Econometrics Basics: Investigating the Stationarity of Time Series

John E. Floyd
University of Toronto
July 22, 2013

It is often useful to think of a time series as a sequence of numbers generated by some mathematical process and to attempt to determine which process best describes the behavior of that series.[1] For example, a particular series might be generated according to the equation

$$y_t = \beta\, y_{t-1} + \epsilon_t \tag{1}$$

where y_t is the level of the series at time t and ϵ_t is a series of drawings of a zero-mean, constant-variance non-autocorrelated random variable. The above equation is a first-order autoregressive process—first order because there is one lag of y_t on the right-hand side and autoregressive because the y_t are autocorrelated in the sense that the level of the variable in a particular period depends on its level in one or more previous periods.

Lagging the above equation repeatedly and substituting these lags back into (1) produces the expression

$$\begin{aligned} y_t = {} & \beta^t\, y_0 + \epsilon_t + \beta\, \epsilon_{t-1} + \beta^2 \epsilon_{t-2} + \beta^3\, \epsilon_{t-3} \\ & + \beta^4\, \epsilon_{t-4} + \cdots\cdots\cdots + \beta^t\, \epsilon_0. \end{aligned} \tag{2}$$

The time path of y_t depends critically on the parameter β. If this parameter equals zero then

$$y_t = \epsilon_t. \tag{3}$$

[1] A very basic discussion of time series analysis can be found in Walter Enders, *Applied Time Series Analysis*, John Wiley and Sons, 1995. For a deeper analysis, see James Hamilton, *Time Series Analysis*, Princeton University Press, 1994.

and y_t is itself a white noise process with mean equal to 0 and variance equal to the variance of ϵ_t, denoted here by σ^2. As can be seen from (1), when $\beta = 1$ equation (2) reduces to

$$y_t \;=\; y_{t-1} + \epsilon_t \tag{4}$$

and the series becomes a random walk—the level of y_t wanders without limit, moving either up or down in each period compared to its previous value by a random amount ϵ_t as can be seen by rewriting (4) as

$$y_t - y_{t-1} \;=\; \epsilon_t \tag{5}$$

or, by expansion, as

$$y_t \;=\; \epsilon_t + \epsilon_{t-1} + \epsilon_{t-2} + \epsilon_{t-3} + \cdots\cdots + \epsilon_0. \tag{6}$$

The variance of y_t will then equal $(\sigma^2 + \sigma^2 + \sigma^2 + \cdots\cdots)$ which will grow in proportion to the number of periods over which the variance is being calculated.

When $\beta = 1$ the series is said to be non-stationary or have a unit root. Its expected level at any point in time is its current level and its variance in the limit is infinity. Its future path need never pass through the level at which it started or any other level previously achieved, although there is no reason why it could not return to those levels. When $|\beta| > 1$ the series explodes, with the values of y_t getting increasingly larger or smaller with time. When $-1 < \beta < 1$ the series is stationary as can be seen from the fact that $|\beta^t|$ gets smaller in (2) as t increases. If the ϵ_t are zero beyond some point, y_t will approach zero as t increases, with the speed of approach being greater, the smaller is $|\beta|$. The effects of each ϵ_t shock will thus dissipate with time. The variance of y_t will equal $[1 + \beta^2 + (\beta^2)^2 + (\beta^3)^2 + \cdots\cdots\cdots]\,\sigma^2$ which will be finite in the limit as t increases, ultimately reaching $1/(1-\beta^2)$.[2] The series will vary around zero with a persistence that will be greater as β gets larger.

[2]This uses the following relationships.

$$\begin{aligned} Var\{u+v\} &= Var\{u\} + Var\{v\} \\ Var\{a\,x\} &= a^2\,Var\{x\} \\ 1 + a + a^2 + a^3 + a^4 + \cdots\cdots &= \frac{1}{1-a}. \end{aligned}$$

Equation (1) is a first-order autoregressive process. An autoregressive process of second-order would be represented by

$$y_t \;=\; \beta_1\, y_{t-1} + \beta_2\, y_{t-2} + \epsilon_t, \tag{7}$$

with two lags of y_t, and third and higher order processes simply inolve the addition of further lags.

Time series can also be moving average processes such as, for example,

$$y_t \;=\; \xi_0\, \epsilon_t + \xi_1\, \epsilon_{t-1} + \xi_2\, \epsilon_{t-2} \tag{8}$$

which is a second-order moving average process—second-order because it contains two lags of the error term. Moving average processes are always stationary because ϵ_t is stationary and any average of stationary processes must itself be a stationary process.

Of course, time series processes can have both autoregressive and moving average components as in the expression

$$y_t \;=\; \beta_1\, y_{t-1} + \beta_2\, y_{t-2} + \xi_0\, \epsilon_t + \xi_1\, \epsilon_{t-1} + \xi_2\, \epsilon_{t-2} \tag{9}$$

which defines an ARMA(2,2) process—a process that is second-order autoregressive and second-order moving average. In general, ARMA(p, q) processes have p autoregressive lags and q moving average lags.

It is possible that y_t above could be a stationary process that is actually the first difference of another series, $y_t = z_t - z_{t-1}$, where z_t is a non-stationary autoregressive moving average process that has to be differenced once to produce the stationary ARMA(2,2) process. It is said to be integrated of order 1 because it has to be differenced once to produce a stationary process. If it had to be differenced twice to produce a stationary process it would be integrated of order 2, and so forth. The process z_t is thus an autoregressive-integrated-moving-average ARIMA(2,1,2) process—differencing it once produces an ARMA(2,2) process. In general, an ARIMA(p, d, q) process is one whose d-th difference is a stationary autoregressive-moving-average process with p autoregressive lags and q moving average lags.

Subtraction of y_{t-1} from both sides of (9) converts it to

$$\begin{aligned} y_t - y_{t-1} &= -(1-\beta_1)\, y_{t-1} + \beta_2\, y_{t-2} \\ &\quad + \xi_0\, \epsilon_t + \xi_1\, \epsilon_{t-1} + \xi_2\, \epsilon_{t-2} \end{aligned} \tag{10}$$

and a further addition and subtraction of $\beta_2\, y_{t-1}$ on the right side yields

$$\begin{aligned} y_t - y_{t-1} &= -(1-\beta_1-\beta_2)\, y_{t-1} + \beta_2\, (y_{t-2} - y_{t-1}) \\ &\quad + \xi_0\, \epsilon_t + \xi_1\, \epsilon_{t-1} + \xi_2\, \epsilon_{t-2}. \end{aligned} \tag{11}$$

This series will be stationary as long as $\beta_1 + \beta_2 < 1$ so that a fraction $(1 - \beta_1 - \beta_2)$ of any change in y_t will be removed in each subsequent period. A random-walk will only occur if $\beta_1 + \beta_2 = 1$, in which case expression reduces to

$$y_t - y_{t-1} = \xi_0\, \epsilon_t + \xi_1\, \epsilon_{t-1} + \xi_2\, \epsilon_{t-2} \tag{12}$$

and any change in y_t from period to period will be permanent.

It turns out that an equation like (9) that includes autoregressive and moving-average terms can be expressed in the form of a pure autoregressive process containing an infinite number of autoregressive lags.[3] Simply reorganize (9) to move ϵ_t to the left of the equality and y_t to the right, lag the resulting equation repeatedly to obtain expressions for e_{t-1}, e_{t-2}, e_{t-3} ... etc. and substitute these expressions successively into (9) and simplify. The resulting infinite order autoregressive process can then be converted into an equation like

$$\begin{aligned} \Delta y_t &= -(1-\rho)\, y_{t-1} + (\beta_2 + \beta_3 + \beta_4 + \cdots + \beta_\infty)\, \Delta y_{t-1} \\ &\quad + (\beta_3 + \beta_4 + \cdots + \beta_\infty)\, \Delta y_{t-2} \\ &\quad + (\beta_4 + \cdots + \beta_\infty)\, \Delta y_{t-3} + \cdots\cdots\cdots + \epsilon_t \end{aligned} \tag{13}$$

containing an infinite succession of lags of $\Delta y_t = (y_t - y_{t-1})$ where the stationarity parameter $\rho = \beta_1 + \beta_2 + \beta_3 + \cdots\cdots + \beta_\infty$. Stationarity occurs where $\rho < 1$ and therefore $-(1-\rho) < 0$.

[3] See pages 225–227 of the book by Enders.

