Package ‘partialCI’

March 6, 2018

Type Package

Title Partial Cointegration

Version 1.1.1

Date 2018-03-05

Author Matthew Clegg [aut],
Christopher Krauss [aut],
Jonas Rende [cre, aut]

Maintainer Jonas Rende <jonas.rende@fau.de>

Description
A collection of time series is partially cointegrated if a linear combination of these time series can be found so that the residual spread is partially autoregressive - meaning that it can be represented as a sum of an autoregressive series and a random walk. This concept is useful in modeling certain sets of financial time series and beyond, as it allows for the spread to contain transient and permanent components alike. Partial cointegration has been introduced by Clegg and Krauss (2017) <doi:10.1080/14697688.2017.1370122>, along with a large-scale empirical application to financial market data. The partialCI package comprises estimation, testing, and simulation routines for partial cointegration models in state space. Clegg et al. (2017) <https://hdl.handle.net/10419/150014> provide an in in-depth discussion of the package functionality as well as illustrating examples in the fields of finance and macroeconomics.

License GPL-2 | GPL-3

Depends partialAR

Imports zoo, parallel, ggplot2, grid, MASS, TTR, data.table, glmnet,
methods, Rcpp, KFAS

Suggests egcm, knitr, rmarkdown

RoxygenNote 6.0.1

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-03-06 11:53:03 UTC

R topics documented:

`partialCI-package` *Partial Cointegration*

Description

A collection of time series is said to be partially cointegrated if they have a linear combination that is partially autoregressive, e.g., that can be represented as a sum of an autoregressive series and a random walk. This may be useful in modeling certain sets of financial time series.

To find the partially cointegrated model that best fits two series X and Y, use:

```
> fit.pci(Y, X)
```

An interface to Yahoo! Finance permits you to find the best fits for two particular stocks of interest:

```
> yfit.pci("RDS-B", "RDS-A")
Fitted values for PCI model
  Y[t] = alpha + X[t]
  M[t] = rho * M[t-1] + eps_M [t], eps_M[t] ~ N(0, sigma_M^2)
  R[t] = R[t-1] + eps_R [t], eps_R[t] ~ N(0, sigma_R^2)

           Estimate Std. Err
alpha        0.2063   0.8804
beta_RDS-A   1.0531   0.0133
rho          0.9055   0.0355
sigma_M      0.2431   0.0162
sigma_R      0.0993   0.0350

-LL = 41.30, R^2[MR] = 0.863
```

This example was run on 1/7/2016. RDS-A and RDS-B are two classes of shares offered by Royal Dutch Shell that differ slightly in aspects of their tax treatment. The above fit shows that the spread between the two shares is mostly mean-reverting but that it contains a small random walk component. The mean-reverting component accounts for 86.3% of the variance of the daily returns. The value of 0.9055 for rho corresponds to a half-life of mean reversion of about 7 trading days.

To test the goodness of fit, the `test.pci` function can be used:

```
> h <- yfit.pci("RDS-B", "RDS-A")
> test.pci(h)

Likelihood ratio test of [Random Walk or CI(1)] vs Almost PCI(1) (joint penalty method)

data:  h

Hypothesis              Statistic    p-value
Random Walk                 -4.94      0.010
AR(1)                       -4.08      0.010
Combined                               0.010
```

The `test.pci` function tests each of two different null hypotheses: (a) the residual series is purely a random walk, and (b) the residual series is purely autoregressive. In addition, the union of these hypothesis is also tested. For practical applications, one is usually most interested in rejecting the first of these null hypotheses, e.g., that the residual series is purely a random walk.

