Package ‘breakfast’

October 18, 2022

Title Methods for Fast Multiple Change-Point Detection and Estimation
Version 2.3
Description A developing software suite for multiple change-point detection/estimation (data segmentation) in data sequences.
Depends R (>= 3.0.0)
License GPL-2
Imports plyr, Rcpp, ggplot2
LinkingTo Rcpp
Encoding UTF-8
Suggests testthat, knitr, rmarkdown
VignetteBuilder knitr
RoxygenNote 7.1.1
NeedsCompilation yes
Author Andreas Anastasiou [aut],
Yining Chen [aut, cre],
Haeran Cho [aut],
Piotr Fryzlewicz [aut]
Maintainer Yining Chen <y.chen101@lse.ac.uk>
Repository CRAN
Date/Publication 2022-10-18 13:45:08 UTC

R topics documented:

breakfast-package .................................................. 2
breakfast ............................................................. 3
model.ic ............................................................... 5
model.lp ............................................................... 6
model.sdll ............................................................ 8
model.thresh .......................................................... 10
plot.breakfast.cpts ............................................... 11
print.breakfast.cpts .............................................. 12
**breakfast-package**

---

**Description**

A developing software suite for multiple change-point detection/estimation (data segmentation) in data sequences.

**Details**

The current version implements the Gaussian mean-shift model, in which the data are assumed to be a piecewise-constant signal observed with i.i.d. Gaussian noise. Change-point detection in breakfast is carried out in two stages: (i) computation of a solution path, and (ii) model selection along the path. A variety of solution path and model selection methods are included, which can be accessed individually, or through breakfast. Currently supported solution path methods are: `sol.idetect`, `sol.idetect_seq`, `sol.wbs`, `sol.wbs2`, `sol.not` and `sol.tguh`.

Currently supported model selection methods are: `model.ic`, `model.lp`, `model.sdll` `model.thresh`.

Check back future versions for more change-point models and further methods.

**Author(s)**

- Andreas Anastasiou
- Yining Chen
- Haeran Cho
- Piotr Fryzlewicz

We would like to thank Shakeel Gavioli-Akilagun, Anica Kostic, Shuhan Yang and Christine Yuen for their comments and suggestions that helped improve this package.

**See Also**

`browseVignettes(package = "breakfast")` contains a detailed comparative simulation study of various methods implemented in breakfast for the Gaussian mean-shift model.
Methods for fast multiple change-point detection and estimation

Description

This function estimates the number and locations of change-points in a data sequence, which is modelled as a piecewise-constant function plus i.i.d. Gaussian noise. This is carried out via a two-stage procedure combining solution path generation and model selection methodologies.

Usage

breakfast(x, solution.path = NULL, model.selection = NULL)

Arguments

x
A numeric vector containing the data to be processed

solution.path
A string or a vector of strings containing the name(s) of solution path generating method(s); if individual methods are accessed via this option, default tuning parameters are used. Alternatively, you can directly access each solution path generating method via sol.[method], see below. If both solution.path and model.selection are unspecified, we return the output from the suggested combinations based on their performance, which are: ("idetect", "ic"), ("idetect_seq", "thresh"), ("not", "ic"), ("tguh", "lp"), ("wbs", "ic") and ("wbs2", "sdll"). If solution.path is specified but model.selection is not, we return the output from the specified solution.path methods combined with the suggested model selection methods (respectively) as above.

• "idetect" IDetect, see sol.idetect
• "idetect_seq" Sequential IDetect, see sol.idetect_seq
• "not" Narrowest-Over-Threshold, see sol.not
• "tguh" Tail-Greedy Unbalanced Haar, see sol.tguh
• "wbs" Wild Binary Segmentation, see sol.wbs
• "wbs2" Wild Binary Segmentation 2, see sol.wbs2
• "all" All of the above

model.selection
A string or a vector of strings containing the name(s) of model selection method(s); if individual methods are accessed via this option, default tuning parameters are used. Alternatively, you can directly access each model selection method via model.[method], see below. If both solution.path and model.selection are unspecified, we return the output from the suggested combinations based on their performance, which are: ("idetect", "ic"), ("idetect_seq", "thresh"), ("not", "ic"), ("tguh", "lp"), ("wbs", "ic") and ("wbs2", "sdll"). If model.selection is specified but solution.path is not, we return the output from the specified model.selection methods combined with the suggested solution path methods (respectively) as above.

