standardized smoothed signal errors.

@sm_state ..... matrix or vector of smoothed states.

@sm_statecov ..... matrix where each row is the @vech of the smoothed state covariances.

@sm_statese ..... matrix or vector of the standard errors of the smoothed state.

@sm_stateerr ..... matrix or vector of the smoothed state errors.

@sm_stateerrcov .. matrix where each row is the @vech of the smoothed state error covariance.

@sm_stateerrse .... matrix or vector of the standard errors of the smoothed state errors.

@sm_stateerrstd... matrix or vector of the standardized smoothed state errors .

@sm_crosserrcov . matrix where each row is the @vec of the smoothed error cross-covariance.

Sspace Examples

The one-step-ahed state values and variances from SS01 may be saved using

```
vector ss_state=ss01.@pred_state
matrix ss_statecov=ss01.@pred_statecov
```

Chapter 4. Miscellaneous Features

Extended Model Commands

EViews 4.1 extends the set of modelling tools to aid you in model building and solution:

- New options features provide greater control over the model solution procedure. You can now initialize solution values and excluded variables, control terminal conditions for forward solution, and set roundoff conditions from the command line. See solveopt (p. 39).
- Expanded programming language support has been provided for working with scenarios. Updated documentation and notes regarding backward compatibility are provided in the entry for scenario (p. 37).
- You may now unlink selected dependent variable links using commands. For details, see unlink (p. 42).
- EViews 4.1 now allows you to force recompilation of a model and the updating of all model and equation links from the command line. See update (p. 43).

Enhanced Wald Tests

The Wald test computes a test statistic based on the unrestricted regression. The Wald statistic measures how close the unrestricted estimates come to satisfying the restrictions under the null hypothesis. If the restrictions are in fact true, then the unrestricted estimates should come close to satisfying the restrictions.

The output of the Wald test has been expanded to provide additional information about the restrictions and the restriction variances. As a result, you can use the expanded output to find the standard errors of functions of your coefficients.

How to Perform Wald Coefficient Tests

To demonstrate the calculation of Wald tests in EViews, we consider simple examples. Suppose a Cobb-Douglas production function has been estimated in the form:

$$\log Q = A + \alpha \log L + \beta \log K + \epsilon \,, \tag{4.1}$$

where Q, K and L denote value-added output and the inputs of capital and labor respectively. The hypothesis of constant returns to scale is then tested by the restriction: $\alpha + \beta = 1$.

Estimation of the Cobb-Douglas production function using annual data from 1947 to 1971 provided the following result:

Dependent Variable: LOG(Q)
Method: Least Squares
Date: 08/11/97 Time: 16:56
Sample: 1947 1971
Included observations: 25

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	-2.327939	0.410601	-5.669595	0.0000
LOG(L)	1.591175	0.167740	9.485970	0.0000
LOG(K)	0.239604	0.105390	2.273498	0.0331

R-squared	0.983672	Mean dependent var	4.767586
Adjusted R-squared	0.982187	S.D. dependent var	0.326086
S.E. of regression	0.043521	Akaike info criterion	-3.318997
Sum squared resid	0.041669	Schwarz criterion	-3.172732
Log likelihood	44.48746	F-statistic	662.6819
Durbin-Watson stat	0.637300	Prob(F-statistic)	0.000000

The sum of the coefficients on LOG(L) and LOG(K) appears to be in excess of one, but to determine whether the difference is statistically relevant, we will conduct the hypothesis test of constant returns.

To carry out a Wald test, choose **View/Coefficient Tests/Wald-Coefficient Restrictions...** from the equation toolbar. Enter the restrictions into the edit box, with multiple coefficient restrictions separated by commas. The restrictions should be expressed as equations involving the estimated coefficients and constants (you may not include series names). The coefficients should be referred to as C(1), C(2), and so on, unless you have used a different coefficient vector in estimation.