Our econometric problem here is to determine whether the time-series processes that can reasonably describe the evolution of particular time-series variables are stationary. The standard procedure, based on path-breaking work by Dickey and Fuller[4] is to perform a **Dickey-Fuller Test** which uses ordinary-least-squares to estimate an equation of the form

$$\begin{aligned} \Delta q_t &= \alpha + \gamma\, t - (1-\rho)\, q_{t-1} + \delta_1\, \Delta q_{t-1} + \delta_2\, \Delta q_{t-2} \\ &\qquad + \delta_3\, \Delta q_{t-3} + \cdots\cdots\cdots + \epsilon_t \end{aligned} \tag{14}$$

which is equivalent to the infinite autoregressive process discussed above with the addition of a constant term and trend. It turns out that, under the null hypothesis that there is no mean reversion and $\rho = 1$, this process can be well approximated by an AR process containing no more than $T^{1/3}$ lags, where T is the number of observations. The deterministic terms α and $\gamma\, t$ are dropped if there is no evidence of a trend in the series—if α is significantly different from zero, the rejection of the null-hypothesis of $\rho = 1$ indicates stationarity of the series around a drift or trend and if γ is significantly different from zero the series is stationary around an increasing or decreasing drift or trend. Two equations additional to the one above are estimated, one without $\gamma\, t$ and a second without $\alpha + \gamma\, t$.

$$\begin{aligned} \Delta q_t &= \alpha - (1-\rho)\, q_{t-1} + \delta_1\, \Delta q_{t-1} + \delta_2\, \Delta q_{t-2} \\ &\qquad + \delta_3\, \Delta q_{t-3} + \cdots\cdots\cdots + \epsilon_t \end{aligned} \tag{15}$$

$$\begin{aligned} \Delta q_t &= -(1-\rho)\, q_{t-1} + \delta_1\, \Delta q_{t-1} + \delta_2\, \Delta q_{t-2} \\ &\qquad + \delta_3\, \Delta q_{t-3} + \cdots\cdots\cdots + \epsilon_t\,. \end{aligned} \tag{16}$$

In selecting the number of lags to be included, an appropriate procedure is to start with an unreasonably large number and progressively drop the longest lag if that lag turns out to be statistically insignificant. An alternative is to choose the number of lags that minimizes an information criterion such as the Akaike information criterion (AIC) or Schwartz Bayesian information criterion (SBC). These give calculated optimal balances between the gain associated with the reduction in the residual sum of squares when a lag is

[4]David Dickey and Wayne A. Fuller, "Distribution of the Estimates for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association* Vol. 74, June 1979, 427-431, and "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica* Vol. 49, July 1981, 1957-72.

added and the loss associated with having one less degree of freedom. The relevant formulae to be calculated for each regression are

$$AIC(n) = ln\left(\frac{SSR(n)}{T}\right) + (n+1)\frac{2}{T} \quad (17)$$

$$SBC(n) = ln\left(\frac{SSR(n)}{T}\right) + (n+1)\frac{ln(T)}{T} \quad (18)$$

were $SSR(n)$ is the sum of squared residuals, n is the number of lags and T is the number of observations and where $ln()$ is represents the natural logarithm of the expression in the brackets.[5] Of course, all these significance tests and criteria comparisons must apply to regressions estimated from the same number of observations.

It turns out that under the null-hypothesis that $\rho = 1$ the estimator of $(1-\rho)$ is not distributed according to the t-distribution. A table of critical values constructed by Dickey and Fuller must be used instead of the standard t-tables.[6]

Our first task in writing a function in `XLispStat` to do Dickey-Fuller tests is to develop an easy method of deciding how many lags to use. To do this, we construct a `DFlags` function. As a prelude to this, we modify our `makelags` function very slightly, inserting the word `lagmat` on a line by itself at the very end of the function.

```
) ; end of dotimes
lagmat
) ; end of function
```

This code line makes the matrix `lagmat` the designated output of the function, rather than just one of a number of objects left in the workspace. As a result, we can use the `def` function to give it another name as follows.

```
(def newmatrix (makelags 5 variable))
```

[5] See James H. Stock and Mark W. Watson, *Introduction to Econometrics*, Addison Wesley, 2003, pages 453–467, for a discussion of these criteria. The authors recommend the AIC over the SBC for the purpose at hand because the former tends to overestimate the number of lags and studies of the performance of Dickey-Fuller tests suggest that having too many lags is better than having too few.

[6] A collection of these and other tables can be found in the file `statabs.pdf`.

We now construct our function to aid in deciding the number of lags to pass to the Dickey-Fuller unit root function we will subsequently construct. The argument `y` is the variable we are checking for stationarity and the argument `m` denotes the maximum allowable number of lags. We also use the `regression-model` function provided by `XLispStat` instead of our `runOLS` function in both of the new functions we create.

```
(defun DFlags (y m)
"Args: (y m)
Provides information to enable us to decide how many lags to use in a
subsequent Dickey-Fuller unit root test of the series y starting with
a maximum lag m. The AIC and SBC statistics are expressed as fractions
of the levels that resulted with the maximum lag."
(def dimlist (list (+ m 2) m))
```

Our first line of code above creates a list giving the dimensions of the matrix we will construct to contain the P-Values of the lags in a series of regressions starting with a lag of `m` and sequentially reducing the number of lags by one until only a single lag remains. This matrix will have along the top of it the AIC and SBC values associated with each of the regressions. We now make the shell of that matrix, into which the proper values will be later inserted, together with a list that will represent its first (left-most) column.

```
(def bigmat (make-array dimlist :initial-element " "))
(def col1list (combine "AIC" "BIC" (iseq 1 m)))
(def lagy (remove-last 1 y))
(def diffy (- (remove-first 1 y)(remove-last 1 y)))
(def lagmat (makelags m diffy))
(def lagy (remove-first m lagy))
(def diffy (select lagslist 0))
```

The last five lines of code above create the first difference of the variable to which we are going to apply our Dickey-Fuller test, then set up a matrix of the maximum number of lags of $\mathbf{\Delta y}$, and finally, the two series representing the current values of $\mathbf{\Delta y}$ and $\mathbf{y_{t-1}}$ with lengths appropriate for the regressions that will follow. We then run the first regression (the one with the most lags) in the first code line below and in the subsequent four lines of code extract the t-ratios and calculate the P-values of the lags and then calculate the values of the AIC and SBC for this first regression.

```
(def dfreg (regression-model (bind-columns lagy lagmat) diffy :print nil))
(def trats (/ (send dfreg :coef-estimates)(send dfreg :coef-standard-errors)))
(def Pvals (- 1 (t-cdf (abs trats)(send dfreg :df))))
(def AICr (+ (* (length diffy)(log (send dfreg :residual-sum-of-squares)))
(* 2 (length Pvals))))
(def SBCr (+ (* (length diffy)(log (send dfreg :residual-sum-of-squares)))
(* (length Pvals)(log (length diffy)))))
```

Next we set these first values of the AIC and SBC as base values and then, as we will do with all subsequent AIC and SBC values, take the ratios of them to these first-period base levels, which will produce, of course, AIC and SBC values of unity for the regression with the longest lag. We express all AIC and SBC as ratios of these initial values to narrow the columns of our matrix, enabling us to print as many as 24 columns on the maximized `XLispStat` screen. In the process we insert these first values of the AIC and SBC as the first two elements of the first column of our matrix. And then, using a `dotimes` loop, we insert the P-Values of the `m` lags in the first regression into that column. We should note here that, while the other t-statistics will have to be evaluated using the Dickey-Fuller tables, the conventional t-statistics are appropriate for determining the statistical significance of the lags.

```
(def AICbase AICr)
(def SBCbase SBCr)
(setf (aref bigmat 0 0) (/ AICr AICbase))
(setf (aref bigmat 1 0) (/ SBCr SBCbase))
(dotimes (j m)
(setf (aref bigmat (+ j 2) 0)(select Pvals (+ j 2)))
) ;end dotimes j
```

Now we begin a `dotimes` loop of length equal to the number of lags minus one, dropping one lag and re-running the regression at each pass through the loop, calculating the AIC and SBC for each regression and inserting the values appropriately into our matrix of results. And within each of these loops we embed another `dotimes` loop to select the P-Values for the each of the lags associated with the current regression and insert them into the column of results for that regression.

```
(dotimes (i (- m 1))
(def lagmat (remove-last-columns 1 lagmat))
(def dfreg (regression-model (bind-columns lagy lagmat) diffy :print nil))
(def trats (/ (send dfreg :coef-estimates)(send dfreg :coef-standard-errors)))
(def Pvals (- 1 (t-cdf (abs trats)(send dfreg :df))))
(def AICr (+ (* (length diffy)(log (send dfreg :residual-sum-of-squares)))
(* 2 (length Pvals))))
(def SBCr (+ (* (length diffy)(log (send dfreg :residual-sum-of-squares)))
(* (length Pvals)(log (length diffy)))))
(setf (aref bigmat 0 (+ i 1)) (/ AICr AICBASE))
(setf (aref bigmat 1 (+ i 1)) (/ SBCr SBCBASE))
(dotimes (j (- m i 1))
(setf (aref bigmat (+ j 2)(+ i 1))(select Pvals (+ j 2)))
) ;end dotimes j
) ;end dotimes i
```