• "ic" Strengthened Schwarz information criterion, see model.ic
• "lp" Localised pruning, see `model.lp`
• "sdll" Steepest Drop to Low Levels method, see `model.sdll`
• "thresh" Thresholding, see `model.thresh`
• "all" All of the above

Details

Please also take a look at the vignette for tips/suggestions/examples of using the breakfast package.

Value

An S3 object of class `breakfast.cpts`, which contains the following fields:

• x Input vector \( x \)

• cptmodel.list A list containing S3 objects of class `cptmodel`; each contains the following fields:
  – solution.path The solution path method used
  – model.selection The model selection method used to return the final change-point estimators object
  – no.of.cpt The number of estimated change-points in the piecewise-constant mean of the vector `cptpath.object$x`
  – cpts The locations of estimated change-points in the piecewise-constant mean of the vector `cptpath.object$x`. These are the end-points of the corresponding constant-mean intervals
  – est An estimate of the piecewise-constant mean of the vector `cptpath.object$x`; the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points

References


model.ic

Examples

```r
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f)) * .5
breakfast(x)
```

model.ic       Estimating change-points in the piecewise-constant mean of a noisy data sequence via the strengthened Schwarz information criterion

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of a noisy data sequence via the sSIC (strengthened Schwarz information criterion) method.

Usage

```r
model.ic(cptpath.object, alpha = 1.01, q.max = NULL)
```

Arguments

- `cptpath.object`: A solution-path object, returned by a `sol.[name]` routine. Note that the field `cptpath.object$x` contains the input data sequence.
- `alpha`: The parameter associated with the sSIC. The default value is 1.01. Note that the SIC is recovered when `alpha = 1`.
- `q.max`: The maximum number of change-points allowed. If nothing or `NULL` is provided, the default value of `\min(100, n/\log(n))` (rounded to an integer) will be used.

Details


Value

An S3 object of class `cptmodel`, which contains the following fields:

- `solution.path`: The solution path method used to obtain `cptpath.object`
- `model.selection`: The model selection method used to return the final change-point estimators object, here its value is "ic"
- `no.of.cpt`: The number of estimated change-points in the piecewise-constant mean of the vector `cptpath.object$x`
The locations of estimated change-points in the piecewise-constant mean of the vector \texttt{cptpath.object$x}. These are the end-points of the corresponding constant-mean intervals.

An estimate of the piecewise-constant mean of the vector \texttt{cptpath.object$x}; the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points.

References


See Also

\texttt{sol.idetect}, \texttt{sol.not}, \texttt{sol.tguh}, \texttt{sol.wbs}, \texttt{sol.wbs2}, \texttt{breakfast}

Examples

```r
x <- c(rep(0, 100), rep(1, 100), rep(0, 100)) + rnorm(300)
model.ic(sol.wbs(x))
model.ic(sol.not(x))
```

---

**Description**

This function estimates the number and locations of change-points in the piecewise-constant mean of a noisy data sequence via the localised pruning method, which performs a Schwarz criterion-based model selection on the given candidate set in a localised way.

**Usage**

```r
model.lp(
  cptpath.object,
  min.d = 5,
  penalty = c("log", "polynomial"),
  pen.exp = 1.01,
  do.thr = TRUE,
  th.const = 0.5
)
```
Arguments

cptpath.object A solution-path object, returned by a sol.[name] routine. Note that the field
cptpath.object$x contains the input data sequence.

min.d A number specifying the minimal spacing between change points; min.d = 5 by
default

penalty A string specifying the type of penalty term to be used in Schwarz criterion;
possible values are:
• "log" Use penalty = log(length(x))^pen.exp
• "polynomial" Use penalty = length(x)^pen.exp

pen.exp Exponent for the penalty term (see penalty)

do.thr If do.thr = TRUE, mild thresholding on the CUSUM test statistics is performed
after internal standardisation step in order to "pre-prune down" the candidates

th.const A constant multiplied to sqrt(2*log(length(x))) to form a mild threshold;
if not supplied, a default value (0.5* a value suggested in Fryzlewicz (2020) is
used, see th.const in model.sdll

Details

Further information can be found in Cho and Kirch (2021).