To test the hypothesis of constant returns to scale, type the following restriction in the dialog box:

```
c(2) + c(3) = 1
```

and click **OK**. EViews reports the following result of the Wald test:

Wald Test:
Equation: EQ1

Test Statistic	Value	df	Probability
Chi-square	120.0177	1	0.0000
F-statistic	120.0177	(1, 22)	0.0000

Null Hypothesis Summary:

Normalized Restriction (= 0)	Value	Std. Err.
-1 + C(2) + C(3)	0.830779	0.075834

Restrictions are linear in coefficients.

EViews reports an F-statistic and a Chi-square statistic with associated p-values. See "Wald Test Details" on page 30 for a discussion of these statistics. In addition, EViews reports the value of the normalized (homogeneous) restriction and an associated standard error. In this example, we have a single linear restriction so the two test statistics are identical, with the p-value indicating that we can decisively reject the null hypothesis of constant returns to scale.

To test more than one restriction, separate the restrictions by commas. For example, to test the hypothesis that the elasticity of output with respect to labor is 2/3 and the elasticity with respect to capital is 1/3, enter the restrictions as

```
c(2)=2/3, c(3)=1/3
```

and EViews reports

Wald Test:
Equation: EQ1

Test Statistic	Value	df	Probability
Chi-square	53.99105	2	0.0000
F-statistic	26.99553	(2, 22)	0.0000

Null Hypothesis Summary:

Normalized Restriction (= 0)	Value	Std. Err.
-2/3 + C(2)	0.924508	0.167740
-1/3 + C(1)	-2.661272	0.410601

Restrictions are linear in coefficients.

Note that in addition to the test statistic summary, we report the values of both of the normalized restrictions, along with their standard errors (the square roots of the diagonal elements of the restriction covariance matrix).

As an example of a nonlinear model with a nonlinear restriction, we estimate a production function of the form

$$\log Q = \beta_1 + \beta_2 \log(\beta_3 K^{\beta_4} + (1-\beta_3) L^{\beta_4}) + \epsilon \tag{4.2}$$

and test the constant elasticity of substitution (CES) production function restriction $\beta_2 = 1/\beta_4$. This is an example of a nonlinear restriction. To estimate the (unrestricted) nonlinear model, you should select **Quick/Estimate Equation…** and then enter the following specification:

```
log(q) = c(1) + c(2)*log(c(3)*k^c(4)+(1-c(3))*l^c(4))
```

To test the nonlinear restriction, choose **View/Coefficient Tests/Wald-Coefficient Restrictions…** from the equation toolbar and type the following restriction in the Wald Test dialog box:

```
c(2)=1/c(4)
```

The results are presented below:

Wald Test:
Equation: EQ2

Test Statistic	Value	df	Probability
Chi-square	0.028508	1	0.8659
F-statistic	0.028508	(1, 21)	0.8675

Null Hypothesis Summary:

Normalized Restriction (= 0)	Value	Std. Err.
C(2) - 1/C(4)	1.292163	7.653088

Delta method computed using analytic derivatives.

Since this is a nonlinear equation, we focus on the Chi-square statistic which fails to reject the null hypothesis. Note that EViews reports that it used the delta method (with analytic derivatives) to compute the Wald restriction variance for the nonlinear restriction.

It is well-known that nonlinear Wald tests are not invariant to the way that you specify the nonlinear restrictions. In this example, the nonlinear restriction $\beta_2 = 1/\beta_4$ may equivalently be written as $\beta_2\beta_4 = 1$ or $\beta_4 = 1/\beta_2$ (for nonzero β_2 and β_4). For example, entering the restriction as

```
c(2)*c(4)=1
```

yields:

Wald Test:
Equation: EQ2

Test Statistic	Value	df	Probability
Chi-square	104.5599	1	0.0000
F-statistic	104.5599	(1, 21)	0.0000

Null Hypothesis Summary:

Normalized Restriction (= 0)	Value	Std. Err.
-1 + C(2)*C(4)	0.835330	0.081691

Delta method computed using analytic derivatives.

so that the test now decisively rejects the null hypothesis. We hasten to add that type of inconsistency is not unique to EViews, but is a more general property of the Wald test. Unfortunately, there does not seem to be a general solution to this problem (see Davidson and MacKinnon, 1993, Chapter 13).

