Package ‘breakfast’

November 23, 2020

Title Methods for Fast Multiple Change-Point Detection and Estimation

Version 2.1

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

Depends R (>= 3.0.0)

LazyData true

License GPL-2

Imports plyr, Rcpp

LinkingTo Rcpp

Encoding UTF-8

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.0

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 2020-11-23 05:50:06 UTC

R topics documented:

`breakfast-package` *Breakfast: Methods for Fast Multiple Change-point Detection and Estimation*

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.

breakfast *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,
  sd.fn = NULL,
  option = c("suggested", "user", "all")[1],
  solution.path = list(name = "idetect", args = list()),
  model.selection = list(name = "ic", args = list())
)
```

Arguments

`x` A numeric vector containing the data to be processed

`sd.fn` Either a positive numeric constant containing the square root of the error variance, or a function for generating this quantity that takes `x` as an input; if `sd.fn = NULL`, a mean absolute deviation estimator is generated internally.

`option` A string specifying the option for selecting the combination of solution path and model selection procedures. The default choice is `option = "suggested"` with combinations of solution path and model selection methods suggested for their good properties.

- "suggested" Suggested combinations based on their performance, which are `("wbs2","sdll")`, `("idetect","ic")`, `("idetect_seq","thresh")` and `("tguh","lp")`
- "user" User-specified combination (to be supplied by `solution.path` and `model.selection`)
- "all" All currently supported combinations; we exclude the following combinations of the solution path and model selection methods based on their performance: `("not","sdll")`, `("idetect","thresh")`, `("idetect","lp")`, `("idetect_seq","ic")`, `("idetect_seq","lp")`, `("idetect_seq","sdll")`

`solution.path` A list containing the name of a solution path generating method and any arguments to be used with them; currently supported methods are

- "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

`model.selection`
A list of containing the name of a model selection method and any arguments to be used with them; currently supported methods are
- "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

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 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

A. Anastasiou & P. Fryzlewicz (2019). Detecting multiple generalized change-points by isolating single ones. *arXiv preprint arXiv:1901.10852*.

R. Baranowski, Y. Chen & P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society: Series B*, 81(3), 649–672.

H. Cho & C. Kirch (2020) Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. *arXiv preprint arXiv:1910.12486*.

P. Fryzlewicz (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6), 2243–2281.

P. Fryzlewicz (2020). Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. *To appear in Journal of the Korean Statistical Society*.

P. Fryzlewicz (2018). Tail-greedy bottom-up data decompositions and fast multiple change-point detection. *The Annals of Statistics*, 46(6B), 3390–3421.

Examples

```
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f))
breakfast(x, option = 'suggested')
```

`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

```
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 around `min(25,n/log(n))` will be used.

Details

The model selection method for algorithms that produce nested solution path is described in "Wild binary segmentation for multiple change-point detection", P. Fryzlewicz (2014), The Annals of Statitics, 42: 2243–2281. The corresponding description for those that produce non-nested solution set can be found in "Narrowest-over-threshold detection of multiple change points and change-point-like features", R. Baranowski, Y. Chen and P. Fryzlewicz (2019), Journal of Royal Statistical Society: Series B, 81(3), 649–672.

Value

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

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

`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

P. Fryzlewicz (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6), 2243–2281.

R. Baranowski, Y. Chen & P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society: Series B*, 81(3), 649–672.

See Also

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

Examples

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

`model.lp` *Estimating change-points in the piecewise-constant mean of a noisy data sequence via the localised pruning*

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

```
model.lp(
  cptpath.object,
  min.d = 5,
  penalty = c("log", "polynomial")[1],
  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 threshoding 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 (2020), arXiv preprint arXiv:1910.12486.

Value

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

solution.path	The solution path method used to obtain cptpath.object
model	The model selection method used to return the final change-point estimators object, here its value is "ip"
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 (2020) Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. *arXiv preprint arXiv:1910.12486*.

See Also

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

Examples