Value

An S3 object of class cptmodel, which contains the following fields:

solution.path The solution path method used to obtain cptpath.object

model.selection The model selection method used to return the final change-point estimators
object, here its value is "lp"

no.of.cpt The number of estimated change-points in the piecewise-constant mean of the
vector cptpath.object$x

cpts The locations of estimated change-points in the piecewise-constant mean of
the vector cptpath.object$x. These are the end-points of the corresponding
constant-mean intervals

est An estimate of the piecewise-constant mean of the vector cptpath.object$x;
the values are the sample means of the data (replicated a suitable number of
times) between each pair of consecutive detected change-points

References

H. Cho & C. Kirch (2021) Two-stage data segmentation permitting multiscale change points, heavy

See Also

sol.idetect, sol.idetect_seq, sol.not, sol.tguh, sol.wbs, sol.wbs2, breakfast
Examples

```r
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f)) * .5
model.lp(sol.not(x))
```

---

**model.sdll**

*Estimating change-points in the piecewise-constant mean of a noisy data sequence via the Steepest Drop to Low Levels method*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of a noisy data sequence via the Steepest Drop to Low Levels method.

Usage

```r
model.sdll(
  cptpath.object,
  sigma = stats::mad(diff(cptpath.object$x)/sqrt(2)),
  universal = TRUE,
  th.const = NULL,
  th.const.min.mult = 0.3,
  lambda = 0.9
)
```

Arguments

- **cptpath.object** A solution-path object, returned by a `sol.[name]` routine. In particular, SDLL model selection should work well when `cptpath.object` is an object returned by the `sol.wbs2` routine. Note that the field `cptpath.object$x` contains the input data sequence.

- **sigma** An estimate of the standard deviation of the noise in the data `cptpath.object$x`. Can be a functional of `cptpath.object$x` or a specific value if known. The default is the Median Absolute Deviation of the vector `diff(cptpath.object$x)/sqrt(2)`, tuned to the Gaussian distribution. Note that `model.sdll` works particularly well when the noise is i.i.d. Gaussian.

- **universal** If TRUE, then the threshold that decides if there are any change-points is chosen automatically, so that the probability of type-I error (i.e. indicating change-points if there are none) is approximately $1 - \alpha$ when the number $M$ of intervals drawn in the `sol.wbs2` solution path routine is 1000. If FALSE, then `th.const` must be specified.

- **th.const** Only relevant if `universal` == FALSE; in that case a numerical value must be provided. Used to create the threshold (applicable to the CUSUM magnitudes stored in `cptpath.object`) that decides if there are any change-points in the mean vector; that threshold is then `th.const * sqrt(2 * log(n)) * sigma`, where $n$ is the length of the data vector `cptpath.object$x`. 


th.const.min.mult
A fractional multiple of the threshold, used to decide the lowest magnitude of CUSUMs from \texttt{cptpath.object} still considered by the SDLL model selection criterion as potentially change-point-carrying.

\texttt{lambda}
Only relevant if \texttt{universal == TRUE}; can be set to 0.9 or 0.95. The approximate probability of not detecting any change-points if the truth does not contain any, when the number \(M\) of intervals drawn in the \texttt{sol.wbs2} solution path routine is 1000.

Details

Value
An S3 object of class \texttt{cptmodel}, which contains the following fields:

\texttt{solution.path}
The solution path method used to obtain \texttt{cptpath.object}

\texttt{model.selection}
The model selection method used to return the final change-point estimators \texttt{object}, here its value is "sdll"

\texttt{no.of.cpt}
The number of estimated change-points in the piecewise-constant mean of the vector \texttt{cptpath.object}$x$

\texttt{cpts}
The locations of estimated change-points in the piecewise-constant mean of the vector \texttt{cptpath.object}$x$. These are the end-points of the corresponding constant-mean intervals

\texttt{est}
An estimate of the piecewise-constant mean of the vector \texttt{cptpath.object}$x$; the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points

References

See Also
\texttt{sol.idetect}, \texttt{sol.idetect_seq}, \texttt{sol.not}, \texttt{sol.tguh}, \texttt{sol.wbs}, \texttt{sol.wbs2}, \texttt{breakfast}

Examples
\begin{verbatim}
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f))
model.sdll(sol.wbs2(x))
\end{verbatim}
model.thresh

Estimating change-points in the piecewise-constant mean of a noisy data sequence via thresholding

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of a noisy data sequence via thresholding.

Usage

model.thresh(
  cptpath.object,
  sigma = stats::mad(diff(cptpath.object$x)/sqrt(2)),
  th_const = 1.15
)

Arguments

cptpath.object A solution-path object, returned by a sol.[name] routine. Note that the field sols.object$x contains the input data sequence.

sigma An estimate of the standard deviation of the noise in the data cptpath.object$x. Can be a functional of cptpath.object$x or a specific value if known. The default is the Median Absolute Deviation of the vector diff(cptpath.object$x)/sqrt(2), tuned to the Gaussian distribution. Note that model.thresh works particularly well when the noise is i.i.d. Gaussian.

th_const A positive real number with default value equal to 1. It is used to define the threshold for the detection process.