Wald Test Details

Consider a general nonlinear regression model

$$y = f(\beta) + \epsilon \tag{4.3}$$

where y and ϵ are T-vectors and β is a k-vector of parameters to be estimated. Any restrictions on the parameters can be written as

$$H_0\colon\ g(\beta) = 0\,, \tag{4.4}$$

where g is a smooth function, $g\colon R^k \to R^q$, imposing q restrictions on β. The Wald statistic is then computed as

$$W = g(\beta)'\left(\frac{\partial g(\beta)}{\partial \beta}\hat{V}(b)\frac{\partial g(\beta)}{\partial \beta'}\right)g(\beta)|_{\beta = b} \tag{4.5}$$

where T is the number of observations and b is the vector of unrestricted parameter estimates, and where $\hat{V}$ is an estimate of the b covariance. In the standard regression case, $\hat{V}$ is given by

$$\hat{V}(b) = s^2\left(\frac{\partial f(\beta)}{\partial \beta}\frac{\partial f(\beta)}{\partial \beta'}\right)^{-1}\Bigg|_{\beta = b} \tag{4.6}$$

where u is the vector of unrestricted residuals, and s^2 is the usual estimator of the unrestricted residual variance, $s^2 = (u'u)/(N-k)$, but the estimator of V may differ. For example, $\hat{V}$ may be a robust variance matrix estimator computing using White or Newey-West techniques.

More formally, under the null hypothesis H_0, the Wald statistic has an asymptotic $\chi^2(q)$ distribution, where q is the number of restrictions under H_0.

For the textbook case of a linear regression model

$$y = X\beta + \epsilon \tag{4.7}$$

and linear restrictions

$$H_0\colon\ R\beta - r = 0\,, \tag{4.8}$$

where R is a known $q \times k$ matrix, and r is a q-vector, respectively. The Wald statistic in Equation (4.5) reduces to

$$W = (Rb - r)'(Rs^2(X'X)^{-1}R')^{-1}(Rb - r)\,, \tag{4.9}$$

which is asymptotically distributed as $\chi^2(q)$ under H_0.

If we further assume that the errors ϵ are independent and identically normally distributed, we have an exact, finite sample F-statistic:

$$F = \frac{W}{q} = \frac{(\tilde{u}'\tilde{u} - u'u)/q}{(u'u)/(T-k)}, \tag{4.10}$$

where $\tilde{u}$ is the vector of residuals from the restricted regression. In this case, the F-statistic compares the residual sum of squares computed with and without the restrictions imposed.

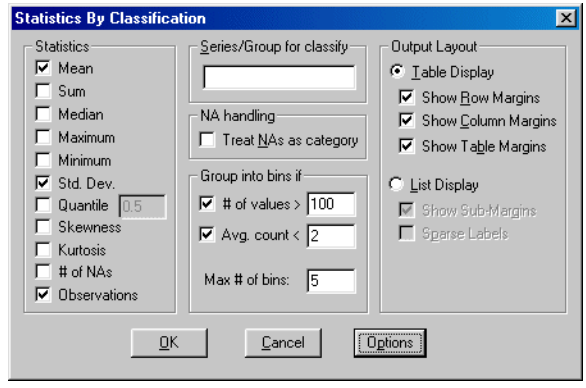
We remind you that the expression for the finite sample F-statistic in (4.10) is for standard linear regression, and is not valid for more general cases (nonlinear models, ARMA specifications, or equations where the variances are estimated using other methods such as Newey-West or White). In non-standard settings, the reported F-statistic (which EViews always computes computes as W/q), does not possess the desired finite-sample properties. In these cases, while asymptotically valid, the F-statistic results should be viewed as illustrative and for comparison purposes only.

