Package ‘breakfast’

April 26, 2024

Title Methods for Fast Multiple Change-Point/Break-Point Detection and Estimation

Version 2.4

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

Depends R (>= 3.0.0)

License GPL-2

Imports plyr, Rcpp, ggplot2, splines

LinkingTo Rcpp

Encoding UTF-8

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

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 2024-04-26 15:10:02 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 methods for detecting changes in the data sequence modelled as (i) a piecewise-constant function plus i.i.d. Gaussian noise, (ii) a piecewise-constant function plus autoregressive time series, (iii) a piecewise-linear and continuous function plus i.i.d. Gaussian noise, and (iv) a piecewise-linear and discontinuous function plus i.i.d. Gaussian noise. This is carried out via a two-stage procedure combining solution path generation and model selection methodologies. Change-point detection in breakfast is carried out in two stages, first the computation of a solution path, followed by a model selection step 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, sol.tguh and sol.wcm.

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

Check back future versions for more change-point models and the corresponding 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 models (i), (iii) and (iv).

`breakfast` *Methods for fast detection of multiple change-points*

Description

This function estimates the number and locations of change-points in a univariate data sequence, which is modelled as (i) a piecewise-constant function plus i.i.d. Gaussian noise, (ii) a piecewise-constant function plus autoregressive time series, (iii) a piecewise-linear and continuous function plus i.i.d. Gaussian noise, or (iv) a piecewise-linear and discontinuous 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,
  type = c("const", "lin.cont", "lin.discont"),
  solution.path = NULL,
  model.selection = NULL
)
```

Arguments

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

`type` The type of change-point models fitted to the data; currently supported models are: piecewise constant signals (`type = "const"`, chosen by default), piecewise linear and continuous signals (`type = "lin.cont"`) and piecewise linear and discontinuous signals (`type = "lin.discont"`).

`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]`. If both `solution.path` and `model.selection` are unspecified, we return the output from the suggested combinations based on their performance, which depends on `type` as below:

When `type = "const"`: `("idetect", "ic")`, `("idetect_seq", "thresh")`, `("not", "ic")`, `("tguh", "lp")`, `("wbs", "ic")`, `("wbs2", "sdll")` and `("wcm", "gsa")`.

When `type = "lin.cont"` or `type = "lin.discont"`: `("idetect_seq", "thresh")`, `("not", "ic")` and `("idetect", "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, supporting `type = "const", "lin.cont", "lin.discont"`, see sol.idetect

"idetect_seq" Sequential IDetect, supporting `type = "const", "lin.cont", "lin.discont"`, see sol.idetect_seq

"not" Narrowest-Over-Threshold, supporting `type = "const", "lin.cont", "lin.discont"`, see sol.not

"tguh" Tail-Greedy Unbalanced Haar, supporting `type = "const"`, see sol.tguh

"wbs" Wild Binary Segmentation, supporting `type = "const"`, see sol.wbs

"wbs2" Wild Binary Segmentation 2, supporting `type = "const"`, see sol.wbs2

"wcm" Wild Contrast Maximisation, supporting `type = "const"` in combination with model.gsa handling model (ii), see sol.wcm

"all" All of the above that support the `type`

`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]`. If both `solution.path` and `model.selection` are unspecified, we return the output from the suggested combinations based on their performance, see `solution.path`. 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). Not all `solution.path` methods are supported by all `model.selection` methods; check the individual functions for more information.

"ic" Strengthened Schwarz information criterion, supporting `type = "const", "lin.cont", "lin.discont"`, see model.ic

"lp" Localised pruning, supporting `type = "const"`, see model.lp

"sdll" Steepest Drop to Low Levels method, supporting `type = "const", "lin.cont", "lin.discont"`, see model.sdll

"thresh" Thresholding, supporting `type = "const", "lin.cont", "lin.discont"`, see model.thresh

"gsa" gappy Schwarz algorithm, supporting `type = "const"` in combination with sol.wcm handling model (ii), see model.gsa

"all" All of the above that support the given `type`

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

A. Anastasiou & P. Fryzlewicz (2022) Detecting multiple generalized change-points by isolating single ones. *Metrika*, 85(2), 141–174.

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 (2022) Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. *Annals of the Institute of Statistical Mathematics*, 74(4), 653–684.

H. Cho & P. Fryzlewicz (2024) Multiple change point detection under serial dependence: Wild contrast maximisation and gappy Schwarz algorithm. *Journal of Time Series Analysis*, 45(3): 479–494.

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

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

P. Fryzlewicz (2020) Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 49(4), 1027–1070.

Examples

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

`model.fixednum` *Estimate the location of change-points when the number of them is fixed*

Description

Return a solution with the given number of change-points or change-point-type features from the solution path

Usage

