Package ‘egcm’

September 18, 2017

Type Package
Title Engle-Granger Cointegration Models
Version 1.0.12
Date 2017-08-28
Description An easy-to-use implementation of the Engle-Granger two-step procedure for identifying pairs of cointegrated series. It is geared towards the analysis of pairs of securities. Summary and plot functions are provided, and the package is able to fetch closing prices of securities from Yahoo. A variety of unit root tests are supported, and an improved unit root test is included.
Depends zoo, xts
Imports grid, ggplot2, tseries, MASS, urca, parallel, pracma, stats, quantmod, methods
License GPL-2 | GPL-3
RoxygenNote 6.0.1
NeedsCompilation no
Author Matthew Clegg [aut, cre, cph]
Maintainer Matthew Clegg <matthewcleggphd@gmail.com>
Repository CRAN
Date/Publication 2017-09-18 13:00:53 UTC

R topics documented:

  egcm-package ................................................. 2
  acor ............................................................. 4
  allpairs.egcm ................................................. 5
  bvr.test ....................................................... 7
  detrend ......................................................... 8
  egcm ........................................................... 9
  egcm.defaults ............................................... 15
  pgff.test ..................................................... 17
  rar1 ........................................................... 19
Description

This package provides a simplified implementation of the Engle-Granger cointegration model that is geared towards the analysis of securities prices. Summary and plot functions are provided, and a convenient interface to \texttt{quantmod} is given. A variety of standard unit root tests are supported, and an improved unit root test is included.

Details

This package implements a test for a simplified form of cointegration. Namely, it checks whether or not a linear combination of two time series follows an autoregressive model of order one. In other words, given two series $X$ and $Y$, it searches for parameters $\alpha$, $\beta$ and $\rho$ such that:

\begin{align*}
Y[i] &= \alpha + \beta \times X[i] + R[i] \\
R[i] &= \rho \times R[i-1] + \epsilon
\end{align*}

If $|\rho| < 1$, then $X$ and $Y$ are cointegrated.

Cointegration is a useful tool in many areas of economics, but this implementation is especially geared towards the analysis of securities prices. Testing for cointegration has been proposed as means for assessing whether or not two securities are suitable candidates for pairs trading. In addition, this package implements two previously unavailable unit root tests. A test based upon the weighted symmetric estimator $\rho_{ws}$ of Pantula, Gonzales-Farias and Fuller is implemented as \texttt{pgffNtest}. This test seems to provide superior performance to the standard Dickey-Fuller test \texttt{adfNtest} and also improves upon the performance of a number of other tests previously available in \texttt{R}.

The variance ratio test proposed by J. Breitung is implemented as \texttt{bvrNtest}. It has the advantage that it is a non-parametric test, and it seems to provide superior performance to other variance ratio tests available in \texttt{R}, although it does not perform as well as \texttt{pgffNtest}.

Users who wish to explore more general models for cointegration are referred to the \texttt{urca} package of Bernard Pfaff.

Disclaimer

DISCLAIMER: The software in this package is for general information purposes only. It is hoped that it will be useful, but it is provided WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. It is not intended to form the basis of any investment decision. USE AT YOUR OWN RISK!
Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

References


See Also

- `egcm` Further documentation of the Engle-Granger cointegration model
- `pgff.test` Unit root test based on the weighted symmetric estimator of Pantula, Gonzalez-Farias and Fuller
- `bvr.test` Unit root test based on Breitung’s variance ratio
- `adf.test`, `pp.test` Unit root tests included in the base R distribution
- `urca` An extensive collection of unit root tests and cointegration tests implemented by Bernard Pfaff
- `hurstexp` Unit root tests based on variance ratios

Examples

```r
## Not run:
library(quantmod)
prices.spy <- getSymbols("SPY", from="2013-01-01", to="2014-01-01",
  auto.assign = FALSE)$SPY.Adjusted
prices.voo <- getSymbols("VOO", from="2013-01-01", to="2014-01-01",
  auto.assign = FALSE)$VOO.Adjusted
egcm(prices.spy, prices.voo)
plot(egcm(prices.spy, prices.voo))
summary(egcm(prices.spy, prices.voo))

# The yegcm method provides a convenient interface to the TTR
# package, which can fetch closing prices from Yahoo. Thus,
# the above can be simplified as follows:
```

```
Description

autocorrelation of a sequence

Usage

```r
acor(X, k = 1, na.rm = FALSE)
```

Arguments

- `X`: a numeric vector or `zoo` vector
- `k`: the number of lags for which to compute the autocorrelation. Default: 1
- `na.rm`: a boolean, which if TRUE, indicates that NA values should be removed from the series prior to computing the autocorrelation. Default: FALSE

Value

Returns the lag k autocorrelation of X, e.g., the correlation of X[i] with X[i-k].

Note

It’s a bit surprising that this is not a part of the core `R` distribution, but I can’t find it. Perhaps it was thought to be too trivial to include.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

See Also

- `acf`
Examples