Quantiles by Classification

The "Stats by Classification" view of a series now allows you to compute arbitrary quantiles of your data by group. An options menu allows you to choose the quantile definition. Command support for this new calculation is documented in `statby` (p. 40).

Stats by Classification

This view allows you to compute the descriptive statistics of a series for various subgroups of your sample. If you select **View/Descriptive Statistics/Stats by Classification…** a Statistics by Classification dialog box appears:



The **Statistics** option at the left allows you to choose the statistics you wish to compute.

In the **Series/Group for Classify** field enter series or group names that define your subgroups. You must type at least one name. Descriptive statistics will be calculated for each unique value of the classification series unless binning is selected. You may type more than one series or group name; separate each name by a space. The quantile statistic requires an additional argument (a number between 0 and 1) corresponding to the desired quantile value. Click on the options button to choose between various methods of computing the quantiles.

By default, EViews excludes observations which have missing values for any of the classification series. To treat NA values as a valid subgroup, select the **NA handling** option.

The **Layout** option allows you to control the display of the statistics. Table layout arrays the statistics in cells of two-way tables. The list form displays the statistics in a single line for each classification group.

The **Table** and **List** options are only relevant if you use more than one series as a classifier.

The **Sparse Labels** option suppresses repeating labels in list mode to make the display less cluttered.

The **Row Margins**, **Column Margins**, and **Table Margins** instruct EViews to compute statistics for aggregates of your subgroups. For example, if you classify your sample on the basis of gender and age, EViews will compute the statistics for each gender/age combination. If you elect to compute the marginal statistics, EViews will also compute statistics corresponding to each gender, and each age subgroup.

A classification may result in a large number of distinct values with very small cell sizes. By default, EViews automatically groups observations to maintain moderate cell sizes and numbers of categories. **Group into Bins** provides you with control over this process.

Setting the **# of values** option bins tell EViews to group data if the classifier series takes more than the specified number of distinct values.

The **Avg. count** option bins the series if the average count for each distinct value of the classifier series is less than the specified number.

The **Max # of bins** specifies the maximum number of subgroups to bin the series. Note that this number only provides you with approximate control over the number of bins.

The default setting is to bin the series into 5 subgroups if either the series takes more than 100 distinct values or if the average count is less than 2. If you do not want to bin the series, unmark both options.

For example, consider the following stats by classification view in table form:

Descriptive Statistics for LWAGE
Categorized by values of MARRIED and UNION
Date: 10/15/97 Time: 01:11
Sample: 1 1000
Included observations: 1000

Mean Median Std. Dev. Obs.		UNION 0	 1	 All
	0	1.993829 1.906575 0.574636 305	2.387019 2.409131 0.395838 54	2.052972 2.014903 0.568689 359
MARRIED	1	2.368924 2.327278 0.557405 479	2.492371 2.525729 0.380441 162	2.400123 2.397895 0.520910 641
	All	2.223001 2.197225 0.592757 784	2.466033 2.500525 0.386134 216	2.275496 2.302585 0.563464 1000

The header indicates that the table cells are categorized by two series MARRIED and UNION. These two series are dummy variables that take only two values and no binning was made. If the series were binned, intervals rather than a number would be displayed in the margins.

The upper left cell of the table indicates the reported statistics in each cell; in this case the median and the number of observations are reported in each cell. The row and column labeled **All** correspond to the **Row Margin** and **Column Margin** options described above.

Here is the same view in list form with sparse labels:

Descriptive Statistics for LWAGE
Categorized by values of MARRIED and UNION
Date: 10/15/97 Time: 01:08
Sample: 1 1000
Included observations: 1000

UNION	MARRIED	Mean	Median	Std. Dev.	Obs.
0	0	1.993829	1.906575	0.574636	305
	1	2.368924	2.327278	0.557405	479
	All	2.223001	2.197225	0.592757	784
1	0	2.387019	2.409131	0.395838	54
	1	2.492371	2.525729	0.380441	162
	All	2.466033	2.500525	0.386134	216
All	0	2.052972	2.014903	0.568689	359
	1	2.400123	2.397895	0.520910	641
	All	2.275496	2.302585	0.563464	1000

Added Functions

EViews 4.1 now supports a family of percentage change functions that complement the existing functions. These functions eliminate the need to rescale function values when working in percentage terms. In addition, we have added functions for computing base-10 and arbitrary base logarithms.