The partialCI package also contains a function for searching for hedging portfolios. Given a particular stock (or time series), a search can be conducted to find the set of stocks that best replicate the target stock. In the following example, a hedge is sought for SPY using sector ETF's.

```
> sectorETFS <- c("XLB", "XLE", "XLF", "XLI", "XLK", "XLP", "XLU", "XLV", "XLY")
> prices <- multigetYahooPrices(c("SPY", sectorETFS), start=20140101)
> hedge.pci(prices[,"SPY"], prices)
    -LL   LR[rw]   p[rw]   p[mr]     rho  R^2[MR]  Factor |   Factor coefficients
  490.67  -1.7771   0.1782   0.0100   0.9587   0.8246      XLF |    6.8351
  283.26  -4.3988   0.0137   0.0786   0.9642   1.0000      XLK |   3.6209   2.2396
  168.86  -6.4339   0.0100   0.0100   0.7328   0.6619     XLI |   2.3191   1.6542   1.1391

Fitted values for PCI model
  Y[t] = alpha + X[t]
  M[t] = rho * M[t-1] + eps_M [t], eps_M[t] ~ N(0, sigma_M^2)
  R[t] = R[t-1] + eps_R [t], eps_R[t] ~ N(0, sigma_R^2)

           Estimate Std. Err
alpha       14.2892   1.5598
beta_XLF     2.3191   0.1439
beta_XLK     1.6542   0.0804
beta_XLI     1.1391   0.0662
rho          0.7328   0.1047
sigma_M      0.2678   0.0315
sigma_R      0.2056   0.0401
```

```
-LL = 168.86, R^2[MR] = 0.662
```

The top table displays the quality of the fit that is found as each new factor is added to the fit. The best fit consisting of only one factor is found by using XLF (the financials sector). The negative log likelihod score for this model is 490.67. However, the random walk hypothesis (p[rw]) cannot be rejected at the 5% level. When adding XLK (the technology sector), the negative log likelihood drops to 283.26 and the random walk hypothesis for the spread can now be rejected. This means that SPY is at least partially cointegrated and possibly fully cointegrated with a portfolio consisting of XLF and XLK in the right proportions. The best overall fit is obtained by also adding XLI (industrials) to the hedging portfolio. The final fit is

```
SPY = $14.29 + 2.32 XLF + 1.65 XLK + 1.14 XLI
```

For this fit, the proportion of variance attributable to the mean reverting component is 66.2%, and the half life of mean reversion is about 2.2 days.

Please feel free to contact us if you have questions or suggestions.

Matthew Clegg, Christopher Krauss and Jonas Rende

April 21, 2017

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

fit.pci yfit.pci test.pci hedge.pci yhedge.pci

`fit.pci` *Fits the partial cointegration model to a collection of time series*

Description

Fits the partial cointegration model to a collection of time series

Usage

```
fit.pci(Y, X,
  pci_opt_method = c("jp", "twostep"),
  par_model = c("par", "ar1", "rw"),
  lambda = 0,
  robust = FALSE, nu = 5,
  include_alpha=FALSE)
```

Arguments

`Y`	The time series that is to be modeled. A plain or `zoo` vector of length `n`.
`X`	A (possibly `zoo`) matrix of dimensions `n` x `k`. If `k=1`, then this may be a plain or `zoo` vector.
`pci_opt_method`	Specifies the method that will be used for finding the best fitting model. One of the following: • "jp" The joint-penalty method (see below) • "twostep" The two-step method (see below) Default: `jp`
`par_model`	The model used for the residual series. One of the following: • "par" The residuals are assumed to follow a partially autoregressive model. • "ar1" The residuals are assumed to be autoregressive of order one. • "rw" The residuals are assumed to follow a random walk. Default: `par`
`lambda`	The penalty parameter to be used in the joint-penalty (`jp`) estimation method. Default: 0.
`robust`	If `TRUE`, then the residuals are assumed to follow a t-distribution with `nu` degrees of freedom. Default: `FALSE`.
`nu`	The degrees-of-freedom parameter to be used in robust estimation. Default: 5.
`include_alpha`	If TRUE, then a constant term is estimated with the model. If FALSE, the constant term is omitted. Default: `FALSE`.

Details

The partial cointegration model is given by the equations:

$$Y_t = \alpha + \beta_1 * X_{t,1} + beta_2 * X_{t,2} + ... + beta_k * X_{t,k} + M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

$$\epsilon_{M,t} \sim N(0, \sigma_M^2)$$

$$\epsilon_{R,t} \sim N(0, \sigma_R^2)$$

Given the input series `Y` and `X`, this function searches for the parameter values `alpha`, `beta`, `rho` that give the best fit of this model when using a Kalman filter.

