Package 'egcm'

February 19, 2015

Type Package

Title Engle-Granger Cointegration Models

Version 1.0.6

Date 2014-01-24

Description This package provides 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, TTR

Imports grid, ggplot2, tseries, MASS, urca, parallel, fArma

License GPL-2 | GPL-3

NeedsCompilation no

Author Matthew Clegg [aut, cre, cph]

Maintainer Matthew Clegg <matthewcleggphd@gmail.com>

Repository CRAN

Date/Publication 2015-02-06 15:50:51

R topics documented:

`egcm-package` *Simplified Engle-Granger Cointegration Models*

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 **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 α, β and ρ such that:

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

$$R[i] = \rho * R[i-1] + \epsilon$$

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 ρ_{ws} of Pantula, Gonzales-Farias and Fuller is implemented as `pgff.test`. This test seems to provide superior performance to the standard Dickey-Fuller test `adf.test` and also improves upon the performance of a number of other tests previously available in R.

The variance ratio test proposed by J. Breitung is implemented as `bvr.test`. 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 R, although it does not perform as well as `pgff.test`.

Users who wish to explore more general models for cointegration are referred to the `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

Breitung, J. (2002). Nonparametric tests for unit roots and cointegration. *Journal of econometrics*, 108(2), 343-363.

Chan, E. (2013). *Algorithmic trading: winning strategies and their rationale*. (Vol. 625). John Wiley & Sons.

Clegg, M. (2014). On the Persistence of Cointegration in Pairs Trading (January 28, 2014). Available at SSRN: http://ssrn.com/abstract=2491201

Ehrman, D.S. (2006). *The handbook of pairs trading: strategies using equities, options, and futures*. (Vol. 240). John Wiley & Sons.

Engle, R. F. and C. W. Granger. (1987) Co-integration and error correction: representation, estimation, and testing. *Econometrica*, 251-276.

Pantula, S. G., Gonzalez-Farias, G., and Fuller, W. A. (1994). A comparison of unit-root test criteria. *Journal of Business & Economic Statistics*, 12(4), 449-459.

Pfaff, B. (2008) *Analysis of Integrated and Cointegrated Time Series with R. Second Edition*. Springer, New York. ISBN 0-387-27960-1

Vidyamurthy, G. (2004). *Pairs trading: quantitative methods and analysis*. (Vol 217). Wiley.com.

See Also

`egcm` Further documentation of the Engle-Granger cointegration model

`pgff.test` Unit root test based on the weighted symmetric estimator of Pantula, Gonzales-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

`aggvarFit` Unit root tests based on variance ratios

Examples

```
library(TTR)
prices.spy <- getYahooData("SPY", 20130101, 20140101)$Close
prices.voo <- getYahooData("VOO", 20130101, 20140101)$Close
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:

e <- yegcm("SPY", "VOO", 20130101, 20140101)
print(e)
plot(e)
summary(e)
```

acor *autocorrelation*

Description

autocorrelation of a sequence

Usage

```
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

```
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, a1=0.8))  # slightly less than 0.8
acor(rar1(1000, a1=0.8), k=2)  # about 0.64
acor(rar1(1000, a1=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

```
allpairs.egcm(tickers,
  startdate = as.numeric(format(Sys.Date() - 365, "%Y%m%d")),
  enddate = as.numeric(format(Sys.Date(), "%Y%m%d")), ...)
```

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 `YYYYMMDD`. Defaults to one year ago.
`enddate`	The ending date for which to download the data. Given in the form `YYYYMMDD`. Defaults to today.
`...`	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.
- 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

egcm

Examples

```
## 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 ρ_T. A separate table has been computed of the distribution of values of ρ_T after detrending.

Value

bvr_rho returns the value ρ_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

Breitung, J. (2002). Nonparametric tests for unit roots and cointegration. *Journal of econometrics*, 108(2), 343-363.

Breitung, J. and Taylor, A.M.R. (2003) Corrigendum to "Nonparametric tests for unit roots and cointegration" [J. Econom. 108 (2002) 343-363] *Journal of econometrics*, 117(2), 401-404.

See Also

aggvarFit egcm

Examples

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

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

# Test with an autoregressive sequence where rho = 0.8
bvr.test(rar1(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 -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,
  i1test = egcm.default.i1test(),
  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`)
- `"sp-r"` Schmidt and Phillips rho statistic (see `ur.sp`)
- `"hurst"` Hurst exponent calculated using the aggregated variance method (see `aggvarFit`)

`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 α, β, and ρ that yield the best fit to the following model:

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

$$R[i] = \rho * R[i-1] + \epsilon[i]$$

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

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, ρ 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 $R[i] = \rho R[i-1] + \epsilon[i]$ 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 α and β 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 α and β are not only consistent but also unbiased. Unfortunately, the estimate obtained for ρ may be biased. Therefore, a bias correction has been implemented for ρ. A precomputed 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 ϵ of innovations was found to be 0.0779. The standard errors of α, β and ρ 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 (ϵ[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 α
alpha.se	standard error of the estimate of α
beta	the computed value of β
beta.se	standard error of the estimate of β
rho	the computed and debiased value of ρ
rho.raw	the value of ρ determined prior to debiasing
rho.se	standard error of the estimate of ρ
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

Chan, E. (2013). *Algorithmic trading: winning strategies and their rationale*. (Vol. 625). John Wiley & Sons.

Clegg, M. (2014). On the Persistence of Cointegration in Pairs Trading (January 28, 2014). Available at SSRN: http://ssrn.com/abstract=2491201

Ehrman, D.S. (2006). *The handbook of pairs trading: strategies using equities, options, and futures*. (Vol. 240). John Wiley & Sons.

Engle, R. F. and C. W. Granger. (1987) Co-integration and error correction: representation, estimation, and testing. *Econometrica*, 251-276.

Pfaff, B. (2008) *Analysis of Integrated and Cointegrated Time Series with R. Second Edition*. Springer, New York. ISBN 0-387-27960-1

Vidyamurthy, G. (2004). *Pairs trading: quantitative methods and analysis*. (Vol 217). Wiley.com.

See Also

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

Examples

