Package ‘breakfast’

September 28, 2017

Title Multiple Change-Point Detection and Segmentation

Version 1.0.0

Description The breakfast package performs multiple change-point detection in data sequences, or sequence segmentation, using computationally efficient multiscale methods. This version of the package implements the ``Tail-Greedy Unbalanced Haar'', ``Wild Binary Segmentation'' and ``Adaptive Wild Binary Segmentation'' change-point detection and segmentation methodologies. To start with, see the function segment.mean.

Depends R (>= 3.4.0)

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports plyr

NeedsCompilation no

Author Piotr Fryzlewicz [aut, cre]

Maintainer Piotr Fryzlewicz <p.fryzlewicz@lse.ac.uk>

Repository CRAN

Date/Publication 2017-09-28 15:49:36 UTC

R topics documented:

`breakfast` *breakfast: Multiple change-point detection and segmentation for data sequences*

Description

The breakfast package performs multiple change-point detection in data sequences, or sequence segmentation, using computationally efficient multiscale methods. This version of the package implements the "Tail-Greedy Unbalanced Haar", "Wild Binary Segmentation" and "Adaptive Wild Binary Segmentation" change-point detection and segmentation methodologies. To start with, see the function `segment.mean`.

Author(s)

Piotr Fryzlewicz, <`p.fryzlewicz@lse.ac.uk`>

References

"Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint. "Wild Binary Segmentation for multiple change-point detection", P. Fryzlewicz (2014), Annals of Statistics, 42, 2243-2281. "Data-adaptive Wild Binary Segmentation", P. Fryzlewicz (2017), in preparation as of September 28th, 2017.

See Also

`segment.mean`

Examples

```
#See Examples for segment.mean
```

`hybrid.cpt` *Multiple change-point detection in the mean of a vector using a hybrid between the TGUH and Adaptive WBS methods.*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, combining the Tail-Greedy Unbalanced Haar and Adaptive Wild Binary Segmentation methods (see Details for the relevant literature references). The constant means between each pair of neighbouring change-points are also estimated. The method works best when the noise in the input vector is independent and identically distributed Gaussian.

Usage

```
hybrid.cpt(x, M = 1000, sigma = stats::mad(diff(x)/sqrt(2)), th.const = 1,
  p = 0.01, minseglen = 1, bal = 1/20, num.zero = 10^(-5))
```

Arguments

x	A vector containing the data in which you wish to find change-points.
M	The same as the corresponding parameter in wbs.K.cpt.
sigma	The same as the corresponding parameter in tguh.cpt.
th.const	The same as the corresponding parameter in tguh.cpt.
p	The same as the corresponding parameter in tguh.cpt.
minseglen	The same as the corresponding parameter in tguh.cpt.
bal	The same as the corresponding parameter in tguh.cpt.
num.zero	The same as the corresponding parameter in tguh.cpt.

Details

This is a hybrid method, which first estimates the number of change-points using tguh.cpt and then estimates their locations using wbs.K.cpt.

The change-point detection algorithms used in tguh.cpt are: the Tail-Greedy Unbalanced Haar method as described in "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint, and Adaptive Wild Binary Segmentation as described in "Data-adaptive Wild Binary Segmentation", P. Fryzlewicz (2017), in preparation as of September 28th, 2017.

Value

A list with the following components:

est	The estimated piecewise-constant mean of x.
no.of.cpt	The estimated number of change-points in the piecewise-constant mean of x.
cpt	The estimated locations of change-points in the piecewise-contant mean of x (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, wbs.bic.cpt, wbs.thresh.cpt, wbs.cpt, tguh.cpt, wbs.K.cpt

Examples

```
teeth <- rep(rep(0:1, each=5), 20)
teeth.noisy <- teeth + rnorm(200)/5
teeth.cleaned <- hybrid.cpt(teeth.noisy)
ts.plot(teeth.cleaned$est)
```

segment.mean *Multiple change-point detection in the mean of a vector*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using a method that puts more emphasis either on "speed" (i.e. is faster but possibly less accurate) or on "accuracy" (i.e. is possibly more accurate but slower). It also estimates the constant means between each pair of neighbouring change-points. It works best when the noise in the input vector is independent and identically distributed Gaussian.

