DEPARTMENT OF ECONOMICS AND FINANCE

COLLEGE OF BUSINESS AND ECONOMICS

UNIVERSITY OF CANTERBURY

CHRISTCHURCH, NEW ZEALAND

Unit Root Tests, Size Distortions, and Cointegrated Data

W. Robert Reed

WORKING PAPER

No. 28/2014

Department of Economics and Finance
College of Business and Economics
University of Canterbury
Private Bag 4800, Christchurch
New Zealand

WORKING PAPER No. 28/2014

Unit Root Tests, Size Distortions, and Cointegrated Data

W. Robert Reed[1]

December 14, 2014

Abstract: This paper demonstrates that unit root tests can suffer from inflated Type I error rates when data are cointegrated. Results from Monte Carlo simulations show that three commonly used unit root tests – the ADF, Phillips-Perron, and DF-GLS tests – frequently overreject the true null of a unit root for at least one of the cointegrated variables. The findings extend previous research which reports size distortions for unit roots tests when the associated error terms are serially correlated (Schwert, 1989; DeJong et al., 1992; Harris, 1992). While the addition to the Dickey-Fuller-type specification of the correct number of lagged differenced (LD) terms can eliminate the size distortion, I demonstrate that determining the correct number of LD terms is unachievable in practice. Standard diagnostics such as testing for serial correlation in the residuals, and using information criteria to compare different lag specifications, are unable to identify the requird number of lags. A unique feature of this study is that it includes programs (an Excel spreadsheet and Stata .do files) that allow the reader to simulate their own cointegrated data -- using parameters of their own choosing -- to confirm the findings reported in this paper.

Keywords: Unit root testing, cointegration, DF-GLS test, Augmented Dickey-Fuller test, Phillips-Perron test, simulation

JEL Classifications: C32, C22, C18

Acknowledgements: This paper has been much improved as a result of comments from David Giles, Peter Phillips, Morten Nielsen, Marco Reale, and seminar participants at the "1st Conference on Recent Developments in Financial Econometrics and Applications" held at Deakin University, 4-5 December, 2014. Remaining errors are the sole property of the author.

[1] Department of Economics and Finance, University of Canterbury, New Zealand, Email: bob.reed@canterbury.ac.nz

I. INTRODUCTION

Unit root testing is central to much of financial econometrics. Many variables of interest in finance are conjectured to have unit roots, raising challenges for both estimation and inference. As a result, unit root tests are one of the essential tools in the financial econometrician's toolkit. It is well known that unit root tests can suffer from both size distortions and poor power . Schwert (1989), Agiakloglou and Newbold (1992), Dejong et al., (1992), and Harris (1992), among others, have demonstrated that ARMA components in the error term of Dickey-Fuller-type unit root test specifications are problematic. This paper demonstrates that similar problems arise when data are cointegrated.

I illustrate this using a simple autoregressive, distributed lag (ARDL) system of two variables. The ARDL framework has a number of features which make it attractive for modelling dynamic relationships. It allows for interactions between variables, and incorporates both endogeneity and own and cross-lagged effects. These features capture likely behaviours of real financial time series. The ARDL framework can be "solved" to identify parameter values that cause the two variables to be cointegrated. Furthermore, the ARDL framework is easily transformed to an error correction specification, which facilitates interpretation of dynamic relationships.

TABLE 1 illustrates the problem with size. X and Y are two simulated data series where the parameter values for the data generating process (DGP) have been chosen to ensure that they are cointegrated. Because the series are cointegrated, we know that each of the series must have a unit root. I subject each series to three unit root tests: the augmented Dickey-Fuller test (ADF), the Phillips-Perron test, and the DF-GLS test. 1000 simulations of sample sizes 100 were conducted. Significance levels were set equal to 0.05. The table reports the associated Type I error rates.

While the ADF and DF-GLS tests produce Type I error rates for *X* close to 0.05, the Phillips-Perron test produces an error rate close to 0.30. For *Y*, the results are much worse. Type I error rates are 0.325, 1.000, and 0.831 for the ADF, Phillips-Perron,and DF-GLS tests, respectively. While I will give more details below, I note for now that the ADF regressions show good diagnostics, with little serial correlation evident in the residuals. Lag lengths for the Phillips-Perron and DF-GLS tests were chosen automatically using commonly accepted algorithms.[1] In other words, in each case, good practice was followed in properly specifying the respective tests.

I proceed as follows. Section II presents the theory that motivates the simulation work. Section III presents additional Monte Carlo evidence of size distortions for cointegrated data. Section IV provides a brief discussion of the results. A unique feature of this study is that it includes programs (an Excel spreadsheet and Stata .do files) that allow the reader to simulate their own cointegrated data, using parameters of their own choosing, to confirm the findings reported in this paper. Section V introduces these programs and explains how they can be used. Section VI concludes.

II. THEORY

Consider the following ARDL(1,1) model.

$$y_t = \beta_{10} + \beta_{12}x_t + \gamma_{11}y_{t-1} + \gamma_{12}x_{t-1} + \varepsilon_{yt} \ , \varepsilon_{yt} \sim NID(0,1) \ ,$$
1)
$$x_t = \beta_{20} + \beta_{21}y_t + \gamma_{21}y_{t-1} + \gamma_{22}x_{t-1} + \varepsilon_{xt} \ , \varepsilon_{xt} \sim NID(0,1) \ ,$$

$t = 1,2, \dots, T$. This can be rewritten in VAR form as:

$$(2) \qquad \begin{bmatrix} y_t \\ x_t \end{bmatrix} = \begin{bmatrix} a_{10} \\ a_{20} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{xt} \end{bmatrix}.$$

[1] The Phillips-Perron test uses the Newey-West procedure to correct for serial correlation, with the number of lags equal to the integer part of $4\left(\frac{T}{100}\right)^{2/9}$ (Newey and West, 1987). The DF-GLS procedure adds lagged, first-differenced terms to the ADF-like regression equation, where the number of lags is chosen as the integer part of $12\left(\frac{T+1}{100}\right)^{0.25}$, as recommended by Schwert (1989).

where the parameters a_{ij}, *i=1,2, j=1,2,3,4* are each functions of the β and γ terms of Equation (1).

Define $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$. The determinant of the matrix $(A - \lambda I) = \begin{bmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{bmatrix}$ is the characteristic equation of A, and the values of λ that set this equation equal to zero are the associated characteristic roots, or eigenvalues:

(3) $\lambda^2 + (-a_{11} - a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) = 0$.

A necessary condition for y_t and x_t to be CI(1,1) is that the corresponding solutions to (3) be given by $\lambda_1 = 1,\ |\lambda_2| < 1$.

The following conditions on a_{11}, a_{12}, a_{21} and a_{22} are sufficient to ensure that $\lambda_1 = 1,\ |\lambda_2| < 1$.[2]

(4a) $0 < a_{22} < 1$

(4b) $0 < a_{12}a_{21} < 1 - a_{22}$

(4c) $a_{11} = 1 - \frac{a_{12}a_{21}}{1-a_{22}}$.

We can work backwards from (4a) – (4c) to obtain β and γ values consistent with $\lambda_1 = 1$, $|\lambda_2| < 1$.

Let β_{12} and β_{21} take any values such that $\beta_{12}\beta_{21} \neq 1$. Then

(5a) $\gamma_{21} = a_{21} - a_{11}\beta_{21}$

(5b) $\gamma_{11} = a_{11}(1 - \beta_{12}\beta_{21}) - \beta_{12}\gamma_{21}$

(5c) $\gamma_{22} = a_{22} - a_{12}\beta_{21}$

(5d) $\gamma_{12} = a_{12}(1 - \beta_{12}\beta_{21}) - \beta_{12}\gamma_{22}$

will produce a_{11}, a_{12}, a_{21} and a_{22} values such that $\lambda_1 = 1,\ |\lambda_2| < 1$.

Equation (2) can be arranged in vector error correction (VEC) model form as:

[2] These conditions are taken from Enders (2010, page 369). I have made the conditions more restrictive to make sure that the speed of adjustment parameters have the correct sign and size.

