Package ‘kcpRS’

May 6, 2019

Type Package

Title Kernel Change Point Detection on the Running Statistics

Version 1.0.0

Description The running statistics of interest is first extracted using a time window which is slid across the time series, and in each window, the running statistics value is computed. KCP (Kernel Change Point) detection proposed by Arlot et al. (2012) <arXiv:1202.3878> is then implemented to flag the change points on the running statistics (Cabrieto et al., 2018, <doi:10.1016/j.ins.2018.03.010>). Change points are located by minimizing a variance criterion based on the pairwise similarities between running statistics which are computed via the Gaussian kernel. KCP can locate change points for a given k number of change points. To determine the optimal k, the KCP permutation test is first carried out by comparing the variance of the running statistics extracted from the original data to that of permuted data. If this test is significant, then there is sufficient evidence for at least one change point in the data. Model selection is then used to determine the optimal k>0.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.0)

Depends RColorBrewer, stats, utils, graphics, roll, foreach, doParallel

LinkingTo Rcpp

RoxygenNote 6.1.1

NeedsCompilation yes

Author Jedelyn Cabrieto [aut, cre],
Kristof Meers [aut],
Janne Adolf [ctb],
Peter Kuppens [ctb],
Francis Tuerlinckx [ctb],
Eva Ceulemans [ctb]

Maintainer Jedelyn Cabrieto <jed.cabrieto@kuleuven.be>

Repository CRAN

Date/Publication 2019-05-06 09:00:03 UTC
Description

Flagging change points on a user-specified running statistics through KCP (Kernel Change Point) detection. A KCP permutation test is first implemented to confirm whether there is at least one change point ($k>0$) in the running statistics. If this permutation test is significant, a model selection procedure is implemented to choose the most optimal number of change points.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>Type</th>
<th>Version</th>
<th>Date</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>kcpRS</td>
<td>Package</td>
<td>1.0.0</td>
<td>15-03-2019</td>
<td>GPL (&gt;=2)</td>
</tr>
</tbody>
</table>

This package contains the function `kcpRS` that can accept a user-defined function, `RS_fun`, which should derive the running statistics of interest. For examples, see `runMean`, `runVar`, `runAR` and `runCorr`. `kcpRS` performs a full change point analysis on the running statistics starting from locating the optimal change points given $k$, significance testing if $k>0$, and finally, determining the most optimal $k$. This function calls the function `kcpa` to find the most optimal change points given $k$ and then the `permTest` function to carry out the permutation test. The model selection step is embedded in the `kcpRS` function.

This package also contains the function `kcpRS_workflow` which carries out a stepwise change point analysis to flag changes in 4 basic time series statistics: mean, variance, autocorrelation (lag 1) and correlations.

Two illustrative data sets are included: MentalLoad and CO2Inhalation
Author(s)

Jedelyn Cabrieto (<jed.cabrieto@kuleuven.be>) and Kristof Meers

For the core KCP analysis, the authors built upon the codes from the Supplementary Material available in https://doi.org/10.1080/01621459.2013.849605 by Matteson and James (2012).

References


See Also

kcprs
kcprs_workflow
MentalLoad
CO2Inhalation

CO2Inhalation

CO2 Inhalation Data

Description

Nine physiological measures during a CO2-inhalation experiment.

Usage

data(CO2Inhalation)

Format

Dataframe with 239 rows and 10 columns. The first column indicates the experimental phase and the last nine columns correspond to the nine physiological measures tracked during the experiment: Breathing volume variables (ViVol, VeVol, Vent, PiaAB), breathing duration variables (Ti, Te, Tt), heart rate (HR) and RR interval (RR) or cardiac beat interval.
**References**


**Examples**

```r
data(CO2Inhalation)
```

---

**kcpa**

**KCP (Kernel Change Point) Detection**

**Description**

Finds the most optimal change point(s) in the running statistic time series `RunStat` by looking at their kernel-based pairwise similarities.

