

Package ‘cointmonitoR’

July 22, 2025

Type Package

Title Consistent Monitoring of Stationarity and Cointegrating Relationships

Date 2016-06-14

Version 0.1.0

Description We propose a consistent monitoring procedure to detect a structural change from a cointegrating relationship to a spurious relationship. The procedure is based on residuals from modified least squares estimation, using either Fully Modified, Dynamic or Integrated Modified OLS. It is inspired by Chu et al. (1996) <DOI:10.2307/2171955> in that it is based on parameter estimation on a pre-break ``calibration" period only, rather than being based on sequential estimation over the full sample. See the discussion paper <DOI:10.2139/ssrn.2624657> for further information. This package provides the monitoring procedures for both the cointegration and the stationarity case (while the latter is just a special case of the former one) as well as printing and plotting methods for a clear presentation of the results.

URL <https://github.com/aschersleben/cointmonitoR>

BugReports <https://github.com/aschersleben/cointmonitoR/issues>

License GPL-3

Depends cointReg (>= 0.2.0)

Imports stats, graphics, matrixStats (>= 0.14.1)

RoxygenNote 5.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Philipp Aschersleben [aut, cre],
Martin Wagner [aut] (Author of underlying paper.),
Dominik Wied [aut] (Author of underlying paper.)

Maintainer Philipp Aschersleben <aschersleben@statistik.tu-dortmund.de>

Repository CRAN

Date/Publication 2016-06-14 20:34:46

Contents

cointmonitoR-package	2
monitorCointegration	3
monitorStationarity	5
plot.cointmonitoR	8
print.cointmonitoR	10
Index	12

cointmonitoR-package *The cointmonitoR package*

Description

Consistent Monitoring of Stationarity and Cointegrating Relationships

Details

See the vignette:

`vignette("cointmonitoR")`

See the DESCRIPTION:

`help(package = cointmonitoR)`

See the README:

<https://github.com/aschersleben/cointmonitoR/blob/master/README.md>

Open the package documentation page:

`package?cointmonitoR`

Further information and bug reporting:

<https://github.com/aschersleben/cointmonitoR>

Functions

- `monitorCointegration`
This procedure is able to monitor a cointegration model for level or trend cointegration and returns the corresponding break point, if available. It is based on parameter estimation on a pre-break "calibration" period at the beginning of the sample that is known or assumed to be free of structural change.
- `monitorStationarity`
This procedure is a special case of `monitorCointegration`, since it's able to monitor a one-dimensional vector for level or trend stationarity.
- `print`
Print clear results.
- `plot`
Plot the test statistics and the values/residuals of a cointmonitoR model.

Dependencies

This package mainly depends on our `cointReg` package.

monitorCointegration *Procedure for Monitoring Level and Trend Cointegration*

Description

This procedure is able to monitor a cointegration model for level or trend cointegration and returns the corresponding break point, if available. It is based on parameter estimation on a pre-break "calibration" period at the beginning of the sample that is known or assumed to be free of structural change and can be specified exactly via the `m` argument (see Details for further information).

Usage

```
monitorCointegration(x, y, m = 0.25, model = c("FM", "D", "IM"),
  trend = FALSE, kernel = c("ba", "pa", "qs", "tr"), bandwidth = c("and",
    "nw"), D.options = NULL, signif.level = 0.05, return.stats = TRUE,
  return.input = TRUE, check = TRUE, ...)
```

Arguments

<code>x</code>	[numeric matrix data.frame] Data on which to apply the monitoring procedure (RHS).
<code>y</code>	[numeric matrix data.frame] Data on which to apply the monitoring procedure (LHS). Has to be one-dimensional. If matrix, it may have only one row or column, if data.frame just one column.
<code>m</code>	[numeric(1)] Length of calibration period as fraction of the data's length (between 0.1 and 0.9) or as number of observations (see Details).
<code>model</code>	[character(1)] The model to be used for modified OLS calculations. Should be one of FM-OLS ("FM"), D-OLS ("D") or IM-OLS ("IM").
<code>trend</code>	[logical] Should an intercept and a linear trend be included? If FALSE (default), only an intercept is included.
<code>kernel</code>	[character(1)] The kernel function to use for calculating the long-run variance. Default is Bartlett kernel ("ba"), see Details for alternatives.
<code>bandwidth</code>	[character(1) numeric(1)] The bandwidth to use for calculating the long-run variance. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw"). You can also set the bandwidth manually.