```r
acor(1:10)  # a perfect correlation
acor(rnorm(100))  # should be close to zero
acor(cumsum(rnorm(100)))  # slightly less than one
acor(rar1(1000, al=0.8))  # slightly less than 0.8
acor(rar1(1000, al=0.8), k=2)  # about 0.64
acor(rar1(1000, al=0.8), k=3)  # about 0.51
```

---

**allpairs.egcm**

*Perform cointegration tests for all pairs of securities in a list*

**Description**

Given a list of ticker symbols, downloads the adjusted daily closing prices of each of the symbols from Yahoo, and performs a cointegration test for each pair of symbols. Returns a data.frame containing the results of the tests.

**Usage**

```r
allpairs.egcm(tickers,
  startdate = format(Sys.Date() - 365, "%Y-%m-%d"),
  enddate = format(Sys.Date(), "%Y-%m-%d"), clear.na.inf=TRUE, ...)
```

**Arguments**

- `tickers` A list of ticker symbols whose data is to be downloaded from Yahoo!. Alternatively, this may be a data.frame containing the price series to be checked, one series per column.
- `startdate` The starting date for which to download the data. Given in the form "YYYY-MM-DD". Defaults to one year ago.
- `enddate` The ending date for which to download the data. Given in the form "YYYY-MM-DD". Defaults to today.
- `clear.na.inf` if TRUE, NA and Inf price values are replaced by the last available price. Default:TRUE.
- `...` Other parameters to be passed to `egcm`

**Value**

A data.frame containing the following columns:

- `series1`: Name of the first ticker in this cointegration test
- `series2`: Name of the second ticker in this cointegration test
- `log`: Boolean which if TRUE indicates that the cointegration test is performed on the logs of the series
- `i1test`: Name of the test used for checking that the series are integrated.
allpairs.egcm

- urtest: Name of the test used for checking for a unit root in the residual series
- alpha: Constant term of the linear relation between the series
- alpha.se: Standard error of alpha
- beta: Linear term of the linear relation between the series
- beta.se: Standard error of beta
- rho: Coefficient of mean reversion
- rho.se: Standard error of rho
- s1.i1.stat: Statistic computed for integration test of first series
- s1.i1.p: p-value for integration test of first series
- s2.i1.stat: Statistic computed for integration test of second series
- s2.i1.p: p-value for integration test of second series
- r.stat: Statistic computed for cointegration test (e.g. whether the residual series contains a unit root)
- r.p: p-value associated with r.stat
- eps.ljungbox.stat: Ljung-Box statistic computed on the innovations of the series
- eps.ljungbox.p: p-value associated with the Ljung-Box statistic
- s1.dsd: Standard deviation of the first differences of the first series
- s2.dsd: Standard deviation of the first differences of the second series
- residuals.sd: Standard deviation of the residual series
- eps.sd: Standard deviation of the innovations
- is.cointegrated: TRUE if the pair is cointegrated at the 5% confidence level

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

See Also
eegcm

Examples

```r
## Not run:
# Check if any of the oil majors are cointegrated:
allpairs.egcm(c("BP","CVX","RDS.A","TOT","XOM"))