```
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f))
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

```
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 100. 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 `cptpath.object` still considered by the SDLL model selection criterion as potentially change-point-carrying.

`lambda`
: Only relevant if `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 `sol.wbs2` solution path routine is 100.

Details

The Steepest Drop to Low Levels method is described in "Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection", P. Fryzlewicz (2020), Journal of the Korean Statistical Society, to appear.

Value

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

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

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

`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

P. Fryzlewicz (2020). Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society (to appear).*

See Also

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

Examples

```
f <- rep(rep(c(0, 1), each = 50), 10)
x <- f + rnorm(length(f))
model.sdll(sol.wbs2(x))
```

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	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

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

---

print.breakfast.cpts *Change-points estimated by breakfast*

Description

Print method for objects of class `breakfast.cpts`

Usage

```
## S3 method for class 'breakfast.cpts'
print(x, ...)
```

Arguments

x	a `breakfast.cpts` object
...	not in use

Examples

```
f <- rep(rep(c(0, 1), each = 50), 5)
x <- f + rnorm(length(f))
print(breakfast(x, option = 'user'))
```

---

sol.idetect *Solution path generation via the Isolate-Detect method*

Description

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

Usage

```
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

The Isolate-Detect method and its algorithm is described in "Detecting multiple generalized change-points by isolating single ones", A. Anastasiou & P. Fryzlewicz (2019), arXiv preprint arXiv:1901.10852.

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 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" here

References

A. Anastasiou & P. Fryzlewicz (2019). Detecting multiple generalized change-points by isolating single ones. *arXiv preprint arXiv:1901.10852*.

See Also

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

Examples

```
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

```
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 (2019), arXiv preprint arXiv:1901.10852.

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 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

A. Anastasiou & P. Fryzlewicz (2019). Detecting multiple generalized change-points by isolating single ones. *arXiv preprint arXiv:1901.10852*.

See Also

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

Examples

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

---

sol.not — *Solution path generation via the Narrowest-Over-Threshold method*

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

```
sol.not(x, M = 10000, rand.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
rand.intervals	When drawing the sub-intervals, whether to use a random or a fixed scheme. The default is rand.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

The Narrowest-Over-Threshold method and its algorithm is described in "Narrowest-over-threshold detection of multiple change points and change-point-like features", R. Baranowski, Y. Chen and P. Fryzlewicz (2019), Journal of Royal Statistical Society: Series B, 81(3), 649–672.

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

R. Baranowski, Y. Chen & P. Fryzlewicz (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society: Series B*, 81(3), 649–672.

See Also

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

Examples

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

---

sol.tguh *Solution path generation via the Tail-Greedy Unbalanced Haar method*

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

```
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

The Tail-Greedy Unbalanced Haar decomposition algorithm is described in "Tail-greedy bottom-up data decompositions and fast multiple change-point detection", P. Fryzlewicz (2018), The Annals of Statistics, 46: 3390–3421.

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 "tguh" here

References

P. Fryzlewicz (2018). Tail-greedy bottom-up data decompositions and fast multiple change-point detection. *The Annals of Statistics*, 46(6B), 3390–3421.

See Also

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

Examples

```
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*

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 (WBS) method.

Usage

```
sol.wbs(x, M = 10000, rand.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
rand.intervals	When drawing the sub-intervals, whether to use a random or a fixed scheme. The default is rand.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

The Wild Binary Segmentation algorithm is described in "Wild binary segmentation for multiple change-point detection", P. Fryzlewicz (2014), The Annals of Statistics, 42: 2243–2281.

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

P. Fryzlewicz (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6), 2243–2281.

See Also

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

Examples

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

`sol.wbs2` *Solution path generation via the Wild Binary Segmentation 2 method*

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

```
sol.wbs2(x, M = 100, cusums = "systematic")
```

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 = 100`. Setting `M = 0` executes the standard binary segmentation.
`cusums`	Whether data sub-intervals for CUSUM computation are drawn systematically (start- and end-points taken from an approximately equispaced grid) or randomly (obtained uniformly with replacement). Use "systematic" (default) or "random".

Details

The Wild Binary Segmentation 2 algorithm is described in "Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection", P. Fryzlewicz (2020), Journal of the Korean Statistical Society, to appear.

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 "wbs2" here

References

P. Fryzlewicz (2020). Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society (to appear)*.

See Also

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

Examples

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

Index