D.options	[list NULL] Options for the D-OLS calculations. A list with elements <code>n.lead</code> , <code>n.lag</code> , <code>kmax</code> and <code>info.crit</code> – or NULL (then default arguments are the same as in cointRegD . See that help page for further information.) Missing list elements will be replaced automatically.
signif.level	[numeric(1)] Level of significance (between 0.01 and 0.1). Detection time will be calculated only if the estimated p-value is smaller than <code>signif.level</code> . Default is 0.05.
return.stats	[logical] Whether to return all test statistics. Default is TRUE.
return.input	[logical] Whether to return the input data, default is TRUE.
check	[logical] Whether to check (and if necessary convert) the arguments. See checkVars for further information.
...	Arguments passed to getBandwidthNW (<code>inter</code> , <code>weights</code>), if <code>bandwidth = "nw"</code> .

Details

The calibration period can be set by setting the argument `m` to the number of the last observation, that should be inside this period. The corresponding fraction of the data's length will be calculated automatically. Alternatively you can set `m` directly to the fitting fraction value, but you should pay attention to the fact, that the calibration period may become smaller than intended: The last observation is calculated as $\text{floor}(m * N)$ (with N the length of x).

The kernel that is used for calculating the long-run variance can be one of the following:

- "ba": Bartlett kernel
- "pa": Parzen kernel
- "qs": Quadratic Spectral kernel
- "tr": Truncated kernel

Value

[cointmonitoR] object with components:

`Hsm` [numeric(1)] value of the test statistic

`time` [numeric(1)] detected time of structural break

`p.value` [numeric(1)] estimated p-value of the test (between 0.01 and 0.1)

`cv` [numeric(1)] critical value of the test

`sig` [numeric(1)] significance level used for the test

`residuals` [numeric] residuals of the modified OLS model to be used for calculating the test statistics

`model` [character(1)] cointOLS model ("FM", "D", or "IM")

`trend` [character(1)] trend model ("level" or "trend")

name [character(1)] name(s) of data
 m [list(2)] list with components:
 \$m.frac [numeric(1)]: calibration period (fraction)
 \$m.index [numeric(1)]: calibration period (length)
 kernel [character(1)] kernel function
 bandwidth [list(2)] \$name [character(1)]: bandwidth function (name)
 \$number [numeric(1)]: bandwidth
 statistics [numeric] values of test statistics with the same length as data, but NA during calibration period (available if return.stats = TRUE)
 input [numeric | matrix | data.frame] copy of input data (available if return.stats = TRUE)
 D.options [list] information about further parameters (available if model = "D")

References

- Wagner, M. and D. Wied (2015): "Monitoring Stationarity and Cointegration," *Discussion Paper*, DOI:10.2139/ssrn.2624657.

See Also

Other cointmonitoR: [monitorStationarity](#), [plot.cointmonitoR](#), [print.cointmonitoR](#)

Examples

```

set.seed(42)
x = data.frame(x1 = cumsum(rnorm(200)), x2 = cumsum(rnorm(200)))
eps1 = rnorm(200, sd = 2)
eps2 = c(eps1[1:100], cumsum(eps1[101:200]))

y = x$x1 - x$x2 + 10 + eps1
monitorCointegration(x = x, y = y, m = 0.5, model = "FM")

y2 = y + seq(1, 30, length = 200)
monitorCointegration(x = x, y = y2, m = 0.5, model = "FM")
monitorCointegration(x = x, y = y2, m = 0.5, trend = TRUE, model = "FM")

y3 = x$x1 - x$x2 + 10 + eps2
monitorCointegration(x = x, y = y3, m = 0.5, model = "FM")
monitorCointegration(x = x, y = y3, m = 0.5, model = "D")
monitorCointegration(x = x, y = y3, m = 0.5, model = "IM")

```

Description

This procedure is able to monitor a one-dimensional vector for level or trend stationarity and returns the corresponding break point, if available. It is based on parameter estimation on a pre-break "calibration" period at the beginning of the sample that is known or assumed to be free of structural change and can be specified exactly via the `m` argument (see Details for further information).