Time Series Functions

`@pc(x)`	one-period percentage change (in percent)	equals @pch(x)*100
`@pch(x)`	one-period percentage change (in decimal)	$(X - X(-1))/X(-1)$
`@pca(x)`	one-period percentage change—annualized (in percent)	equals @pcha(x)*100
`@pcha(x)`	one-period percentage change—annualized (in decimal)	@pcha(x) $= (1 + @\text{pch}(x))^n - 1$ where n is the lag associated with one-year ($n = 4$) for quarterly data, etc.).
`@pcy(x)`	one-year percentage change (in percent)	equals @pchy(x)*100
`@pchy(x)`	one-year percentage change (in decimal)	$(X - X(-n))/X(-n)$, where n is the lag associated with one-year ($n = 12$) for annual data, etc.).

Basic Mathematical Functions

`@log10(x)`	base-10 logarithm, $\log_{10}(x)$	@log10(100) $= 2$
`@logx(x, b)`	base-*b* logarithm, $\log_b(x)$	@log(256, 2) $= 8$

Chapter 5. Updated Command Entries

The following is an alphabetical listing of the commands, views, and procedures in EViews that have been updated for Version 4.1.

scenario	Model Proc

Manage the model scenarios.

The scenario procedure is used to set the active and comparison scenarios for a model, to create new scenarios, to initialize one scenario with settings from another scenario, to delete scenarios, and to change the variable aliasing associated with a scenario.

Syntax

Model Proc: model_name.**scenario***(options)* "*name*"

performs scenario options on a scenario given by the "name". By default the scenario procedure also sets the active scenario to the specified name.

Options

c	Set the comparison scenario to the named scenario.
n	Create a new scenario with the specified name.
i = "*name*"	Copy the Excludes and Overrides from the named scenario.
d	Delete the named scenario.
a = *string*	Set the scenario alias string to be used when creating aliased variables (*string* is a 1 to 3 alphanumeric string to be used in creating aliased variables). If an underscore is not specified, one will be added to the beginning of the string. Examples: "_5", "_T", "S2". The string "A" may not be used since it may conflict with add factor specifications.

Examples

The command string

```
mod1.scenario "baseline"
```

sets the active scenario to the baseline, while

```
mod1.scenario(c) "actuals"
```

sets the comparison scenario to the actuals (warning: this will overwrite any historical data in the solution period).

A newly created scenario will become the active scenario. Thus,

```
mod1(n) "Peace Scenario"
```

creates a scenario called "Peace Scenario" and makes it the active scenario. The scenario will automatically be assigned a unique numeric alias. To change the alias, simply use the "a = " option:

```
mod1(a=_ps) "Peace Scenario"
```

changes the alias for "Peace Scenario" to "_PS" and makes this scenario the active scenario.

The command:

```
mod1.scenario(n, a=w, i="Peace Scenario", c) "War Scenario"
```

creates a scenario called "War Scenario", initializes it with the Excludes and Overrides contained in "Peace Scenario", associates it with the alias "_W", and makes this scenario the comparison scenario.

```
mod1.scenario(i="Scenario 1") "Scenario 2"
```

copies the Excludes and Overrides in "Scenario 1" to "Scenario 2" and makes "Scenario 2" the active scenario.

Compatibility Notes

For backward compatibility with EViews 4.0, the option "a" may be used to set the comparison scenario, but is method not guaranteed to be supported in the future.)

In all of the arguments above the quotation marks around scenario name are currently optional. Support for the non-quoted names is provided for backward compatibility, but may be dropped in the future, thus

```
mod1.scenario Scenario 1
```

is currently valid, but may not be in future versions of EViews.

solveopt	Model Proc

Solve options for models.

