

Package ‘BinSegBstrap’

January 20, 2025

Title Piecewise Smooth Regression by Bootstrapped Binary Segmentation

Version 1.0-1

Depends R (>= 3.0.0)

Imports Rcpp (>= 0.12.3), stats

LinkingTo Rcpp

Suggests knitr

VignetteBuilder knitr

Description Provides methods for piecewise smooth regression. A piecewise smooth signal is estimated by applying a bootstrapped test recursively (binary segmentation approach). Each bootstrapped test decides whether the underlying signal is smooth on the currently considered subsegment or contains at least one further change-point.

License GPL-3

NeedsCompilation yes

Author McDaid Kate [aut],
Pein Florian [aut, cre]

Maintainer Pein Florian <f.pein@lancaster.ac.uk>

Repository CRAN

Date/Publication 2022-01-27 23:10:10 UTC

Contents

BinSegBstrap-package	2
BinSegBstrap	3
BstrapTest	4
estimateSingleCp	6
Index	8

BinSegBstrap-package *Piecewise smooth regression by bootstrapped binary segmentation*

Description

Provides methods for piecewise smooth regression. The main function `BinSegBstrap` estimates a piecewise smooth signal by applying a bootstrapped test recursively (binary segmentation approach). A single bootstrapped test for the hypothesis that the underlying signal is smooth versus the alternative that the underlying signal contains at least one change-point can be performed by the function `BstrapTest`. A single change-point is estimated by the function `estimateSingleCp`. More details can be found in the vignette. Parts of this work were inspired by Gijbels and Goderniaux (2004).

Acknowledgement

This work results from a summer research project at the University of Cambridge in 2019. Kate McDaid was supported by a bursary from the summer research programme of the Centre of Mathematics at the University of Cambridge. Florian Pein's position is funded by the EPSRC programme grant 'StatScale: Statistical Scalability for Streaming Data'.

References

Gijbels, I., Goderniaux, A-C. (2004) Bootstrap test for change-points in nonparametric regression. *Journal of Nonparametric Statistics* **16**(3-4), 591–611.

See Also

[BinSegBstrap](#), [BstrapTest](#), [estimateSingleCp](#)

Examples

```
n <- 200
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5
signal[151:200] <- signal[151:200] + 5

y <- rnorm(n) + signal

est <- BinSegBstrap(y = y)

plot(y)
lines(signal)
lines(est$est, col = "red")

n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5
```

```

y <- rnorm(n) + signal

test <- BstrapTest(y = y)
est <- estimateSingleCp(y = y)

plot(y)
lines(signal)
lines(est$est, col = "red")

```

BinSegBstrap

Estimates a piecewise smooth signal

Description

A piecewise smooth signal is estimated by applying `BstrapTest` recursively (binary segmentation approach). The final estimator is estimated by kernel smoothing on each segment separately; a joint bandwidth is selected by crossvalidation. More details can be found in the vignette.

Usage

```

BinSegBstrap(y, bandwidth, nbandwidth = 30L, B = 500L, alpha = 0.05,
             kernel = c("epanechnikov", "gaussian", "rectangular",
                       "triangular", "biweight", "silverman"))

```

Arguments

<code>y</code>	a numeric vector containing the data points
<code>bandwidth</code>	the bandwidth, i.e. a numeric with values between $1 / \text{length}(y)$ and 0.5 . If missing <code>exp(seq(log(10 / length(y)), log(0.25), length.out = nbandwidth))</code> will be used. Crossvalidation will be performed if it is not a single numeric. Note that the test has almost no power when the bandwidth for the kernel smoother is too small, since then a change-point can be approximated well by a quickly changing smooth function.
<code>nbandwidth</code>	a single integer giving the number of bandwidths (see above) if bandwidth is missing
<code>B</code>	a single integer giving the number of bootstrap samples
<code>alpha</code>	a probability, i.e. a single numeric between 0 and 1, giving the significance level of the test
<code>kernel</code>	the kernel function, i.e. either a string or a function that takes a single numeric vector and returns the values of the kernel at those locations

Value

a `list` with the following components:

- `est`: the estimated signal
- `cps`: the estimated change-point locations
- `bandwidth`: the selected bandwidth

Examples

```

n <- 200
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5
signal[151:200] <- signal[151:200] + 5

y <- rnorm(n) + signal

# default bandwidth and kernel
est <- BinSegBstrap(y = y)

plot(y)
lines(signal)
lines(est$est, col = "red")

# fixed bandwidth
est <- BinSegBstrap(y = y, bandwidth = 0.1)

# user specified kernel
kernel <- function(x) 1 - abs(x) # triangular kernel
est <- BinSegBstrap(y = y, kernel = kernel)

```

BstrapTest

Bootstrap test for a single change-point

Description

Tests whether the underlying signal is smooth or contains at least one change-point. The smooth alternative is estimated by a (crossvalidated) kernel smoother. The single change-point alternative is estimated by `estimateSingleCp`. Its estimated jump size is used as a test statistic and the critical value is obtained by bootstrapping. More details can be found in the vignette.