Usage

```
segment.mean(x, attribute = "speed", M = 1000,
  sigma = stats::mad(diff(x)/sqrt(2)), th.const = 1, p = 0.01,
  minseglen = 1, bal = 1/20, num.zero = 10^(-5))
```

Arguments

`x`	A vector containing the data in which you wish to find change-points.
`attribute`	As described in the Details section of this help file.
`M`	The same as the corresponding parameter in `hybrid.cpt`.
`sigma`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.
`th.const`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.
`p`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.
`minseglen`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.
`bal`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.
`num.zero`	The same as the corresponding parameter in `tguh.cpt` and `hybrid.cpt`.

Details

In the current version of the package, `attribute="speed"` triggers the function `tguh.cpt` and `attribute="accuracy"` triggers the function `hybrid.cpt`. **Warning:** this can change in future versions of the package. Note that `tguh.cpt` and `hybrid.cpt` return the same number of change-points and the only difference lies in their estimated locations.

Value

A list with the following components:

`est`	The estimated piecewise-constant mean of `x`.
`no.of.cpt`	The estimated number of change-points in the piecewise-constant mean of `x`.
`cpt`	The estimated locations of change-points in the piecewise-contant mean of `x` (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

tguh.cpt, hybrid.cpt, wbs.cpt

Examples

```
stairs <- rep(1:50, each=10)
stairs.noisy <- stairs + rnorm(500)/5
stairs.cleaned <- segment.mean(stairs.noisy)
ts.plot(stairs.cleaned$est)
stairs.cleaned$no.of.cpt
stairs.cleaned$cpt
```

---

tguh.cpt — *Multiple change-point detection in the mean of a vector using the TGUH method*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using the Tail-Greedy Unbalanced Haar method (see Details for the relevant literature reference). It also estimates the constant means between each pair of neighbouring change-points. It works best when the noise in the input vector is independent and identically distributed Gaussian.

Usage

```
tguh.cpt(x, sigma = stats::mad(diff(x)/sqrt(2)), th.const = 1, p = 0.01,
  minseglen = 1, bal = 1/20, num.zero = 10^(-5))
```

Arguments

x	A vector containing the data in which you wish to find change-points.
sigma	The estimate or estimator of the standard deviation of the noise in x; the default is the Median Absolute Deviation of x computed under the assumption that the noise is independent and identically distributed Gaussian.
th.const	Tuning parameter. Change-points are estimated by connected thresholding (of the Tail-Greedy Unbalanced Haar decomposition of x) in which the threshold has magnitude `sigma * sqrt(2 * (1 + 0.01) * log(n)) * th.const`, where n is the length of x. The default value of th.const is 1.
p	Specifies the number of region pairs merged in each pass through the data, as the proportion of all remaining region pairs. The default is 0.01.
minseglen	The minimum permitted length of each segment of constancy in the estimated mean of x; the default is 1.

`bal`	Specifies the minimum ratio of the length of the shorter wing of each Unbalanced Haar wavelet whose coefficient survives the thresholding, to the length of its support. The default is 0.05.
`num.zero`	Numerical zero; the default is 0.00001.

Details

The change-point detection algorithm used in `tguh.cpt` is the Tail-Greedy Unbalanced Haar method as described in "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint. This paper describes two optional post-processing steps; neither of them is implemented in this package.