Finally, after binding the first column to the rest of our matrix we print the results that have been inserted above into that matrix. The code for doing this is straight-forward, using the `princ`, `terpri` and `format` functions as usual.

```
(def bigmat (bind-columns col1list bigmat))
(def rcnum (array-dimensions bigmat))
(terpri)
(princ "AIC and SBC (first-two rows) and P-Values of up to ")(princ m)
(princ " Dickey-Fuller Lags")(terpri)
(terpri)
(dotimes (j (select rcnum 0))
(dotimes (i (select rcnum 1))(if (= i 0)(format t "~3d" (aref bigmat j i))
(format t "~6,3f" (aref bigmat j i)))
) ; end dotimes i
(terpri)
) ; end dotimes j
) ; end function
```

We now run our `DFlags` function, after setting up our variables, as follows, using 12 as the maximum number of lags.[7]

[7]This code is in the file `unitroot.lsp` and uses the data file `causdata.lsp`, producingthe output in `unitroot.lou`.

```
; XLISPSTAT BATCH FILE FOR UNIT ROOT TESTS
;
(load "ourfuncs")
(load "causqdat")
(variables)
;
; SET UP CANADA/U.S. REAL EXCHANGE RATE
;
(def rexcaus (remove-first 8 (* 100 (/ cpica (* exrcaus cpius)))))
;
(def plotrex (plot-lines (- dates74 1900) rexcaus
:title "Canada vs. U.S. Real Exchange Rate: Index -- 1974 = 100"))
;
(DFlags rexcaus 12)
```

In addition to a plot of the real exchange rate series in a separate window, this code produces the following results.

```
; loading E:\DSLMHTML\SPURREG\unitroot.lsp
; loading ourfuncs.lsp
; loading causqdat.lsp

AIC and SBC (first-two rows) and P-Values of up to 12 Dickey-Fuller Lags

AIC 1.000 0.998 0.997 0.994 0.992 0.990 0.990 0.990 0.988 0.989 0.990 0.989
SBC 1.000 0.995 0.990 0.985 0.980 0.975 0.972 0.969 0.963 0.961 0.958 0.955
  1 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  2 0.058 0.052 0.052 0.050 0.051 0.051 0.045 0.025 0.028 0.048 0.128
  3 0.020 0.019 0.017 0.018 0.017 0.017 0.028 0.016 0.018 0.048
  4 0.062 0.063 0.075 0.073 0.070 0.076 0.107 0.060 0.067
  5 0.415 0.389 0.384 0.379 0.391 0.414 0.465 0.322
  6 0.275 0.270 0.284 0.270 0.248 0.220 0.085
  7 0.055 0.056 0.070 0.072 0.073 0.076
  8 0.297 0.268 0.334 0.341 0.360
  9 0.392 0.364 0.390 0.411
 10 0.337 0.321 0.405
 11 0.233 0.188
 12 0.321
```

The maximum number of significant lags is three and, it turns out, the AIC is almost minimized with that number of lags and with one lag (although the minimum is with four lags), while the SBC is minimized with one lag.

Another way to decide how many lags to use is to examine the statistical significance of the autocorrelations and partial-autocorrelations of the first difference of the time-series variable, being tested for stationarity. For doing this we develop a function called `acfpacf`. The econometrics underlying this function is quite complicated—it is insufficient to simply calculate the correlations between the current and all lagged values of the series to obtain the autocorrelations and to run a single OLS regression including all lagged values to obtain the partial-autocorrelations. The standard approach, which is outlined by Davidson and MacKinnon in their graduate-level textbook, is to calculate the j^{th} autocorrelation as the covariance of the j^{th} lag of the variable with its current level divided by the variance of the current level.[8]

$$\hat{\rho}\,(j) \;=\; \frac{Cov(y_t, y_{t-j})}{Var(y_t)} \tag{19}$$

where

$$Cov(y_t, y_{t-j}) \;=\; \frac{1}{n-1}\sum_{t=j+1}^{n}(y_t - \bar{y})(y_{t-j} - \bar{y}) \tag{20}$$

and

$$Var(y_t) \;=\; \frac{1}{n-1}\sum_{t=1}^{n}(y_t - \bar{y})^2\,. \tag{21}$$

The function $Cov(\quad)$ defines the covariance of the two time-series in the brackets and the function $Var(\quad)$ defines the variance of the single series in the brakets. In a standard calculation of a correlation coefficient, the denominator of (19) would equal the product of the standard-deviations of the two time series in the numerator rather than the variance of one of them. And the number of elements summed in (20) is less than the number summed in (21) by an amount equal to the number of lags involved in the calculation of each particular element, or correlation coefficient, in the autocorrelation function. But the calculations in the above equations make good sense when

[8]Russell Davidson and James G. MacKinnon, *Econometric Theory and Methods*, Oxford University Press, 2004, pages 564 and 565.

we note that the purpose of calculating these autocorrelations is to see if we can reject the null-hypotheses that they are zero—lags included in our `dfunit` function will equal the set of lags up to the the longest statistically significant one. Under the null-hypothesis that there are no correlations between the lags, the expected variances, and therefore standard deviations, of the current series and all lags of that series should be the same in a large enough sample. Hence, the square-root of the variance of the entire unlagged series will give the best estimate of the standard deviations of all lags of that series. And the variance of the entire unlagged series will thus give the best estimate of the product of the standard deviations of the series and any of its lags. It is also desirable to use the maximum possible number of elements in calculating each of the covariances, taking account of the fact that in the numerator of (19) the creation of lags uses up, in each case, a number of elements equal to the length of the longest lag.

The reason why we need to examine the partial correlations is that, for example, the fourth lag will be correlated with the unlagged series because its level will be determined in part by the level of the thrice-lagged series, which will be determined in part by the level of the twice-lagged series, which will be determined in part by level of the the once-lagged series, even if only the first lag is statistically significant in regressions of the series on its first four lags. However, we do not want to determine the statistical significance of, say, the third lag of a series by regressing the series on the first ten of its lags because lags longer than three should not be allowed to influence the coefficients of the first three lags, as they might well do, because a decision to use three lags must be based on the statistical significance of those three lags, uncontaminated by the effects of adding further lags which, themselves, might not be statistically significant. Accordingly, the partial-autocorrelation function—that is the list of partial-autocorrelations—must be obtained by regressing the current level of the series on one lag and then using the coefficient of that lag as a measure of the first partial-autocorrelation and then adding additional lags in sequence, measuring the partial-autocorrelation of each of these lags by its coefficient when it is the longest lag. To calculate twelve partial-autocorrelations, therefore, we have to run twelve regressions, starting with one lag and adding additional lags in turn, recording in each case the coefficient of the added lag.

A final issue is the setting of confidence limits beyond which the null-hypotheses of zero autocorrelation and partial-autocorrelation can be re-

jected. Here we utilize the fact that the variance of each correlation is equal to $1/T$ where T is the number of observations. A two-tailed 5% confidence interval can thus be obtained by taking the square root of $1/T$ to obtain the standard deviation and then multiplying that standard deviation by 2 and assigning to the result both positive and negative values.[9]

The code for our `acfpacf` function, with interspersed comments, follows.

```
(defun acfpacf (y n l)
"Args: (y n l)
Calculates autocorrelation and partial-autocorrelation functions
for the variable y having name n with number of lags truncated
to l and plots the functions to the screen."
(def meany (/ (sum y)(length y)))
(def numlags l)
(def covars (repeat 0 (+ numlags 1)))
```

The last line above creates a list of zeros, called `covars`, of length equal to the number of lags plus 1. Each element of the list will be replaced by the corresponding covariance as it is calculated in the seven code lines below.

```
(dotimes (j (+ numlags 1))
(def covsum 0)
(dotimes (i (- (length y) j))
(def covsum (+ covsum (* (- (select y (+ i j)) meany)(- (select y  i) meany))))
) ; end dotimes i
(setf (select covars j)(* (/ 1 (- (length y) 1)) covsum))
) ; end dotimes j
```

A `dotimes j` loop with number of passes equal to the number of lags plus one is constructed above—the first pass creates the covariance of current value of the series with itself, followed by one pass for each lag. Within the `dotimes j` loop is embedded a `dotimes i` loop which sums up products of the deviations of the current value from its mean and the corresponding lagged value from its mean for each element in the series. This sum is then divided by the the number of observations minus one to produce the covariance for that lag

[9]For references relating to basis for this calculation as well as an excellent basic discussion of the theory underlying time series analysis, read the first four chapters of Chris Chatfield, *The Analysis of Time Series: An Introduction*, Chapman and Hall, 1997. The variance of the autocorrelation coefficients is discussed on page 51.

which is then appropriately inserted into the `covars` list. Then the first code line below calculates the variance of the series, calling it `vary`, and the second code line calculates the list `acf` containing the series of autocorrelations, calculated by dividing each element of `covars` by `vary`. In the process the first element of the resulting `acf` list is removed because it represents simply the covariance of `y` with itself. This element was initially included as a way of testing the code—if the correlation of the series with itself is not unity, the coding is is obviously wrong.

```
(def vary (* (/ 1 (- (length y) 1))(sum (^ (- y meany) 2))))
(def acf (remove-first 1 (/ covars vary)))
```