Usage

```
monitorStationarity(x, m = 0.25, trend = FALSE, kernel = c("ba", "pa",
  "qs", "tr"), bandwidth = c("and", "nw"), signif.level = 0.05,
  return.stats = TRUE, return.input = TRUE, check = TRUE, ...)
```

Arguments

<code>x</code>	[numeric matrix data.frame] Data on which to apply the monitoring procedure. If <code>matrix</code> , it may have only one row or column, if <code>data.frame</code> just one column.
<code>m</code>	[numeric(1)] Length of calibration period as fraction of the data's length (between 0.1 and 0.9) or as number of observations (see Details).
<code>trend</code>	[logical] Should an intercept and a linear trend be included? If <code>FALSE</code> (default), only an intercept is included.
<code>kernel</code>	[character(1)] The kernel function to use for calculating the long-run variance. Default is Bartlett kernel ("ba"), see Details for alternatives.
<code>bandwidth</code>	[character(1) numeric(1)] The bandwidth to use for calculating the long-run variance. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw"). You can also set the bandwidth manually.
<code>signif.level</code>	[numeric(1)] Level of significance (between 0.01 and 0.1). Detection time will be calculated only if the estimated p-value is smaller than <code>signif.level</code> . Default is 0.05.
<code>return.stats</code>	[logical] Whether to return all test statistics. Default is <code>TRUE</code> .
<code>return.input</code>	[logical] Whether to return the input data, default is <code>TRUE</code> .
<code>check</code>	[logical] Whether to check (and if necessary convert) the arguments. See checkVars for further information.
<code>...</code>	Arguments passed to getBandwidthNW (<code>inter</code> , <code>weights</code>), if <code>bandwidth = "nw"</code> .

Details

The calibration period can be specified by setting the argument `m` to the number of its last observation. The corresponding fraction of the data's length will be calculated automatically. Alternatively

you can set `m` directly to the fitting fraction value. Attention: The calibration period may become smaller than intended: The last observation is calculated as `floor(m * N)` (with `N` = length of `x`).

The kernel that is used for calculating the long-run variance can be one of the following:

- "ba": Bartlett kernel
- "pa": Parzen kernel
- "qs": Quadratic Spectral kernel
- "tr": Truncated kernel

Value

[cointmonitoR] object with components:

`Hsm` [numeric(1)] value of the test statistic

`time` [numeric(1)] detected time of structural break

`p.value` [numeric(1)] estimated p-value of the test (between 0.01 and 0.1)

`cv` [numeric(1)] critical value of the test

`sig` [numeric(1)] significance level used for the test

`trend` [character(1)] trend model ("level" or "trend")

`name` [character(1)] name(s) of data

`m` [list(2)] list with components:

`$m.frac` [numeric(1)]: calibration period (fraction)

`$m.index` [numeric(1)]: calibration period (length)

`kernel` [character(1)] kernel function

`bandwidth` [list(2)] `$name` [character(1)]: bandwidth function (name)

`$number` [numeric(1)]: bandwidth

`statistics` [numeric] values of test statistics with the same length as data, but NA during calibration period (available if `return.stats = TRUE`)

`input` [numeric | matrix | data.frame] copy of input data (available if `return.stats = TRUE`)

References

- Wagner, M. and D. Wied (2015): "Monitoring Stationarity and Cointegration," *Discussion Paper*, DOI:10.2139/ssrn.2624657.

See Also

Other cointmonitoR: [monitorCointegration](#), [plot.cointmonitoR](#), [print.cointmonitoR](#)

Examples

```
set.seed(1909)
x <- rnorm(200)
x2 <- c(x[1:100], cumsum(x[101:200]) / 2)

# Specify the calibration period
# as fraction of the total length of x:
monitorStationarity(x, m = 0.25)
monitorStationarity(x2, m = 0.465)

# Specify the calibration period
# by setting its last observation exactly:
monitorStationarity(x, m = 50)
monitorStationarity(x2, m = 93)
```

plot.cointmonitoR	<i>Plot Method for Monitoring Procedures.</i>
-------------------	---

Description

Plotting objects of class "cointmonitoR".