## End(Not run)
```
bvr.test

Unit root test based upon Breitung’s variance ratio

Description

Unit root test based upon Breitung’s variance ratio

Usage

bvr.test(Y, detrend = FALSE)
bvr_rho(Y, detrend = FALSE)

Arguments

Y A vector or zoo-vector
detrend A boolean, which if TRUE, indicates that the test should be performed after removing a linear trend from Y

Details

Breitung’s variance ratio is given by the formula:

$$\rho_T = \frac{T^{-2} \sum_{t=1}^{T} Y_t^2}{T^{-1} \sum_{t=1}^{T} y_t^2}$$

where $T$ is the length of the vector $Y$. (See equation (5) of his paper.)

The advantage of Breitung’s variance ratio is that, in contrast to the Dickey-Fuller test and other related tests, it is a nonparametric statistic. In simulations, it seems to perform favorably with respect to the Hurst exponent.

Simulation has been used to determine the distribution of the statistic, and table lookup is used to determine p-values.

If detrend=TRUE, then a linear trend is removed from the data prior to computing the estimator $\rho_T$. A separate table has been computed of the distribution of values of $\rho_T$ after detrending.

Value

bvr_rho returns the value $\rho_T$ of Breitung’s variance ratio.
bvr.test returns a list with class "htest" containing the following components:

- statistic the value of the test statistic.
- parameter the truncation lag parameter.
- p.value the p-value of the test.
- method a character string indicating what type of test was performed.
- data.name a character string giving the name of the data.
Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

References


See Also

hurstexp egcm

Examples

# The following should produce a low p-value
bvr_rhoo(rnorm(100))
bvr.test(rnorm(100))

# The following should produce a high p-value
bvr_rhoo(cumsum(rnorm(100)))
bvr.test(cumsum(rnorm(100)))

# Test with an autoregressive sequence where rho = 0.8
bvr.test(ar1(100, a1=0.8))

# If there is a linear trend, bvr.test with detrend=FALSE
# is likely to find a unit root when there is none:
bvr.test(1:100 + rnorm(100))
bvr.test(1:100 + rnorm(100), detrend=TRUE)

# Display the power of the test for various values of rho and n:
bvr_power(a1=0.8, n=100, nrep=100)
bvr_power(a1=0.9, n=250, nrep=100)
bvr_power(a1=0.95, n=250, nrep=100)

# This is to be compared to the power of the adf.test at this level:
adf_power(a1=0.8, n=100, nrep=100)
adf_power(a1=0.9, n=250, nrep=100)
adf_power(a1=0.95, n=250, nrep=100)

---

**detrend**

*Remove a linear trend from a vector*

**Description**

Given a numeric vector Y, removes a linear trend from it.
Usage
detrend(Y)

Arguments
Y numeric vector to be de-trended

Value
Returns a vector X where X[i] = Y[i] - a - b * i, where a and b describe the linear trend in Y.

Author(s)
Matthew Clegg <matthewcleggphd@gmail.com>

Examples
detrend(rep(1,10)) # == 0 0 0 0 0 0 0 0 0 0
detrend(1:10) # == 0 0 0 0 0 0 0 0 0 0
detrend((1:10)^2) # == 12 4 -2 -6 -8 -6 -2 4 12

mean(detrend(rnorm(1:100) + 1:100)) # should be very close to 0
sd(rnorm(1:100) + 1:100) # approximately 29
sd(detrend(rnorm(1:100) + 1:100)) # approximately 1

egcm
Simplified Engle-Granger Cointegration Model

Description
Performs the two-step Engle Granger cointegration procedure on a pair of time series, and creates an object representing the results of the analysis.

Usage
egcm(X, Y, na.action, log = FALSE, normalize = FALSE, debias = TRUE, robust=FALSE, include.const=TRUE, iltest = egcm.default.iltest(), urtest = egcm.default.urtest(), p.value = egcm.default.pvalue())
is.cointegrated(E)
is.ar1(E)
Arguments

X  the first time series to be considered in the cointegration test. A plain or zoo vector. Alternatively, a two-column matrix or data.frame, in which case Y should be omitted.

Y  the second time series to be considered in the cointegration test. A plain or zoo vector.

E  an object of class "egcm" returned from a previous call to egcm

na.action  a function that indicates what should happen when the data contain NAs. See lm.

log  a boolean value which if TRUE, indicates that the model should be fit to the logs of the input vectors X and Y. Default: FALSE.

normalize  a boolean value which if TRUE, indicates that each series should be normalized to start at 1. This is performed by dividing the series by its first element. Default: FALSE.

debias  a boolean value which if TRUE, indicates that the value of rho that is reported should be debiased. Default: TRUE.

robust  a boolean value which if TRUE, indicates that the two-step Engle-Granger procedure should be performed using a robust linear model rather than a standard linear model. See rlm. Default: FALSE.

include.const  a boolean which if TRUE, indicates that the constant term alpha should be included in the model. Otherwise, sets alpha = 0. Default: TRUE.

i1test  a mnemonic indicating the name of the test that should be used for checking if the input series X and Y are integrated. If none is specified, then defaults to the value reported by egcm.default.i1test(). The installation default is "pp". The following tests are supported:
- "adf" Augmented Dickey-Fuller test (see adf.test)
- "pp" Phillips-Perron test (see pp.test)
- "pgff" Pantula, Gonzales-Farias and Fuller weighted symmetric estimate (see pgff.test)
- "bvr" Breitung’s variance ratio (see bvr.test)

urtest  a mnemonic indicating the name of the test that should be used for checking if the residual series contains a unit root. If none is specified, then defaults to the value reported by egcm.default.urtest(). The installation default is "pp". The following tests are supported:
- "adf" Augmented Dickey-Fuller test (see adf.test)
- "pp" Phillips-Perron test (see pp.test)
- "pgff" Pantula, Gonzales-Farias and Fuller weighted symmetric estimate (see pgff.test)
- "bvr" Breitung’s variance ratio (see bvr.test)
- "jo-e" Johansen’s eigenvalue test (see ca.jo)
- "jo-t" Johansen’s trace test (see ca.jo)
- "ers-p" Elliott, Rothenberg and Stock point optimal test (see ur.ers)
- "ers-d" Elliott, Rothenberg and Stock DF-GLS test (see ur.ers)
egcm

- "sp-r" Schmidt and Phillips rho statistic (see ur.sp)
- "hurst" Hurst exponent calculated using the corrected empirical method (see hurstexp)

*p.value* the p-value to be used in the above tests. If none is specified, then defaults to the value reported by egcm.default.pvalue(). The installation default is 0.05.

**Details**

The two-step Engle Granger procedure searches for parameters $\alpha$, $\beta$, and $\rho$ that yield the best fit to the following model:

\[
Y[i] = \alpha + \beta \times X[i] + R[i]
\]

\[
R[i] = \rho \times R[i-1] + \epsilon[i]
\]

$\epsilon[i] \sim N(0,\sigma^2)$

In the first step, *alpha* and *beta* are found using a linear fit of $X[i]$ with respect to $Y[i]$. The residual sequence $R[i]$ is then determined. Then, in the second step, $\rho$ is determined, again using a linear fit.

Engle and Granger showed that if $X$ and $Y$ are cointegrated, then this procedure will yield consistent estimates of the parameters. However, there are several ways in which this estimation procedure can fail:

- Either $X$ or $Y$ (or both) may already be mean-reverting. In this case, there is no point in forming the difference $Y - \beta X$. If one series is mean-reverting and the other is not, then any non-trivial linear combination will not be mean-reverting.

- The residual series $R[i]$ may not be mean-reverting. In the language of cointegration theory, it is then said to contain a unit root. In this case, there is no benefit to forming the linear combination $Y - \beta X$.

- The residual series $R[i]$ may be mean-reverting, but the relation $R[i] = \rho \times R[i-1] + \epsilon[i]$ may not be the right model. In other words, the residual series may not be adequately described by an auto-regressive series of order one. In this case, the parameters $\alpha$ and $\beta$ will be correct, however the specification for the residuals $R[i]$ will not be. The user may wish to try fitting the residuals using another function, such as arima.

The egcm function checks for each of the above contingencies, using an appropriate statistical test. If one of the above conditions is found, then a warning message is displayed when the model is printed.

The p-value used in the above tests is given by the parameter *p.value*. This can be changed by setting the value of the parameter, or by changing the default value with egcm.set.default.pvalue. For all of the unit root tests, the p-values of the corresponding test statistics have been recomputed through simulation and a table lookup is used. The Ljung-Box test (see Box.test) is used to assess whether or not the residual series can be adequately fit with an autoregressive series of order one.

The estimates of $\alpha$ and $\beta$ are not only consistent but also unbiased. Unfortunately, the estimate obtained for $\rho$ may be biased. Therefore, a bias correction has been implemented for $\rho$. A pre-computed table of biases has been determined through simulation, and a table lookup is performed to determine the appropriate bias correction. To turn off this feature, set debias = FALSE.
The helper function `is.cointegrated()` takes as input an "egcm" object `E`. It returns TRUE if `E` appears to represent a valid pair of cointegrated series. In other words, it checks that both `X` and `Y` are integrated and that the residual series `R` is free of unit roots. The helper function `is.ar1()` also takes as input an "egcm" object `E`. It returns TRUE if the residual series `R` can be adequately fit by an autoregressive model of order one.

From the standpoint of securities trading, cointegration is thought to provide a useful model for pairs trading. If the price series of two securities are cointegrated, then the corresponding residual series `R[i]` will be mean-reverting. When the magnitude of the residual `R[N]` is large, a trader might establish a long position in the undervalued security and a short position in the overvalued security. With high probability, the positions will converge in value, and a profit can be collected. Numerous scholarly articles and several books have been written on pairs trading.

Data mining for cointegrated pairs is not recommended, though. As with any statistical test, the cointegration test will generate false positives. Experience shows that at least in the case of the components of the S&P 500, the number of false positives overwhelms the number of truly cointegrated series.

**Value**

Returns an S3 object of class "egcm". This can then be printed or plotted. There is also a summary method.

The following is a copy of the printed output that was obtained from running the first example below:

```
VOO[i] = 0.9201 SPY[i] - 0.6845 + R[i],
       (0.0005)    (0.0845)  
R[i] = -0.0004 R[i-1] + eps[i], eps ~ N(0, 0.0779^2)
       (0.0633)  
R[2013-12-31] = -0.0987 (t = -1.265)
```

The first line of the output shows the fit that was found. The parameters were determined to be $\beta = 0.9201$, $\alpha = -0.6845$ and $\rho = -0.0004$. The standard deviation of the sequence $\epsilon$ of innovations was found to be $0.0779$. The standard errors of $\alpha$, $\beta$ and $\rho$ were found to be $0.0845$, $0.0005$ and $0.0633$ respectively.

The third line of output shows the value of the residual as of the last observation in the series. The sign of the value $-0.0987$ indicates that VOO was relatively undervalued on this date and that the difference between the two series was $-1.265$ standard deviations from their historical mean.

The fields of the "egcm" object are as follows:

- **S1**: the first data series ($X[i]$)
- **S2**: the second data series ($Y[i]$)
- **residuals**: the residual series ($R[i]$)
- **innovations**: the sequence of innovations ($\epsilon[i]$)
- **index**: the index vector for the series
- **i1test**: the name of the test used for verifying that $X$ and $Y$ are integrated
urtest the name of the test used for verifying that the residual series does not contain a unit root
pvalue the p-value that is used for the various tests used by this model
log Boolean, which if true indicates that S1 and S2 are logged
alpha the computed value of $\alpha$
alpha.se standard error of the estimate of $\alpha$
beta the computed value of $\beta$
beta.se standard error of the estimate of $\beta$
rho the computed and debiased value of $\rho$
rho.raw the value of $\rho$ determined prior to debiasing
rho.se standard error of the estimate of $\rho$
s1.i1.stat test statistic found when checking that S1 is integrated
s1.i1.p p-value associated to s1.i1.stat
s2.i1.stat test statistic found when checking that S2 is integrated
s2.i1.p p-value associated to s2.i1.stat
r.stat test statistic found when checking whether the residual series contains a unit root
r.p p-value associated to r.stat
eps.ljungbox.stat test statistic found when checking whether an AR(1) model adequately fits the residual series
eps.ljungbox.p p-value associated to eps.ljungbox.stat
s1.dsd standard deviation of diff(S1)
s2.dsd standard deviation of diff(S2)
r.sd standard deviation of residuals
eps.sd standard deviation of the innovations $\epsilon_i$

Disclaimer

The software in this package is for general information purposes only. It is hoped that it will be useful, but it is provided WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. It is not intended to form the basis of any investment decision. USE AT YOUR OWN RISK!

Note

Cointegration is a more general concept than has been presented here. Users who wish to explore more general models for cointegration are referred to the urca package of Bernard Pfaff.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>
References


See Also

yeegcm egcm.default.i1test egcm.default.urtest egcm.default.pvalue sim.egcm pgff.test bvr.test ca.jo

Examples