In the code below creates `pacf`, list of zeros each of which will be replaced by the appropriate individual partial-autocorrelation as it is calculated. A `dotimes` loop is then constructed which sequentially constructs matrices of lags, starting with one lag and adding additional ones until the maximum specified by `numlags` is reached. On each pass an OLS regression is calculated and the coefficient of the last lag is extracted and inserted appropriately into the `pacf` list.

```
(def pacf (repeat 0 numlags))
(dotimes (j numlags)
(def lagmat (makelags (+ j 1) y))
(def pacreg (regression-model lagmat (select lagslist 0) :print nil))
(def coeffs (send pacreg :coef-estimates))
(setf (select pacf j)(select coeffs (- (length coeffs) 1)))
) ; end dotimes j
```

The next two lines below calculate the upper and lower confidence levels, which are the same for all lags and equal to two times the reciprocal of the square root of the length of the series being examined. After adding an additional line to reconstruct the sequence of lag numbers and rename that sequence, we proceed to print out the results in the usual way.

```
(def UCL (/ 2 (sqrt (length y))))
(def LCL (/ -2 (sqrt (length y))))
;
(def lagnum (iseq 1 numlags)) ; insert line here to overwrite variable
;                              left in memory with the same name.
```

```
(princ "AUTOCORRELATIONS AND PARTIAL AUTOCORRELATIONS")(terpri)
(terpri)(princ "VARIABLE: ")(princ n)(terpri)
(terpri)
(format t "~4d ~6a ~6a ~6a ~6a ~6a ~6a ~6a"
" Lag" "  LCL" "  ACF" "  UCL" " " "  LCL" "  PACF" "  UCL")(terpri)
(dotimes (i numlags)
(format t "~4d ~6,3f ~6,3f ~6,3f ~6a ~6,3f ~6,3f ~6,3f"
(+ i 1) LCL (select acf i) UCL " "  LCL (select pacf i) UCL)(terpri)
) ;end dotimes i
```

Finally, we plot the autocorrelations and partial-correlations along with the confidence limits. It is easy to determine the number of statistically significant lags from examining the plots. As Chatfield notes in the book cited in the last footnote, a single significant quite-long lag when all shorter lags are insignificant probably represents a random occurrence that should not be taken seriously—we can expect the occasional randomly significant lag when the underlying model being estimated contains no autocorrelations.

```
(def acfplot (plot-points lagnum acf :title "ACF"))
(send acfplot :add-lines lagnum (repeat LCL numlags))
(send acfplot :add-lines lagnum (repeat UCL numlags))
(send acfplot :range 1 -1.0 1.0)
(send acfplot :abline 0 0)
;
(def pacfplot (plot-points lagnum pacf :title "PACF"))
(send pacfplot :add-lines lagnum (repeat LCL numlags))
(send pacfplot :add-lines lagnum (repeat UCL numlags))
(send pacfplot :range 1 -1.0 1.0)
(send pacfplot :abline 0 0)
;
) ; end of functions
```

We use our `acfpacf` function as follows, preceding it with a code line that takes, as required by our subsequent Dickey-Fuller test, the first difference of the variable we are testing for stationarity.

```
; XLISPSTAT BATCH FILE FOR UNIT ROOT TESTS
;
(load "ourfuncs")
(load "causqdat")
(variables)
;
; SET UP CANADA/U.S. REAL EXCHANGE RATE
;
(def rexcaus (remove-first 8 (* 100 (/ cpica (* exrcaus cpius)))))
;
(def plotrex (plot-lines (- dates74 1900) rexcaus
:title "Canada vs. U.S. Real Exchange Rate: Index -- 1974 = 100"))
;
(DFlags rexcaus 12)
;
(def drexcaus (- (remove-first 1 rexcaus)(remove-last 1 rexcaus)))
(acfpacf drexcaus "First Difference of Canada vs. U.S. Real Exchange Rate" 12)
```

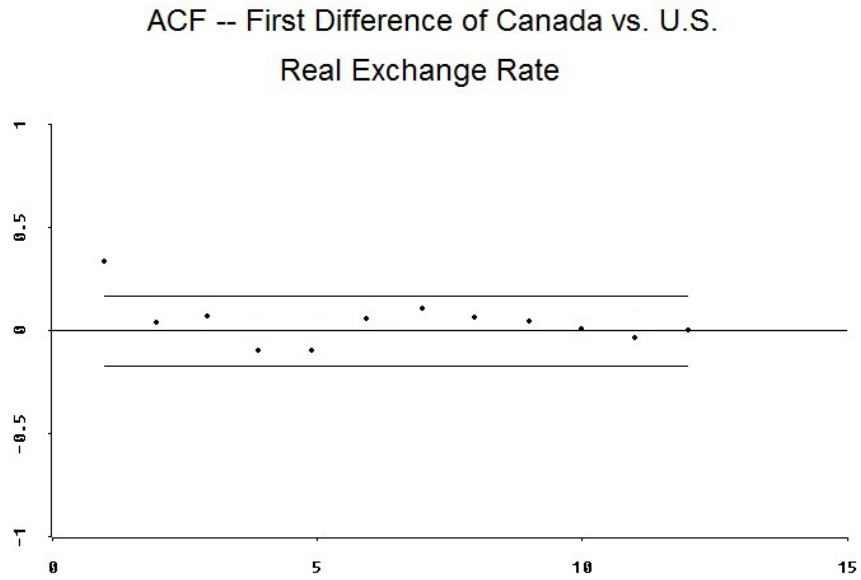
The addition to our output as a result of these last two lines of code is

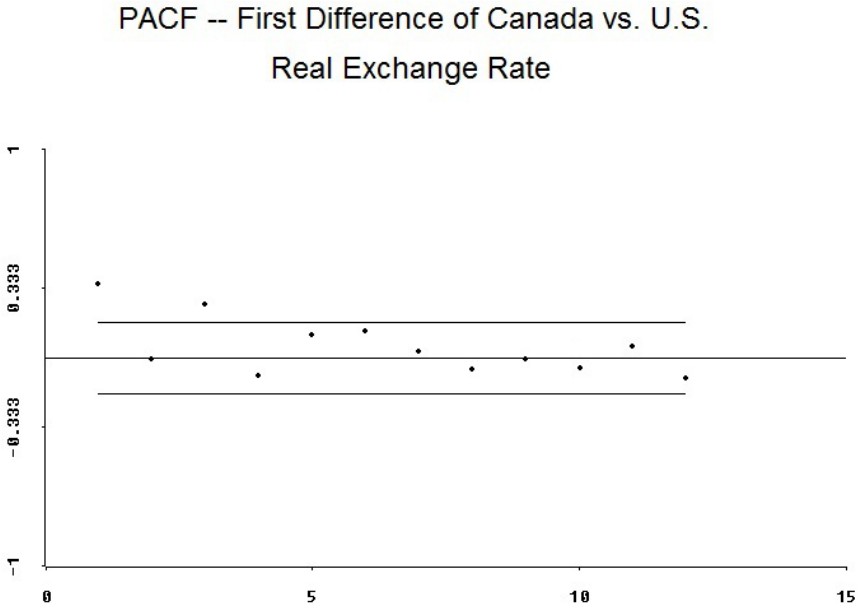
```
AUTOCORRELATIONS AND PARTIAL AUTOCORRELATIONS

VARIABLE: First Difference of Canadian vs. U.S. Real Exchange Rate

 Lag   LCL     ACF    UCL           LCL     PACF    UCL
   1 -0.165  0.338  0.165        -0.165  0.346  0.165
   2 -0.165  0.012  0.165        -0.165 -0.127  0.165
   3 -0.165  0.046  0.165        -0.165  0.106  0.165
   4 -0.165 -0.092  0.165        -0.165 -0.174  0.165
   5 -0.165 -0.097  0.165        -0.165  0.022  0.165
   6 -0.165  0.064  0.165        -0.165  0.110  0.165
   7 -0.165  0.134  0.165        -0.165  0.127  0.165
   8 -0.165  0.051  0.165        -0.165 -0.053  0.165
   9 -0.165  0.047  0.165        -0.165 -0.015  0.165
  10 -0.165  0.002  0.165        -0.165 -0.061  0.165
  11 -0.165 -0.013  0.165        -0.165  0.045  0.165
  12 -0.165  0.004  0.165        -0.165  0.003  0.165
```

followed by the following two graphs.





It seems reasonable to go with the partial-autocorrelation function and set the lag at three, which confirms the results suggested by our `DFlags` function.