Value

A list with the following components:

`est`	The estimated piecewise-constant mean of `x`.
`no.of.cpt`	The estimated number of change-points in the piecewise-constant mean of `x`.
`cpt`	The estimated locations of change-points in the piecewise-contant mean of `x` (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, hybrid.cpt, tguh.decomp, tguh.denoise, tguh.reconstr

Examples

```
stairs <- rep(1:50, each=10)
stairs.noisy <- stairs + rnorm(500)/5
stairs.cleaned <- tguh.cpt(stairs.noisy)
ts.plot(stairs.cleaned$est)
stairs.cleaned$no.of.cpt
stairs.cleaned$cpt
```

---

tguh.decomp — *The Tail-Greedy Unbalanced Haar decomposition of a vector*

Description

This function performs the Tail-Greedy Unbalanced Haar decomposition of the input vector.

Usage

```
tguh.decomp(x, p = 0.01)
```

Arguments

x	A vector you wish to decompose.
p	Specifies the number of region pairs merged in each pass through the data, as the proportion of all remaining region pairs. The default is 0.01.

Details

The Tail-Greedy Unbalanced Haar decomposition algorithm is described in "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint.

Value

A list with the following components:

n	The length of x.
decomp.hist	The decomposition history: the complete record of the n-1 steps taken to decompose x. This is an array of dimensions 4 by 2 by n-1. Each of the n-1 matrices of dimensions 4 by 2 contains the following: first row - the indices of the regions merged, in increasing order (note: the indexing changes through the transform); second row - the values of the Unbalanced Haar filter coefficients used to produce the corresponding detail coefficient; third row - the (detail coefficient, smooth coefficient) of the decomposition; fourth row - the lengths of (left wing, right wing) of the corresponding Unbalanced Haar wavelet.
tguh.coeffs	The coefficients of the Tail-Greedy Unbalanced Haar transform of x.

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

tguh.cpt, tguh.denoise, tguh.reconstr

Examples

```
rnoise <- rnorm(10)
tguh.decomp(rnoise)
```

---

tguh.denoise — *Noise removal from Tail-Greedy Unbalanced Haar coefficients via connected thresholding*

Description

This function performs the connected thresholding of the Tail-Greedy Unbalanced Haar coefficients.

Usage

```
tguh.denoise(tguh.decomp.obj, lambda, minseglen = 1, bal = 1/20)
```

Arguments

tguh.decomp.obj	A variable returned by tguh.decomp or tguh.denoise.
lambda	The threshold value.
minseglen	The minimum permitted length of either wing of any Unbalanced Haar wavelet whose corresponding coefficient survives the thresholding.
bal	The minimum permitted ratio of the length of either wing to the sum of the lengths of both wings of any Unbalanced Haar wavelet whose corresponding coefficient survives the thresholding.

Details

Typically, the first parameter of tguh.denoise will be an object returned by tguh.decomp. The function tguh.denoise performs the "connected thresholding" of this object, in the sense that if a Tail-Greedy Unbalanced Haar detail coefficient does not have any surviving children coefficients, then it gets set to zero if it falls under the threshold, or if the corresponding Unbalanced Haar wavelet is too unbalanced or has too short a wing. See "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint, for details.

Value

Modified object tguh.decomp.obj; the modification is that the detail coefficients in the decomp.hist field that do not survive the thresholding get set to zero.

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

tguh.cpt, tguh.decomp, tguh.reconstr

Examples

```
rnoise <- rnorm(10)
rnoise.tguh <- tguh.decomp(rnoise)
print(rnoise.tguh)
rnoise.denoise <- tguh.denoise(rnoise.tguh, 3)
rnoise.clean <- tguh.reconstr(rnoise.denoise)
print(rnoise.clean)
```

`tguh.reconstr` *The inverse Tail-Greedy Unbalanced Haar transformation*

Description

This function performs the inverse Tail-Greedy Unbalanced Haar transformation, also referred to as reconstruction.

Usage

```
tguh.reconstr(tguh.decomp.obj)
```

Arguments

`tguh.decomp.obj`
: A variable returned by `tguh.decomp` or `tguh.denoise`.

Details

The Tail-Greedy Unbalanced Haar decomposition and reconstruction algorithms are described in "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2017), preprint.

Value

A vector being the result of the inverse Tail-Greedy Unbalanced Haar transformation of `tghu.decomp.obj`.

