Package 'dSTEM'

May 8, 2026

Type Package

Title Multiple Testing of Local Extrema for Detection of Change Points

Version 2.0-1

Date 2023-6-20

Author Zhibing He <zhibingh@asu.edu>

Maintainer Zhibing He <zhibingh@asu.edu>

Description

Simultaneously detect the number and locations of change points in piecewise linear models under stationary Gaussian noise allowing autocorrelated random noise. The core idea is to transform the problem of detecting change points into the detection of local extrema (local maxima and local minima)through kernel smoothing and differentiation of the data sequence, see Cheng et al. (2020) <doi:10.1214/20-EJS1751>. A low-computational and fast algorithm call 'dSTEM' is introduced to detect change points based on the 'STEM' algorithm in D. Cheng and A. Schwartzman (2017) <doi:10.1214/16-AOS1458>.

Depends R (>= 3.1.0)

Imports MASS

URL https://doi.org/10.1214/20-EJS1751,
https://doi.org/10.1214/16-AOS1458

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-21 09:00:07 UTC

Contents

---

conv *Compute convolution function using FFT*

Description

Compute convolution function using FFT, similar to `'conv'` in matlab

Usage

```
conv(u, v, shape = c("same", "full"))
```

Arguments

u	numerical vector
v	numerical vector, don't need to have the same length as u
shape	if 'same', return central part of the convolution and has the same size as u; otherwise return the whole sequence of size $lenth(u) + length(v) - 1$.

Value

a vector of convolution, as specified by shape.

References

Matlab document on `'conv'`: https://www.mathworks.com/help/matlab/ref/conv.html

Examples

```
u = c(-1,2,3,-2,0,1,2)
v = c(2,4,-1,1)
w = conv(u,v,'same')
```

`cp.plt` *Plot data sequence, the first and second-order derivatives, and their local extrema*

Description

Plot data sequence, the first and second-order derivatives, and their local extrema

Usage

```
cp.plt(x, order, icd.noise, H)
```

Arguments

`x`	numerical vector of signal or signal-plus-noise data
`order`	order of derivative of data
`icd.noise`	logical value indicating if `x` includes noise
`H`	optional, vector of change-point locations

Value

a plot

Examples

```
l = 1200
h = seq(150,by=150,length.out=6)
jump = c(0,1.5,2,2.2,1.8,2,1.5)*3
beta1 = c(2,-1,2.5,-3,-0.2,2.5,-0.5)/50
signal = gen.signal(l,h,jump,beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
sdata = smth.gau(signal+noise,gamma)
dy = diff(sdata)
ddy = diff(sdata,differences=2)
cp.plt(signal,0,FALSE)
points(signal+noise,col="grey")
cp.plt(dy,1,H=h)
cp.plt(ddy,2,H=h)
```

`cpTest` *Multiple testing of change points for kernel smoothed data*

Description

Multiple testing of change points for kernel smoothed data

Usage

```
cpTest(
  x,
  order,
  alpha,
  gamma,
  sigma,
  breaks,
  slope,
  untest,
  nu,
  is.constant,
  margin
)
```

Arguments

`x`	vector of kernel smoothed data
`order`	order of derivative of data
`alpha`	global significant level
`gamma`	bandwidth of Gaussian kernel
`sigma`	standard deviation of kernel smoothed noise
`breaks`	vector of rough estimate of change-point locations, only required when order is 1.
`slope`	vector of rough estimate of slopes associated with `breaks`, only required when order is 1.
`untest`	vector of locations unnecessary to test
`nu`	standard deviation of Gaussian kernel used to generate autocorrelated Gaussian noise, it is 0 if the noise is Gaussian white noise.
`is.constant`	logical value indicating if the signal is piecewise constant, if TRUE, `breaks` and `slope` are not necessary.
`margin`	length of one period of data `x`