**Usage**

```r
kcpa(RunStat, Kmax = 10, wsize = 25)
```

**Arguments**

- `RunStat` : Dataframe of running statistics with rows corresponding to the windows and the columns corresponding to the variable(s) on which these running statistics were computed.
- `Kmax` : Maximum number of change points
- `wsize` : Window size

**Value**

- `kcpsoln` : A matrix comprised of the minimized variance criterion \( R_{min} \) and the optimal change point location(s) for each \( k \) from 1 to \( K_{max} \)

**References**


**kcpRS**

**KCP on the running statistics**

**Description**

Given a user-specified function `RS_fun` to compute the running statistics (see `runMean`, `runVar`, `runAR` and `runCorr`), a KCP permutation test (see `permTest`) is first implemented to test whether there is at least one significant change point, then through model selection most optimal number of change points is chosen.

**Usage**

```r
kcpRS(data, RS_fun, RS_name, wsize = 25, nperm = 1000, Kmax = 10,
       alpha = 0.05, varTest = FALSE, ncpu = 1)

## S3 method for class 'kcpRS'
plot(x, ...)

## S3 method for class 'kcpRS'
print(x, kcp_details = TRUE, ...)

## S3 method for class 'kcpRS'
summary(object, ...)
```

**Arguments**

- **data**
  - data `N x v` dataframe where `N` is the number of time points and `v` the number of variables.

- **RS_fun**
  - Running statistics function: Should require `wsize` and `wstep` as input and return a dataframe of running statistics as output. The rows of this dataframe should correspond to the windows and the columns should correspond to the variable(s) on which the running statistics were computed.

- **RS_name**
  - Name of the monitored running statistic.

- **wsize**
  - Window size

- **nperm**
  - Number of permutations used in the permutation test

- **Kmax**
  - Maximum number of change points desired

- **alpha**
  - Significance level of the permutation test

- **varTest**
  - If set to FALSE, only the variance DROP test is implemented, and if set to TRUE, both the variance test and the variance DROP tests are implemented.

- **ncpu**
  - Number of cpu cores to use

- **x**
  - An object of the type produced by `kcpRS`

- **...**
  - Further plotting arguments.

- **kcp_details**
  - If `TRUE`, then the matrix of optimal change points solutions given `k` is displayed. If `FALSE`, then this output is suppressed.

- **object**
  - An object of the type produced by `kcpRS_workflow`
value

RS_name Name indicated for the monitored running statistic
RS Dataframe of running statistics with rows corresponding to the time window and columns corresponding to the (combination of) variable(s) on which the running statistics were computed
wsize Selected window size
varTest Selected choice of implementation for varTest
nperm Selected number of permutations
alpha Selected significance level of the permutation test
subTest_alpha Significance level of each subtest. If varTest=0, subTest_alpha is equal to alpha since only the variance drop test is implemented. If varTest=1, subTest_alpha=alpha/2 since two subtests are carried out and Bonferroni correction is applied.
BestK Optimal number of change points
changePoints Change point location(s)
p_var_test P-value of the variance test
p_varDrop_test P-value of the variance drop test
CPs_given_K A matrix comprised of the minimized variance criterion \( R_{min} \) and the optimal change point location(s) for each \( k \) from 1 to \( K_{max} \)

References


Examples

```r
phase1=cbind(rnorm(50,0,1),rnorm(50,0,1)) #phase1: Means=0
phase2=cbind(rnorm(50,1,1),rnorm(50,1,1)) #phase2: Means=1
X=rbind(phase1,phase2)
res=kcpRS(data=X,RD_fun=runMean,RD_name="Mean",wsize=25,
nperm=1000,Kmax=10,alpha=.05,varTest=FALSE,ncpu=1)

summary(res)
plot(res)
```
**Description**

Any of the four basic running statistics (i.e., running means, running variances, running autocorrelations and running correlations) or a combination thereof can be scanned for change points.

