Package ‘slopeOP’

November 23, 2020

Type Package

Title Change-in-Slope OP Algorithm with a Finite Number of States

Version 1.0.1

Maintainer Vincent Runge <vincent.runge@univ-evry.fr>

Description Optimal partitioning algorithm for change-in-slope problem with continuity constraint and a finite number of states. Some constraints can be enforced in the inference: isotonic, unimodal or smoothing. With the function slopeSN() (segment neighborhood) the number of segments to infer is fixed by the user and does not depend on a penalty value.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp

RoxygenNote 6.1.1

NeedsCompilation yes

Author Vincent Runge [aut, cre],
Marco Pascucci [aut]

Repository CRAN

Date/Publication 2020-11-23 16:40:02 UTC

R topics documented:

linearOP 2
plot.slopeOP 2
sdHallDiff 3
slopeData 3
slopeOP 4
slopeSN 5

Index 7

`linearOP` *linearOP*

Description

An optimal partitioning algorithm with a linear fit for each segment

Usage

```
linearOP(x, data, penalty, cc = FALSE)
```

Arguments

`x`	a vector (see data)
`data`	a vector defining the data points (x[i], data[i])
`penalty`	the penalty for introducing a new segment
`cc`	a boolean to impose a continuity constraint

`plot.slopeOP` *plot.slopeOP*

Description

Plot the result of the slopeOP function and the data

Usage

```
## S3 method for class 'slopeOP'
plot(x, ..., data, chpt = NULL, states = NULL)
```

Arguments

`x`	a slopeOP class object
`...`	other parameters
`data`	the data from which we get the slopeOP object x
`chpt`	vector of changepoints of the model
`states`	vector of states of the model

Value

plot data and the inferred slopeOP result (and the model if specified in 'chpt' and 'states' parameters)

Examples

```
myData <- slopeData(index = c(1,100,200,300), states = c(0,5,3,6), noise = 2)
s <- slopeOP(data = myData, states = 0:6, penalty = 20)
plot(s, data = myData, chpt = c(1,100,200,300), states = c(0,5,3,6))
```

`sdHallDiff` *sdHallDiff*

Description

Estimation of the standard deviation using the HallDiff estimator

Usage

```
sdHallDiff(data)
```

Arguments

`data`	vector of data to segment: a univariate time series

Value

an estimation of the sd

Examples

```
myData <- slopeData(index = c(1,100,200,300), states = c(0,5,3,6), noise = 1)
sdHallDiff(data = myData)
```

`slopeData` *slopeData*

Description

Generate data with a given continuous piecewise linear model

Usage

```
slopeData(index, states, noise = 0, outlierDensity = 0,
  outlierNoise = 50)
```

Arguments

`index`	a vector of increasing changepoint indices
`states`	vector of successive states
`noise`	noise level = standard deviation of an additional normal noise
`outlierDensity`	probability for a datapoint to be an outlier (has to be close to 0)
`outlierNoise`	noise level for outlier data points

Value

a vector of simulated data

Examples

```
myData <- slopeData(index = c(1,100,200,300), states = c(0,5,3,6), noise = 1)
```

`slopeOP` *slopeOP*

Description

Optimal partitioning algorithm for change-in-slope problem with a finite number of states (beginning and ending values of each segment is restricted to a finite set of values called states). The algorithm takes into account a continuity constraint between successive segments and infers a continuous piecewise linear signal.

Usage

```
slopeOP(data, states, penalty = 0, constraint = "null", minAngle = 0,
  type = "channel", testMode = FALSE)
```

Arguments

`data`	vector of data to segment: a univariate time series
`states`	vector of states = set of accessible starting/ending values for segments in increasing order.
`penalty`	the penalty value (a non-negative real number)
`constraint`	string defining a constraint : "null", "isotonic", "unimodal" or "smoothing"
`minAngle`	a minimal inner angle in degree between consecutive segments in case constraint = "smoothing"
`type`	string defining the pruning type to use. "null" = no pruning, "channel" = use monotonicity property, "pruning" = pelt-type property
`testMode`	a boolean, if true the function also returns the percent of elements to scan (= ratio scanned elements vs. scanned elements if no pruning)

Value

a list of 3 elements = (changepoints, states, globalCost). (Pruning is optional)

`changepoints` is the vector of changepoints (we return the extremal values of all segments from left to right)

`states` is the vector of successive states. states[i] is the value we inferred at position changepoints[i]

`globalCost` is a number equal to the global cost of the non-penalized change-in-slope problem. That is the value of the fit to the data ignoring the penalties for adding changes

pruning is the percent of positions to consider in cost matrix Q (returned only if testMode = TRUE)

Examples

```
myData <- slopeData(index = c(1,100,200,300), states = c(0,5,3,6), noise = 1)
slopeOP(data = myData, states = 0:6, penalty = 10)
```

---

slopeSN *slopeSN*

Description

Segment neighborhood algorithm for change-in-slope problem with a finite number of states (beginning and ending values of each segment is restricted to a finite set of values called states). The algorithm takes into account a continuity constraint between successive segments and infers a continuous piecewise linear signal with a given number of segments.

Usage

```
slopeSN(data, states, nbSegments = 1, constraint = "null",
  testMode = FALSE)
```

Arguments

`data`	vector of data to segment: a univariate time series
`states`	vector of states = set of accessible starting/ending values for segments in increasing order.
`nbSegments`	the number of segments to infer
`constraint`	string defining a constraint : "null", "isotonic"
`testMode`	a boolean, if true the function also returns the percent of elements to scan (= ratio scanned elements vs. scanned elements if no pruning)

Value

a list of 3 elements = (changepoints, states, globalCost). (Pruning is optional)

`changepoints` is the vector of changepoints (we return the extremal values of all segments from left to right)

`states` is the vector of successive states. states[i] is the value we inferred at position changepoints[i]

`globalCost` is a number equal to the global cost of the non-penalized change-in-slope problem. That is the value of the fit to the data ignoring the penalties for adding changes

pruning is the percent of positions to consider in cost matrix Q (returned only if testMode = TRUE)

Examples

```
myData <- slopeData(index = c(1,100,200,300), states = c(0,5,3,6), noise = 1)
slopeSN(data = myData, states = 0:6, nbSegments = 2)
```

Index

linearOP, 2

plot.slopeOP, 2

sdHallDiff, 3
slopeData, 3
slopeOP, 4
slopeSN, 5