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

`tguh.cpt`, `tguh.decomp`, `tguh.denoise`

Examples

```
rnoise <- rnorm(10)
rnoise.tguh <- tguh.decomp(rnoise)
print(rnoise.tguh)
rnoise.denoise <- tguh.denoise(rnoise.tguh, 3)
rnoise.clean <- tguh.reconstr(rnoise.denoise)
print(rnoise.clean)
```

`wbs.bic.cpt` *Multiple change-point detection in the mean of a vector using the WBS method, with the number of change-points chosen by BIC*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using the Wild Binary Segmentation method (see Details for the relevant literature reference). The number of change-points is chosen via the Bayesian Information Criterion. The constant means between each pair of neighbouring change-points are also estimated. The method works best when the noise in the input vector is independent and identically distributed Gaussian, and when the number change-points is small.

Usage

```
wbs.bic.cpt(x, M = 20000, Kmax = ceiling(length(x)/5))
```

Arguments

`x`	A vector containing the data in which you wish to find change-points.
`M`	The number of randomly selected sub-segments of the data on which to build the CUSUM statistics in the Wild Binary Segmentation algorithm; generally, the larger the value of M, the more accurate but slower the algorithm - but see the remarks below about the BIC penalty.
`Kmax`	The maximum number of change-points that can be detected.

Details

The BIC penalty is unsuitable as a model selection tool in long signals with frequent change-points; if you need a more versatile function that works well regardless of the number of change-points, try `segment.mean` (for a default recommended estimation technique), `wbs.thresh.cpt`, `wbs.cpt` (if you require an (Adaptive) WBS-based technique), `tguh.cpt` (if you require a TGUH-based technique), or `hybrid.cpt` (to use a hybrid between TGUH and Adaptive WBS). If you are unsure where to start, try `segment.mean`. (If you know how many change-points you wish to detect, try `wbs.K.cpt`.)

The change-point detection algorithm used in `wbs.bic.cpt` is the Wild Binary Segmentaton method as described in "Wild Binary Segmentation for multiple change-point detection", P. Fryzlewicz (2014), Annals of Statistics, 42, 2243-2281.

Value

A list with the following components:

`est`	The estimated piecewise-constant mean of `x`.
`no.of.cpt`	The estimated number of change-points in the piecewise-constant mean of `x`.
`cpt`	The estimated locations of change-points in the piecewise-contant mean of `x` (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, wbs.thresh.cpt, wbs.cpt, tguh.cpt, hybrid.cpt, wbs.K.cpt

Examples

```
teeth <- rep(rep(0:1, each=5), 20)
teeth.noisy <- teeth + rnorm(200)/5
teeth.cleaned <- wbs.bic.cpt(teeth.noisy)
ts.plot(teeth.cleaned$est)
teeth.cleaned$no.of.cpt
teeth.cleaned$cpt
```

---

wbs.cpt — *Multiple change-point detection in the mean of a vector using the (Adaptive) WBS method.*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using the (Adaptive) Wild Binary Segmentation method (see Details for the relevant literature references). The constant means between each pair of neighbouring change-points are also estimated. The method works best when the noise in the input vector is independent and identically distributed Gaussian.

Usage

```
wbs.cpt(x, sigma = stats::mad(diff(x)/sqrt(2)), M.bic = 20000,
  Kmax = ceiling(length(x)/5), universal = TRUE, M.thresh = NULL,
  th.const = NULL, th.const.min.mult = 0.825, adapt = TRUE,
  lambda = 0.9)
```

Arguments

`x`	A vector containing the data in which you wish to find change-points.
`sigma`	Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.
`M.bic`	Only relevant to the wbs.bic.cpt part (see Details below); the same as the M parameter in wbs.bic.cpt.
`Kmax`	Only relevant to the wbs.bic.cpt part (see Details below); the same as the corresponding parameter in wbs.bic.cpt.
`universal`	Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.