We can now proceed to construct our `dfunit` function to do the Dickey-Fuller unit-root tests. The code lines are as follows with, as usual, my comments in between. Our function takes four arguments. The first, `y`, is the series being examined and the second, `n`, is a text element denoting the name of that series. The third argument, `l`, is the number of lags and the final argument, `s`, is the the starting observation number, counting from 1. This latter argument allows us to start additional calculations at later dates without changing the number of lags.

```
(defun dfunit (y n l s)
"Args: (y n l s)
Performs an augmented Dickey-Fuller unit root text on the series y having
name n with l lags, starting at observation s (counting from 1)."
(terpri)
(princ "DICKEY-FULLER TEST  --- ")(princ n)
(terpri)
(princ "Lags = ")
(princ l)
(terpri)
(princ "Starting observation = ")
(princ s)
(terpri)
(def lagy (remove-last 1 y))
(def newy (remove-first 1 y))
(def Dy (- newy lagy))
(def lagy (remove-first (- s 1) lagy))
(def newy (remove-first (- s 1) newy))
(def Dy (remove-first (- s l 1) Dy))
```

The first lines of code above print to screen the information about the test. Then we calculate the first difference of our variable and adjust both it and the one-period lag of that variable to take into account the length of the lag chosen. Next, in the code below, we create the chosen number of lags of $\mathbf{\Delta y}$ if the number of lags chosen is positive, calculate the number of observations and print the corresponding information to the screen. We then construct the trend variable and calculate the sum of squared deviations of $\mathbf{\Delta y}$ from its mean along with the sum of the squared values of that variable itself.

```
(if (> l 0)
(def Dymat (makelags l Dy)))
(def Dy (remove-first l Dy))
(def nobs (length Dy))
(princ "Number of observations = ")
(princ nobs)
(terpri)
(def trend (iseq 1 nobs))
(def SSDy (sum (^ (- Dy (mean Dy)) 2)))
(def DYSQ (sum (^ Dy 2)))
```

We then run the three regressions, using the `if` function to add the lagged values if one or more lags has been chosen and run the regression without lags otherwise. Our first regression includes both a constant term (drift) and a trend. The second includes drift but no trend and the third includes neither drift (specified by the constant term) nor trend.

```
(if (> l 0)
(def dfreg1 (regression-model (list trend lagy Dymat) Dy :print nil))
(def dfreg1 (regression-model (list trend lagy) Dy :print nil))
) ; end if
(if (> l 0)
(def dfreg2 (regression-model (list lagy Dymat) Dy :print nil))
(def dfreg2 (regression-model (list lagy) Dy :print nil ))
) ; end if
(if (> l 0)
(def dfreg3 (regression-model (list lagy Dymat) Dy :intercept nil :print nil))
(def dfreg3 (regression-model (list lagy) Dy :intercept nil :print nil))
) ; end if
(if (> l 0)         ; run regression with lags only
(def dfreg1r (regression-model (list Dymat) Dy :intercept nil :print nil))
) ; end if
(if (> l 0)         ; run regression with constant and lags only
(def dfreg2r (regression-model (list Dymat) Dy :print nil))
) ; end if
```

Two restricted regressions are then added above, one with lags only and the second with a constant and lags only. Next we send the regression objects messages to retrieve the residual sums of squares, the coefficient estimates,

and the coefficient standard errors and construct the conventional t-statistics, the magnitudes of which will have to be compared with the critical values in the Dickey-Fuller tables. In addition we must keep in mind that the standard error of a restricted regression with no lags, no trend and no lagged value of `y` is equal to the sum of the squared deviations of the dependent variable from its mean, and that the standard error of a restricted regression containing no variables is the sum of the squared values of the dependent variable. Then we conduct F-tests of the null hypotheses that coefficients of the trend and lagged value of `y`, and the coefficients of both the constant and trend and lagged value of `y`, are zero respectively in the first regression and that the coefficients of the constant and lagged value of `y` are both zero the second regression. A necessarily negative t-statistic for the coefficient of the lagged value of `y` in the third regression, which includes neither a constant or tend, can be checked in the Dickey-Fuller table to determine whether the null hypothesis of a zero coefficient of lagged `y` can be rejected and the series is therefore stationary with no drift or trend. Thus, in the first two regressions, we determine whether it is possible to establish stationary around trend and/or drift, keeping in mind that the trend variable really measures the change in drift through time. Notice that the F-statistics are calculated by taking the ratio of the increase in the standard error of the regression when variables are removed, divided by the number of variables removed, over the standard error of the unrestricted regression divided by the degrees of freedom. And the F-statistic has as its two parameters the number of degrees of freedom in the numerator, represented by the number of restrictions (or variables removed), and the number of degrees of freedom in the unrestricted regression in the denominator.

```
(def SSE1 (send dfreg1 :residual-sum-of-squares))
(def SSE1 (send dfreg1 :residual-sum-of-squares))
(def df1 (send dfreg1 :df))
(def SSE2 (send dfreg2 :residual-sum-of-squares))
(def df2 (send dfreg2 :df))
(def SSE3 (send dfreg3 :residual-sum-of-squares))
(def df3 (send dfreg3 :df))
(def coefs1 (send dfreg1 :coef-estimates))
(def trat1 (/ (send dfreg1 :coef-estimates)(send dfreg1
   :coef-standard-errors)))
(def coefs2 (send dfreg2 :coef-estimates))
```

```
(def trat2 (/ (send dfreg2 :coef-estimates)(send dfreg2
   :coef-standard-errors)))
(def coefs3 (send dfreg3 :coef-estimates))
(def trat3 (/ (send dfreg3 :coef-estimates)(send dfreg3
   :coef-standard-errors)))
(def SSE1r (send dfreg1r :residual-sum-of-squares))
(def SSE2r (send dfreg2r :residual-sum-of-squares))
; DYSQ is the restricted sum of squares with no lags and no constant
; SSDy is the restricted sum of squares with constant only
(if (> l 0)    ; remove constant trend and lag y
(def Fnctly (/ (/ (- SSE1r SSE1) 3)(/ SSE1 df1)))
(def Fnctly (/ (/ (- DYSQ SSE1) 3)(/ SSE1 df1)))
) ; end if
(if (> l 0)    ; remove trend and lag y
(def Fntly (/ (/ (- SSE2r SSE1) 2)(/ SSE1 df1)))
(def Fntly (/ (/ (- SSDy SSE1) 2)(/ SSE1 df1)))
) ; end if
(if (> l 0)    ; remove constant and lagy (regression without trend)
(def Fncly (/ (/ (- SSE1r SSE3) 2)(/ SSE3 df3)))
(def Fncly (/ (/ (- DYSQ SSE3) 2)(/ SSE3 df3))))
```

Finally the remaining code below prints the results to screen using the three basic functions noted in the case of our `Dflags` function.

```
(princ "Coefficients:")(terpri)(terpri)
(format t "   Constant                   ~10,6f ~10,6f"
(select coefs1 0)(select coefs2 0))
(terpri)
(format t "   Trend                      ~10,6f" (select coefs1 1))
(terpri)
(format t "   Y(t-1)                     ~10,6f ~10,6f ~10,6f"
(select coefs1 2)(select coefs2 1)(select coefs3 0))
(terpri)
(terpri)
(princ "t-Statistics:")(terpri)(terpri)
(format t "   Constant                   ~10,6f ~10,6f" (select trat1 0)
(select trat2 0))
(terpri)
(format t "   Trend                      ~10,6f" (select trat1 1))
```

```
(terpri)
(format t "   Y(t-1)                      ~10,6f ~10,6f ~10,6f"
(select trat1 2)(select trat2 1)(select trat3 0))
(terpri)
(if (> l 0)
(format t "   Lagged (Y(t)-Y(t-1))        ~10,6f ~10,6f ~10,6f"
(select trat1 3)(select trat2 2)(select trat3 1)))
(terpri)
(if (> l 1)
(dotimes (i (- l 1))
(format t "        ........               ~10,6f ~10,6f ~10,6f"
(select trat1 (+ i 4))(select trat2 (+ i 3))(select trat3 (+ i 2)))
(terpri)))
(terpri)
(princ "F-Statistics:")(terpri)(terpri)
(format t "   All Three Coefficients = 0 ~10,6f" Fnctly)
(terpri)
(format t "   Constant & Y(t-1) Coef = 0 ~20,6f" Fntly)
(terpri)
(format t "   Trend & Y(t-1) Coefs = 0   ~10,6f" Fncly)
(terpri)
(terpri)
) ;end function
```

We can now apply our `dfunit` function by adding the following line of code to our batch file,

```
(dfunit rexcaus "Canadian Real Exchange Rate" 3 4)
```

obtaining the following addition to our output.

```
DICKEY-FULLER TEST  --- Canadian Real Exchange Rate
Lags = 3
Starting observation = 4
Number of observations = 144

Coefficients:

   Constant                      1.917298   2.773078
   Trend                         0.003669
   Y(t-1)                       -0.026069  -0.033138  -0.000283

t-Statistics:

   Constant                      1.029580   1.841325
   Trend                         0.783369
   Y(t-1)                       -1.295709  -1.845327  -0.138492
   Lagged (Y(t)-Y(t-1))          4.785584   4.965487   4.832090
        ........                -1.798384  -1.721166  -1.869615
        ........                 1.314176   1.479683   1.241849

F-Statistics:

   All Three Coefficients  = 0   1.338059
   Trend & Y(t-1) Coefs    = 0   2.004717
   Constant & Y(t-1) Coefs = 0             1.704993
```

The t-ratios above are clearly smaller than would be necessary for us to reject null hypotheses using the confidence limits in the Dickey-Fuller table. It is also obvious that the F-statistics are far too low to enable us to reject the null hypotheses in question.