```r
## Not run:
library(quantmod)

# SPY and IVV are both ETF's that track the S&P 500.
# One would expect them to be cointegrated, and in 2013 they were.
spy2013 <- getSymbols("SPY", from = "2013-01-01",
                       to = "2013-12-31",auto.assign = FALSE)$SPY.Adjusted
ivv2013 <- getSymbols("IVV", from = "2013-01-01",
                      to = "2013-12-31",auto.assign = FALSE)$IVV.Adjusted
egcm(spy2013, ivv2013)

# egcm has a plot method, which can be useful
# In this plot, it appears that there is only one price series,
# but that is because the two price series are so close to each
# other that they are indistinguishable.
plot(egcm(spy2013, ivv2013))

# The yeegcm method provides a convenient interface to the quantmod
# package, which can fetch closing prices from Yahoo. Thus,
# the above can be simplified as follows:

e <- yeegcm("SPY", "VOD", start="2013-01-01", end="2014-01-01")
print(e)
plot(e)
sim(e)

# GLD and IAU both track the price of gold.
# They too tend to be very tightly cointegrated.
gld.iau.2013 <- yeegcm("GLD", "IAU", start="2013-01-01", end="2013-12-31")
gld.iau.2013
```
# Coca-cola and Pepsi are often mentioned as an example of a pair of securities for which pairs trading may be fruitful. However, at least in 2013, they were not cointegrated.

```r
doctest
ko.pep.2013 <- yegcm("KO", "PEP", start="2013-01-01", end="2013-12-31")
ko.pep.2013
plot(ko.pep.2013)

# Ford and GM seemed to be even more tightly linked. Yet, the degree of linkage was not high enough to pass the cointegration test.

f.gm.2013 <- yegcm("F","GM", start="2013-01-01", end="2013-12-31")
f.gm.2013
plot(f.gm.2013)
```

## End(Not run)

---

**egcm.defaults**

Set and get defaults for Engle-Granger cointegration models

### Description

Set and get defaults for Engle-Granger cointegration models

### Usage

```r
egcm.set.default.itest(itest)
egcm.default.itest()
egcm.itests()

egcm.set.default.urtest(urtest)
egcm.default.urtest()
egcm.urtests()