`solveopt` sets options for model solution but does not solve the model. The same options can be set directly in a `solve` procedure.

Syntax

Model Proc: model_name.**solveopt***(options)*

Options

s = *arg* (default = d)	Solution type: “d” (deterministic), “m” (stochastic – collect means only), “s” (stochastic – collect means and s.d.), “b” (stochastic – collect means and confidence bounds), “a” (stochastic – collect all; means, s.d. and confidence bounds).
d = *arg* (default = d)	Model solution dynamics: “d” (dynamic solution), “s” (static solution), “f” (fitted values – single equation solution).
m = *integer* (default = 5000)	Maximum number of iterations for solution (maximum 100,000).
c = *number* (default = 1e-8)	Convergence criterion. Based upon the maximum change in any of the endogenous variables in the model. You may set a number between 1e-15 and 0.01.
r = *integer* (default = 1000)	Number of stochastic repetitions (used with stochastic “s = ” options).
b = *number* (default = .95)	Size of stochastic confidence intervals (used with stochastic “s = ” options).
a = *arg* (default = f)	Alternate scenario solution: “t” (true - solve both active and alternate scenario and collect deviations for stochastic), “f” (false - solve only the active scenario).
o = *arg* (default = g)	Solution method: “g” (Gauss-Seidel), “e” (Gauss-Seidel with extended search/reduced step size), “n” (Newton), “m” (Newton with extended search/reduced step size).

i = *arg*	Set initial (starting) solution values: “a” (actuals), “p” (values in period prior to start of solution period).
n = *arg* (default = t)	NA behavior: “t” (true - stop on “NA” values), “f” (false - do not stop when encountering “NA” values). Only applies to deterministic solution; EViews will always stop on “NA” values in stochastic solution.
e = *arg*	Excluded variables initialized from actuals: “t” (true), “f” (false).
t = *arg*	Terminal condition for forward solution: “u” (user supplied), “l” (constant level), “d” (constant difference), “g” (constant growth rate).
g = *arg* (default = 7)	Number of digits to round solution: an integer value (number of digits), “n” (do not roundoff).
z = *arg* (default = 1e-7)	Zero value: a positive number below which the solution (absolute value) is set to zero), “n” (do not set to zero).

statby — Series View

Basic statistics by classification.

The `statby` view displays descriptive statistics for the elements of a series classified into categories by one or more other series.

Syntax

Series View: series_name.**statby***(options)* classifier_name

Follow the series name with a period, the `statby` keyword, and a name (or a list of names) for the series or group by which to classify. The options control which statistics to display and in what form. By default, `statby` displays the means, standard deviations, and counts for the series.

Options

Options to control statistics to be displayed

sum	Display sums.
med	Display medians.
max	Display maxima.
min	Display minima.

quant = *arg* (default = .5)	Display quantile with value given by the argument.
q = *arg* (default = "r")	Compute quantiles using the definition: "b" (Blom), "r" (Rankit-Cleveland), "o" (simple fraction), "t" (Tukey), "v" (van der Waerden).
skew	Display skewness.
kurt	Display kurtosis.
na	Display counts of NAs.
nomean	Do not display means.
nostd	Do not display standard deviations.
nocount	Do not display counts.

Options to control layout

l	Display in list mode (for more than one classifier).
nor	Do not display row margin statistics.
noc	Do not display column margin statistics.
nom	Do not display table margin statistics (unconditional tables); for more than two classifier series.
nos	Do not display sub-margin totals in list mode; only used with "l" option and more than two classifier series.
sp	Display sparse labels; only with list mode option, "l".

Options to control binning

dropna (default), keepna	[Drop/Keep] NA as a category.
v = *integer* (default = 100)	Bin categories if classification series take on more than the specified number of distinct values.
nov	Do not bin based on the number of values of the classification series.
a = *number* (default = 2)	Bin categories if average cell count is less than the specified number.
noa	Do not bin based on the average cell count.
b = *integer* (default = 5)	Set maximum number of binned categories.