The Dickey-Fuller tests assume that the shocks ϵ_t are statistically independent of each other and have a constant variance. An alternative procedure, developed by Peter Phillips and Pierre Perron, can be used to conduct the tests under the assumption that there is some interdependence of the shocks and they are heterogeneously distributed. Theirs is a modified version of the Dickey-Fuller approach that incorporates adjustments for serial correlation and heteroskedasticity of the sort involved in HAC adjustments to coefficient-standard-errors. It turns out, however, that the probability of distortions in these estimates in finite samples makes them a bit controversial and has prompted continuing extensions of the process through which they are calculated. Accordingly, given the complexity of figuring out and programming the optimal formulation of the test, we will here focus entirely on Dickey-Fuller tests, as consistent with the modern applied econometrics literature.[10]

Suppose, alternatively, that we want to run the Dickey-Fuller test in `Gretl`. The simplest way to do this is to fire-up `Gretl` and load the data file `causdata.gdt`. If no real exchange rate series is present in the file, we simply click on `add` and scroll down to `Define new variable` and type the following code in the window that appears:

```
REXCAUS = 100*CPICA/(EXRCAUS*CPIUS
```

Then click on `variable` and scroll down to and click on `Augmented Dickey-Fuller test` (the word augmented refers to the option of including lags of $\mathbf{\Delta y}$). A window will appear in which we give `Gretl` the appropriate instructions. We should highlight the following options

```
test without constant
test with constant
test with constant and trend
test down from maximum lag order
use level of variable
```

and then, acting as if we did not know how many lags to use, set the lag-order at 12, the maximum number we chose when working with `XLispStat`. Upon clicking on the OK button we obtain the following results in a separate window.

[10]For example, Russell Davidson and James G. MacKinnon in their peviously cited graduate-level textbook, do not include the Phillips-Perron test for reasons stated on page 623.

```
Augmented Dickey-Fuller test for REXCAUS
including 4 lags of (1-L)REXCAUS
sample size 143
unit-root null hypothesis: a = 1

   test without constant
   model: (1-L)y = (a-1)*y(-1) + ... + e
   1st-order autocorrelation coeff. for e: 0.002
   lagged differences: F(4, 138) = 7.116 [0.0000]
   estimated value of (a - 1): -0.000167519
   test statistic: tau_nc(1) = -0.0823193
   asymptotic p-value 0.6553

Augmented Dickey-Fuller regression
OLS, using observations 1975:2-2010:4 (T = 143)
Dependent variable: d_REXCAUS

               coefficient    std. error   t-ratio    p-value
  ------------------------------------------------------------
  REXCAUS_1     -0.000167519   0.00203499   -0.08232   0.6553
  d_REXCAUS_1    0.431535      0.0852664     5.061     1.31e-06 ***
  d_REXCAUS_2   -0.206480      0.0921666    -2.240     0.0267   **
  d_REXCAUS_3    0.182883      0.0918769     1.991     0.0485   **
  d_REXCAUS_4   -0.174496      0.0848528    -2.056     0.0416   **
```

```
   test with constant
   model: (1-L)y = b0 + (a-1)*y(-1) + ... + e
   1st-order autocorrelation coeff. for e: 0.005
   lagged differences: F(4, 137) = 7.262 [0.0000]
   estimated value of (a - 1): -0.0276231
   test statistic: tau_c(1) = -1.51726
   asymptotic p-value 0.5251

Augmented Dickey-Fuller regression
OLS, using observations 1975:2-2010:4 (T = 143)
Dependent variable: d_REXCAUS

               coefficient   std. error   t-ratio   p-value
  ------------------------------------------------------------
  const          2.31419      1.52502      1.517    0.1314
  REXCAUS_1     -0.0276231    0.0182059   -1.517    0.5251
  d_REXCAUS_1    0.436541     0.0849309    5.140    9.28e-07 ***
  d_REXCAUS_2   -0.189634     0.0924040   -2.052    0.0421   **
  d_REXCAUS_3    0.190346     0.0915785    2.079    0.0395   **
  d_REXCAUS_4   -0.155718     0.0853568   -1.824    0.0703   *
```

```
   with constant and trend
   model: (1-L)y = b0 + b1*t + (a-1)*y(-1) + ... + e
   1st-order autocorrelation coeff. for e: 0.003
   lagged differences: F(4, 136) = 6.875 [0.0000]
   estimated value of (a - 1): -0.017401
   test statistic: tau_ct(1) = -0.850501
   asymptotic p-value 0.9597

Augmented Dickey-Fuller regression
OLS, using observations 1975:2-2010:4 (T = 143)
Dependent variable: d_REXCAUS

               coefficient   std. error   t-ratio   p-value
  ------------------------------------------------------------
  const          0.710292     2.11645      0.3356   0.7377
  REXCAUS_1     -0.0174010    0.0204597   -0.8505   0.9597
  d_REXCAUS_1    0.423749     0.0856757    4.946    2.20e-06 ***
  d_REXCAUS_2   -0.203544     0.0932134   -2.184    0.0307   **
  d_REXCAUS_3    0.178846     0.0921180    1.941    0.0543   *
  d_REXCAUS_4   -0.172032     0.0865950   -1.987    0.0490   **
  time           0.00518317   0.00474610   1.092    0.2767

   test with constant and trend
   model: (1-L)y = b0 + b1*t + (a-1)*y(-1) + ... + e
   1st-order autocorrelation coeff. for e: 0.020
   lagged differences: F(3, 126) = 7.521 [0.0001]
   estimated value of (a - 1): -0.0109607
   test statistic: tau_ct(1) = -0.55294
   asymptotic p-value 0.9811
```

`Gretl` picks a lag order of 4, one more than we chose in the case of `XLispStat`.

`Gretl` does not seem to have a function equivalent to our `DFlags` function. But it does permit us to estimate autocorrelations and partial-autocorrelations by clicking on `variable` and then `correlogram`. Before doing this, however, we must set up the first-difference of our `REXCAUS` variable by clicking on `add` and then `Define new variable`, inserting the code line

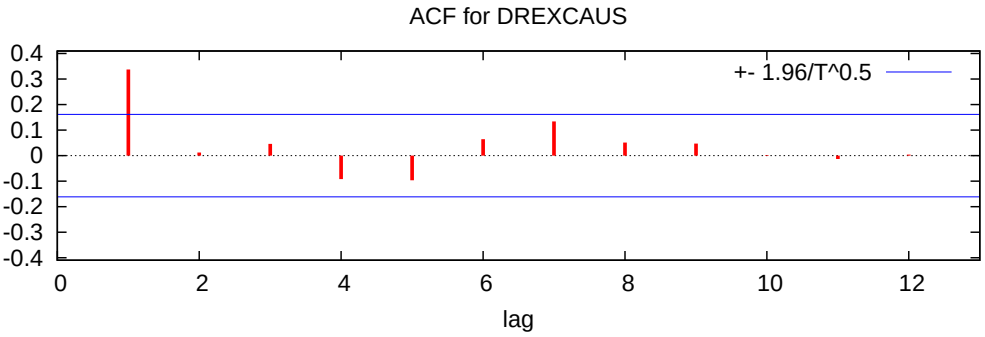
```
DREXCAUS = REXCAUS - REXCAUS_1
```

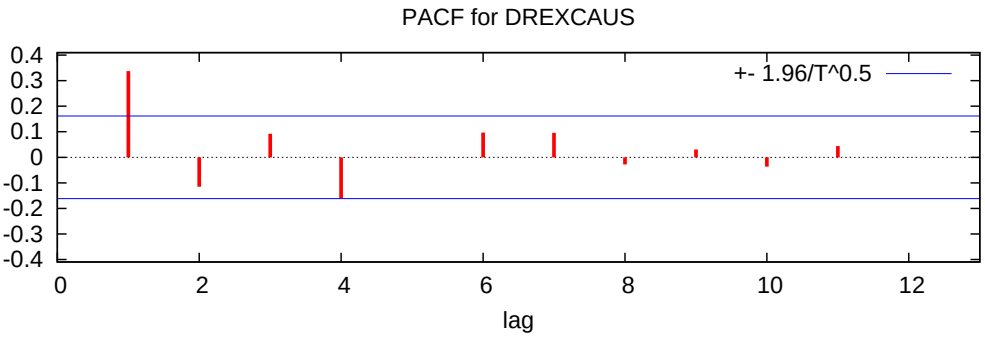
And prior to this, we must have first created the one-period lag of REXCAUS, which appears in the right-most position in the code-line above, by clicking on add and Lags of selected variable, setting the lag at 1. This procedure, as well as the one that creates the correlogram, requires that the selected variable be highlighted when the commands are given. The result of all this is a window containing the following autocorrelation and partial autocorrelation functions (i.e., the correlogram),

Autocorrelation function for DREXCAUS

LAG	ACF		PACF		Q-stat.	[p-value]
1	0.3375	***	0.3375	***	17.0922	[0.000]
2	0.0118		-0.1152		17.1134	[0.000]
3	0.0460		0.0920		17.4348	[0.001]
4	-0.0920		-0.1604	*	18.7307	[0.001]
5	-0.0967		0.0006		20.1745	[0.001]
6	0.0645		0.0970		20.8202	[0.002]
7	0.1342		0.0957		23.6370	[0.001]
8	0.0512		-0.0278		24.0505	[0.002]
9	0.0469		0.0309		24.3994	[0.004]
10	0.0023		-0.0362		24.4002	[0.007]
11	-0.0131		0.0442		24.4278	[0.011]
12	0.0038		0.0003		24.4302	[0.018]

together with the figure on the next page. The results are essentially the same as we obtained using our autocorrelation and partial-autocorrelation functions in XLispStat.