If `pci_opt_method` is `twostep`, then a two-step procedure is used. In the first step, a linear regression is performed of `X` on `Y` to determine the parameters `alpha` and `beta`. From this regression, a series of residuals is determined. In the second step, a model is fit to the residual series. If `par_model` is `par`, then a partially autoregressive model is fit to the residual series. If `par_model` is `ar1`, then an autoregressive model is fit to the residual series. If `par_model` is `rw` then a random walk model is fit to the residual series. Note that if `pci_opt_method` is `twostep` and `par_model` is `ar1`, then this reduces to the Engle-Granger two-step procedure.

If `pci_opt_method` is `jp`, then the joint-penalty procedure is used. In this method, the parameters `alpha` and `beta` are estimated jointly with the parameter `rho` using a gradient-search optimization function. In addition, a penalty value of $\lambda * \sigma_R^2$ is added to the Kalman filter likelihood score when searching for the optimum solution. By choosing a positive value for `lambda`, you can drive the solution towards a value that places greater emphasis on the mean-reverting component.

Because the joint-penalty method uses gradient search, the final parameter values found are dependent upon the starting point. There is no guarantee that a global optimum will be found. However, the joint-penalty method chooses several different starting points, so as to increase the chance of finding a global optimum. One of the chosen starting points consists of the parameters found through the two-step procedure. Because of this, the joint-penalty method is guaranteed to find parameter values which give a likelihood score at least as good as those found using the two-step procedure. Sometimes the improvement over the two-step procedure is substantial.

The parameter `include_alpha` determines whether or not a constant term is included with the fit. Note that the model `alpha = a, R0 = 0` is equivalent to the model `alpha = 0, R0 = a`. Thus, the constant term can be interpreted as an estimate of the mean of the system at time `t=0`. As the random walk evolves, the mean will drift correspondingly.

Value

An object of class `pci.fit` containing the fit that was found. The following components may be of interest

`alpha`	The constant term of the fit
`alpha.se`	The estimated standard error of `alpha`
`beta`	The vector of weights
`beta.se`	The standard errors of the components of `beta`
`rho`	The estimated coefficient of mean reversion
`rho.se`	The standard error of `rho`
`negloglik`	The negative of the log likelihood
`pvmr`	The proportion of variance attributable to mean reversion

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

References

Clegg, Matthew, 2015. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

Clegg, Matthew and Krauss, Christopher, 2016. Pairs trading with partial cointegration. FAU Discussion Papers in Economics, University of Erlangen-Nuernberg. *Available at https://www.iwf.rw.fau.de/files/2016/05/05-2016.pdf*

See Also

`egcm` Engle-Granger cointegration model

`partialAR` Partially autoregressive models

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.


YX <- rpci(n=1000, alpha=1, beta=c(2,3,4), sigma_C=c(1,1,1), rho=0.9,sigma_M=0.1, sigma_R=0.2)
fit.pci(YX[,1], YX[,2:ncol(YX)])
```

`hedge.pci` *Searches for a partially cointegrated hedge for a given time series*

Description

Given a time series and a collection of possible factors, finds a subset of the factors that provides the best fit to the given time series using the partially cointegrated model.

Usage

