

# Package ‘detectR’

May 8, 2026

**Type** Package

**Title** Change Point Detection

**Version** 0.3.0

**Maintainer** Changryong Baek <crbaek@skku.edu>

**Description** Time series analysis of network connectivity. Detects and visualizes change points between networks.

Methods included in the package are discussed in depth in Baek, C., Gates, K. M., Leinwand, B., Pipiras, V. (2021)

``Two sample tests for high-dimensional auto-covariances" <doi:10.1016/j.csda.2020.107067> and Baek, C., Gampe, M., Leinwand B., Lindquist K., Hopfinger J. and Gates K. (2023)

“Detecting functional connectivity changes in fMRI data” <doi:10.1007/s11336-023-09908-7>.

**License** Unlimited

**Encoding** UTF-8

**LazyData** true

**Imports** signal, lavaan, doParallel, glasso, LogConcDEAD, foreach, parallel

**Depends** R (>= 2.10)

**URL** <https://github.com/crbaek/detectR>

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Changryong Baek [aut, cre],  
Matthew Gampe [aut],  
Kathleen M. Gates [aut],  
Seok-Oh Jeong [aut],  
Vladas Pipiras [aut]

**Repository** CRAN

**Date/Publication** 2024-03-08 20:30:02 UTC

## Contents

changesim . . . . .	2
detectBinary . . . . .	2
detectGlasso . . . . .	4
detectMaxChange . . . . .	5
detectSliding . . . . .	6
global . . . . .	7
preprocess . . . . .	8
testGlasso . . . . .	9
testMax . . . . .	10
testPCA . . . . .	11
testsim . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

changesim	<i>Changepoint Example Data</i>
-----------	---------------------------------

---

### Description

This dataset contains a simulated multivariate time series with two changepoints at time point 150 and 300. The dimension of the data is  $T=450$  and  $p=20$ .

### Usage

changesim

### Format

An object of class `matrix` (inherits from `array`) with 450 rows and 20 columns.

---

detectBinary	<i>Change point detection using PCA and binary segmentation</i>
--------------	---

---

### Description

This function uses PCA-based method to find breaks. Simultaneous breaks are found from binary segmentation.

**Usage**

```

detectBinary(
  Y,
  Del,
  L,
  q = "fixed",
  alpha = 0.05,
  nboot = 199,
  n.cl,
  bsize = "log",
  bootTF = TRUE,
  scaleTF = TRUE,
  diagTF = TRUE,
  plotTF = TRUE
)

```

**Arguments**

Y	data: Y = length*dim
Del	Delta away from the boundary restriction
L	the number of factors
q	methods in calculating long-run variance of the test statistic. Default is "fixed" = length^(1/3). Adaptive selection method is also available via "andrews", or user specify the length
alpha	significance level of the test
nboot	the number of bootstrap sample for p-value. Default is 199.
n.cl	number of cores in parallel computing. The default is (machine cores - 1)
bsize	block size for the Block Wild Bootstrapping. Default is log(length), "sqrt" uses sqrt(length), "adaptive" determines block size using data dependent selection of Andrews
bootTF	determine whether the threshold is calculated from bootstrap or asymptotic
scaleTF	scale the variance into 1
diagTF	include diagonal term of covariance matrix or not
plotTF	Draw plot to see test statistic and threshold

**Value**

**tstathist** The complete history of test statistics.

**Brhist** The sequence of breakpoints found from binary splitting

**L** The number of factors used in the procedure

**q** The estimated vectorized autocovariance on each regime.

**crit** The critical value to identify change point

**bsize** The block size of the bootstrap

**diagTF** If TRUE, the diagonal entry of covariance matrix is used in detecting connectivity changes.

**bootTF** If TRUE, bootstrap is used to find critical value

**scaleTF** If TRUE, the multivariate signal is studentized to have zero mean and unit variance.

### Examples

```
out3= detectBinary(changesim, L=2, n.cl=1)
```

---

detectGlasso	<i>Change point detection using Graphical lasso as in Cribben et al. (2012)</i>
--------------	---

---

### Description

This function implements the Dynamic Connectivity Regression (DCR) algorithm proposed by Cribben et al. (2012) to locate changepoints.