Value

a list of estimated change-point locations and threshold for p-value

Examples

```
## piecewise linear signal
l = 1200
h = seq(150,by=150,length.out=6)
jump = rep(0,7)
beta1 = c(2,-1,2.5,-3,-0.2,2.5)/50
beta1 = c(beta1,-sum(beta1*(c(h[1],diff(h))))/(l-tail(h,1)))
signal = gen.signal(l,h,jump,beta1)
noise = rnorm(length(signal),0,2)
gamma = 25
sdata = smth.gau(signal+noise,gamma)
ddy = diff(sdata,differences=2)
model2 = cpTest(x=ddy,order=2,gamma=gamma,alpha=0.05)
## piecewise constant
l = 1200
h = seq(150,by=150,length.out=6)
jump = c(0,1.5,2,2.2,1.8,2,1.5)
beta1 = rep(0,length(h)+1)
signal = gen.signal(l,h,jump,beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
sdata = smth.gau(signal+noise,gamma)
dy = diff(sdata)
model1 = cpTest(x=dy,order=1,alpha=0.05,gamma=gamma,is.constant=TRUE)
## piecewise linear with jump
l = 1200
h = seq(150,by=150,length.out=6)
jump = c(0,1.5,2,2.2,1.8,2,1.5)*3
beta1 = c(2,-1,2.5,-3,-0.2,2.5,-0.5)/50
signal = gen.signal(l=l,h=h,jump=jump,b1=beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
sdata = smth.gau(signal+noise,gamma)
dy = diff(sdata)
ddy = diff(sdata,differences=2)
model2 = cpTest(x=ddy,order=2,gamma=gamma,alpha=0.1)
breaks = est.pair(vall=model2$vall,peak=model2$peak,gamma=gamma)$cp
slope = est.slope(x=(signal+noise),breaks=breaks)
```

`dstem` *Detection of change points based on 'dSTEM' algorithm*

Description

Detection of change points based on 'dSTEM' algorithm

Usage

```
dstem(
```

```
  data,
  type = c("I", "II-step", "II-linear", "mixture"),
  gamma = 20,
  alpha = 0.05
)
```

Arguments

data	vector of data sequence
type	"I" if the change points are piecewise linear and continuous; "II-step" if the change points are piecewise constant and noncontinuous; "II-linear" if the change points are piecewise linear and noncontinuous; "mixture" if both type I and type II change points are include in data
gamma	bandwidth of Gaussian kernel
alpha	global significant level

Value

if type is 'mixture', the output is a list of type I and type II change points, otherwise, it is a list of change points

See Also

cpTest

Examples

```
## piecewise linear signal
l = 1200
h = seq(150,by=150,length.out=6)
jump = rep(0,7)
beta1 = c(2,-1,2.5,-3,-0.2,2.5)/50
beta1 = c(beta1,-sum(beta1*(c(h[1],diff(h))))/(l-tail(h,1)))
signal = gen.signal(l,h,jump,beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
model = dstem(signal + noise,"I",gamma=gamma,alpha=0.05)
## piecewise constant
l = 1200
h = seq(150,by=150,length.out=6)
jump = c(0,1.5,2,2.2,1.8,2,1.5)
beta1 = rep(0,length(h)+1)
signal = gen.signal(l,h,jump,beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
model = dstem(signal + noise, "II-step",gamma,alpha=0.05)
## piecewise linear with jump
l = 1200
h = seq(150,by=150,length.out=6)
jump = c(0,1.5,2,2.2,1.8,2,1.5)*3
```

```
beta1 = c(2,-1,2.5,-3,-0.2,2.5,-0.5)/50
signal = gen.signal(l=l,h=h,jump=jump,b1=beta1)
noise = rnorm(length(signal),0,1)
gamma = 25
model = dstem(signal + noise, "II-linear",gamma,alpha=0.05)
```

`est.pair` *Identify pairwise local maxima and local minima of the second-order derivative*

Description

Identify pairwise local maxima and local minima of the second-order derivative

Usage

```
est.pair(vall, peak, gamma)
```

Arguments

`vall`	vector of locations of significant local minima
`peak`	vector of locations of significant local maxima
`gamma`	bandwidth of Gaussian kernel smoothing function

Value

a list of detected pairs and detected change-point locations through second-order derivative testing

---

`est.sigma2` *Estimate variance of smoothed Gaussian noise*

Description

Estimate variance of smoothed Gaussian noise through its second-order derivative

Usage

```
est.sigma2(x, gamma, k = 0.5)
```

Arguments

`x`	numerical vector of second-order derivative of kernel smoothed data
`gamma`	bandwidth of Gaussian kernel
`k`	numerical value, local maxima (minima) are presumed beyond $Mean(x)k * SD(x)$