6)
$$\Delta y_t = a_{10} + \delta_y(y_{t-1} + \theta x_{t-1}) + \epsilon_{yt} \ ,$$
$$\Delta x_t = a_{20} + \delta_x(y_{t-1} + \theta x_{t-1}) + \epsilon_{xt} \ ,$$

where the coefficients θ, δ_y, and δ_x, as well as the error terms ϵ_{yt} and ϵ_{xt}, are functions of the a_{ij} terms, *i=1,2, j=1,2,3,4.* This allows the long-run equilibrium relationship between y_t and x_t, represented by the parameter θ; and the speed-of-adjustment parameters δ_y and δ_x, to all be expressed as functions of a_{11}, a_{12}, a_{21} and a_{22}.

7a) $\theta = \frac{a_{22}-1}{a_{21}}$

7b) $\delta_y = -\frac{a_{12}a_{21}}{1-a_{22}}$

7c) $\delta_x = a_{21}$.

The parameter values chosen in this way will ensure that (i) $-1 < \delta_y < 0$, and (ii) $-1 < \delta_x\theta < 0$, so that our VEC model is well-behaved.

III. RESULTS

TABLE 2 reports more simulation findings demonstrating that the results from TABLE 1 are not isolated outcomes. Three different sets of parameter values are used in each of the panels of TABLE 2. The top panel reports Monte Carlo results using the parameter values:

$\beta_{10} = 0, \beta_{12} = 0.2, \gamma_{11} = 0.4$, $\gamma_{12} = -0.08$, $\beta_{20} = 0$, $\beta_{21} = 0.8$, $\gamma_{21} = 0.1, \gamma_{22} = 0.82$.

The fact that both β_{12} and β_{21} are nonzero implies that if one of the series is I(1), the other must be as well.

These values were substituted into Equation (1) to produce cointegrated time series data for *Y* and *X*. The corresponding eigenvalues are $\lambda_1 = 1$, $\lambda_2 = 0.4$. The values generate a VEC model with long-run equilibrium and speed of adjustment parameters $\theta = -0.2$, $\delta_y = -0.5$, and $\delta_x = 0.5$. This implies that the long-run relationship between *Y* and *X* is given by $y_t = 0.2x_t$. A one-unit increase in y_t from its equilibrium value causes the next period's

value of Y to decrease by 0.5 units. A one-unit increase in x_t from its equilibrium value causes the next period's value of X to decrease by 0.1 units (=$\delta_x\theta$). As in TABLE 1, the Monte Carlo results are based on 1000 simulations of sample sizes 100.

TABLE 2 also reports the results of unit root tests for a random walk variable, $z_t = z_{t-1} + \varepsilon_{zt}$. $\varepsilon_{zt} \sim NID(0,1)$ These results are provided as a benchmark. The associated Type I error rates should be 0.05. However, as only 1000 simulations were run, deviations from 0.05 are to be expected. The Z column illustrates the kinds of deviations that can be expected from sampling error.

Turning first to the results for the X variable, there is no indication that the unit root tests are not performing well, other than perhaps a rather low Type I error rate for the Phillips-Perron test. Variable Y is a different story. The Type I error rates for the ADF and DF-GLS tests are 15.9 and 54 percent, respectively. The Phillips-Perron test produces a Type I error rate of over 90 percent. The latter two results indicate that over half the time, a researcher would conclude that the Y variable was stationary, when in fact it is nonstationary.

As is well-known, unit root tests can differ substantially depending on the number of lags included in the Dickey-Fuller-type, unit root specification. The Phillips-Perron and DF-GLS tests provided by Stata automatically select the number of lags. The ADF test requires the user to supply the number of lags. All of the results in TABLE 2 use 5 lagged, differenced (LD) terms for the ADF test. It is standard procedure in unit root testing to test the residuals to make sure that the number of included LD terms are sufficient to control for serial correlation. The last row of the panel reports the results of a Breusch-Godfrey test where the null hypothesis is no serial correlation. There is no indication in the associated size values that serial correlation is a problem. Based on these results, a researcher would conclude that the ADF test was correctly specified.

The next two panels report more unit root test results. In the second panel, the *X* variable shows high rejection rates for the null hypothesis of a unit root. 100 percent of the Phillips-Perron tests conclude that the *X* variable is stationary. The ADF tests for the *Y* variable are consistent with the data having a unit root. The DF-GLS tests are somewhat on the high side, while the Phillips-Perron test yielding a rejection rate of over 50 percent. The results from the bottom panel indicate that a researcher would generally conclude that the *X* variable was nonstationary while the *Y* variable was stationary, and hence not cointegrated.

It turns out that adding enough lagged, differenced terms to the ADF specification gets Type I error rates down to 5 percent. However, knowing the correct number of LD terms to add is difficult, if not impossible. In practice, researchers may use a number of criteria to determine the number of LD terms. For example, they may test the residuals for serial correlation. Or they may use information criteria such as the AIC and SIC (Harris, 1992).

TABLE 3A reports the results of successively adding LD terms using simulated data produced by the parameter values in the top panel of TABLE 2. The first panel of TABLE 3A reports rejection rates associated with the Breusch-Godfrey test for serial correlation. The null hypothesis is no serial correlation. The next two panels report average AIC and SIC values. The last panel reports Type I error rates for the different number of LD terms.

TABLE 3A is designed to address this thought experiment: based on the results from the first three panels, how many LD terms would a researcher think is the "correct" number of terms to add? For example, when LAGS = 1, the null hypothesis of no serial correlation is rejected approximately 12.5 percent of the time for the *Y* series. When LAGS =2 , this falls to 6.4 percent, which is consistent with a significance level of 5 percent once one accounts for sampling error. The average AIC value for the *Y* series when LAGS =1 is 320.43. This falls to 320.15 when LAGS =2, and then successively increases for additional lags. Average SIC

values for the *Y* series have their minimum when LAGS = 1. Given these diagnostics, a researcher might conclude that the "correct" number of LD terms to add was 1 or 2. Turning now to the ADF unit root tests results of the fourth panel, we see that the Type I error rates are 70.6 percent when LAGS = 1, and 47.4 percent when LAGS = 2. In fact, one must add 9 or 10 lags before Type I error rates display correct size values.

TABLE 3B repeats this experiment, using the parameter values from the middle panel of TABLE 2. The Breusch-Godfrey tests indicate that it takes approximately 4 lags to eliminate serial correlation from a DF specification involving the *X* series. In contrast, both the AIC and SIC values indicate that two lags are best. Turning now to the unit root tests in the bottom panel of TABLE 3B, we see that with LAGS = 2, the ADF test rejects the null of a unit root 57.1 percent of the time. When LAGS = 4, the Type I error rate is 16.2 percent. In fact, it takes 7 or more lags to reduce the Type I error rate to the 5 percent range.

TABLE 3C repeats the exercise one more time, using the parameter values from the bottom panel of TABLE 2. Focussing on the *Y* series, we see that the Breusch-Godfrey tests, AIC, and SIC values all indicate that the "correct" number of lags is 1. In fact, it takes more than 10 lags to reduce Type I error rates to 5 percent.

TABLES 3A through 3C demonstrate that it is possible to add sufficient lagged differenced terms to where the respective unit root tests produce the correct Type I error rates. The problem is that it is not possible, based upon standard diagnostic tests, to know what this number needs to be.

IV. DISCUSSION

The explanation for the poor size performance of the unit root tests above has to do with how critical values are determined in the Dickey-Fuller framework. Given Monte Carlo simulation of the random walk process,

$z_t = z_{t-1} + \varepsilon_t$,

repeated OLS estimation of the DF specification below

$\Delta z_t = \gamma + \rho z_{t-1} + error_t.$

produces an empirical distribution of t values, $t = \frac{\hat{\rho}}{s.e.(\hat{\rho})}$ associated with testing the null hypothesis: $H_0: \rho = 0$. The 1st, 5th, and 10th percentile points of the empirical cumulative distribution function of *t*-statistics correspond to critical values associated with the 99th, 95th, and 90th confidence levels.

In my simulations of the ARDL(1,1) data, the DGP is given by

$y_t = \beta_{10} + \beta_{12} x_t + \gamma_{11} y_{t-1} + \gamma_{12} x_{t-1} + \varepsilon_{yt}$,

$x_t = \beta_{20} + \beta_{21} y_t + \gamma_{21} y_{t-1} + \gamma_{22} x_{t-1} + \varepsilon_{xt}$,

and the corresponding VEC formulation is given by

$\Delta y_t = a_{10} + \delta_y y_{t-1} + \delta_y \theta x_{t-1} + \epsilon_{yt}$

$\Delta x_t = a_{20} + \delta_x x_{t-1} + \delta_x \theta y_{t-1} + \epsilon_{xt}$.