egcm.set.default.pvalue(p)
egcm.default.pvalue()
```

### Arguments

- **itest**: a mnemonic indicating the name of the test that should be used for checking if the input series are integrated. The following tests are supported:
  - "adf" Augmented Dickey-Fuller test (see `adf.test`)
  - "pp" Phillips-Perron test (see `pp.test`)
  - "pgff" Pantula, Gonzales-Farias and Fuller weighted symmetric estimate (see `pgff.test`)
  - "bvr" Breitung’s variance ratio (see `bvr.test`)
urtest

A mnemonic indicating the name of the test that should be used for checking if the residual series contains a unit root. The following tests are supported:

- "adf" Augmented Dickey-Fuller test (see adf.test)
- "pp" Phillips-Perron test (see pp.test)
- "pgff" Pantula, Gonzales-Farias and Fuller weighted symmetric estimate (see pgff.test)
- "bvr" Breitung’s variance ratio (see bvr.test)
- "jo-e" Johansen’s eigenvalue test (see ca.jo)
- "jo-t" Johansen’s trace test (see ca.jo)
- "ers-p" Elliott, Rothenberg and Stock point optimal test (see ur.ers)
- "ers-d" Elliott, Rothenberg and Stock DF-GLS test (see ur.ers)
- "sp-r" Schmidt and Phillips rho statistic (see ur.sp)
- "hurst" Hurst exponent calculated using the corrected empirical method (see hurstexp)

p

The p-value should be used for rejecting the null hypothesis in the various statistical tests conducted by egcm.

Value

For egcm.default.itest, returns the string representing the currently selected default I(1) test. For egcm.itests, returns a list of all available I(1) tests.

For egcm.default.urtest, returns the string representing the currently selected unit root test. For egcm.urtests, returns a list of all available unit root tests.

For egcm.default.pvalue, returns the default p-value that will be used for rejecting the null hypothesis in the various statistical tests conducted by egcm.

The setter functions do not return a value.

Note

Changing the default value only affects egcm objects created after the change is made.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

See Also

egcm

Examples

# Get and set the current default I(1) test
egcm.default.itest()
egcm.set.default.itest("adf")

# Get and set the current default unit root test
egcm.default.urtest()
egcm.set.default.urtest("pp")

# Get and set the current default p-value
egcm.default.pvalue()
egcm.set.default.pvalue(0.01)

---

pgff.test  
Unit root test of Pantula, Gonzales-Farias and Fuller

**Description**

Unit root test based upon the weighted symmetric estimator of Pantula, Gonzales-Farias and Fuller

**Usage**

pgff.test(Y, detrend = FALSE)  
pgff_rh0.ws(Y, detrend = FALSE)

**Arguments**

Y  
A vector or zoo-vector

detrend  
A boolean, which if TRUE, indicates that the test should be performed after removing a linear trend from Y

**Details**

The weighted symmetric estimator $\hat{\rho}_{WS}$ of Pantula, Gonzales-Farias and Fuller is given as follows:

$$
\hat{\rho}_{WS} = \frac{\sum_{t=2}^{n} Y_{t-1}Y_{t}}{\sum_{t=2}^{n-1} Y_{t}^2 + n^{-1} \sum_{t=1}^{n} Y_{t}^2}
$$

where $n$ is the length of the sequence $Y$.

The authors give an associated pivotal statistic and derive the limiting distribution for it, however the approach taken in this implementation was simply to determine the distribution of $\rho_{WS}$ through simulation. Table lookup is used to determine the p-value associated with a given value of the statistic.

If detrend=True, then a linear trend is removed from the data prior to computing the estimator $\rho_{WS}$. A separate table has been computed of the distribution of values of $\rho_{WS}$ after detrending.

This unit root test is intended to identify autoregressive sequences of order one. However, the authors state that, "A Monte Carlo study indicates that the weighted symmetric estimator performs well in second order processes."
Value

pgff_rho_ws returns the value $\rho_{WS}$ of the weighted symmetric estimator.

pgff.test returns a list with class "htest" containing the following components:

- statistic: the value of the test statistic.
- parameter: the truncation lag parameter.
- p.value: the p-value of the test.
- method: a character string indicating what type of test was performed.
- data.name: a character string giving the name of the data.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

References


See Also

adf.test, egcm

Examples

# The following should produce a low p-value
pgff_rho_ws(rnorm(100))
pgff.test(rnorm(100))

# The following should produce a high p-value
pgff_rho_ws(cumsum(rnorm(100)))
pgff.test(cumsum(rnorm(100)))

# Test with an autoregressive sequence where rho = 0.8
pgff.test(ar1(100, a1=0.8))

# If there is a linear trend, pgff.test with detrend=FALSE
# is likely to find a unit root when there is none:
pgff.test(1:100 + rnorm(100))
pgff.test(1:100 + rnorm(100), detrend=TRUE)

# Display the power of the test for various values of rho and n:
pgff_power(a1=0.8, n=100, nrep=100)
pgff_power(a1=0.9, n=250, nrep=100)
pgff_power(a1=0.95, n=250, nrep=100)

# This is to be compared to the power of the adf.test at this level:
adf_power(a1=0.8, n=100, nrep=100)
adf_power(a1=0.9, n=250, nrep=100)
adf_power(a1=0.95, n=250, nrep=100)
Random AR(1) vector

Description

Generates a random realization of an AR(1) sequence

Usage

rar1(n, a0 = 0, a1 = 1, trend = 0, sd = 1, x0 = 0)

Arguments

n Length of vector to produce
a0 Constant term in AR(1) sequence
a1 Coefficient of mean-reversion
trend Linear trend
sd Standard deviation of sequence of innovations
x0 Starting value of sequence

Value

If trend=0, returns a vector of length n representing a simulation of an AR(1) process

\[ X[k] = a_0 + a_1 \times X[k - 1] + \epsilon[t] \]

where \( \epsilon[t] \) is a sequence of independent and identically distributed samples from a normal distribution with mean zero and standard deviation sd.

If trend \( \neq 0 \), returns a vector of length n representing a simulation of a trend-stationary AR(1) process

\[ R[k] = a_0 + a_1 \times R[k - 1] + \epsilon[t] \]
\[ X[k] = k \times trend + R[k] \]

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

See Also

rcoint

Examples

rar1(100, 0, 0) # Equivalent to rnorm(100)
rar1(100, 0, 1) # Equivalent to cumsum(rnorm(100))
acor(rar1(100, 1, .5)) # Should be about 0.5
tseries::adf.test(rar1(100, 0, .5)) # Should have a low p-value
rcoint

Random generation of cointegrated sequences

Description

Generates a random pair of cointegrated sequences

Usage