```
library(TTR)

# 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 <- getYahooData("SPY", 20130101, 20131231)$Close
ivv2013 <- getYahooData("IVV", 20130101, 20131231)$Close
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 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:

e <- yegcm("SPY", "VOO", 20130101, 20140101)
print(e)
plot(e)
summary(e)

# GLD and IAU both track the price of gold.
# They too tend to be very tightly cointegrated.
gld.iau.2013 <- yegcm("GLD", "IAU", 20130101, 20131231)
gld.iau.2013
plot(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.
ko.pep.2013 <- yegcm("KO", "PEP", 20130101, 20131231)
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", 20130101, 20131231)
f.gm.2013
plot(f.gm.2013)
```

`egcm.defaults` *Set and get defaults for Engle-Granger cointegration models*

Description

Set and get defaults for Engle-Granger cointegration models

Usage

```
egcm.set.default.i1test(i1test)
egcm.default.i1test()
egcm.i1tests()

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

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

Arguments

i1test	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 aggregated variance method (see aggvarFit)
p	the p-value should be used for rejecting the null hypothesis in the various statistical tests conducted by egcm.

Value

For `egcm.default.i1test`, returns the string representing the currently selected default I(1) test. For `egcm.i1tests`, returns a list of all available I(1) tests.

For `egcm.default.urtest`, returns the string represeting 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.i1test()
egcm.set.default.i1test("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_rho_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 ρ_{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 ρ_{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 ρ_{WS}. A separate table has been computed of the distribution of values of ρ_{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 ρ_{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

Pantula, S. G., Gonzalez-Farias, G., and Fuller, W. A. (1994). A comparison of unit-root test criteria. *Journal of Business & Economic Statistics*, 12(4), 449-459.

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(rar1(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)
```

---

rar1 *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

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

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 != 0`, returns a vector of length `n` representing a simulation of a trend-stationary AR(1) process

$$R[k] = a_0 + a_1 * R[k-1] + \epsilon[t]$$

$$X[k] = k * trend + R[k]$$

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

```
rcoint(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 <- rcoint(1000, alpha = 1, beta = 2, rho = 0.8)
egcm(xy)
```

sim.egcm *Generate simulated data from an Engle-Granger cointegration model*

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 rcoint

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,
    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)
```

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 adf.test, pp.test, pgff.test and 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
...	Additional arguments to be passed to the unit root test 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.

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

Breitung, J. (2002). Nonparametric tests for unit roots and cointegration. *Journal of econometrics*, 108(2), 343-363.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a), 427-431.

Pantula, S. G., Gonzalez-Farias, G., and Fuller, W. A. (1994). A comparison of unit-root test criteria. *Journal of Business & Economic Statistics*, 12(4), 449-459.

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)
```

`yegcm` *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

```
yegcm(ticker1, ticker2,
  start = as.numeric(format(Sys.Date() - 365, "%Y%m%d")),
  end = as.numeric(format(Sys.Date(), "%Y%m%d")),
  ...)
```

Arguments

ticker1	the ticker symbol of the first security
ticker2	the ticker symbol of the second security
start	starting date, given in the format YYYYMMDD. Default: One year ago.
end	ending date, given in the format YYYYMMDD. Default: Today.
...	additional parameters passed to egcm

Details

Uses the getYahooData function of the TTR 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

Engle, R. F. and C. W. Granger. (1987) Co-integration and error correction: representation, estimation, and testing. *Econometrica*, 251-276.

See Also

egcm getYahooData

Examples

```
e <- yegcm("SPY", "VOO", 20130101, 20140101)
print(e)
plot(e)
summary(e)
```

Index