A comparison of the Δy_t and Δx_t specifications with that of Δz_t shows that the first two specifications contain an additional term not contained in the Dickey-Fuller specification: x_{t-1} and y_{t-1}, respectively. Said and Dickey (1984) show that a sufficiently large number of AR terms can approximate an unknown ARMA process. This provides the rationale for adding LD terms to the Dickey-Fuller-type specification when serial correlation in suspected in the error terms. Indeed, as demonstrated above, one can produce appropriate size results with the addition of the right number of lags. Unfortunately, standard diagnostic tools are unable to identify what that number is.

V. DESCRIPTION OF PROGRAMS

Introduction. The unit root test results in TABLES 1 through 3 report some of the more egregious examples of poor size behaviour. While Type I error rates greater than 50 percent are moderately unusual, it is quite easy to generate cointegrated data where the Type I error

rates for one of the two variables is larger than 10 percent. Included with this paper is (i) a spreadsheet for choosing DGP parameter values that produce cointegrated data, (ii) programs to confirm that the data are cointegrated, and (iii) simulation programs that generate the Monte Carlo results of TABLES 2 and 3. In this manner, the reader can produce his/her own results – using parameter values of their own choosing – to confirm the findings reported here.[3]

The Excel spreadsheet "Parameter Values". The worksheet "ParameterValues" allows one to select parameter values that produce CI(1,1) series. The yellow-highlighted cells identify where the researcher enters values for a_{22} (STEP ONE), and a_{12} and a_{21} (STEP TWO) and gives instructions to ensure they satify the necessary conditions.

STEP ONE	Choose 0 < a22 < 1		
a22 =	0.9		
STEP TWO	Choose a12 and a21 such that 0 < a12a21 < 1-a22		
a12 =	-0.1		
a21 =	-0.9		
a12a21 =	0.09	1-a22=	0.1

From here, the spreadsheet calculates the corresponding value of a_{11} that guarantees that the largest eigenvalue = 1 (STEP THREE).

STEP THREE	a11 is set equal to 1 - (a12a21/1-a22) to guarantee that the second eigenvalue is < 1
a11 =	0.1

The next step works backwards to calculate the values of β and γ that will produce the values of a_{11}, a_{12}, a_{21} and a_{22} that one selected in STEPS ONE through THREE. It is straightforward to show that the values of a_{11}, a_{12}, a_{21} and a_{22} in Equation (2) are related to the values of β and γ in Equations (1) and (2) by the following:

[3] The examples in this section relate to the Monte Carlo results of TABLE 1.

(8a) $a_{11} = \frac{\gamma_{11}+\beta_{12}\gamma_{21}}{1-\beta_{12}\beta_{21}}$

(8b) $a_{12} = \frac{\gamma_{12}+\beta_{12}\gamma_{22}}{1-\beta_{12}\beta_{21}}$

(8c) $a_{21} = \frac{\gamma_{21}+\beta_{21}\gamma_{11}}{1-\beta_{12}\beta_{21}}$

(8d) $a_{22} = \frac{\gamma_{22}+\beta_{21}\gamma_{12}}{1-\beta_{12}\beta_{21}}$

(8a) – (8d) provide 4 equations with 6 unknowns, β_{12}, β_{21}, γ_{11}, γ_{12}, γ_{21}, and γ_{22}. Accordingly, the yellow-highlighted cells in STEP FOUR allow one to set β_{12} and β_{21} to any value so long as their product does not equal 1.

STEP FOUR	Let b12 and b21 take any values such that b12*b21 is not equal to 1
b12 =	2
b21 =	5
b12b21 =	10
X =	-9

Once β_{12} and β_{21} have been set, one can use Equations (8a) – (8d) to solve for the corresponding values of γ_{11}, γ_{12}, γ_{21}, and γ_{22} (STEP FIVE).[4]

STEP FIVE	This uses the definitions of a11, a12, a21, and a22 to solve for g11, g12, g21, and g22, given b12 and b22	
g21 =	-1.4000	NOTE: It is important that the last two digits be zero so that
g11 =	1.9000	these EXACT parameter values can be entered into Stata
g22 =	1.4000	
g12 =	-1.9000	

Note that these parameter values will be entered into the Stata .do files. The values need to be exact in order for the series to be cointegrated. Approximate values, such as 0.333 for 1/3, will not suffice. So this section warns that the user must be sure that the respective parameter values do not have trailing values.

The last part of the worksheet reports the corresponding values of $\theta, \delta_y,$ and δ_x.

[4] Note that g11, g12, g21, and g22 represent γ_{11}, γ_{12}, γ_{21}, and γ_{22}, respectively.

THIS REPORTS LR EQUILIBRIUM RELATIONSHIP (theta) AND SPEED OF ADJUSTMENT VALUES (deltay/deltax)		
theta =	0.111111	
deltay =	-0.9	NOTE: Need to confirm that this is between -1 and 0.
deltax =	-0.9	
deltax*theta=	-0.1	NOTE: Need to confirm that this is between -1 and 0.

This allows one to check that the speed of adjustment parameters satisfy the conditions (i) $-1 < \delta_y < 0,$ and (ii) $-1 < \delta_x\theta < 0$, so that the series do not explode.

Also included in the Excel spreadsheet is a worksheet by the name of "PasteValues." It summarizes all the parameter values that were calculated on the "ParameterValues" worksheet (see below).

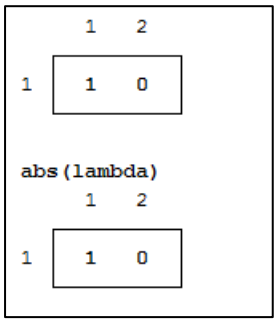
local b10 =	0		b10 =	0
local b12 =	2		b12 =	2
local g11 =	1.9		g11 =	1.9
local g12 =	-1.9		g12 =	-1.9
local b20 =	0		b20 =	0
local b21 =	5		b21 =	5
local g21 =	-1.4		g21 =	-1.4
local g22 =	1.4		g22 =	1.4

These are formatted for easy copy-and-pasting into the subsequent .do files.

The Stata .do file "CharRoots". The .do file "CharRoots" takes the parameter values from the spreadsheet and calculates the associated eigenvalues. It also provides a plot of some of the y_t and x_t values so that one can visualize how they relate. All one has to do is copy the appropriate cells from the "PasteValues" worksheet, paste them into the corresponding section of the .do file, and then run the program (see below).

```
// This section fixes the parameter values of the DGP
b10 =   0
b12 =   2
g11 =   1.9
g12 =   -1.9
b20 =   0
b21 =   5
g21 =   -1.4
g22 =   1.4
```

The associated output confirms that the two eigenvalues satisfy the conditions $\lambda_1 = 1,\ |\lambda_2| < 1$:



The Stata .do file "VECrank". The .do file "VECrank" takes the parameters from the spreadsheet and uses the respective testing procedures in Stata to test for the rank of the matrix $(A - I) = \begin{bmatrix} a_{11} - 1 & a_{12} \\ a_{21} & a_{22} - 1 \end{bmatrix}$. A rank of 1 is consistent with the variables *Y* and *X* being cointegrated. To run this program, follow the same procedure as above and copy and paste the parameter values from the "PasteValues" worksheet into the appropriate section of the .do file, as discussed above.

This program produces output in two parts. The first part calculates the rank of $(A - I)$ directly from the population parameter values of the DGP:

```
A1I = A1-I(2)

rank(A1I)
1
```

The second part simulates 1000 observations of *X* and *Y* and performs (i) the trace test, (ii) the maximum eigenvalue test, and (iii) presents a series of information criterion values using the Schwarz Bayesian information criterion (SBIC), the Hannan and Quinn information criterion (HQIC), and the Akaike information criterion (AIC). All the results from the table below indicate that $(A - I) = \begin{bmatrix} a_{11} - 1 & a_{12} \\ a_{21} & a_{22} - 1 \end{bmatrix}$ has a rank of 1, consistent with the data being cointegrated.