```
hedge.pci(Y, X,
  maxfact = 10,
  lambda = 0,
  use.multicore = TRUE,
  minimum.stepsize = 0,
  verbose = TRUE,
  exclude.cols = c(),
  search_type = c("lasso", "full", "limited"),
  pci_opt_method=c("jp", "twostep"),
  ...)
```

Arguments

`Y`	An `N x 1` column vector or data `data.frame`, representing the series that is to be hedged.
`X`	An `N x L` `data.frame`, where each column represents a possible factor to be used in a partially cointegrated fit.
`maxfact`	The maximum number of columns from `X` that will be selected for modeling `Y`. Default: 10
`lambda`	A penalty to be applied to the random walk portion of the partialAR model. A positive value for `lambda` will drive the model towards a solution with a smaller random walk component. Default: 0

use.multicore	If TRUE, parallel processing will be used to improve performance. See parallel:mclapply Default: TRUE
minimum.stepsize	If this is non-NA, then the search stops if an improvement cannot be found of at least this much. Default: 0
verbose	If TRUE, then detailed information is printed about the execution. Default: TRUE
exclude.cols	A list of column indexes specifying columns from X which should be excluded from consideration. Alternatively, the list of excluded columns may be given as a list of strings, in which case they are interepreted as column names. Default: c()
search_type	If "lasso", then the lasso algorithm (see glmnet) is used to identify the factors that provide the best linear fit to the target sequence. If "full", then a greedy algorithm is used to search for factors to be used in the hedge. At each step, all possible additions to the portfolio are considered, and the best one is chosen for inclusion. If "limited", then at each iteration, a preliminary screening step is performed to identify the securities with the highest correlations to the residuals of the currently selected portfolio. The top securities from this list are then checked for whether they would improve the portfolio, and the best one included.
pci_opt_method	Specifies the method that will be used for finding the best fitting model. One of the following: • "jp" The joint-penalty method (see fit.pci) • "twostep" The two-step method (see fit.pci) Default: jp
...	Other parameters to be passed onto the search function. See the source code.

Details

The hedge is constructed by searching for column indices i1,i2, ..., iN from among the columns of X which yield the best fit to the partially cointegrated fit:

$$Y_t = \alpha + \beta_1 * X_{t,i1} + beta_2 * X_{t,i2} + ... + beta_N * X_{t,iN} + M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

$$\epsilon_{M,t} \sim N(0, \sigma_M^2)$$

$$\epsilon_{R,t} \sim N(0, \sigma_R^2)$$

if search_type="lasso" is specified, then the lasso algorithm (see glmnet) is used to search for the factors that give the best linear fit to the target sequence Y. Having determined the list of factors, the cutoff point is determined based successive improvements to the likelihood score of the fitted model.

Otherwise, a greedy algorithm (search_type="full") or a modified greedy algorithm (search_type="limited") is used. This proceeds by searching through all columns of X (except those listed in exclude.cols) to find the column that gives the best fit to Y, as determined by the likelihood score of the partially cointegrated model. This column becomes the initial hedging portfolio. Having selected columns `i1, i2, ..., iK`, the next column is found by searching through all remaining columns of X (except those listed in `exclude.cols`) for the column which gives the best improvement to the partially cointegrated fit. However, if the best improvement is less than `minimum.stepsize`, or if `maxfact` columns have already been added, then the search terminates.

In the case of the modified greedy algorithm (`search_type="limited"`), a preprocessing step is used at the beginning of each iteration. In this preprocessing step, the correlation is computed between each unused column of X and the residual series of the currently computed best fit. The top B choices are then considered for inclusion in the portfolio, where B is a branching factor. The branching factor can be controlled by setting the value of the optional parameter `max.branch`. Its default value is 10.

The `lasso` algorithm is by far the fastest, followed by the `limited` greedy search. So, the best strategy is probably to start by using the `lasso`. If it fails to produce acceptable results, then move on to the `limited` greedy algorithm and finally the `full` search.

Value

Returns an S3 object of class `pci.hedge` containing the following fields

`pci`	The best partially cointegrated fit that was found
`indexes`	The indexes of the columns from X that were selected
`index_names`	The names of the columns from X that were selected

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

`fit.pci` Fitting of partially cointegrated models

`partialAR` Partially autoregressive models

`egcm` Engle-Granger cointegration model

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## Not run: YX <- rpci(n=1000, alpha=1, beta=c(2,3,4,5,6),
  sigma_C=c(0.1,0.1,0.1,0.1,0.1), rho=0.9, sigma_M=1, sigma_R=1)
YXC <- cbind(YX, matrix(rnorm(5000), ncol=5))
hedge.pci(YX[,1], YX[,2:ncol(YX)])
hedge.pci(YXC[,1], YXC[,2:ncol(YXC)])
## End(Not run)
```

`likelihood_ratio.pci` *Computes the likelihood ratio of the partially cointegrated model vs the null model*

Description

Computes the likelihood ratio of the partially cointegrated model vs the null model

Usage

```
likelihood_ratio.pci(Y, X,
  robust = FALSE,
  null_model = c("rw", "ar1"),
  pci_opt_method = c("jp", "twostep"),
  nu = 5)
```