### Usage

```
detectGlasso(
  Y,
  Del,
  p,
  lambda = "bic",
  nboot = 100,
  n.cl,
  bound = c(0.001, 1),
  gridTF = FALSE,
  plotTF = TRUE
)
```

### Arguments

Y	Input data of dimension length*dim (T times d)
Del	Delta away from the boundary restriction
p	Gep(p) distribution controls the size of stationary bootstrap. The mean block length is 1/p
lambda	two selections possible for optimal parameter of lambda. "bic" finds lambda from bic criteria, or user can directly input the penalty value
nboot	the number of bootstrap sample for p-value. Default is 100.
n.cl	number of cores in parallel computing. The default is (machine cores - 1)
bound	bound of bic search in "bic" rule. Default is (.001, 1)
gridTF	minimum bic is found by grid search. Default is FALSE
plotTF	Draw plot to see test statistic

**Value**

A list with component

- br** The estimated breakpoints including boundary (0, T)
- brhist** The sequence of breakpoints found from binary splitting
- diffhist** The history of BIC reduction on each step
- W** The estimated vectorized autocovariance on each regime.
- WI** The estimated vectorized precision matrix on each regime.
- lambda** The penalty parameter estimated on each regime.
- pvalhist** The empirical p-values on each binary splitting.
- fitzero** Detailed output at first stage. Useful in producing plot.

**Examples**

```
out1= detectGlasso(changesim, p=.2, n.cl=1)
```

---

detectMaxChange	<i>Change point detection using max-type statistic as in Jeong et. al (2016)</i>
-----------------	--

---

**Description**

Change point detection using max-type statistic as in Jeong et. al (2016)

**Usage**

```
detectMaxChange(
  Y,
  m = c(30, 40, 50),
  margin = 30,
  thre.localfdr = 0.2,
  design.mat = NULL,
  plotTF = TRUE,
  n.cl
)
```

**Arguments**

Y	Input data matrix
m	window sizes
margin	margin
thre.localfdr	threshold for local fdr
design.mat	design matrix for analyzing task data
plotTF	Draw plot to see test statistic and threshold
n.cl	number of clusters for parallel computing

**Value**

**CLX** Test statistic corresponding to window size arranged in column

**CLXLocalFDR** The Local FDR calculated for each time point

**br** The final estimated break points

**Examples**

```
out2= detectMaxChange(changesim, m=c(30, 35, 40, 45, 50), n.cl=1)
```

---

detectSliding	<i>Change point detection using PCA and sliding method</i>
---------------	--

---

**Description**

Change point detection using PCA and sliding method

**Usage**

```
detectSliding(
  Y,
  wd = 40,
  L,
  Del,
  q = "fixed",
  alpha = 0.05,
  nboot = 199,
  n.cl,
  bsize = "log",
  bootTF = TRUE,
  scaleTF = TRUE,
  diagTF = TRUE,
  plotTF = TRUE
)
```

**Arguments**

Y	data: Y = length*dim
wd	window size for sliding averages
L	the number of factors
Del	Delta away from the boundary restriction
q	methods in calculating long-run variance of the test statistic. Default is "fixed" = length^(1/3) or "andrews" implements data adaptive selection, or user specify the length
alpha	significance level of the test

nboot	the number of bootstrap sample for p-value. Default is 199.
n.c1	number of cores in parallel computing. The default is (machine cores - 1)
bsize	block size for the Block Wild Bootstrapping. Default is log(length), "sqrt" uses sqrt(length), "adaptive" determines block size using data dependent selection of Andrews
bootTF	determine whether the threshold is calculated from bootstrap or asymptotic
scaleTF	scale the variance into 1
diagTF	include diagonal term of covariance matrix or not
plotTF	Draw plot to see test statistic and threshold

### Value

**sW** The test statistic

**L** The number of factors used in the procedure

**q** The estimated vectorized autocovariance on each regime.

**crit** The critical value to identify change point

**bsize** The block size of the bootstrap

**diagTF** If TRUE, the diagonal entry of covariance matrix is used in detecting connectivity changes.

**bootTF** If TRUE, bootstrap is used to find critical value

**scaleTF** If TRUE, the multivariate signal is studentized to have zero mean and unit variance.

### Examples

```
out4 = detectSliding(changesim, wd=40, L=2, n.c1=1)
```

---

global

*Global Variables and functions*

---

### Description

Defining variables and functions used in the internal functions

---

preprocess	<i>Data preparation for changepoint detection using functions in this package..</i>
------------	---

---

## Description

Id

## Usage

```
preprocess(file = NULL,
           header = NULL,
           sep = NULL,
           signal = NULL,
           noise = NULL,
           butterfreq = NULL,
           model = NULL)
```

## Arguments

file	a data matrix or file name with columns as variables and rows as observations across time.
header	logical for whether or not there is a header in the data file.
sep	The spacing of the data files. "" indicates space-delimited, "/t" indicates tab-delimited, "," indicates comma delimited. Only necessary to specify if reading data in from physical directory.
signal	(optional) a character vector containing the names of variables that contain signal i.e., which variables to use to detect change point. The default (NULL) indicates all variables except those in 'noise' argument are considered signal. Example: signal = c("dDMN4", "vDMN5", "vDMN1",
noise	(optional) a character vector containing the names of variables that contain noise. The signal variables will be regressed on these variables and residuals used in change point detection. The default (NULL) indicates there are no noise variables. Example: noise = c("White.Matter1", "CSF1")
butterfreq	(optional) bandpass filter frequency ranges. Example: c(.04,.4)
model	(optional) syntax indicating which variables belong to which networks for first pass of data reduction that is user-specified. If no header naming convention follows "V#". Notation should follow lavaan syntax style.