**Usage**

```r
kcprs_workflow(data, RS_funs = c("runMean", "runVar", "runAR", "runCorr"),
    wsize = 25, nperm = 1000, Kmax = 10, alpha = 0.05,
    varTest = FALSE, bcorr = TRUE, ncpu = 1)
```

```r
# S3 method for class 'kcprs_workflow'
plot(x, ...)
```

```r
# S3 method for class 'kcprs_workflow'
print(x, ...)
```

```r
# S3 method for class 'kcprs_workflow'
summary(object, ...)
```

**Arguments**

- `data` data N x v dataframe where N is the number of time points and v the number of variables.
- `RS_funs` a vector of names of the functions that correspond to the running statistics to be monitored. Options available: "runMean"=running mean, "runVar"=running variance, "runAR"=running autocorrelation and "runCorr"=running correlation.
- `wsize` Window size
- `nperm` Number of permutations used in the permutation test
- `Kmax` Maximum number of change points desired
- `alpha` Significance level for the full KCP-RS workflow analysis if bcorr=1. Otherwise, this is the significance level for each running statistic.
- `varTest` If set to TRUE, only the variance DROP test is implemented, and if set to FALSE, both the variance test and the variance DROP tests are implemented.
- `bcorr` Set to TRUE if Bonferonni correction is desired for the workflow analysis and set to FALSE otherwise.
- `ncpu` number of cpu cores to use
- `x` An object of the type produced by kcprs_workflow
- `...` Further plotting arguments
- `object` An object of the type produced by kcprs_workflow
Details

The workflow proceeds in two steps: First, the mean change points are flagged using KCP on the running means. If there are significant change points, the data is centered based on the yielded change points. Otherwise, the data remains untouched for the next analysis. Second, the remaining running statistics are computed using the centered data in the first step. The user can specify which running statistics to scan change points for (see RS_funs and examples). Bonferroni correction for tracking multiple running statistics can be implemented using the bcorr option.

Value

- **kcpMean**: kcpRS solution for the running means. See output of kcpRS for further details.
- **kcpVar**: kcpRS solution for the running variances. See output of kcpRS for further details.
- **kcpAR**: kcpRS solution for the running autocorrelations. See output of kcpRS for further details.
- **kcpCorr**: kcpRS solution for the running correlations. See output of kcpRS for further details.

References


Examples

```r
phase1=cbind(rnorm(50,0,1),rnorm(50,0,1)) #phase1: Means=0
phase2=cbind(rnorm(50,1,1),rnorm(50,1,1)) #phase2: Means=1
X=rbind(phase1,phase2)

#scan all statistics
res=kcpRS_workflow(data=X,rs_funs=c("runMean","runVar","runAR","runCorr"),wsize=25,nperm=1000,Kmax=10,alpha=.05, varTest=FALSE, bcorr=TRUE, ncpu=1)
summary(res)
plot(res)

#scan the mean and the correlation only
res=kcpRS_workflow(data=X,rs_funs=c("runMean","runCorr"),wsize=25,nperm=1000,Kmax=10, alpha=.05, varTest=FALSE, bcorr=TRUE, ncpu=1)
summary(res)
plot(res)
```
MentalLoad

Mental Load Data

Description
Three physiological measures during a mental load assessment experiment on aviation pilots

Usage
data(MentalLoad)

Format
Dataframe with 1393 rows and 4 columns. The first column indicates the experimental period, while the last three columns correspond to the three physiological measures monitored during the experiment: Heart rate (HR), respiration rate (RR) and petCO2.

References

Examples
data(MentalLoad)

permTest

KCP Permutation Test

Description
The KCP permutation test implements the variance test and the variance drop test to determine if there is at least one change point in the running statistics

Usage
permTest(data, RS_fun, wsize = 25, nperm = 1000, Kmax = 10, alpha = 0.05, varTest = FALSE)
Arguments

- **data**: data $N \times v$ dataframe where $N$ is the number of time points and $v$ the number of variables.
- **RS_fun**: Running statistics function: Should require the time series and wsize as input and return a dataframe of running statistics as output. This output dataframe should have rows that correspond to the time windows and columns that correspond to the variable(s) on which the running statistics were computed.
- **wsize**: Window size
- **nperm**: Number of permutations to be used in the permutation test
- **Kmax**: Maximum number of change points desired
- **alpha**: Significance level of the permutation test
- **varTest**: If FALSE, only the variance DROP test is implemented, and if TRUE, both the variance and the variance DROP tests are implemented.

Value

- **sig**: Significance of having at least one change point. 0 - Not significant, 1 - Significant
- **p_var_test**: P-value of the variance test.
- **p_vardrop_test**: P-value of the variance drop test.
- **perm_rmin**: A matrix of minimized variance criterion for the permuted data.