The essence of the above results can also be obtained by constructing and running in `Gretl` the following script after loading the data.[11]

```
# GRETL SCRIPT FOR UNIT ROOT TESTS
#
smpl 1974:1 2010:4
#
REXCAUS = 100*CPICA/(EXRCAUS*CPIUS)
#
adf 12 REXCAUS --nc --c --ct --verbose --test-down
#
lags 1; REXCAUS
DREXCAUS = REXCAUS - REXCAUS_1
corrgm DREXCAUS 12
```

Unfortunately, running the above script produces only a crude text graph containing the autocorrelations only and no delineation of the critical values.

[11] The script file is `unitroot.inp`, the data file in which the variables and there sources are appropriately described is `causqdat.gdt`, and the output file is `unitroot.gou`.

It is necessary to highlight the variable DREXCAUS and then click on variable and then correlogram to get the pretty graph above containing both the autocorrelations and partial-autocorrelations with the 5% critical values highlighted. The Q-statistics in the results above test whether the current and previous autocorrelations are zero—if that statistic is larger than some critical value we can reject the null hypothesis of no significant autocorrelation. The P-Values above clearly indicate that the series is not white-noise.

Finally, we can also run Dickey-Fuller tests in the statistical program R. We load the program and then click on File and then Change dir... to change the directory to the one we are working out of. Then we load the following script, after making sure that the package urca is installed on our system.[12]

```
# R SCRIPT FOR INVESTIGATING THE STATIONARITY OF THE REAL EXCHANGE RATES
#
rexdata <- read.table("causqdat.tab",header=TRUE)
names(rexdata)
#
attach(rexdata)
#
#Set up the relevant variables in the object rexdata as time-series
#
EXRCAUS <- ts(rexdata$EXRCAUS,start=c(1972,1),end=c(2010,4),frequency=4)
CPICA <- ts(rexdata$CPICA,start=c(1972,1),end=c(2010,4),frequency=4)
CPIUS <- ts(rexdata$CPIUS,start=c(1972,1),end=c(2010,4),frequency=4)
#
REXCAUS <- 100*CPICA/(EXRCAUS*CPIUS)
REXCAUS <- REXCAUS[9:156]    # shorten series to start in 1974Q1
REXCAUS <- ts(REXCAUS,start=c(1974,1),end=c(2010,4),frequency=4)
#
library(urca)
#
summary(ur.df(REXCAUS,lags=3,type="trend"))
summary(ur.df(REXCAUS,lags=3,type="drift"))
summary(ur.df(REXCAUS,lags=3,type="none"))
summary(ur.df(REXCAUS,selectlags = "AIC",type="trend"))
summary(ur.df(REXCAUS,selectlags = "AIC",type="drift"))
```

[12]The R script file we use here is unitroot.R, the data file is causqdat.tab and the output file is unitroot.Rou.

```
summary(ur.df(REXCAUS,selectlags = "AIC",type="none"))
```

We perform the Dickey-Fuller tests using the `ur.df` function in two ways—once with our previously chosen lag of 3 and once letting `R` choose the optimal lag using the AIC. We obtain the following results.

```
R version 2.10.1 (2009-12-14)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> # R SCRIPT FOR INVESTIGATING THE STATIONARITY OF THE REAL EXCHANGE RATES
> #
> rexdata <- read.table("causqdat.tab",header=TRUE)
> names(rexdata)
 [1] "X.YEAR"  "IPDCA"   "IPDUS"   "EXRCAUS" "GDPCA"   "GDPUS"   "EXPGSUS"
 [8] "IMPGSUS" "EXPGSCA" "IMPGSCA" "CPICA"   "PEXPUS"  "PIMPUS"  "CPIUS"
[15] "PCOMM"   "PCOMXEN" "PENERGY" "PCROIL"  "M1CA"    "M2CA"    "M1US"
[22] "M2US"
> #
> #Set up the relevant variables in the object rexdata as time-series
> #
> EXRCAUS <- ts(rexdata$EXRCAUS,start=c(1972,1),end=c(2010,4),frequency=4)
> CPICA <- ts(rexdata$CPICA,start=c(1972,1),end=c(2010,4),frequency=4)
> CPIUS <- ts(rexdata$CPIUS,start=c(1972,1),end=c(2010,4),frequency=4)
> #
```

```
> REXCAUS <- 100*CPICA/(EXRCAUS*CPIUS)
> REXCAUS <- REXCAUS[9:156]    # shorten series to start in 1974Q1
> REXCAUS <- ts(REXCAUS,start=c(1974,1),end=c(2010,4),frequency=4)
> #
> library(urca)
> #
```

```
> summary(ur.df(REXCAUS,lags=3,type="trend"))
>
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.4288  -1.0699  -0.1282   1.0384   5.9248

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.906292   1.870491   1.019   0.3099
z.lag.1     -0.026069   0.020120  -1.296   0.1972
tt           0.003669   0.004683   0.783   0.4348
z.diff.lag1  0.411959   0.086083   4.786 4.33e-06 ***
z.diff.lag2 -0.164778   0.091625  -1.798   0.0743 .
z.diff.lag3  0.113852   0.086634   1.314   0.1910
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.041 on 138 degrees of freedom
Multiple R-squared: 0.169,      Adjusted R-squared: 0.1389
F-statistic: 5.614 on 5 and 138 DF,  p-value: 9.658e-05


Value of test-statistic is: -1.2957 1.3381 2.0047

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -3.99 -3.43 -3.13
phi2  6.22  4.75  4.07
```

```
phi3  8.43  6.49  5.47

> summary(ur.df(REXCAUS,lags=3,type="drift"))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.0604  -0.9645  -0.1542   1.1179   6.1789

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.77308    1.50602   1.841   0.0677 .
z.lag.1     -0.03314    0.01796  -1.845   0.0671 .
z.diff.lag1  0.42205    0.08500   4.965 1.98e-06 ***
z.diff.lag2 -0.15641    0.09087  -1.721   0.0874 .
z.diff.lag3  0.12596    0.08513   1.480   0.1412
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.038 on 139 degrees of freedom
Multiple R-squared: 0.1653,     Adjusted R-squared: 0.1413
F-statistic: 6.883 on 4 and 139 DF,  p-value: 4.386e-05


Value of test-statistic is: -1.8453 1.705

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
```

```
phi1  6.52  4.63  3.81

> summary(ur.df(REXCAUS,lags=3,type="none"))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.4192  -1.0240  -0.0656   1.1632   6.1402

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
z.lag.1     -0.000283   0.002043  -0.138   0.8901
z.diff.lag1  0.413597   0.085594   4.832 3.50e-06 ***
z.diff.lag2 -0.170716   0.091311  -1.870   0.0636 .
z.diff.lag3  0.105719   0.085130   1.242   0.2164
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.056 on 140 degrees of freedom
Multiple R-squared: 0.145,      Adjusted R-squared: 0.1205
F-statistic: 5.934 on 4 and 140 DF,  p-value: 0.0001932


Value of test-statistic is: -0.1385

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

```
> summary(ur.df(REXCAUS,selectlags = "AIC",type="trend"))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.8480  -1.0609  -0.1068   0.9765   6.0598

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.019015   1.781982   1.133    0.259
z.lag.1     -0.027608   0.019209  -1.437    0.153
tt           0.003788   0.004489   0.844    0.400
z.diff.lag   0.347649   0.080237   4.333 2.77e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.045 on 142 degrees of freedom
Multiple R-squared: 0.1457,     Adjusted R-squared: 0.1277
F-statistic: 8.074 on 3 and 142 DF,  p-value: 5.279e-05


