Package 'breakfast'

May 26, 2017

Title Multiple Change-Point Detection and Segmentation

Version 0.1.0

Description Performs multiple change-point detection in data sequences, or data sequence segmentation, using computationally efficient multiscale methods. This version only implements the ``Tail-Greedy Unbalanced Haar'' change-point detection methodology; more methods will be added in future versions. 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 2017-05-26 13:12:07 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 only implements the "Tail-Greedy Unbalanced Haar" change-point detection and segmentation methodology; more methods will be added in future versions of the package. 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.

See Also

`segment.mean`

Examples

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

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 user-specified method. 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, method = "tguh", 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.
method	Specifies the method to be used for change-point detection. In this version of the package, the only option (and the default) is "tguh": the Tail-Greedy Unbalanced Haar method, see Details for the relevant reference.
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. If method=="tguh", then 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	Only relevant if method=="tguh". 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	Only relevant if method=="tguh". 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

If method=="tguh", then the change-point detection algorithm used 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

tguh.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`, `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)
```

Index