```r
coint(n,
    alpha = runif(1, -10, 10),
    beta = runif(1, -10, 10),
    rho = runif(1, 0, 1),
    sd_eps = 1,
    sd_delta = 1,
    x0=0,
    y0=0)
```

Arguments

- `n`: number of observations in each sequence
- `alpha`: constant term of linear relation
- `beta`: slope term of linear relation
- `rho`: coefficient of mean reversion
- `sd_eps`: standard deviation of innovations in first sequence
- `sd_delta`: standard deviation of innovations in residual sequence
- `x0`: initial value of first sequence
- `y0`: initial value of second sequence

Details

Generates a random pair of cointegrated sequences. The sequences are constructed by first generating two random sequences that are independent and normally distributed. The elements of the first sequence, $\epsilon[i]$, have standard deviation `sd_eps`, while those of the second sequence, $\delta[i]$, have standard deviation `sd_delta`. Having generated these two sequences, the cointegrated sequences $X[i]$ and $Y[i]$ are generated according to the following relations:

$$X[i] = X[i - 1] + \epsilon[i]$$

$$R[i] = \rho R[i - 1] + \delta[i]$$

$$Y[i] = \alpha + \beta X[i] + R[i]$$

Value

Returns a two-column data.frame containing the randomly generated cointegrated sequences.
Author(s)
Matthew Clegg <matthewcleggphd@gmail.com>

See Also
rar1 sim.egcm egcm

Examples
xy <- rcount(1000, alpha = 1, beta = 2, rho = 0.8)
egcm(xy)

Description
Given an Engle-Granger cointegration model and the number of steps to simulate, generates a simulated realization of that model for the specified number of steps.

Usage
sim.egcm(E, nsteps, X0, Y0)

Arguments
E        the Engle-Granger model to be simulated. See egcm
nsteps   the number of steps to simulate
X0       the starting value of X to be used in the simulation. If not specified, uses the last value of X in E.
Y0       the starting value of Y to be used in the simulation. If not specified, uses the last value of Y in E.

Value
Returns a two-column data.frame, where the first column contains the simulated values of X, and the second column contains the simulated values of Y.

Author(s)
Matthew Clegg <matthewcleggphd@gmail.com>

See Also
egcm rcount
Examples

# Generate a random pair of cointegrated vectors
cv1 <- rcoint(1000)
# Construct a cointegration model from them
e1 <- egcm(cv1)
# Simulate the model for an additional 1000 steps
cv2 <- sim.egcm(e1, 1000)
# Construct a cointegration model from the simulated data
e2 <- egcm(cv2)
# Compare the original model to the model obtained from simulation
e1
e2

ur_power

Power assessment for unit root tests

Description

A collection of functions designed to assist in determining the power of various unit root tests

Usage

ur_power (ur_test, a0 = 0, a1 = 0.95, trend=0, n = 250,
nrep = 10000, p.value = 0.05, ...)
adf_power (a0=0, a1=0.95, trend=0, n=250,
nrep=10000, p.value=0.05, k=1)
bvr_power (a0=0, a1=0.95, trend=0, n=250,
nrep=10000, p.value=0.05, detrend=FALSE)
pgff_power (a0=0, a1=0.95, trend=0, n=250,
nrep=10000, p.value=0.05, detrend=FALSE)

ur_power_table (ur_test, nrep=1000, p.value=0.05,
a1=c(0.995, 0.99, 0.98, 0.97, 0.96, 0.95),
trend=0,
n=c(100, 250, 500, 750, 1000, 1250),
...)
adf_power_table (nrep=1000, p.value=0.05,
a1=c(0.995, 0.99, 0.98, 0.97, 0.96, 0.95),
trend=0,
n=c(250, 500, 750, 1000, 1250),
k=1)
bvr_power_table (nrep=1000, p.value=0.05,
a1=c(0.995, 0.99, 0.98, 0.97, 0.96, 0.95),
trend=0,
n=c(100, 250, 500, 750, 1000, 1250),
detrend=FALSE)
pgff_power_table (nrep=1000, p.value=0.05,
Arguments

ur_test A function that performs a unit root test. It should accept an argument consisting of a vector of real numbers, and it should return an object with the p-value stored in the field p.value. Example functions that satisfy this criterion include 

\texttt{adf.test, pp.test, pgff.test} and \texttt{bvr.test}

a0 Constant term of AR(1) series

a1 Linear term of AR(1) series (e.g. coefficient of mean reversion). For the \_power_table variants, this may be a vector of numbers, representing different values of the linear term that should be tried.

trend Trend parameter. This may either be a scalar or it may be a vector of length nrep. In the latter case, each replication of the test is performed with a different value from trend.

n Length of AR(1) series. For the \_power_table variants, this may be a vector of numbers, representing different sequence lengths that should be tried.

nrep Number of repetitions to perform

p.value p-value used as cutoff point for rejecting the null hypothesis

detrend A boolean which, if TRUE, indicates that linear trends should be removed from the AR(1) series prior to performing the unit root test.

k Number of lags to consider in Dickey-Fuller test

\ldots Additional arguments to be passed to the unit root test \texttt{ur.test}.

Details

The purpose of this family of functions is to provide a means for investigating the power of various unit root tests. The power of a statistical test is the probability that it will reject the null hypothesis when the null hypothesis is false.

For unit root tests, a common practice for assessing power is to randomly generate AR(1) sequences of a fixed length and with a fixed coefficient of mean reversion, and to quantify the power in terms of these two parameters. That is the approach taken here.

The \_power functions generate nrep random AR(1) sequences of length n having the parameters a0 and a1. For each such sequence, the unit root test is performed and a check is made to see if the null hypothesis is rejected at the level given by p.value. The frequency of rejections is then reported.

The \_power_table functions generate a table of powers for various choices of n and a1. These functions can take quite a while to run.

\texttt{adf_power} and \texttt{adf_power_table} report the power of the augmented Dickey-Fuller test as implemented in \texttt{adf.test}. \texttt{bvr_power} and \texttt{bvr_power_table} report the power of Breitung’s variance ratio as implemented in \texttt{bvr.test}. \texttt{pgff_power} and \texttt{pgff_power_table} report the power of the weighted symmetric estimator of Pantula, Gonzalez-Farias and Fuller as implemented in \texttt{pgff.test}.
Value

For the *.power functions, returns the frequency of rejections of the null hypothesis.

For the *.power_table functions, returns a data.frame. Each column corresponds to a value of the mean reversion coefficient given in the vector a1, and each row corresponds to a sample length given in the vector n. An entry in the table records the frequency of rejections of the null hypothesis for the given sample length and coefficient of mean reversion.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

References


See Also

adf.test pp.test bvr.test pgff.test

Examples

# The following examples may take a long time to run

# Compare the power of various unit root tests for specific parameter values:
# adf.power(a1=0.9, n=125, p.value=0.1)
# bvr.power(a1=0.9, n=125, p.value=0.1)
# pgff.power(a1=0.9, n=125, p.value=0.1)

# library(tseries)
# ur_power(pp.test, a1=0.9, n=125, p.value=0.1)

# The following illustrates the importance of de-trending
# pgff.power(a1=0.9, n=125, p.value=0.1, trend=10)
# pgff.power(a1=0.9, n=125, p.value=0.1, trend=10, detrend=TRUE)

# Generate tables comparing the powers of various unit root tests:
# adf.power_table()
# bvr.power_table()
# pgff.power_table()
# ur.power_table(pp.test)
Engle-Granger cointegration model from Yahoo! price series

Description

Fetches the Yahoo! price series for two securities and constructs an Engle-Granger cointegration model from them.

Usage