---

testGlasso	<i>Test for for the equality of connectivity based on the Graphical lasso estimation.</i>
------------	---

---

### Description

This function utilizes Dynamic Connectivity Regression (DCR) algorithm proposed by Cribben et al. (2012) to test the equality of connectivity in two fMRI signals.

### Usage

```
testGlasso(
  subY1,
  subY2,
  p,
  lambda = "bic",
  nboot = 100,
  n.c1,
  bound = c(0.001, 1),
  gridTF = FALSE
)
```

### Arguments

subY1	a sample of size length*dim
subY2	a sample of size length*dim
p	Gep(p) distribution controls the size of stationary bootstrap. The mean block length is 1/p
lambda	two selections possible for optimal parameter of lambda. "bic" finds lambda from bic criteria, or user can directly input the penalty value.
nboot	the number of bootstrap sample for p-value. Default is 100.
n.c1	number of cores in parallel computing. The default is (machine cores - 1)
bound	bound of bic search in "bic" rule. Default is (.001, 1)
gridTF	Utilize a grid search to optimize hyperparameters

### Value

**pval** The empirical p-value for testing the equality of connectivity structure  
**rho** The sequence of penalty parameter based on the combined sample, subY1 and subY2.  
**fit0** Output of glasso for combined sample  
**fit1** Output of glasso for subY1  
**fit2** Output of glasso for subY2

### Examples

```
test1= testGlasso(testsim$X, testsim$Y, n.c1=1)
```

---

testMax	<i>Max-type test for for the equality of connectivity</i>
---------	---

---

### Description

This function produces three test results based on max-type block bootstrap (BMB), long-run variance block bootstrapping with lagged-window estimator (LVBWR) and sum-type block bootstrap (BSUM). See Baek et al. (2019) for details.

### Usage

```
testMax(subY1, subY2, diagTF = TRUE, nboot, q = "fixed", n.c1)
```

### Arguments

subY1	a sample of size length*dim
subY2	a sample of size length*dim
diagTF	include diagonal term of covariance matrix or not
nboot	number of bootstrap sample, default is 2000
q	methods in calculating long-run variance of the test statistic. Default is "fixed" = length^(1/3) or "andrews" implements data adaptive selection, or user specify the length
n.c1	number of cores in parallel computing. The default is (machine cores - 1)

### Value

**tstat** Test statistic for testing the equality of connectivity structure

**pval** The p-value for testing the equality of connectivity structure

**q** The tuning parameter used in calculating long-run variance

### Examples

```
test2 = testMax(testsim$X, testsim$Y, n.c1=1)
```

---

testPCA	<i>PCA-based test for the equality of connectivity</i>
---------	--

---

**Description**

This function performs PCA-test for testing the equality of connectivity in two fMRI signals

**Usage**

```
testPCA(subY1, subY2, L = 2, nlag, diagTF = TRUE)
```

**Arguments**

subY1	a sample of size length*dim
subY2	a sample of size length*dim
L	the number of factors
nlag	is the number of ACF lag to be used in the test, default is 2, Default is nlag = floor(N^(1/3))
diagTF	include diagonal term of covariance matrix or not

**Value**

**tstat** Test statistic  
**pval** Returns the p-value  
**df** The degree of freedom in PCA-best test  
**L** The number of factors used in the test  
**diagTF** If true, the diagonal entry of covariance matrix is used in testing

**Examples**

```
test3 = testPCA(testsim$X, testsim$Y, L=2)
```

---

testsim	<i>Test Example Data</i>
---------	--------------------------

---

**Description**

This dataset contains a simulated multivariate time series with two different autocovariances. It is a list data with two variables X and Y. Each multivariate time series had dimension of T=150 and p=20

**Usage**

```
testsim
```

**Format**

An object of class `list` of length 2.

# Index

## \* datasets

changesim, [2](#)

testsim, [11](#)

changesim, [2](#)

detectBinary, [2](#)

detectGlasso, [4](#)

detectMaxChange, [5](#)

detectSliding, [6](#)

global, [7](#)

preprocess, [8](#)

testGlasso, [9](#)

testMax, [10](#)

testPCA, [11](#)

testsim, [11](#)