Johansen tests for cointegration

Trend: constant — Number of obs = 998

Sample: 3 - 1000 — Lags = 2

maximum rank	parms	LL	eigenvalue	trace statistic	5% critical value
0	6	-832.09042	.	430.6955	15.41
1	9	-616.91072	0.35029	0.3362*	3.76
2	10	-616.74265	0.00034		

maximum rank	parms	LL	eigenvalue	max statistic	5% critical value
0	6	-832.09042	.	430.3594	14.07
1	9	-616.91072	0.35029	0.3362	3.76
2	10	-616.74265	0.00034		

maximum rank	parms	LL	eigenvalue	SBIC	HQIC	AIC
0	6	-832.09042		1.709033	1.690751	1.67954
1	9	-616.91072	0.35029	1.29857*	1.271146*	1.25433
2	10	-616.74265	0.00034	1.305153	1.274682	1.255997

The Stata .do file "VECstable". The .do file "VECstable" takes the parameters from the spreadsheet, simulates 1000 observations using those parameter values, estimates the VEC model, and then reports the estimated eigenvalues of the system. This program also has output in two parts. The first part estimates of the VEC model and allows one to compare the estimated values of $\theta, \delta_y,$ and δ_x with their true values, as reported at the bottom of the "ParameterValues" worksheet.

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
D_y						
_ce1						
L1.	-.9075725	.0403104	-22.51	0.000	-.9865793	-.8285656
y						
LD.	-.0016972	.0332669	-0.05	0.959	-.0668991	.0635047
x						
LD.	-.0019733	.0196572	-0.10	0.920	-.0405007	.0365541
_cons	.0124837	.0079948	1.56	0.118	-.0031858	.0281533
D_x						
_ce1						
L1.	-.935039	.0906559	-10.31	0.000	-1.112721	-.7573566
y						
LD.	.0216589	.0748156	0.29	0.772	-.1249769	.1682947
x						
LD.	-.0001907	.044208	-0.00	0.997	-.0868369	.0864554
_cons	-.012117	.0179799	-0.67	0.500	-.047357	.0231229

Johansen normalization restriction imposed

beta	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_ce1						
y	1	.	.	.	.	.
x	.1106794	.0012192	90.78	0.000	.1082897	.113069
_cons	.0069962	.	.	.	.	.

From the output above, we see that the VEC estimates of δ_y, δ_x, and θ are -0.908, -0.935, and 0.111. The true values are reported on the bottom of the "Parameter Values" worksheet:

THIS REPORTS LR EQUILIBRIUM RELATIONSHIP (theta) AND SPEED OF ADJUSTMENT VALUES (deltay/deltax)		
theta =	0.111111	
deltay =	-0.9	NOTE: Need to confirm that this is between -1 and 0.
deltax =	-0.9	
deltax*theta=	-0.1	NOTE: Need to confirm that this is between -1 and 0.

The last part of the program estimates the value of the second eigenvalue, assuming that the larger of the two eigenvalues equals one:

Eigenvalue stability condition

Eigenvalue	Modulus
1	1
.04103287	.041033
-.0269914 + .01791487i	.032396
-.0269914 - .01791487i	.032396

The VECM specification imposes a unit modulus.

$|\lambda_2|$ is estimated to be 0.041. We know from "CharRoots" program above that the true value of the second eigenvalue is zero. While standard errors are not calculated, the estimated value of $|\lambda_2|$ is well below 1, confirming again that the series are cointegrated.

The Stata .do files "function_data", "Table2A", and "Table2B". The .do files "function_data", "Table2A", and "Table2B" are a suite of three programs that are designed to be used together. These programs perform the simulation exercises that test each of the series for unit root using the (i) ADF, (ii) Phillips-Perron, and (iii) DF-GLS tests. The programs simulate 1000 data sets of 100 observations each of *X*, *Y*, and a third variable *Z*. *X* and *Y* are simulated using the parameter values pasted into the program "Table2B":

```
// This section fixes the parameter values of the DGP
local b10 = 0
local b12 = 2
local g11 = 1.9
local g12 = -1.9
local b20 = 0
local b21 = 5
local g21 = -1.4
local g22 = 1.4
```

The variable *Z* is a classic random walk variable:

(9) $z_t = z_{t-1} + \varepsilon_{zt}$, $\varepsilon_{zt} \sim NID(0,1)$.

It is included in the simulations for comparison's sake.

The programs must be run in order (first "function_data", then "Table2A", and then "Table2B") and take less than 5 minutes to run on a standard issue laptop. The output

consists of two parts. The first part are the Type I error rates associated with the three unit root tests:

```
RESULTS[3,3]
             X      Y      Z
DFuller   .035   .325   .053
PPerron   .291      1   .066
  DFGLS   .075   .831   .039
```

While the Phillips-Perron and DF-GLS tests automatically select lag lengths, the default option in Stata is to include no lagged differenced terms in the Dickey-Fuller specification. As serial correlation in the error term distort test results, it is advisable to add lagged differenced terms.

My simulations add 5 lags to the Dickey-Fuller test. To ensure that this is sufficient to account for serial correlation, I perform a Breusch-Godfrey test following each Dickey-Fuller test. The associated rejection rates are reported in the second part of the output:

```
BGODFREY[1,3]
              X      Y      Z
BGodfrey   .056   .056   .055

. display as text "Number of lags = `lagz'"
Number of lags = 5
```

This provides a check to ensure that I have added sufficient lagged differenced terms to ensure that the Dickey-Fuller test results are not distorted by the presence of serial correlation.

The Stata .do files "function_data", "Table3A", and "Table3B". The last set of programs are designed to illustrate the effect of increasing the number of lagged differenced terms to the Dickey-Fuller unit root specification. As with the TABLE 2 programs, they must be run in order (first "function_data", then "Table3A", and then "Table3B") and take less than 5 minutes to run on a standard issue laptop.

VI. CONCLUSION

This paper demonstrates that unit root tests can suffer from inflated Type I error rates when data are cointegrated. The results extend previous findings which report size distortions for unit roots tests when the associated error terms are characterized by serial correlation (Schwert, 1989; DeJong et al., 1992; Harris, 1992). While the addition of sufficiently many lagged differenced (LD) terms to the Dickey-Fuller-type specification will produce correct size results, standard diagnostic tools are unable to identify the correct number of LD terms in practice.

My results should be of interest to researchers who are interested in estimating relationships between nonstationary variables. Standard procedure calls for testing variables for unit roots before proceeding to the estimation of error correction models. My results indicate that the very fact that the data are cointegrated causes unit root tests to be unreliable. This suggests that researchers should be conservative in the weight they attach to individual unit root tests.

REFERENCES

Agiakloglou, C. and Newbold, P. 1992. Empirical evidence on Dickey-Fuller-type tests. *Journal of Time Series Analysis* 13: 471-483.

DeJong, D.N., Nankervis, J.C., Savin, N.E., and Whiteman, C.H. 1992. The power problems of unit root tests in time series with autoregressive errors. *Journal of Econometrics* 53: 323-343.

Enders, W. 2010. *Applied Econometric Time Series*, 3rd Edition, New York: John Wiley & Sons.

Harris, R.I.D. 1992. Testing for unit roots using the augmented Dickey-Fuller test: Some issues relating to the size, power and the structure of the test. *Economics Letters* 38: 381-386.

Newey, W. K., and K. D. West. 1987. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708.

Said, S.E. and Dickey, D.A. 1984. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika* 71: 599-607.

Schwert, G. W. 1989. Tests for unit roots: A Monte Carlo investigation. *Journal of Business and Economic Statistics* 2: 147-159.

TABLE 1
Example of Unit Root Test Results Using Cointegrated Data

UNIT ROOT TEST	***X***	***Y***
ADF	0.035	0.325
Phillips-Perron	0.291	1.000
DF-GLS	0.075	0.831

NOTE: Values in the table are Type I error rates associated with the null hypothesis that the data series have a unit root. The simulations underlying TABLE 1 used the following parameter values for the DGP (see Equation 1):

$\beta_{10} = 0, \beta_{12} = 2, \gamma_{11} = 1.9,\ \gamma_{12} = -1.9,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -1.4,\ \gamma_{22} = 1.4$

The corresponding long-run equilibrium and speed of adjustment parameters (see Equations 7a-7c) are given by: $\theta = 0.11, \delta_y = -0.9, \delta_x = -0.9, \theta\delta_x = -0.1$.