Other options

p	Print the descriptive statistics table.

Examples

```
wage.statby(max,min) sex race
```

displays the mean, standard deviation, max, and min of the series WAGE by (possibly binned) values of SEX and RACE.

Cross-references

See "Stats by Classification" on page 32.

unlink — Model Proc

Break model links.

Breaks equation links in the model. Follow the name of the model object by a period, the keyword `unlink`, and a specification for the variables to unlink.

Syntax

Model Proc: object.**unlink** *spec*

where *spec* is

@all	Unlinks all equations in the model.
list of endogenous vars	Unlink equations for the listed endogenous variables.

Note: If a link is to another Model or a System object, then more than one endogenous variable may be associated with the link. If the *spec* contains any of the endogenous variables in a linked Model or System, EViews will break the link for all of the variables found in the link.

Examples

```
mod1.unlink @all
mod2.unlink z1 z2
```

unlinks all of equations in MOD1, and all of the variables associated with the links for Z1 and Z2 in MOD2.

update	Model Proc

Update model specification.

Recompiles the model and updates all links.

Syntax

Model Proc: model.**update**

Follow the name of the model object by a period and the keyword update.

Examples

```
mod1.update
```

recompiles and updates all of the links in MOD1.

uroot	Command | Series View

Unit root tests.

Carries out the Augmented Dickey-Fuller (ADF), GLS detrended Dickey-Fuller (DFGLS), Phillips-Perron (PP), Kwiatkowski, et. al. (KPSS), Elliot, Rothenberg, and Stock (ERS) Point Optimal, or Ng and Perron (NP) tests for a unit root in the series (or its first or second difference).

Syntax

Command: **uroot***(options)* series_name

Series View: series_name.**uroot***(options)*

You should enter the keyword `uroot` followed by the series name, or the series name followed by a period and the keyword `uroot`.

Options

Specify the test type using one of the following keywords:

adf (default)	Augmented Dickey-Fuller.
dfgls	GLS detrended Dickey-Fuller (Elliot, Rothenberg, and Stock).
pp	Phillips-Perron.

kpss	Kwiatkowski, Phillips, Schmidt, and Shin.
ers	Elliot, Rothenberg, and Stock (Point Optimal).
np	Ng and Perron.

Specify the exogenous variables in the test equation from the following:

const (default)	Include a constant in the test equation.
trend	Include a constant and a linear time trend in the test equation.
none	Do not include a constant or time trend (only available for the ADF and PP tests).

For backward compatibility, the shortened forms “c”, “t”, and “n” are presently supported. However for future compatiblity we recommend that you use the longer forms.

Other Options:

dif = *integer* (default = 0)	Order of differencing of the series prior to running the test. Valid values are {0, 1, 2}.
hac = *arg*	Method of estimating the frequency zero spectrum: “bt” (Bartlett kernel), “pr” (Parzen kernel), “qs” (Quadratic Spectral kernel), “ar” (AR spectral), “ardt (AR spectral - OLS detrended data), “argls” (AR spectral - GLS detrended data). *The default settings are test specific:* “bt” for PP and KPSS, “ar” for ERS, “argls” for NP. Applicable to PP, KPSS, ERS and NP tests.
band = *arg*, b = *arg* (default = “nw”)	Method of selecting the bandwidth: “nw” (Newey-West automatic variable bandwidth selection), “a” (Andrews automatic selection), “*number*” (user specified bandwidth). Applicable to PP, KPSS, ERS, ERS and NP tests when using kernel sums-of-covariances estimators (where “hac = ” is one of {bt, pz, qs}).

lag = *arg* (default = "a")	Method of selecting lag length (number of first difference terms) to be included in the regression: "a" (automatic information criterion based selection), "*integer*" (user specified lag length) Applicable to ADF and DFGLS tests, and for the other tests when using AR spectral density estimators (where "hac = " is one of {ar, ardt, argls}).
info = *arg* (default = "maic")	Information criterion 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). Applicable to ADF and DFGLS tests, and for other tests when using AR spectral density estimators (where "hac = " is one of {ar, ardt, argls}).
maxlag = *integer*	Maximum lag length to consider when performing automatic lag length selection (default = $\text{int}(12(T/100)^{1/4})$). Applicable to ADF and DFGLS tests, and for other tests when using AR spectral density estimators (where "hac = " is one of {ar, ardt, argls}).
p	Print output from the test.