Value

An S3 object of class cptmodel, which contains the following fields:

solution.path The solution path method used to obtain cptpath.object

model.selection The model selection method used to return the final change-point estimators object, here its value is "thresh"

no.of.cpt The number of estimated change-points in the piecewise-constant mean of the vector cptpath.object$x

cpts The locations of estimated change-points in the piecewise-constant mean of the vector cptpath.object$x. These are the end-points of the corresponding constant-mean intervals

est An estimate of the piecewise-constant mean of the vector cptpath.object$x; the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points
See Also

sol.idetect_seq, sol.idetect_seq, sol.not, sol.tguh, sol.wbs, sol.wbs2, breakfast

Examples

```r
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f))
model.thresh(sol.idetect_seq(x))
```

Description

Plot method for objects of class `breakfast.cpts`

Usage

```r
## S3 method for class 'breakfast.cpts'
plot(x, display.data = TRUE, ...)
```

Arguments

- `x`: a `breakfast.cpts` object
- `display.data`: if `display.data = TRUE`, change-point estimators are plotted against the data by method. If `display.data = FALSE`, only the estimators are plotted; this option is recommended when `length(x)` is large.
- `...`: current not in use

Examples

```r
f <- rep(rep(c(0, 1), each = 50), 5)
x <- f + rnorm(length(f)) * .5
plot(breakfast(x, solution.path = 'all', model.selection = 'all'), display.data = TRUE)
plot(breakfast(x), display.data = FALSE)
```
print.breakfast.cpts  Change-points estimated by breakfast

Description

Print method for objects of class breakfast.cpts

Usage

```r
## S3 method for class 'breakfast.cpts'
print(x, by = c("method", "estimator"), ...)
```

Arguments

- `x`: a breakfast.cpts object
- `by`: if `by = 'method'`, change-point estimators are printed by method; if `by = 'estimator'`, each change-point estimator is printed with the methods that detect it.
- `...`: current not in use

Examples

```r
f <- rep(rep(c(0, 1), each = 50), 5)
x <- f + rnorm(length(f)) * .5
print(breakfast(x, solution.path = 'all', model.selection = 'all'), by = 'method')
print(breakfast(x), by = 'estimator')
```

print.cptmodel Change-points estimated by solution path generation + model selection methods

Description

Print method for objects of class cptmodel

Usage

```r
## S3 method for class 'cptmodel'
print(x, ...)  
```

Arguments

- `x`: a cptmodel object
- `...`: current not in use
Examples

```r
f <- rep(rep(c(0, 1), each = 50), 5)
x <- f + rnorm(length(f)) * .5
print(model.ic(sol.idetect(x)))
```

Description

This function arranges all possible change-points in the mean of the input vector in the order of importance, via the Isolate-Detect (ID) method. It is developed to be used with the sdll and information criterion (ic) model selection rules.

Usage

```r
sol.idetect(x, thr_ic = 0.9, points = 3)
```

Arguments

- **x**: A numeric vector containing the data to be processed.
- **thr_ic**: A positive real number with default value equal to 0.9. It is used to create the solution path. The lower the value, the larger the solution path vector.
- **points**: A positive integer with default value equal to 3. It defines the distance between two consecutive end- or start-points of the right- or left-expanding intervals, as described in the Isolate-Detect methodology.

Details


Value

An S3 object of class cptpath, which contains the following fields:

- **solutions.nested**: TRUE, i.e., the change-point outputs are nested
- **solution.path**: Locations of possible change-points in the mean of x, arranged in decreasing order of change-point importance
- **solution.set**: Empty list
- **x**: Input vector x
cands

Matrix of dimensions length(x) - 1 by 4. The first two columns are (start, end)-
points of the detection intervals of the corresponding possible change-point lo-
cation in the third column. The fourth column is a measure of strength of the
corresponding possible change-point. The order of the rows is the same as the
order returned in solution.path

method

The method used, which has value "idetect" here

References

A. Anastasiou & P. Fryzlewicz (2021). Detecting multiple generalized change-points by isolating

See Also

sol.idetect_seq, sol.not, sol.wbs, sol.wbs2, sol.tguh,

Examples

```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.idetect(r3)
```

---

**sol.idetect_seq**

Solution path generation using the sequential approach of the Isolate-
Detect method

Description

This function uses the Isolate-Detect method in its original sequential way in order to create the
solution path. It is developed to be used with the thresholding model selection rule.