Arguments

`Y`	The time series that is to be modeled. A plain or `zoo` vector of length `n`.
`X`	A (possibly `zoo`) matrix of dimensions `n` x `k`. If `k=1`, then this may be a plain or `zoo` vector.
`robust`	If `TRUE`, then the residuals are assumed to follow a t-distribution with `nu` degrees of freedom. Default: `FALSE`.
`null_model`	This specifies the model that is assumed under the null hypothesis. • rwRandom walk. Assumes sigma_M = rho = 0. Default. • ar1Autoregressive of order one. Assumes sigma_R=0.
`pci_opt_method`	Method to be used for fitting Y to X. • jpThe coefficients of Y are jointly optimized with the parameters of the AAR fit of the residuals. Default. • twostepA modified Engle-Granger procedure is used, where the coefficients of Y are first estimated, and then an AAR model is fit to the residuals.
`nu`	If `robust` is `TRUE`, then this is the degrees of freedom parameter used in fitting the t-distribution. Default: 5.

Details

First searches for the optimal fit under the null model, and computes the log of the likelihood score of this fit. Then, searches for the optimal fit under the full model, and computes the log of the likelihood score of this fit. Returns the difference of the two likelihood scores. Since the null model is nested in the full model, the log likelihood ratio score is guaranteed to be negative.

Value

The log of the ratio of the likelihoods of the two models.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

References

Clegg, Matthew, 2015. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

See Also

`fit.pci` Fitting partially cointegrated models

Examples

```
YX <- rpci(n=1000, alpha=1, beta=c(2,3,4), sigma_C=c(1,1,1), rho=0.9, sigma_M=0.1, sigma_R=0.2)
likelihood_ratio.pci(YX[,1], YX[,2:ncol(YX)])
```

---

loglik.pci — *Computes the log likelihood of a partially cointegrated model*

Description

Computes the log likelihood of a partially cointegrated model

Usage

```
loglik.pci(Y, X, alpha, beta, rho, sigma_M, sigma_R,
  M0 = 0, R0 = 0,
  calc_method = c("css", "kfas", "ss", "sst", "csst"),
  nu = pci.nu.default())
```

Arguments

`Y`	The time series that is to be modeled. A plain or `zoo` vector of length `n`.
`X`	A (possibly `zoo`) matrix of dimensions `n` x `k`. If `k=1`, then this may be a plain or `zoo` vector.
`alpha`	The constant term to be used in the fit.
`beta`	A vector of length `k` representing the weightings to be given to the components of `X`.
`rho`	The coefficient of mean reversion.
`sigma_M`	The standard deviation of the innovations of the mean-reverting component of the model.

sigma_R	The standard deviation of the innovations of the random walk component of the model.
M0	The initial value of the mean-reverting component. Default = 0.
R0	The initial value of the random walk component. Default = 0.
calc_method	Specifies the Kalman filter implementation that will be used for computing the likelihood score: • "ss" Steady-state Kalman filter • "css" C++ implementation of steady-state Kalman filter • "kfas" Kalman filter implementation of the KFAS package • "sst" Steady-state Kalman filter using t-distributed innovations • "csst" C++ implementation of steady-state Kalman filter using t-distributed innovations Default: `css`
nu	The degrees-of-freedom parameter to be used if `calc_method` is "sst" or "csst".

Details

The partial cointegration model is given by the equations:

$$Y_t = \alpha + \beta_1 * X_{t,1} + beta_2 * X_{t,2} + ... + beta_k * X_{t,k} + M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

$$\epsilon_{M,t} \sim N(0, \sigma_M^2)$$

$$\epsilon_{R,t} \sim N(0, \sigma_R^2)$$

Given the input series `Y` and `X`, and given the parameter values `alpha`, `beta`, `rho`, `M0` and `R0`, the innovations `epsilon_M[t]` and `epsilon_R[t]` are calculated using a Kalman filter. Based upon these values, the log-likelihood score is then computed and returned.