M.thresh — Only relevant to the wbs.thresh.cpt part (see Details below); the same as the M parameter in wbs.thresh.cpt.

th.const — Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.

th.const.min.mult — Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.

adapt — Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.

lambda — Only relevant to the wbs.thresh.cpt part (see Details below); the same as the corresponding parameter in wbs.thresh.cpt.

Details

This is a hybrid method, which returns the result of wbs.thresh.cpt or wbs.bic.cpt, whichever of the two detect the larger number of change-points. If there is a tie, wbs.bic.cpt is returned.

The change-point detection algorithms used in wbs.thresh.cpt are: standard Wild Binary Segmentation [see "Wild Binary Segmentation for multiple change-point detection", P. Fryzlewicz (2014), Annals of Statistics, 42, 2243-2281] and Adaptive Wild Binary Segmentation [see "Data-adaptive Wild Binary Segmentation", P. Fryzlewicz (2017), in preparation as of September 28th, 2017].

Value

A list with the following components:

est — The estimated piecewise-constant mean of x.

no.of.cpt — The estimated number of change-points in the piecewise-constant mean of x.

cpt — The estimated locations of change-points in the piecewise-contant mean of x (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, wbs.bic.cpt, wbs.thresh.cpt, tguh.cpt, hybrid.cpt, wbs.K.cpt

Examples

```
teeth <- rep(rep(0:1, each=5), 20)
teeth.noisy <- teeth + rnorm(200)/5
teeth.cleaned <- wbs.cpt(teeth.noisy)
ts.plot(teeth.cleaned$est)
```

wbs.K.cpt *Detecting exactly K change-points in the mean of a vector using the Adaptive WBS method*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using the Adaptive Wild Binary Segmentation method (see Details for the relevant literature reference). The number of change-points is exactly K. The constant means between each pair of neighbouring change-points are also estimated. The method works best when the noise in the input vector is independent and identically distributed Gaussian. As a by-product, the function also computes the entire solution path, i.e. all estimated n-1 change-point locations (where n is the length of the input data) sorted from the most to the least important.

Usage

```
wbs.K.cpt(x, K, M = 1000)
```

Arguments

x	A vector containing the data in which you wish to find change-points.
K	The number of change-points you wish to detect.
M	The number of randomly selected sub-segments of the data on which to build the CUSUM statistics on each recursively identified interval in the Adaptive Wild Binary Segmentation algorithm.

Details

This function should only be used if (a) you know exactly how many change-points you wish to detect, or (b) you wish to order all possible change-points from the most to the least important. If you need a function to estimate the number of change-points for you, try segment.mean (for a default recommended estimation technique), wbs.thresh.cpt, wbs.bic.cpt, wbs.cpt (if you require an (Adaptive) WBS-based technique), tguh.cpt (if you require a TGUH-based technique), or hybrid.cpt (to use a hybrid between TGUH and Adaptive WBS). If you are unsure where to start, try segment.mean.

The change-point detection algorithm used in wbs.K.cpt is the Adaptive Wild Binary Segmentaton method as described in "Data-adaptive Wild Binary Segmentation", P. Fryzlewicz (2017), in preparation as of September 28th, 2017.

Value

A list with the following components:

est	The estimated piecewise-constant mean of x.
no.of.cpt	The estimated number of change-points in the piecewise-constant mean of x; the minumum of K and n-1, where n is the length of x

cpt — The estimated locations of change-points in the piecewise-contant mean of x (these are the final indices *before* the location of each change-point).

cpt.sorted — The list of all possible change-point locations, sorted from the most to the least likely

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, wbs.thresh.cpt, wbs.cpt, tguh.cpt, hybrid.cpt, wbs.bic.cpt

Examples

```
teeth <- rep(rep(0:1, each=5), 20)
teeth.noisy <- teeth + rnorm(200)/5
teeth.cleaned <- wbs.K.cpt(teeth.noisy, 39)
teeth.cleaned$cpt
teeth.cleaned <- wbs.K.cpt(teeth.noisy, 78)
teeth.cleaned$cpt
teeth.cleaned$cpt.sorted
```

---

wbs.thresh.cpt — *Multiple change-point detection in the mean of a vector using the (Adaptive) WBS method, with the number of change-points chosen by thresholding*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of the noisy input vector, using the (Adaptive) Wild Binary Segmentation method (see Details for the relevant literature references). The number of change-points is chosen via a thresholding-type criterion. The constant means between each pair of neighbouring change-points are also estimated. The method works best when the noise in the input vector is independent and identically distributed Gaussian.