```r
yegcm(ticker1, ticker2, start = format(Sys.Date() - 365, "%Y-%m-%d"), end = format(Sys.Date(), "%Y-%m-%d"), clear.na.inf=TRUE, ...)
```

Arguments

- `ticker1`: the ticker symbol of the first security
- `ticker2`: the ticker symbol of the second security
- `start`: starting date, given in the format "YYYY-MM-DD". Default: One year ago.
- `end`: ending date, given in the format "YYYY-MM-DD". Default: Today.
- `clear.na.inf`: if TRUE, NA and Inf price values are replaced by the last available price. Default: TRUE.
- `...`: additional parameters passed to `egcm`

Details

Uses the `getSymbols` function of the `quantmod` package to retrieve the adjusted closing prices of the two securities over the specified date range. Then, constructs an Engle-Granger cointegration model from this data, and returns it.

Value

An Engle-Granger cointegration model

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

References

See Also

eyegcm getSymbols

Examples

e <- yegcm("SPY", "VOO", start="2013-01-01", end="2014-01-01")
print(e)
plot(e)
summary(e)
Index

*Topic models
   allpairs.eegcm, 5
   eegcm, 9
   eegcm-package, 2
   yegcm, 25
*Topic package
   eegcm-package, 2
*Topic ts
   acor, 4
   allpairs.eegcm, 5
   bvr.test, 7
   eegcm, 9
   eegcm-package, 2
   pgff.test, 17
   rar1, 19
   rcoint, 20
   ur_power, 22
   yegcm, 25

acf, 4
acor, 4
adf.test, 3, 10, 15, 16, 18, 23, 24
adf_power (ur_power), 22
adf_power_table (ur_power), 22
allpairs.eegcm, 5
arima, 11
Box.test, 11
bvr.test, 3, 7, 10, 14–16, 23, 24
bvr_power (ur_power), 22
bvr_power_table (ur_power), 22
bvr_rho (bvr.test), 7
c.a.jo, 10, 14, 16
detrend, 8
eegcm, 3, 5, 6, 8, 9, 16, 18, 21, 25, 26
eegcm-package, 2
eegcm.default.ilttest, 14
eegcm.default.ilttest (eegcm.defaults), 15

egcm.default.pvalue, 14
egcm.default.pvalue (eegcm.defaults), 15
egcm.default.urtest, 14
egcm.default.urtest (eegcm.defaults), 15
egcm.defaults, 15
eegcm.ilttests (eegcm.defaults), 15
eegcm.set.default.ilttest
   (eegcm.defaults), 15
eegcm.set.default.pvalue, 11
egcm.set.default.pvalue
   (eegcm.defaults), 15
egcm.set.default.urtest
   (eegcm.defaults), 15
eegcm.urtests (eegcm.defaults), 15
getSymbols, 25, 26

hurstexp, 3, 8, 11, 16

is.ar1 (eegcm), 9
is.cointegrated (eegcm), 9

lm, 10

pgff.test, 3, 10, 14–16, 17, 23, 24
pgff_power (ur_power), 22
pgff_power_table (ur_power), 22
pgff_rho_ws (pgff.test), 17
plot.eegcm (eegcm), 9
pp.test, 3, 10, 15, 16, 23, 24
rar1, 19, 21
rcoint, 19, 20, 21
rlm, 10

sim.eegcm, 14, 21, 21
summary.eegcm (eegcm), 9

ur.ers, 10, 16
ur.sp, 11, 16
ur_power, 22
ur_power_table (ur_power), 22
urca, 2, 3, 13
yegcm, 14, 25
zoo, 4, 10