Examples

The command

```
gnp.uroot(adf,const,lag=3,save=mout)
```

performs an ADF test on the series GDP with the test equation including a constant term and three lagged first-difference terms. Intermediate results are stored in the matrix MOUT.

```
ip.uroot(dfgls,trend,info=sic)
```

runs the DFGLS unit root test on the series IP with a constant and a trend. The number of lagged difference terms is selected automatically using the Schwarz criterion.

```
unemp.uroot(kpss,const,hac=pr,b=2.3)
```

runs the KPSS test on the series UNEMP The null hypothesis is that the series is stationary around a constant mean. The frequency zero spectrum is estimated using kernel methods (with a Parzen kernel), and a bandwidth of 2.3.

```
sp500.uroot(np,hac=ardt,info=maic)
```

runs the NP test on the series SP500. The frequency zero spectrum is estimated using the OLS AR spectral estimator with the lag length automatically selected using the modified AIC.

Cross-references

See "Unit Root Tests" on page 5 for further discussion.

References

Andrews, Donald W. K. (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation," *Econometrica*, 59, 817–858.

Andrews, Donald W. K. and J. Christopher Monahan (1992). "An Improved Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimator," *Econometrica*, 60, 953–966.

Bhargava, A. (1986). "On the Theory of Testing for Unit Roots in Observed Time Series," *Review of Economic Studies*, 53, 369-384.

Davidson, Russell and James G. MacKinnon (1993). *Estimation and Inference in Econometrics*, Oxford University Press.

Dickey, D.A. and W.A. Fuller (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, 74, 427–431.

Elliott, Graham, Thomas J. Rothenberg and James H. Stock (1996). "Efficient Tests for an Autoregressive Unit Root," *Econometrica* 64, 813-836.

Hamilton, James D. (1994a). *Time Series Analysis*, Princeton University Press.

Hayashi, Fumio. (2000). *Econometrics*, Princeton University Press.

Kwiatkowski, Denis, Peter C. B. Phillips, Peter Schmidt & Yongcheol Shin (1992). "Testing the Null Hypothesis of Stationary against the Alternative of a Unit Root," *Journal of Econometrics,* 54, 159-178.

MacKinnon, J. G. (1991). "Critical Values for Cointegration Tests," Chapter 13 in R. F. Engle and C. W. J. Granger (eds.), *Long-run Economic Relationships: Readings in Cointegration*, Oxford University Press.

MacKinnon, James G. (1996). "Numerical Distribution Functions for Unit Root and Cointegration Tests," *Journal of Applied Econometrics*, 11, 601-618.

Newey, Whitney and Kenneth West (1994). "Automatic Lag Selection in Covariance Matrix Estimation," *Review of Economic Studies*, 61, 631-653.

Ng, Serena and Pierre Perron. 2001. "Lag Length Selection and the Construction of Unit Root Tests with Good Size and Power," *Econometrica*, 69(6), 1519-1554.

Phillips, P.C.B. and P. Perron (1988). "Testing for a Unit Root in Time Series Regression," *Biometrika*, 75, 335–346.

Said, Said E. and David A. Dickey (1984). "Testing for Unit Roots in Autoregressive Moving Average Models of Unknown Order," *Biometrika*, 71, 599–607.

White, Halbert (1980)."A Heteroskedasticity-Consistent Covariance Matrix and a Direct Test for Heteroskedasticity," *Econometrica*, 48, 817–838.

Index