Usage

```

BstrapTest(y, bandwidth, nbandwidth = 30L, B = 500L, alpha = 0.05,
           kernel = c("epanechnikov", "gaussian", "rectangular",
                     "triangular", "biweight", "silverman"))

```

Arguments

y	a numeric vector containing the data points
bandwidth	the bandwidth, i.e. a numeric with values between $1 / \text{length}(y)$ and 0.5. If missing <code>exp(seq(log(10 / length(y)), log(0.25), length.out = nbandwidth))</code> will be used. Crossvalidation will be performed if it is not a single numeric. Note that the test has almost no power when the bandwidth for the kernel smoother is too small, since then a change-point can be approximated well by a quickly changing smooth function.

nbandwidth	a single integer giving the number of bandwidths (see above) if bandwidth is missing
B	a single integer giving the number of bootstrap samples
alpha	a probability, i.e. a single numeric between 0 and 1, giving the significance level of the test
kernel	the kernel function, i.e. either a string or a function that takes a single numeric vector and returns the values of the kernel at those locations

Value

a **list** with the following components:

- piecewiseSignal: the estimated signal with a single change-point
- cp: the estimated change-point location
- size: the estimated jump size
- bandwidth: the selected bandwidth for the piecewise signal
- bandwidthSmooth: the selected bandwidth for the smooth signal
- smoothSignal: the estimated smooth signal
- critVal: the by bootstrapping obtained critical value
- pValue: the p-Value of the test
- outcome: a boolean saying whether the test rejects the hypothesis of a smooth signal

Examples

```
n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5

y <- rnorm(n) + signal

# default bandwidth and kernel
test <- BstrapTest(y = y)

if (test$outcome) {
  # null hypothesis of a smooth signal is rejected
  estimatedSignal <- test$piecewiseSignal
} else {
  # null hypothesis of a smooth signal is accepted
  estimatedSignal <- test$smoothSignal
}

plot(y)
lines(signal)
lines(estimatedSignal, col = "red")

# fixed bandwidth
test <- BstrapTest(y = y, bandwidth = 0.1)

# user specified kernel
kernel <- function(x) 1 - abs(x) # triangular kernel
test <- BstrapTest(y = y, kernel = kernel)
```

 estimateSingleCp

Estimation of a single change-point

Description

Estimates a single change-point in an otherwise smooth function. The change-point location is estimated as the maximum of the differences of left and right sided running means. The estimate left and right of the change-point are obtained by kernel smoothers. Windows of the running mean and kernel bandwidth are chosen by crossvalidation. More details can be found in the vignette.

Usage

```
estimateSingleCp(y, bandwidth, nbandwidth = 30L,
                 kernel = c("epanechnikov", "gaussian", "rectangular",
                            "triangular", "biweight", "silverman"))
```

Arguments

y	a numeric vector containing the data points
bandwidth	the bandwidth, i.e. a numeric with values between $1 / \text{length}(y)$ and 0.5 . If missing $\exp(\text{seq}(\log(2 / \text{length}(y)), \log(0.25), \text{length.out} = \text{nbandwidth}))$ will be used. Crossvalidation will be performed if it is not a single numeric
nbandwidth	a single integer giving the number of bandwidths (see above) if bandwidth is missing
kernel	the kernel function, i.e. either a string or a function that takes a single numeric vector and returns the values of the kernel at those locations

Value

a [list](#) with the following components:

- est: the estimated function with a single change-point
- cp: the estimated change-point location
- size: the estimated jump size
- bandwidth: the selected bandwidth

Examples

```
n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5

y <- rnorm(n) + signal

# default bandwidth and kernel
est <- estimateSingleCp(y = y)

plot(y)
```

```
lines(signal)
lines(est$est, col = "red")

# fixed bandwidth
est <- estimateSingleCp(y = y, bandwidth = 0.1)

# user specified kernel
kernel <- function(x) 1 - abs(x) # triangular kernel
est <- estimateSingleCp(y = y, kernel = kernel)
```

Index

* **nonparametric**

BinSegBstrap-package, [2](#)

* **package**

BinSegBstrap-package, [2](#)

BinSegBstrap, [2](#), [3](#)

BinSegBstrap-package, [2](#)

BstrapTest, [2](#), [3](#), [4](#)

estimateSingleCp, [2](#), [4](#), [6](#)

list, [3](#), [5](#), [6](#)