References


---

runAR | Running Autocorrelations

Description

Extracts the running autocorrelations by sliding a window comprised of wsize time points, and in each window, the autocorrelation for each variable is computed. Each time the window is slid, the oldest time point is discarded and the latest time point is added.

Usage

runAR(data, wsize = 25)

Arguments

- **data**: $N \times v$ dataframe where $N$ is the no. of time points and $v$ the no. of variables
- **wsize**: Window size
runCorr

Value

Running autocorrelations time series

Examples

```r
phase1 = cbind(rnorm(50, 0, 1), rnorm(50, 0, 1))  # phase1: AutoCorr = 0
phase2 = cbind(rnorm(50, 0, 1), rnorm(50, 0, 1))
phase2 = filter(phase2, .5, method = "recursive")  # phase2: AutoCorr = .5
X = rbind(phase1, phase2)
RS = runAR(data = X, wsize = 25)
ts.plot(RS, gpars = list(xlab = "Window", ylab = "Autocorrelation", col = 1:2, lwd = 2))
```

---

### runCorr

**Running Correlations**

**Description**

Extracts the running correlations by sliding a window comprised of \( wsize \) time points, and in each window, the correlation of each pair of variables is computed. Each time the window is slid, the oldest time point is discarded and the latest time point is added.

**Usage**

```r
runCorr(data, wsize = 25)
```

**Arguments**

- **data**: \( N \times v \) dataframe where \( N \) is the no. of time points and \( v \) the no. of variables
- **wsize**: window size

**Value**

Running correlationa time series

**Examples**

```r
data(MentalLoad)
RS <- runCorr(data = MentalLoad, wsize = 25)
ts.plot(RS, gpars = list(xlab = "Window", ylab = "Correlations", col = 1:3, lwd = 2))
```
runMean

Running Means

Description

Extracts the running means by sliding a window comprised of \( wsize \) time points, and in each window, the mean for each variable is computed. Each time the window is slid, the oldest time point is discarded and the latest time point is added.

Usage

\[
\text{runMean(data, wsize = 25)}
\]

Arguments

data : \( N \times v \) dataframe where \( N \) is the no. of time points and \( v \) the no. of variables
wsize : Window size

Value

Running means time series

Examples

phase1=cbind(rnorm(50,0,1), rnorm(50,0,1)) # phase1: Means=0
phase2=cbind(rnorm(50,1,1), rnorm(50,1,1)) # phase2: Means=1
X=rbind(phase1, phase2)
RS=runMean(data=X, wsize=25)
\text{ts.plot(RS, gpars=list(xlab="Window", ylab="Means", col=1:2, lwd=2))}

runVar

Running Variances

Description

Extracts the running variances by sliding a window comprised of \( wsize \) time points, and in each window, the variance for each variable is computed. Each time the window is slid, the oldest time point is discarded and the latest time point is added.

Usage

\[
\text{runVar(data, wsize = 25)}
\]
runVar

Arguments

data N x v dataframe where N is the no. of time points and v the no. of variables
wsize Window size

Value

Running variances time series

Examples

phase1<-cbind(rnorm(50,0,1), rnorm(50,0,1)) #phase1: SD=1
phase2<-cbind(rnorm(50,0,2), rnorm(50,0,2)) #phase2: SD=2
X<-rbind(phase1, phase2)
RS<-runVar(data=X, wsize=25)
ts.plot(RS, gpars=list(xlab="Window", ylab="Variances", col=1:2, lwd=2))
Index

*Topic datasets
  CO2Inhalation, 3
  MentalLoad, 9

CO2Inhalation, 2, 3, 3

kcpa, 2, 4
kcpRS, 2, 3, 5, 8
kcpRS-package, 2
kcpRS_workflow, 2, 3, 7

MentalLoad, 2, 3, 9

permTest, 2, 5, 9
plot.kcpRS (kcpRS), 5
plot.kcpRS_workflow (kcpRS_workflow), 7
print.kcpRS (kcpRS), 5
print.kcpRS_workflow (kcpRS_workflow), 7

runAR, 2, 5, 10
runCorr, 2, 5, 11
runMean, 2, 5, 12
runVar, 2, 5, 12

summary.kcpRS (kcpRS), 5
summary.kcpRS_workflow
  (kcpRS_workflow), 7