Usage

```r
sol.idetect_seq(x, points = 3)
```

Arguments

- **x**
  A numeric vector containing the data to be processed

- **points**
  A positive integer with default value equal to 3. It defines the distance between
two consecutive end- or start-points of the right- or left-expanding intervals, as
described in the Isolate-Detect methodology.

Details

The Isolate-Detect method and its algorithm is described in "Detecting multiple generalized change-
points by isolating single ones", A. Anastasiou & P. Fryzlewicz (2021), Metrika, https://doi.org/10.1007/s00184-
021-00821-6.
sol.not

Value

An S3 object of class cptpath, which contains the following fields:

- solutions.nested: TRUE, i.e., the change-point outputs are nested
- solution.path: Locations of possible change-points in the mean of \( x \), arranged in decreasing order of change-point importance
- solution.set: Empty list
- x: Input vector \( x \)
- cands: Matrix of dimensions \( \text{length}(x) - 1 \) by 4. The first two columns are (start, end)-points of the detection intervals of the corresponding possible change-point location in the third column. The fourth column is a measure of strength of the corresponding possible change-point. The order of the rows is the same as the order returned in solution.path
- method: The method used, which has value "idetect_seq" here

References


See Also

sol.idetect, sol.not, sol.wbs, sol.wbs2, sol.tguh.

Examples

```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.idetect_seq(r3)
```

Description

This function arranges all possible change-points in the mean of the input vector in the order of importance, via the Narrowest-Over-Threshold (NOT) method.

Usage

```r
sol.not(x, M = 10000, systematic.intervals = TRUE, seed = NULL)
```
Arguments

- **x**: A numeric vector containing the data to be processed.
- **M**: The maximum number of all data sub-samples at the beginning of the algorithm. The default is $M = 10000$.
- **systematic.intervals**: When drawing the sub-intervals, whether to use a systematic (and fixed) or random scheme. The default is `systematic.intervals = TRUE`.
- **seed**: If a random scheme is used, a random seed can be provided so that every time the same sets of random sub-intervals would be drawn. The default is `seed = NULL`, which means that this option is not taken.

Details


Value

An S3 object of class `cptpath`, which contains the following fields:

- **solutions.nested**: FALSE, i.e., the change-point outputs are not nested.
- **solution.path**: Empty list.
- **solution.set**: Locations of possible change-points in the mean of $x$ for each threshold level (in the decreasing order), arranged in the form of a list of lists.
- **solution.set.th**: A list that contains threshold levels corresponding to the detections in `solution.set`.
- **x**: Input vector $x$.
- **M**: Input parameter $M$.
- **cands**: Matrix of dimensions `length(x) - 1` by 4. The first two columns are (start, end)-points of the detection intervals of the corresponding possible change-point location in the third column resulted from applying NOT to all threshold levels. The fourth column is a measure of strength of the corresponding possible change-point. The order of the rows reflect the strength of each detection in decreasing order. To avoid repetition, each possible location would appear at most once in the matrix (with the sub-interval that carries its highest possible strength).
- **method**: The method used, which has value "not" here.

References

See Also

sol.idetect, sol.tguh, sol.wbs, sol.wbs2

Examples

```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.not(r3)
```

Description

This function arranges all possible change-points in the mean of the input vector in the order of importance, via the Tail-Greedy Unbalanced Haar method.

Usage

```r
sol.tguh(x, p = 0.01)
```

Arguments

- `x`: A numeric vector containing the data to be processed.
- `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 `p = 0.01`.

Details


Value

An S3 object of class cptpath, which contains the following fields:

- `solutions.nested`: TRUE, i.e., the change-point outputs are nested.
- `solution.path`: Locations of possible change-points in the mean of `x`, arranged in decreasing order of change-point importance.
- `solution.set`: Empty list.
- `x`: Input vector `x`.
- `M`: Input parameter `M`. 