TABLE 2
More Examples of Unit Root Test Results Using Cointegrated Data

	X	***Y***	***Z***
Parameter Values:[1] $\beta_{10} = 0, \beta_{12} = 0.2, \gamma_{11} = 0.4,\ \gamma_{12} = -0.08,\ \beta_{20} = 0,\ \beta_{21} = 0.8,\ \gamma_{21} = 0.1, \gamma_{22} = 0.82$ $\theta = -0.2, \delta_y = -0.5, \delta_x = 0.5, \theta\delta_x = -0.1$			
ADF	0.046	0.159	0.053
Phillips-Perron	0.019	0.912	0.066
DF-GLS	0.039	0.540	0.039
BREUSCH-GODFREY TEST:	0.053	0.057	0.055
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = 2.2,\ \gamma_{12} = -1.4,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -3.8, \gamma_{22} = 4.6$ $\theta = 3.0, \delta_y = -0.3, \delta_x = -0.3, \theta\delta_x = -0.9$			
ADF	0.119	0.051	0.053
Phillips-Perron	1.000	0.527	0.066
DF-GLS	0.508	0.091	0.039
BREUSCH-GODFREY TEST	0.052	0.064	0.055
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = -4.5,\ \gamma_{12} = -4.45,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -1.5, \gamma_{22} = 0.65$ $\theta = -0.1, \delta_y = -0.5, \delta_x = 1, \theta\delta_x = -0.1$			
ADF	0.057	0.335	0.053
Phillips-Perron	0.022	0.998	0.066
DF-GLS	0.045	0.844	0.039
BREUSCH-GODFREY TEST:	0.069	0.05	0.055

[1] The β and γ values correspond to Equation (1) and are the values used to simulate the cointegrated X and Y values in the table. δ_y represents the Δy_t resulting from a unit increase in y_t. $\delta_x\theta$ represents the Δx_t resulting from a unit increase in x_t. θ is associated with the long-run relationship between y_t and x_t, where the long-run relationship is given by $y_t + \theta x_t = 0$.

NOTE: Values in the table are rejection rates of the respective null hypothesis. For the unit root tests, the null hypothesis is that the series has a unit root. For the Breusch-Godfrey tests, the null hypothesis is that the residuals associated with the ADF test (with five lagged, differenced terms) are not serially correlated.

TABLE 3A
The Effect of Adding Lagged Differenced Terms to the Dickey-Fuller Unit Root Regression Equation: Example 1

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 0.2, \gamma_{11} = 0.4,\ \gamma_{12} = -0.08,\ \beta_{20} = 0,\ \beta_{21} = 0.8,\ \gamma_{21} = 0.1, \gamma_{22} = 0.82$			
BREUSCH-GODFREY TESTS:			
LAGS = 1	0.065	0.125	0.051
LAGS = 2	0.064	0.064	0.072
LAGS = 3	0.067	0.079	0.052
LAGS = 4	0.043	0.067	0.066
LAGS = 5	0.065	0.062	0.069
LAGS = 6	0.063	0.049	0.068
LAGS = 7	0.051	0.064	0.066
LAGS = 8	0.057	0.043	0.069
LAGS = 9	0.055	0.060	0.059
LAGS = 10	0.063	0.065	0.073
AIC VALUES:			
LAGS = 1	369.95	320.43	279.46
LAGS = 2	370.03	320.15	280.82
LAGS = 3	372.03	321.92	280.53
LAGS = 4	371.42	322.22	282.62
LAGS = 5	372.86	322.78	283.77
LAGS = 6	374.44	324.31	283.89
LAGS = 7	375.88	325.35	285.78
LAGS = 8	376.71	326.40	286.45
LAGS = 9	376.73	328.07	287.53
LAGS = 10	377.44	327.73	287.53
SIC VALUES:			
LAGS = 1	380.34	330.81	289.84
LAGS = 2	383.00	333.13	293.80
LAGS = 3	387.60	337.49	296.10
LAGS = 4	389.59	340.39	300.78
LAGS = 5	393.63	343.54	304.53
LAGS = 6	397.79	347.66	307.24
LAGS = 7	401.83	351.30	311.74
LAGS = 8	405.26	354.95	315.00
LAGS = 9	407.87	359.21	318.68
LAGS = 10	411.18	361.46	321.26

TABLE 3A (continued)

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 0.2, \gamma_{11} = 0.4,\ \gamma_{12} = -0.08,\ \beta_{20} = 0,\ \beta_{21} = 0.8,\ \gamma_{21} = 0.1, \gamma_{22} = 0.82$			
<u>ADF UNIT ROOT TESTS</u>:			
LAGS = 1	0.039	0.706	0.048
LAGS = 2	0.056	0.474	0.049
LAGS = 3	0.034	0.299	0.053
LAGS = 4	0.055	0.218	0.039
LAGS = 5	0.044	0.139	0.058
LAGS = 6	0.041	0.100	0.066
LAGS = 7	0.049	0.102	0.033
LAGS = 8	0.053	0.078	0.054
LAGS = 9	0.048	0.066	0.050
LAGS = 10	0.042	0.051	0.043

NOTE: Values in the top panel ("Breusch-Godfrey Tests") are the rejection rates associated with the null hypothesis of no serial correlation. Values in the next two panels ("AIC Values" and "SIC Values") are the average information criteria values associated with the respective lag specifications. The number of observations are held constant across the different specifications. The values in the bottom panel ("ADF Unit Root Tests") are the Type I error rates associated with the null hypothesis of a unit root using the ADF test with the designated number of lagged, differenced terms.

TABLE 3B
The Effect of Adding Lagged Differenced Terms to the Dickey-Fuller Unit Root Regression Equation: Example 2

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = 2.2,\ \gamma_{12} = -1.4,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -3.8, \gamma_{22} = 4.6$			
BREUSCH-GODFREY TESTS:			
LAGS = 1	0.402	0.095	0.051
LAGS = 2	0.165	0.050	0.072
LAGS = 3	0.097	0.058	0.052
LAGS = 4	0.057	0.073	0.066
LAGS = 5	0.053	0.068	0.069
LAGS = 6	0.050	0.066	0.068
LAGS = 7	0.060	0.055	0.066
LAGS = 8	0.060	0.063	0.069
LAGS = 9	0.070	0.056	0.059
LAGS = 10	0.065	0.074	0.073
AIC VALUES:			
LAGS = 1	-5.766	30.107	279.462
LAGS = 2	-9.340	30.165	280.823
LAGS = 3	-8.261	32.974	280.533
LAGS = 4	-8.733	32.028	282.615
LAGS = 5	-7.723	33.365	283.774
LAGS = 6	-6.561	34.516	283.887
LAGS = 7	-5.655	35.286	285.785
LAGS = 8	-4.228	36.844	286.453
LAGS = 9	-2.882	37.168	287.535
LAGS = 10	-3.493	37.858	287.527
SIC VALUES:			
LAGS = 1	4.614	40.488	289.842
LAGS = 2	3.636	43.141	293.799
LAGS = 3	7.310	48.545	296.104
LAGS = 4	9.432	50.194	300.781
LAGS = 5	13.038	54.126	304.535
LAGS = 6	16.795	57.872	307.243
LAGS = 7	20.297	61.237	311.736
LAGS = 8	24.319	65.391	315.000
LAGS = 9	28.260	68.309	318.676
LAGS = 10	30.243	71.595	321.264

TABLE 3B (continued)

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = 2.2,\ \gamma_{12} = -1.4,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -3.8, \gamma_{22} = 4.6$			
<u>ADF UNIT ROOT TESTS</u>:			
LAGS = 1	0.886	0.112	0.048
LAGS = 2	0.571	0.057	0.049
LAGS = 3	0.295	0.051	0.053
LAGS = 4	0.162	0.042	0.039
LAGS = 5	0.115	0.045	0.058
LAGS = 6	0.083	0.030	0.066
LAGS = 7	0.068	0.040	0.033
LAGS = 8	0.051	0.045	0.054
LAGS = 9	0.042	0.051	0.05
LAGS = 10	0.043	0.059	0.043

NOTE: Values in the top panel ("Breusch-Godfrey Tests") are the rejection rates associated with the null hypothesis of no serial correlation. Values in the next two panels ("AIC Values" and "SIC Values") are the average information criteria values associated with the respective lag specifications. The number of observations are held constant across the different specifications. The values in the bottom panel ("ADF Unit Root Tests") are the Type I error rates associated with the null hypothesis of a unit root using the ADF test with the designated number of lagged, differenced terms.