Value

The log of the likelihood score of the Kalman filter

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

References

Clegg, Matthew, 2015. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

See Also

`egcm` Engle-Granger cointegration model

`partialAR` Partially autoregressive models

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

    set.seed(1)
   YX <- rpci(n=500, alpha=1, beta=c(2,3,4), sigma_C=c(1,1,1), rho=0.9, sigma_M=0.1, sigma_R=0.2)
   loglik.pci(YX[,1], YX[,2:ncol(YX)], alpha=1, beta=c(2,3,4), rho=0.9, sigma_M=0.1, sigma_R=0.2)
```

---

multigetYahooPrices   *Fetches closing prices of multiple stock tickers*

Description

Fetches a zoo `data.frame` of daily closing prices of multiple stock tickers.

Usage

```
multigetYahooPrices(components, start, end, quiet = FALSE, adjust = TRUE)
```

Arguments

`components`	Character vector of Yahoo ticker symbols
`start`	First date of desired data in YYYYMMDD format. Default is earliest date of all series
`end`	Last date of desired data in YYYYMMDD format. Default is the last date for which data is available
`quiet`	If `FALSE`, then information is printed about the progress of the fetch operation
`adjust`	If `TRUE`, then adjusted closing prices are returned. Otherwise, unadjusted prices are returned.

Value

Returns a `zoo data.frame` containing the closing prices of the series listed in the `components` parameter, one column per price series.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

getYahooData

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

### Note: you must have a working internet
### connection for these examples to work!
spy.voo <- multigetYahooPrices(c("SPY","VOO"))

## End(Not run)
```

---

rpci *Generates a random instance of a partial cointegration model*

Description

Generates a random instance of a partial cointegration model

Usage

```
rpci(n, alpha, beta, sigma_C, rho, sigma_M, sigma_R,
  include.state = FALSE, robust = FALSE, nu = 5)
```

Arguments

n	Number of observations to generate
alpha	Constant term of the model
beta	A vector of factor loadings
sigma_C	A vector of standard deviations
rho	The coefficient of mean reversion
sigma_R	The standard deviation of the innovations of the random walk portion of the residual series
sigma_M	The standard deviation of the innovations of the mean-reverting portion of the residual series
include.state	If TRUE, then the output data.frame contains the innovations to the factors and residual series, as well as the state of the residual series. Default: FALSE
robust	If TRUE, then a t-distribution is used to generate the innovations. Otherwise, the innovations are normally distributed. Default: FALSE.
nu	The degrees of freedom parameter used for t-distributed innovations. Default: 5.

Details

Generates a random set of partially cointegrated vectors. On input, `n` is the length of the sequence to be generated. `beta` is a vector of length k representing the coefficients of the factor loadings, and `sigma_C` is a vector of length k representing the standard deviations of the increments of the factor loadings.

Generates a random realization of the sequence

$$Y_t = \alpha + \beta_1 F_{1,t} + \beta_2 F_{2,t} + ... + \beta_k F_{k,t} + M_t + R_t$$

$$F_{i,j} = F_{i,j-1} + \delta_{i,j}$$

$$M_t = \rho m_{t-1} + \epsilon_{M,t}$$

$$R_t = r_{t-1} + \epsilon_{R,t}$$

$$\delta_{i,j}\ N(0, \sigma^2_{C,i})$$

$$\epsilon_{M,t}\ N(0, \sigma^2_M)$$

$$\epsilon_{R,t}\ N(0, \sigma^2_R)$$

Value

A `data.frame` of `n` rows representing the realization of the partially cointegrated sequence.

If `include.state` is `FALSE`, returns an `n x (k+1)` matrix whose columns are `y, F_1, F_2, ..., F_k`.
If include.state is `TRUE`, returns an `n x (2k + 6)` matrix whose columns are `y, F_1, F_2, ..., F_k, x, M, R, delta_1,`

Author(s)

Matthew Clegg <`matthewcleggphd@gmail.com`>

Christopher Krauss <`christopher.krauss@fau.de`>

Jonas Rende <`jonas.rende@fau.de`>

See Also

`fit.pci`

Examples

```
rpci(10, alpha =1, beta=1, sigma_C=1, rho=0.9, sigma_R=1, sigma_M=1)
```

statehistory.pci *Generates the sequence of inferred states of a partial cointegration model*

Description

Generates the sequence of inferred states of a partial cointegration model

Usage

```
statehistory.pci(A, data = A$data, basis = A$basis)
```

Arguments

A	An object returned by fit.pci representing a partial cointegration fit.
data	The data history for which the inferred states are to be computed. This should be a (k+1) x n matrix, where k is the number of independent variables and n is the number of observations. If this is omitted, then uses the data history that was used in fitting the model A.
basis	The coefficients of the independent variables. This is a vector of length k. If this is omitted, then uses the coefficients that were computed in fitting the model A.