Usage

```
wbs.thresh.cpt(x, sigma = stats::mad(diff(x)/sqrt(2)), universal = TRUE,
  M = NULL, th.const = NULL, th.const.min.mult = 0.825, adapt = TRUE,
  lambda = 0.9)
```

Arguments

`x` A vector containing the data in which you wish to find change-points.

`sigma` The estimate or estimator of the standard deviation of the noise in `x`; the default is the Median Absolute Deviation of `x` computed under the assumption that the noise is independent and identically distributed Gaussian.

`universal` If `TRUE`, then `M` and `th.const` (see below) are chosen automatically in such a way that if the mean of `x` is constant (i.e. if there are no change-points), the probability of no detection (i.e. `est` being constant) is approximately `lambda`. When `universal` is `TRUE`, then `M=1000` for longer signals and `M<1000` for shorter signals to avoid `th.const` being larger than `1.3`, which empirically appears to be too high a value. If `universal` is `FALSE`, then both `M` and `th.const` must be specified.

`M` The number of randomly selected sub-segments of the data on which to build the CUSUM statistics in the (Adaptive) Wild Binary Segmentation algorithm. If you are using Adaptive Wild Binary Segmentation (`adapt=TRUE`) and do not wish to set `universal` to `TRUE` (and therefore have `M` chosen for you), try `M=1000`. If you are using standard Wild Binary Segmentation (`adapt=TRUE`), try `M=20000` or higher.

`th.const` Tuning parameter. Change-points are estimated by thresholding [of the (Adaptive) WBS CUSUMs of `x`] in which the threshold has magnitude `th.const * sqrt(2 * log(n)) * sigm` where `n` is the length of `x`. There is an extra twist if `adapt=TRUE`, see `th.const.min.mult` below.

`th.const.min.mult`
If `adapt=TRUE`, then the threshold gradually decreases in each recursive pass through the data, but in such a way that in never goes below `th.const.min.mult * th.const * sqrt(2`

`adapt` If `TRUE` (respectively, `FALSE`), then Adaptive (respectively, standard) Wild Binary Segmentation is used.

`lambda` See the description for the `universal` parameter above. Currently, the only permitted values are `0.9` and `0.95`.

Details

The change-point detection algorithms used in `wbs.thresh.cpt` are: standard Wild Binary Segmentation [see "Wild Binary Segmentation for multiple change-point detection", P. Fryzlewicz (2014), Annals of Statistics, 42, 2243-2281] and Adaptive Wild Binary Segmentation [see "Data-adaptive Wild Binary Segmentation", P. Fryzlewicz (2017), in preparation as of September 28th, 2017].

Value

A list with the following components:

`est` The estimated piecewise-constant mean of `x`.

`no.of.cpt` The estimated number of change-points in the piecewise-constant mean of `x`.

`cpt` The estimated locations of change-points in the piecewise-contant mean of `x` (these are the final indices *before* the location of each change-point).

Author(s)

Piotr Fryzlewicz, <p.fryzlewicz@lse.ac.uk>

See Also

segment.mean, wbs.bic.cpt, wbs.cpt, tguh.cpt, hybrid.cpt, wbs.K.cpt

Examples

```
teeth <- rep(rep(0:1, each=5), 20)
teeth.noisy <- teeth + rnorm(200)/5
teeth.cleaned <- wbs.thresh.cpt(teeth.noisy)
ts.plot(teeth.cleaned$est)
teeth.cleaned$no.of.cpt
teeth.cleaned$cpt
```

Index