TABLE 3C
The Effect of Adding Lagged Differenced Terms to the Dickey-Fuller Unit Root Regression Equation: Example 3

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = -4.5,\ \gamma_{12} = -4.45,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -1.5, \gamma_{22} = 0.65$			
BREUSCH-GODFREY TESTS:			
LAGS = 1	0.059	0.063	0.051
LAGS = 2	0.057	0.063	0.072
LAGS = 3	0.061	0.067	0.052
LAGS = 4	0.055	0.064	0.066
LAGS = 5	0.056	0.055	0.069
LAGS = 6	0.082	0.055	0.068
LAGS = 7	0.073	0.066	0.066
LAGS = 8	0.064	0.053	0.069
LAGS = 9	0.067	0.05	0.059
LAGS = 10	0.074	0.054	0.073
AIC VALUES:			
LAGS = 1	34.619	-25.486	279.462
LAGS = 2	34.938	-25.379	280.823
LAGS = 3	36.235	-23.861	280.533
LAGS = 4	36.445	-24.410	282.615
LAGS = 5	37.771	-22.322	283.774
LAGS = 6	38.842	-21.251	283.887
LAGS = 7	39.836	-19.844	285.785
LAGS = 8	41.298	-18.913	286.453
LAGS = 9	42.084	-18.837	287.535
LAGS = 10	41.982	-18.002	287.527
SIC VALUES:			
LAGS = 1	45.000	-15.105	289.842
LAGS = 2	47.913	-12.403	293.799
LAGS = 3	51.806	-8.290	296.104
LAGS = 4	54.611	-6.244	300.781
LAGS = 5	58.532	-1.561	304.535
LAGS = 6	62.198	2.105	307.243
LAGS = 7	65.787	6.108	311.736
LAGS = 8	69.844	9.634	315.000
LAGS = 9	73.226	12.305	318.676
LAGS = 10	75.718	15.735	321.264

TABLE 3C (continued)

	X	***Y***	***Z***
Parameter Values: $\beta_{10} = 0, \beta_{12} = 5, \gamma_{11} = -4.5,\ \gamma_{12} = -4.45,\ \beta_{20} = 0,\ \beta_{21} = 5,\ \gamma_{21} = -1.5, \gamma_{22} = 0.65$			
<u>ADF UNIT ROOT TESTS</u>:			
LAGS = 1	0.041	0.959	0.048
LAGS = 2	0.063	0.806	0.049
LAGS = 3	0.046	0.618	0.053
LAGS = 4	0.047	0.451	0.039
LAGS = 5	0.050	0.351	0.058
LAGS = 6	0.044	0.249	0.066
LAGS = 7	0.051	0.186	0.033
LAGS = 8	0.053	0.133	0.054
LAGS = 9	0.045	0.102	0.050
LAGS = 10	0.035	0.085	0.043

NOTE: Values in the top panel ("Breusch-Godfrey Tests") are the rejection rates associated with the null hypothesis of no serial correlation. Values in the next two panels ("AIC Values" and "SIC Values") are the average information criteria values associated with the respective lag specifications. The number of observations are held constant across the different specifications. The values in the bottom panel ("ADF Unit Root Tests") are the Type I error rates associated with the null hypothesis of a unit root using the ADF test with the designated number of lagged, differenced terms.

APPENDIX:
Stata .do files that accompany this paper

Note #1: These .do files are included to make it possible for readers to confirm this study's findings, and investigate alternative DGP specifications. Running times for the longer programs are given at the beginning of the program. The purposes of the respective programs are described in the text.

Note #2: To simulate different data, the reader need only change the parameter values in the yellow-highlighted sections of the programs. Parameters that produce cointegrated data can be obtained from the accompanying Excel spreadsheet, "Parameter Values". Note that these parameter can be cut and pasted directly from the spreadsheet into the Stata .do files.

Note #3: The programs underlying TABLE 2 in the text must be run in this order: (i) "function.data", (ii) "Table2A", and (iii) "Table2B". The programs underlying TABLE 3 in the text must be run in this order: (i) "function.data", (ii) "Table3A", and (iii) "Table3B"

"CharRoots" program

```
clear
set more off
graph drop _all
set scheme s2mono
matrix m=J(100,2,0)
svmat m

mata:

// This section fixes the parameter values of the DGP
b10 =    0
b12 =    2
g11 =    1.9
g12 =    -1.9
b20 =    0
b21 =    5
g21 =    -1.4
g22 =    1.4

// This section maps the b* and g* parameters to the elements of A
a11 = (g11+b12*g21)/(1-b12*b21)
a12 = (g12+b12*g22)/(1-b12*b21)
a21 = (g21+b21*g11)/(1-b12*b21)
a22 = (g22+b21*g12)/(1-b12*b21)

// This section maps the b* and g* parameters to the rest of the VAR
a10 = (b10+b12*b20)/(1-b12*b21)
a20 = (b20+b21*b10)/(1-b12*b21)
a13 = 1/(1-b12*b21)
a14 = b12/(1-b12*b21)
a23 = b21/(1-b12*b21)
a24 = 1/(1-b12*b21)
A0 = a10\a20
A1 = a11,a12\a21,a22
A2 = a13,a14\a23,a24

// This section creates the data
st_view(m=.,.,.)
for (i=2; i<=rows(m); i++) {
    u = rnormal(1,2,0,1)
    m[i,.]=A0'+ m[i-1,.]*A1'+ u*A2'
 }

// This section calculates and reports the characteristic roots of A
lambda = polyroots((a11*a22-a12*a21,-a11-a22,1))
lambda
abs(lambda)

end


rename m1 y
rename m2 x
```

```
// This section produces a time series of y and x
generate t = _n
tsset t
tsline y x in -30/l
```

"VECrank" program

```
clear
set more off
graph drop _all
set scheme s2mono
set matsize 2000
set seed 13
matrix m=J(1000,2,0)
svmat m

mata:

// This section fixes the parameter values of the DGP
b10 =    0
b12 =    2
g11 =    1.9
g12 =    -1.9
b20 =    0
b21 =    5
g21 =    -1.4
g22 =    1.4

// This section maps the b* and g* parameters to the matrix formulation of the VEC
a11 = (g11+b12*g21)/(1-b12*b21)
a12 = (g12+b12*g22)/(1-b12*b21)
a21 = (g21+b21*g11)/(1-b12*b21)
a22 = (g22+b21*g12)/(1-b12*b21)
a10 = (b10+b12*b20)/(1-b12*b21)
a20 = (b20+b21*b10)/(1-b12*b21)
a13 = 1/(1-b12*b21)
a14 = b12/(1-b12*b21)
a23 = b21/(1-b12*b21)
a24 = 1/(1-b12*b21)
A0 = a10\a20
A1 = a11,a12\a21,a22
A2 = a13,a14\a23,a24

// This section creates the data
st_view(m=.,.,.)
for (i=2; i<=rows(m); i++) {
    u = rnormal(1,2,0,1)
    m[i,.]=A0'+ m[i-1,.]*A1'+ u*A2'
}

A1I = A1-I(2)
rank(A1I)

end

rename m1 y
rename m2 x
```

```
// The "vecrank" command estimates the rank of the matrix AI.
generate t = _n
tsset t
vecrank y x, max ic
```

“VECstable” program

```
clear
set more off
graph drop _all
set scheme s2mono
set matsize 2000
set seed 13
matrix m=J(1000,2,0)
svmat m

mata:

// This section fixes the parameter values of the DGP
b10 =    0
b12 =    2
g11 =    1.9
g12 =    -1.9
b20 =    0
b21 =    5
g21 =    -1.4
g22 =    1.4

// This section maps the b* and g* parameters to the matrix formulation of the VEC
a11 = (g11+b12*g21)/(1-b12*b21)
a12 = (g12+b12*g22)/(1-b12*b21)
a21 = (g21+b21*g11)/(1-b12*b21)
a22 = (g22+b21*g12)/(1-b12*b21)
a10 = (b10+b12*b20)/(1-b12*b21)
a20 = (b20+b21*b10)/(1-b12*b21)
a13 = 1/(1-b12*b21)
a14 = b12/(1-b12*b21)
a23 = b21/(1-b12*b21)
a24 = 1/(1-b12*b21)
A0 = a10\a20
A1 = a11,a12\a21,a22
A2 = a13,a14\a23,a24
// This section creates the data
st_view(m=.,.,.)
for (i=2; i<=rows(m); i++) {
    u = rnormal(1,2,0,1)
    m[i,.]=A0'+ m[i-1,.]*A1'+ u*A2'
}

end

rename m1 y
rename m2 x

// The "vecrank" command estimates the rank of the matrix AI.
generate t = _n
tsset t
vec y x
vecstable
```