Details

Computes the expected internal states of the model over the course of the data history.

Value

Returns a data.frame with the following columns:

Y	The variable being modeled
X1,...,X_N	The independent variables
Z	The residual series Y - beta %*% X
M	The estimated state of the mean reverting component
R	The estimated state of the random walk component
eps_M	The innovation to the mean reverting component
eps_R	The innovation to the random walk component

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

egcm Engle-Granger cointegration model

partialAR Partially autoregressive models

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

YX <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.9, sigma_M=1, sigma_R=2)
f <- fit.pci(YX[,1], YX[,2:ncol(YX)])
statehistory.pci(f)
```

---

test.pci — *Tests the goodness of fit of a partial cointegration model*

Description

Tests the goodness of fit of a partial cointegration model

Usage

```
test.pci(Y, X, alpha = 0.05,
  null_hyp = c("rw", "ar1"),
  robust = FALSE,
  pci_opt_method = c("jp", "twostep"))
```

Arguments

`Y`	The time series that is to be modeled. A plain or zoo vector of length n.
`X`	A (possibly zoo) matrix of dimensions n x k. If k=1, then this may be a plain or zoo vector.
`alpha`	The cutoff value to be used for determining whether or not to accept the null hypothesis. If the the p-value computed through the likelihood ratio test is below this value, then the null hypothesis is rejected. Default: 0.05.
`null_hyp`	This specifies the null hypothesis. This can be either "rw", "ar1" or c("rw","ar1"). If "rw", then the null hypothesis is a random walk. If "ar1", then the null hypothesis is an autoregressive process of order 1. (In this case, the null hypothesis calls for Y and X to be cointegrated.) If (c("rw","ar1"), then the null hypothesis is either a random walk or AR(1) process. Default: both.
`robust`	If TRUE, then the residuals are assumed to follow a t-distribution. Default: FALSE.
`pci_opt_method`	The method that will be used for fitting a partially cointegrated model to X and Y. This can be either "jp" (joint penalty) or "twostep" (Engle-Granger two-step). See fit.pci for a complete explanation. Default: "jp".

Details

The likelihood ratio test is used to determine whether the null hypothesis should be accepted or whether the alternative of partial cointegration should be accepted. That is to say, a search is performed for the best fitting model under the null hypothesis, and the log likelihood score of this model is computed. Then a search is performed for the best fitting model under the alternative hypothesis of partial cointegration, and the log likelihood score of this model is computed. The associated p-values have been computed through simulation.

If the null hypothesis is the union `c("rw", "ar1")`, then a table of p-values is used that has been pre-computed, however this table is not unique.

Value

An object of class `"pcitest"` containing the results of the hypothesis test.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

References

Clegg, Matthew, 2015. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

Clegg, Matthew and Krauss, Christopher, 2016. Pairs trading with partial cointegration. FAU Discussion Papers in Economics, University of Erlangen-Nuernberg. *Available at https://www.iwf.rw.fau.de/files/2016/05/05-2016.pdf*

See Also

`fit.pci` Fits a partially cointegrated model

`likelihood_ratio.pci` Computes the likelihood ratio of a PCI model versus a null model

Examples

```
# The following should reject both the random walk and AR(1) models

## Not run:
YX <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=1, sigma_R=1)
test.pci(YX[,1], YX[,2:ncol(YX)])

# The following should accept the random walk model and reject the AR(1) model
YX.rw <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=0, sigma_R=1)
test.pci(YX.rw[,1], YX.rw[,2:ncol(YX.rw)])

# The following should reject the random walk model and accept the AR(1) model
YX.mr <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=1, sigma_R=0)
test.pci(YX.mr[,1], YX.mr[,2:ncol(YX.mr)])
```

```
## End(Not run)
```

---

which.hypothesis.pcitest
Returns the preferred hypothesis when testing for partial cointegration

---

Description

Returns the preferred hypothesis when testing for partial cointegration

Usage