Value of test-statistic is: -1.4372 1.5601 2.3399

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -3.99 -3.43 -3.13
phi2  6.22  4.75  4.07
phi3  8.43  6.49  5.47
```

```
> summary(ur.df(REXCAUS,selectlags = "AIC",type="drift"))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5012  -1.0426  -0.1600   1.0615   6.3060

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.88233    1.45751   1.978   0.0499 *
z.lag.1     -0.03456    0.01733  -1.994   0.0481 *
z.diff.lag   0.36086    0.07862   4.590 9.61e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.043 on 143 degrees of freedom
Multiple R-squared: 0.1414,     Adjusted R-squared: 0.1294
F-statistic: 11.78 on 2 and 143 DF,  p-value: 1.84e-05


Value of test-statistic is: -1.9939 1.9882

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
```

```
> summary(ur.df(REXCAUS,selectlags = "AIC",type="none"))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-10.8114  -1.0104  -0.1352   1.0519   6.4942

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
z.lag.1    -0.0005153  0.0020310  -0.254      0.8
z.diff.lag  0.3463609  0.0790608   4.381 2.26e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.063 on 144 degrees of freedom
Multiple R-squared: 0.118,      Adjusted R-squared: 0.1057
F-statistic: 9.632 on 2 and 144 DF,  p-value: 0.0001186


Value of test-statistic is: -0.2537

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

The best way of interpreting the above results is to compare the t-ratios in the Coefficients section with the Dickey-Fuller critical values in the tables at the end of this document. It is not clear what the R-squares refer to or what to make of the τ and ϕ critical values presented at the end of each run.

When the number of lags is set at three, `R` produces the same results as we obtained with `XLispStat`. When `R` is allowed to chose the lag length on the basis of the AIC, however, it picks a lag of one and therefore produces somewhat different values for the statistics, although stationarity is still clearly rejected. It would appear that since the AIC rises when the lag is increased from one to two, `R` adopts a lag of one, even though the AIC for a lag of four is slightly smaller than that for one lag. The SBC clearly indicates a lag of one but the partial-autocorrelation function indicates a lag of 4, the lag chosen by Gretl. Since of lag selection in `XLispStat` indicated a lag of 3, it would seem best to follow `Gretl` and go with the longer lag.

Statistical Tables

While the simplest way to calculate P-Values is to calculate the relevant cumulative densities using Gretl, R, XLispStat, or other statistical software, the test-statistics for unit root and cointegration tests do not follow standard distributions. Accordingly, the next three pages contain the relevant statistical tables for Dickey-Fuller and Phillips-Perron unit root tests, for cointegration tests based on unit root tests of regression residuals, and for Johansen cointegration tests.

The critical values for the unit root tests in the table that follows were calculated using Monte Carlo methods by David Dickey and Wayne A. Fuller and were obtained from their paper "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica*, Vol. 49, July 1981, pages 1062 and 1063, and from Walter Enders, *Applied Econometric Time Series*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1995, pages 223, 419 and 421, and from James D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994, page 763. These critical values are based on sample sizes of 100 and remain unchanged when the Dickey-Fuller estimating equations are augmented by inclusion of lagged values of Δy_t to improve the fit as indicated by the AIC and SBC. Larger sample sizes will result in critical values that are slightly smaller in absolute value and smaller sample sizes will result in somewhat larger critical values.

STATISTICAL TABLES FOR UNIT ROOT TESTS

True Model Used to Generate the Data: $y_t = y_{t-1} + \epsilon_t$

1. Model Estimated:	Dickey-Fuller	$\Delta y_t = a_0 + a_1 y_{t-1} + a_2 t + \epsilon_t$
	Phillips-Perron	$y_t = \tilde{a}_0 + \tilde{a}_1 y_{t-1} + \tilde{a}_2(t - n/2) + \tilde{\epsilon}_t$

Hypothesis	Test Statistic	Critical Values 10%	5%	1%
$a_1 = 0$, $\tilde{a}_1 = 1$	t-based	-3.15	-3.45	-4.04
$a_0 = 0$, $\tilde{a}_0 = 0$	t-based	2.73	3.11	3.78
$a_2 = 0$, $\tilde{a}_2 = 0$	t-based	2.38	2.79	3.53
$a_1 = a_2 = 0$, $\tilde{a}_1 = 1$ & $\tilde{a}_2 = 0$	F-based	5.47	6.49	8.73
$a_0 = a_1 = a_2 = 0$, $\tilde{a}_0 = \tilde{a}_2 = 0$ & $\tilde{a}_1 = 1$	F-based	4.16	4.88	6.50

2. Model Estimated:	Dickey-Fuller	$\Delta y_t = a_0 + a_1 y_{t-1} + \epsilon_t$
	Phillips-Perron	$y_t = \tilde{a}_0 + \tilde{a}_1 y_{t-1} + \tilde{\epsilon}_t$

Hypothesis	Test Statistic	Critical Values 10%	5%	1%
$a_1 = 0$, $\tilde{a}_1 = 1$	t-based	-2.58	-2.89	-3.51
$a_0 = 0$, $\tilde{a}_0 = 0$	t-based	2.17	2.54	3.22
$a_0 = a_1 = 0$, $\tilde{a}_0 = 0$ & $\tilde{a}_1 = 1$	F-based	3.86	4.71	6.70

3. Model Estimated:	Dickey-Fuller	$\Delta y_t = a_1 y_{t-1} + \epsilon_t$
	Phillips-Perron	$y_t = \tilde{a}_1 y_{t-1} + \tilde{\epsilon}_t$

Hypothesis	Test Statistic	Critical Values 10%	5%	1%
$a_1 = 0$, $\tilde{a}_1 = 1$	t-based	-1.61	-1.95	-2.60

CRITICAL VALUES FOR REGRESSION-RESIDUAL BASED COINTEGRATION TESTS

Estimated Cointegrating Regression Residual:

$$z_t = y_t - \beta_0 - \beta_1 x_{1t} - \beta_2 x_{2t} - \beta_3 x_{3t} - \ldots\ldots - \beta_N x_{Nt}$$

Number of Variables $N+1$	Sample Size	Critical Values 10%	5%	1%
2	50	3.28	3.67	4.32
	100	3.03	3.37	4.07
	200	3.02	3.37	4.00
3	50	3.73	4.11	4.84
	100	3.59	3.93	4.45
	200	3.47	3.78	4.35
4	50	4.02	4.35	4.94
	100	3.89	4.22	4.75
	200	3.89	4.18	4.70
5	50	4.42	4.76	5.41
	100	4.26	4.58	5.18
	200	4.18	4.48	5.02
6	500	4.43	4.71	5.28

Notes and Sources: Standard Dickey-Fuller and Phillips Perron unit-root tests are applied to the regression residuals using the critical values above instead of those on the previous page, focussing entirely on the coefficients of the laggeed residual. Except for the case of 6 variables, these critical values were calculated using Monte Carlo methods by Robert F. Engle and Byung Sam Yoo and obtained from their paper "Forecasting and Testing in Co-Integrated Systems," *Journal of Econometrics*, Vol. 35, 1987, page 157. The critical values for the case of 6 variables using 500 observations were calculated by Peter C. B. Phillips and S. Ouliaris, "Asymptotic Properties of Residual Based Tests for Cointegration," *Econometrica*, Vol. 58, 1990, 165-93, and were obtained from James D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994, page 766, Case 2. The complete set of Phillips-Ouliaris critical values distinguish between whether or not a constant and trend are included in the cointegrating regression. These values are so similar in the three cases to the ones calculated by Engle and Yoo, based on the inclusion of a constant but not trend, that the complexities of including them here are avoided.

CRITICAL VALUES FOR JOHANSEN COINTEGRATION TESTS

	Probability that Statistic Exceeds Entry					
$n-h$	0.10	0.05	0.01	0.10	0.05	0.01
	Unrestricted Estimation: Trend Drift in Data					
	L-max			Trace		
1	2.816	3.962	6.936	2.816	3.962	6.936
2	12.099	14.036	17.936	13.338	15.197	19.310
3	18.697	20.778	25.521	26.791	29.509	35.397
4	24.712	27.169	31.943	43.964	47.181	53.792
5	30.774	33.178	38.341	65.063	68.905	76.955
	Unrestricted Estimation: No Trend Drift in Data					
	L-max			Trace		
1	6.691	8.083	11.576	6.691	8.083	11.576
2	12.783	14.595	18.782	15.583	17.844	21.962
3	18.959	21.279	26.154	28.436	31.256	37.291
4	24.917	27.341	32.616	45.245	48.419	55.551
5	30.818	33.262	38.858	69.956	69.977	77.911

Notes and Sources: n is the number of variables and h is the number of cointegrating vectors under the null hypothesis. The critical values in the table are copied from Walter Enders, *Applied Economic Time Series*, Wiley Series in Probability and Statistics, 1995, page 420, and are identical to those found in James D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994, pages 767 and 768, Cases 2 and 3.