“function.data” program

```
mata

mata drop rvars()

function rvars(b10,b12,g11,g12,b20,b21,g21,g22)
{
a11 = (g11+b12*g21)/(1-b12*b21)
a12 = (g12+b12*g22)/(1-b12*b21)
a21 = (g21+b21*g11)/(1-b12*b21)
a22 = (g22+b21*g12)/(1-b12*b21)
a10 = (b10+b12*b20)/(1-b12*b21)
a20 = (b20+b21*b10)/(1-b12*b21)
a13 = 1/(1-b12*b21)
a14 = b12/(1-b12*b21)
a23 = b21/(1-b12*b21)
a24 = 1/(1-b12*b21)
A0 = a10\a20
A1 = a11,a12\a21,a22
A2 = a13,a14\a23,a24

// This section creates the data
st_view(m=.,.,.)
for (i=2; i<=rows(m); i++) {
    u = rnormal(1,2,0,1)
    m[i,.]=A0'+ m[i-1,.]*A1'+ u*A2'
}

}

end
```

"Table2A" program

```
program drop _all

program define URootprog, rclass
version 13

syntax, b10(real) b12(real) g11(real) g12(real) ///
      b20(real) b21(real) g21(real) g22(real) lagz(real)
drop _all
matrix m=J(150,2,0)
svmat m

mata: rvars(`b10',`b12',`g11',`g12',`b20',`b21',`g21',`g22')

rename m1 y
rename m2 x
generate t=_n
tsset t
generate z = 0
replace z = L.z + rnormal() in 2/l
drop t
keep in -100/l
generate t = _n
tsset t


// This section performs unit root test with "lagz" lags for the dfuller test.
// Note that both the pperron and dfgls tests have automatic selection of lags
// In the dfgls case, we use SIC to choose the appropriate number of lags
// In the program below, we return the pvalue of the unit root test for
// the dfuller and pperron tests.  However, the dfgls test does not provide
// a pvalue.  Accordingly, we save the test statistic and compare it with the
// 5% critical value.  For sample sizes of 100 and all lags, the 5% critical value
// reported by Stata is -3.030

dfuller x, trend lags(`lagz')
return scalar pdfullerX = r(p)

regress D.x L.x t L(1/`lagz')D.x
estat bgodfrey
matrix bob = r(p)
return scalar pbgX = bob[1,1]

pperron x, trend
return scalar pperronX = r(pval)

dfgls x, ers
local sclag = r(sclag)
matrix bob=r(results)
return scalar pdfglsX = bob[13-r(sclag),5]

dfuller y, trend lags(`lagz')
return scalar pdfullerY = r(p)
```

```
regress D.y L.y t L(1/`lagz')D.y
estat bgodfrey
matrix bob = r(p)
return scalar pbgY = bob[1,1]

pperron y, trend
return scalar pperronY = r(pval)

dfgls y, ers
local sclag = r(sclag)
matrix bob=r(results)
return scalar pdfglsY = bob[13-r(sclag),5]

dfuller z, trend lags(`lagz')
return scalar pdfullerZ = r(p)

regress D.z L.z t L(1/`lagz')D.z
estat bgodfrey
matrix bob = r(p)
return scalar pbgZ = bob[1,1]

pperron z, trend
return scalar pperronZ = r(pval)

dfgls z, ers
local sclag = r(sclag)
matrix bob=r(results)
return scalar pdfglsZ = bob[13-r(sclag),5]


end
```

"Table2B" program

```
// This program took approximately 4 mins to run on my laptop.

// The programs must be run in the following order: (i) function_data,
// (ii) Table2A, and (iii) Table2B

// The program "etime" is a user-written .do file that can be obtained online

etime, start
drop _all
clear
set more off
graph drop _all
set scheme s2mono
set seed 13
matrix RESULTS = J(3,3,0)
matrix BGODFREY = J(1,3,0)

// This section fixes the parameter values of the DGP
local b10 =         0
local b12 =         2
local g11 =         1.9
local g12 =         -1.9
local b20 =         0
local b21 =         5
local g21 =         -1.4
local g22 =         1.4

local lagz = 5
local reps = 1000

simulate pdfullerX = r(pdfullerX) pdfullerY = r(pdfullerY) pdfullerZ = r(pdfullerZ) ///
      pperronX=r(pperronX) pperronY=r(pperronY) pperronZ = r(pperronZ) ///
      pdfglsX =r(pdfglsX) pdfglsY =r(pdfglsY) pdfglsZ =r(pdfglsZ) ///
      pbgX =r(pbgX) pbgY =r(pbgY) pbgZ =r(pbgZ) , ///
      reps(`reps'): URootprog, b10(`b10') b12(`b12') g11(`g11') g12(`g12') ///
      b20(`b20') b21(`b21') g21(`g21') g22(`g22') lagz(`lagz')

// The following commands summarize the results of the respective hypothesis
// tests.  The RESULTS matrix collects the tests of unit from the (i) augmented
// Dickey Fuller test, (ii) the Phillips-Perron test, and (iii) the Dickey-Fuller
// GLS test.  The Phillips-Perron and DF GLS tests automatically select the
// lags. For the augmented DF test, we set lags = 5, but then test to see if
// this eliminates serial correlation.

// The results of the Breusch-Godfrey test for serial correlation corresponding
// to a specification of 5 lags in the augmented DF regression are
// reported in the BGODFREY matrix.

// Note that the variables x and y are cointegrated with the parameters given
// above.  The variable z is bog standard random walk variable with no drift.
// The variable z is included for comparison's sake.  All Type I error rates
// should equal 0.05 for z.
```

```
generate RRdfX = 0
replace RRdfX = cond(pdfullerX<0.05,1,0)
summ RRdfX, meanonly
matrix RESULTS[1,1] = r(mean)

generate RRdfY = 0
replace RRdfY = cond(pdfullerY<0.05,1,0)
summ RRdfY, meanonly
matrix RESULTS[1,2] = r(mean)

generate RRdfZ = 0
replace RRdfZ = cond(pdfullerZ<0.05,1,0)
summ RRdfZ, meanonly
matrix RESULTS[1,3] = r(mean)

generate RRppX = 0
replace RRppX = cond(pperronX<0.05,1,0)
summ RRppX, meanonly
matrix RESULTS[2,1] = r(mean)

generate RRppY = 0
replace RRppY = cond(pperronY<0.05,1,0)
summ RRppY, meanonly
matrix RESULTS[2,2] = r(mean)

generate RRppZ = 0
replace RRppZ = cond(pperronZ<0.05,1,0)
summ RRppZ, meanonly
matrix RESULTS[2,3] = r(mean)

generate RRglsX = 0
replace RRglsX = cond(pdfglsX<-3.03,1,0)
summ RRglsX, meanonly
matrix RESULTS[3,1] = r(mean)

generate RRglsY = 0
replace RRglsY = cond(pdfglsY<-3.03,1,0)
summ RRglsY, meanonly
matrix RESULTS[3,2] = r(mean)

generate RRglsZ = 0
replace RRglsZ = cond(pdfglsZ<-3.03,1,0)
summ RRglsZ, meanonly
matrix RESULTS[3,3] = r(mean)

generate RRbgX = 0
replace RRbgX = cond(pbgX<0.05,1,0)
summ RRbgX, meanonly
matrix BGODFREY[1,1] = r(mean)

generate RRbgY = 0
replace RRbgY = cond(pbgY<0.05,1,0)
summ RRbgY, meanonly
matrix BGODFREY[1,2] = r(mean)
```

```
generate RRbgZ = 0
replace RRbgZ = cond(pbgZ<0.05,1,0)
summ RRbgZ, meanonly
matrix BGODFREY[1,3] = r(mean)

// These commands print out the results
matrix colnames RESULTS = X Y Z
matrix rownames RESULTS = DFuller PPerron DFGLS
matrix colnames BGODFREY = X Y Z
matrix rownames BGODFREY = BGodfrey
matrix list RESULTS
matrix list BGODFREY
display as text "Number of lags = `lagz'"

etime
```