Usage

```
## S3 method for class 'cointmonitoR'
plot(x, what = "test", type, main, xlab, ylab,
     axes = TRUE, legend = TRUE, main.val, xlab.val, ylab.val, lines = TRUE,
     ...)
```

Arguments

x	[cointmonitoR] Object of class "cointmonitoR", i.e. the result of monitorStationarity or monitorCointegration .
what	[character] Whether to plot test statistics ("test") (default) or the values/residuals of the tested time series ("values" or "residuals") or "both". Works only, if return.stats = TRUE in the called function that to get x (default setting).
type	[character] Plot type (from plot). Default is "1".
main, xlab, ylab	[character] Title and axis titles (from plot). Default values will be generated from the contents of x.
axes, legend	[logical] Whether to add axes (from plot) and a legend to the plot.

main.val, xlab.val, ylab.val	[character]
	Title and axis titles (from plot) for the second plot, if generating both plots in one step (see argument what). Default values will be generated from the contents of x.
lines	[logical]
	Whether to add lines and annotations to the plot. Default is TRUE.
...	[any]
	Further arguments passed to plot .

See Also

Other cointmonitor: [monitorCointegration](#), [monitorStationarity](#), [print.cointmonitor](#)

Examples

```
### Monitoring stationarity (no break):
set.seed(1909)
x = rnorm(200)
test = monitorStationarity(x, m = 0.5)
plot(test)

oldpar = par(mfrow = c(2, 1), mar = c(4, 4, 1, 1))
plot(test, what = "both", legend = FALSE, main = "", main.val = "")
par(oldpar)

### Monitoring stationarity (break):
x = c(x[1:100], cumsum(rnorm(100, sd = 0.5)) + x[101:200])
test2 = monitorStationarity(x, m = 0.5)
plot(test2)

oldpar = par(mfrow = c(2, 1), mar = c(4, 4, 1, 1))
plot(test2, what = "both", legend = FALSE, main = "", main.val = "")
par(oldpar)

### Monitoring cointegration (no break):
set.seed(42)
x = data.frame(x1 = cumsum(rnorm(200)), x2 = cumsum(rnorm(200)))
eps1 = rnorm(200, sd = 2)
y = x$x1 - x$x2 + 10 + eps1
test3 = monitorCointegration(x = x, y = y, m = 0.5, model = "FM")
plot(test3)

oldpar = par(mfrow = c(2, 1), mar = c(4, 4, 1, 1))
plot(test3, what = "both", legend = FALSE, main = "", main.val = "")
par(oldpar)

### Monitoring cointegration (break):
eps2 = c(eps1[1:100], cumsum(eps1[101:200]))
```

```

y = x$x1 - x$x2 + 10 + eps2
test4 = monitorCointegration(x = x, y = y, m = 0.5, model = "FM")
plot(test4)

oldpar = par(mfrow = c(2, 1), mar = c(4, 4, 1, 1))
plot(test4, what = "both", legend = FALSE, main = "", main.val = "")
par(oldpar)

```

print.cointmonitoR *Print Method for Monitoring Procedures.*

Description

Printing objects of class "cointmonitoR".

Usage

```

## S3 method for class 'cointmonitoR'
print(x, ..., digits = getOption("digits"))

```

Arguments

x	[cointmonitoR] Object of class "cointmonitoR", i.e. the result of monitorStationarity() or monitorCointegration().
...	ignored
digits	[numeric] Number of significant digits to be used.

Value

The invisible x object.

See Also

Other cointmonitoR: [monitorCointegration](#), [monitorStationarity](#), [plot.cointmonitoR](#)

Examples

```

set.seed(42)
test = monitorStationarity(rnorm(100), m = 0.5)
print(test)

x = data.frame(x1 = cumsum(rnorm(200)), x2 = cumsum(rnorm(200)))
eps1 = rnorm(200, sd = 2)
eps2 = c(eps1[1:100], cumsum(eps1[101:200]))
y1 = x$x1 - x$x2 + 10 + eps1

```

```
y2 = x$x1 - x$x2 + 10 + eps2
test1 = monitorCointegration(x = x, y = y1, m = 0.5, model = "FM")
print(test1)
test2 = monitorCointegration(x = x, y = y2, m = 0.5, model = "FM")
print(test2)
```

Index

checkVars, [4](#), [6](#)

cointmonitoR-package, [2](#)

cointRegD, [4](#)

getBandwidthNW, [4](#), [6](#)

monitorCointegration, [2](#), [3](#), [7–10](#)

monitorStationarity, [2](#), [5](#), [5](#), [8–10](#)

plot, [2](#), [8](#), [9](#)

plot.cointmonitoR, [5](#), [7](#), [8](#), [10](#)

print, [2](#)

print.cointmonitoR, [5](#), [7](#), [9](#), [10](#)