Value

value of estimated variance of smoothed noise

Examples

```
l=15000; h = seq(150,l,150)
jump = rep(0,length(h)+1); b1 = seq(from=0,by=0.15,length = length(h)+1)
signal = gen.signal(l,h,jump,b1)
data = signal + rnorm(length(signal),0,1) # standard white noise
gamma = 10
ddy = diff(smth.gau(data,gamma),differences=2)
est.sigma2(ddy,gamma,k=0.5) # true value is \eqn{\frac{1}{2\sqrt{pi}\gamma}}
```

---

est.slope *Estimate piecewise slope for piecewise linear model*

Description

Estimate piecewise slope for piecewise linear model

Usage

```
est.slope(x, breaks)
```

Arguments

x	numerical vector of signal-plus-noise data
breaks	numerical vector of change-point locations

Value

a vector of estimated piecewise slope

Fdr *Compute TPR and FPR*

Description

Compute TPR and FPR

Usage

```
Fdr(uh, th, b)
```

Arguments

uh	numerical vector of estimated change point locations
th	numerical vector of true change point locations
b	location tolerance, usually specified as the bandwidth gamma

Value

a dataframe of FDR (FPR) and Power (TPR)

---

fdrBH — *Compute FDR threshold based on Benjamini-Hochberg (BH) algorithm*

Description

Compute FDR threshold based on Benjamini-Hochberg (BH) algorithm

Usage

```
fdrBH(p, q)
```

Arguments

p	a vector of p-values
q	False Discovery Rate level

Value

p-value threshold based on independence or positive dependence

Examples

```
fdrBH(seq(0.01,0.1,0.01),q=0.1)
```

gen.signal *Generate simulated signals*

Description

Generate simulated signals

Usage

```
gen.signal(l, h, jump, b1, rep = 1, shift = 0)
```

Arguments

l	length of data, if data is periodic then the length in each period
h	numerical vector of true change point locations
jump	numerical vector of jump size at change point locations
b1	numerical vector of piecewise slopes
rep	number of periods if data is periodic, default is 1
shift	numerical vector of vertical shifts for each period, default is 0

Value

a vector of simulated signal

Examples

```
l = 1200
h = seq(150,by=150,length.out=6)
jump = rep(0,7)
beta1 = c(2,-1,2.5,-3,-0.2,2.5)/50
beta1 = c(beta1,-sum(beta1*(c(h[1],diff(h))))/(l-tail(h,1)))
signal = gen.signal(l,h,jump,beta1)
```

HST_stock *Stock price of Host & Hotel Resorts (HST)*

Description

A subset of daily stock price data of HST from November 7, 2011 to November 5, 2021.

Usage

```
HST_stock
```

Format

A data frame with 2,517 rows and 6 columns:

Date date from November 7, 2011 to November 5, 2021

Close, Open, High, Low stock price

Volume stock exchange volume

Source

<https://finance.yahoo.com/quote/HST>

`smth.gau` *Smoothing data using Gaussian kernel*

Description

Smoothing data using Gaussian kernel

Usage

```
smth.gau(x, gamma)
```

Arguments

`x`	numeric vector of values to smooth
`gamma`	bandwidth of Gaussian kernel

Value

vector of smoothed values

Examples

```
smth.gau(x=rnorm(1000), gamma=20)
```

snr *Compute SNR of a certain change point location*

Description

Compute SNR of a certain change point location

Usage

```
snr(order, gamma, is.jump, jump, diffb, addb)
```

Arguments

order	order of derivative of data
gamma	bandwidth of Gaussian kernel
is.jump	logical value indicating if the location to be calculated is a jump point
jump	jump height
diffb	difference of the slopes on left and right sides of the location
addb	sum of the slopes, only used when order is 1

Value

value of SNR

which.peaks *Find local maxima and local minima of data sequence*

Description

Find local maxima and local minima of data sequence

Usage

```
which.peaks(x, partial = FALSE, decreasing = FALSE)
```

Arguments

x	numerical vector contains local maxima (minima)
partial	logical value indicating if the two endpoints will be considered
decreasing	logical value indicating whether to find local minima

Value

a vector of locations of local maxima or minima

Examples

```
a = 100:1
which.peaks(a*sin(a/3))
```

Index