"Table3A" program

```
program drop _all

program define URootprog, rclass
version 13

syntax, b10(real) b12(real) g11(real) g12(real) ///
      b20(real) b21(real) g21(real) g22(real) lagz(real)
drop _all
matrix m=J(150,2,0)
svmat m

mata: rvars(`b10',`b12',`g11',`g12',`b20',`b21',`g21',`g22')

rename m1 y
rename m2 x
generate t=_n
tsset t
generate z = 0
replace z = L.z + rnormal() in 2/l

//I create the lagged differenced terms first, then keep the last 100
//observations. This ensures that all of the regressions have the same number
//of observations, no matter how many lagged differenced terms they have.
//This is necessary to ensure comparability of the AIC and SIC values.

generate LDx1 = L1D.x
generate LDx2 = L2D.x
generate LDx3 = L3D.x
generate LDx4 = L4D.x
generate LDx5 = L5D.x
generate LDx6 = L6D.x
generate LDx7 = L7D.x
generate LDx8 = L8D.x
generate LDx9 = L9D.x
generate LDx10 = L10D.x

generate LDy1 = L1D.y
generate LDy2 = L2D.y
generate LDy3 = L3D.y
generate LDy4 = L4D.y
generate LDy5 = L5D.y
generate LDy6 = L6D.y
generate LDy7 = L7D.y
generate LDy8 = L8D.y
generate LDy9 = L9D.y
generate LDy10 = L10D.y

generate LDz1 = L1D.z
generate LDz2 = L2D.z
generate LDz3 = L3D.z
generate LDz4 = L4D.z
generate LDz5 = L5D.z
generate LDz6 = L6D.z
```

```
generate LDz7 = L7D.z
generate LDz8 = L8D.z
generate LDz9 = L9D.z
generate LDz10 = L10D.z

drop t
keep in -100/l
generate t = _n
tsset t

// This section performs unit root test with "lagz" lags for the dfuller test.
// Note that both the pperron and dfgls tests have automatic selection of lags
// In the dfgls case, we use SIC to choose the appropriate number of lags
// In the program below, we return the pvalue of the unit root test for
// the dfuller and pperron tests.  However, the dfgls test does not provide
// a pvalue.  Accordingly, we save the test statistic and compare it with the
// 5% critical value.  For sample sizes of 100 and all lags, the 5% critical value
// reported by Stata is -3.030

dfuller x, trend lags(`lagz')
return scalar pdfullerX = r(p)

regress D.x L.x t LDx1-LDx`lagz'
estat bgodfrey
matrix bob1 = r(p)
return scalar pbgX = bob1[1,1]
estat ic
matrix bob2= r(S)
return scalar aicX = bob2[1,5]
return scalar sicX = bob2[1,6]

dfuller y, trend lags(`lagz')
return scalar pdfullerY = r(p)

regress D.y L.y t LDy1-LDy`lagz'
estat bgodfrey
matrix bob1 = r(p)
return scalar pbgY = bob1[1,1]
estat ic
matrix bob2= r(S)
return scalar aicY = bob2[1,5]
return scalar sicY = bob2[1,6]

dfuller z, trend lags(`lagz')
return scalar pdfullerZ = r(p)

regress D.z L.z t LDz1-LDz`lagz'
estat bgodfrey
matrix bob1 = r(p)
return scalar pbgZ = bob1[1,1]
estat ic
matrix bob2= r(S)
return scalar aicZ = bob2[1,5]
return scalar sicZ = bob2[1,6]


end
```

"Table3B" program

```
// This program takes approximately 10 mins to run on my laptop.

// The programs must be run in the following order: (i) function_data,
// (ii) Table3A, and (iii) Table3B

// The program "etime" is a user-written .do file that can be obtained online

etime, start
drop _all
clear
set more off
graph drop _all
set scheme s2mono
set seed 13
matrix RESULTS = J(10,3,0)
matrix BGODFREY = J(10,3,0)
matrix AIC = J(10,3,0)
matrix SIC = J(10,3,0)

// This section fixes the parameter values of the DGP
local b10 =         0
local b12 =         2
local g11 =         1.9
local g12 =         -1.9
local b20 =         0
local b21 =         5
local g21 =         -1.4
local g22 =         1.4

local reps = 1000

local i = 1
forvalues lagz = 1/10 {
simulate pdfullerX = r(pdfullerX) pdfullerY = r(pdfullerY) pdfullerZ = r(pdfullerZ) ///
      pbgX =r(pbgX) pbgY =r(pbgY) pbgZ =r(pbgZ) ///
      aicX = r(aicX) aicY = r(aicY) aicZ = r(aicZ) ///
      sicX = r(sicX) sicY = r(sicY) sicZ = r(sicZ) , ///
      reps(`reps'): URootprog, b10(`b10') b12(`b12') g11(`g11') g12(`g12') ///
      b20(`b20') b21(`b21') g21(`g21') g22(`g22') lagz(`lagz')

// The RESULTS matrix collects the results of augmented Dickey Fuller tests
// for different lags.

// The BGODFREY matrix collects the results of Breusch-Godfrey tests
// for serial correlation with the specified number of lagged
// differences in the DF regression equation.

// Note that the variables x and y are cointegrated with the parameters given
// above.  The variable z is bog standard random walk variable with no drift.
// The variable z is included for comparison's sake.  All Type I error rates
// should equal 0.05 for z.
```

```
generate RRdfX = 0
replace RRdfX = cond(pdfullerX<0.05,1,0)
summ RRdfX, meanonly
matrix RESULTS[`i',1] = r(mean)

generate RRdfY = 0
replace RRdfY = cond(pdfullerY<0.05,1,0)
summ RRdfY, meanonly
matrix RESULTS[`i',2] = r(mean)

generate RRdfZ = 0
replace RRdfZ = cond(pdfullerZ<0.05,1,0)
summ RRdfZ, meanonly
matrix RESULTS[`i',3] = r(mean)

generate RRbgX = 0
replace RRbgX = cond(pbgX<0.05,1,0)
summ RRbgX, meanonly
matrix BGODFREY[`i',1] = r(mean)

generate RRbgY = 0
replace RRbgY = cond(pbgY<0.05,1,0)
summ RRbgY, meanonly
matrix BGODFREY[`i',2] = r(mean)

generate RRbgZ = 0
replace RRbgZ = cond(pbgZ<0.05,1,0)
summ RRbgZ, meanonly
matrix BGODFREY[`i',3] = r(mean)

summ aicX, meanonly
matrix AIC[`i',1] = r(mean)

summ aicY, meanonly
matrix AIC[`i',2] = r(mean)

summ aicZ, meanonly
matrix AIC[`i',3] = r(mean)

summ sicX, meanonly
matrix SIC[`i',1] = r(mean)

summ sicY, meanonly
matrix SIC[`i',2] = r(mean)

summ sicZ, meanonly
matrix SIC[`i',3] = r(mean)

local `++i'
}

// These commands print out the results
matrix colnames RESULTS = X Y Z
matrix rownames RESULTS = LAG1 LAG2 LAG3 LAG4 LAG5 LAG6 LAG7 LAG8 LAG9 LAG10
matrix colnames BGODFREY = X Y Z
matrix rownames BGODFREY = LAG1 LAG2 LAG3 LAG4 LAG5 LAG6 LAG7 LAG8 LAG9 LAG10
matrix colnames AIC = X Y Z
```

```
matrix rownames AIC = LAG1 LAG2 LAG3 LAG4 LAG5 LAG6 LAG7 LAG8 LAG9 LAG10
matrix colnames SIC = X Y Z
matrix rownames SIC = LAG1 LAG2 LAG3 LAG4 LAG5 LAG6 LAG7 LAG8 LAG9 LAG10
matrix list RESULTS
matrix list BGODFREY
matrix list AIC
matrix list SIC

etime
```