Matrix of dimensions length(x) - 1 by 4. The first two columns are (start, end)-
points of the detection intervals of the corresponding possible change-point lo-
cation in the third column. The fourth column is a measure of strength of the
corresponding possible change-point. The order of the rows is the same as the
order returned in solution.path

The method used, which has value "tguh" here

References

P. Fryzlewicz (2018). Tail-greedy bottom-up data decompositions and fast multiple change-point

See Also

`sol.idetect, sol.idetect_seq, sol.not, sol.wbs, sol.wbs2`

Examples

```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.tguh(r3)
```

---

### sol.wbs

*Solution path generation via the Wild Binary Segmentation method*

This function arranges all possible change-points in the mean of the input vector in the order of
importance, via the Wild Binary Segmentation (WBS) method.

#### Usage

```r
sol.wbs(x, M = 10000, systematic.intervals = TRUE, seed = NULL)
```

#### Arguments

- **x**: A numeric vector containing the data to be processed
- **M**: The maximum number of all data sub-samples at the beginning of the algorithm. The default is `M = 10000`
- **systematic.intervals**: When drawing the sub-intervals, whether to use a systematic (and fixed) or ran-
dom scheme. The default is `systematic.intervals = TRUE`
- **seed**: If a random scheme is used, a random seed can be provided so that every time the same sets of random sub-intervals would be drawn. The default is `seed = NULL`, which means that this option is not taken
Details

Value
An S3 object of class cptpath, which contains the following fields:
- solutions.nested: TRUE, i.e., the change-point outputs are nested
- solution.path: Locations of possible change-points in the mean of x, arranged in decreasing order of change-point importance
- solution.set: Empty list
- x: Input vector x
- M: Input parameter M
- cands: Matrix of dimensions length(x) - 1 by 4. The first two columns are (start, end)-points of the detection intervals of the corresponding possible change-point location in the third column. The fourth column is a measure of strength of the corresponding possible change-point. The order of the rows is the same as the order returned in solution.path
- method: The method used, which has value "wbs" here

References

See Also
sol.idetect, sol.not, sol.tguh, sol.wbs2

Examples
```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.wbs(r3)
```

Description
This function arranges all possible change-points in the mean of the input vector in the order of importance, via the Wild Binary Segmentation 2 method.

Usage
```r
sol.wbs2(x, M = 1000, systematic.intervals = TRUE)
```
Arguments

x    A numeric vector containing the data to be processed.
M    The maximum number of data sub-samples drawn at each recursive stage of the algorithm. The default is \(M = 1000\). Setting \(M = 0\) executes the standard binary segmentation.

systematic.intervals

Whether data sub-intervals for CUSUM computation are drawn systematically (TRUE; start- and end-points taken from an approximately equispaced grid) or randomly (FALSE; obtained uniformly with replacement). The default is TRUE.

Details


Value

An S3 object of class cptpath, which contains the following fields:

- **solutions.nested**
  - TRUE, i.e., the change-point outputs are nested

- **fmax**

- **solution.path**
  - Locations of possible change-points in the mean of \(x\), arranged in decreasing order of change-point importance

- **solution.set**
  - Empty list

- **x**
  - Input vector \(x\)

- **M**
  - Input parameter \(M\)

- **cands**
  - Matrix of dimensions length(\(x\)) - 1 by 4. The first two columns are (start, end)-points of the detection intervals of the corresponding possible change-point location in the third column. The fourth column is a measure of strength of the corresponding possible change-point. The order of the rows is the same as the order returned in solution.path

- **method**
  - The method used, which has value "wbs2" here

References


See Also

sol.idetect, sol.idetect_seq, sol.not, sol.tguh, sol.wbs

Examples

```r
r3 <- rnorm(1000) + c(rep(0,300), rep(2,200), rep(-4,300), rep(0,200))
sol.wbs2(r3)
```
Index

breakfast, 2, 3, 6, 7, 9, 11
breakfast-package, 2

model.ic, 2, 3, 5
model.lp, 2, 4, 6
model.sdll, 2, 4, 7, 8
model.thresh, 2, 4, 10

plot.breakfast.cpts, 11
print.breakfast.cpts, 12
print.cptmodel, 12

sol.idetect, 2, 3, 6, 7, 9, 13, 15, 17–20
sol.idetect_seq, 2, 3, 7, 9, 11, 14, 14, 18, 20
sol.not, 2, 3, 6, 7, 9, 11, 14, 15, 15, 18–20
sol.tguh, 2, 3, 6, 7, 9, 11, 14, 15, 17, 17, 19, 20
sol.wbs, 2, 3, 6, 7, 9, 11, 14, 15, 17, 18, 18, 20
sol.wbs2, 2, 3, 6, 7, 9, 11, 14, 15, 17–19, 19