```
which.hypothesis.pcitest(AT)
```

Arguments

AT	An object of class "pcitest" that has been returned by a previous call to test.pci.

Details

Based upon the critical value alpha that was given in the call to test.pci and the p-value that was computed, determines which hypothesis best fits the data.

Value

If a non-robust fit was used, then one of the following values is returned:

"PCI"	Partially cointegrated. Both the random walk hypothesis and the AR(1) hypothesis were rejected.
"RW"	Random walk.
"AR1"	Autoregressive of order one.

If a robust fit was used, then one of the following values is returned:

"RPCI"	Partially cointegrated. Both the random walk hypothesis and the AR(1) hypothesis were rejected.
"RRW"	Random walk.
"RAR1"	Autoregressive of order one.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

`fit.pci` Fits a partially cointegrated model

`likelihood_ratio.pci` Computes the likelihood ratio of a PCI model versus a null model

Examples

```
## Not run:
# The following should usually return "PCI"
YX <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=1, sigma_R=1)
which.hypothesis.pcitest(test.pci(YX[,1], YX[,2:ncol(YX)]))

# The following should usally return "RW"
YX.rw <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=0, sigma_R=1)
which.hypothesis.pcitest(test.pci(YX.rw[,1], YX.rw[,2:ncol(YX.rw)]))

# The following should usually return "AR1"
YX.mr <- rpci(n=1000, alpha=1, beta=c(2,3), sigma_C=c(0.1,0.1), rho=0.8, sigma_M=1, sigma_R=0)
which.hypothesis.pcitest(test.pci(YX.mr[,1], YX.mr[,2:ncol(YX.mr)]))

## End(Not run)
```

`yfit.pci` *Fetch series from Yahoo and perform a partial cointegration fit.*

Description

Fetch series from Yahoo and perform a partial cointegration fit.

Usage

```
yfit.pci(target, factors, start, end, na.rm=FALSE, ...)
```

Arguments

`target`	The ticker symbol of the stock price series that is to be modeled.
`factors`	A list of ticker symbols of stock price series to be used in modeling `target`
`start`	The starting date for which data is to be fetched, given in the format YYYYMMDD. Default: 2 years ago today.
`end`	The ending date for which data is to be fetched, given in the format YYYYMMDD. Default: today.
`na.rm`	If TRUE, then `NA`'s will be removed from the `data.frame` of fetched prices. A heuristic approach is used to decide between deleting securities versus deleting days.
`...`	Additional optional parameters to be passed to `fit.pci`

Value

An S3 object of class `pci.fit` representing the best fit that was found.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

`fit.pci`

Examples

```
# Compare a cointegration fit Coca-Cola and Pepsi to a partial cointegration fit.
# Note that yegcm(X, Y) has a different parameter ordering than yfit.pci(Y, X)
# yegcm("PEP", "KO", start=as.numeric(format(Sys.Date() - 365*2, "%Y%m%d")))
# yfit.pci("KO", "PEP")
```

`yhedge.pci` *Hedge portfolio for a stock price series*

Description

Computes the hedge of a stock price series fetched form Yahoo! using one or more other stock price series also fetched form Yahoo!

Usage

```
yhedge.pci(target, factors, start, end, na.rm=FALSE, ...)
```

Arguments

`target`	The ticker symbol of the stock price series that is to be modeled.
`factors`	A list of ticker symbols of stock price series to be used in modeling `target`
`start`	The starting date for which data is to be fetched, given in the format YYYYMMDD. Default: 2 years ago today.
`end`	The ending date for which data is to be fetched, given in the format YYYYMMDD. Default: today.
`na.rm`	If TRUE, then `NA`'s will be removed from the `data.frame` of fetched prices. A heuristic approach is used to decide between deleting securities versus deleting days.
`...`	Additional optional parameters to be passed to `fit.pci`

Value

An S3 object of class pci.hedge representing the best fit that was found.

Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

Christopher Krauss <christopher.krauss@fau.de>

Jonas Rende <jonas.rende@fau.de>

See Also

fit.pci

Examples

```
# Compute the best hedge of Coca-Cola using sector ETFS.
# sectorETFS <- c("XLB","XLE","XLF","XLI","XLK","XLP","XLU","XLV","XLY")
# hedge <- yhedge.pci("KO", sectorETFS)
# hedge
# test.pci(hedge$pci)
# plot(hedge)
```

Index