```
model.fixednum(cptpath.object, fixednum = 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.

fixednum
: The number of change-points or change-point-type features

Details

The model selection method which returns results with a given number of change-points or change-point-type features. If there are multiple such elements on the solution path, the one with the smaller residual sum of squares will be returned. On the other hand, if no such element exists, an empty set (i.e. with no change-points) will be returned.

Value

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

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

type
: The model type used, inherited from the given `cptpath.object`

model.selection
: The model selection method used to return the final change-point or change-point-type feature estimators object, here its value is `"ic"`

no.of.cpt
: The number of estimated features in the mean of the vector `cptpath.object$x` based on the given `type` of the model

cpts
: The locations of estimated features in the mean of the vector `cptpath.object$x`. These are the end-points of the corresponding constant-mean or constant-slope intervals

est
: An estimate of the mean of the vector `cptpath.object$x`; for piecewise-constant signals, the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points; for piecewise-linear but discontinuous signals, the values are the estimated linear trend (replicated a suitable number of times) between each pair of consecutive detected change of slopes; for piecewise-linear and continuous signals, it is similar to the previous case but with the continuity constraint enforced, which envolves solving a global least squares problem.

See Also

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

Examples

```
x <- c(rep(0, 100), rep(1, 100), rep(0, 100)) + rnorm(300)
model.fixednum(sol.wbs(x),2)
model.fixednum(sol.not(x),2)
```

model.gsa — *Estimating change-points in the piecewise-constant mean of a noisy data sequence with auto-regressive noise via gappy Schwarz algorithm*

Description

This function estimates the number and locations of change-points in the piecewise-constant mean of a noisy data sequence with auto-regressive noise via gappy Schwarz algorithm from a candidate model sequence generated by `sol.wcm`.

Usage

```
model.gsa(cptpath.object, p.max = 10, pen = log(length(cptpath.object$x))^1.01)
```

Arguments

`cptpath.object`	A solution-path object, returned by a `sol.wcm` routine. Note that the field `cptpath.object$x` contains the input data sequence.
`p.max`	The maximum AR order. The default is `p.max = 10`.
`pen`	Penalty used for the Schwarz criterion. `log(length(cptpath.object$x))^1.01` is used as default.

Details

From the largest to the smallest (i.e. empty) candidate models generated by `sol.wcm`, gappy Schwarz algorithm locally evaluates the Schwarz criterion (SC, under piecewise constant signal + AR(p) noise model, with the AR order p to be determined adaptively) and its modification SC0 on each segment determined by the next smallest candidate model. It selects the larger model as the final model if over each segment, all newly introduced estimators are deemed 'significant' according to SC and SC0; see Cho and Fryzlewicz (2023) for details.

Value

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

`solution.path`	The solution path method used to obtain `cptpath.object`, here its value is `"wcm"`
`model.selection`	The model selection method used to return the final change-point estimators object, here its value is `"gsa"`
`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 & P. Fryzlewicz (2024) Multiple change point detection under serial dependence: Wild contrast maximisation and gappy Schwarz algorithm. *Journal of Time Series Analysis*, 45(3): 479–494.

See Also

sol.wcm

Examples

```
set.seed(111)
f <- rep(c(0, 5, 2, 8, 1, -2), c(100, 200, 200, 50, 200, 250))
x <- f + arima.sim(list(ar = c(.75, -.5), ma = c(.8, .7, .6, .5, .4, .3)), n = length(f), sd = 1)
model.gsa(sol.wcm(x))
```

`model.ic` *Estimating change-points or change-point-type features in the mean of a noisy data sequence via the strengthened Schwarz information criterion*

Description

This function estimates the number and locations of change-points or change-point-type features in the 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 features allowed. If nothing or `NULL` is provided, the default value of `min(100, n/log(n))` (rounded to an integer) 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`
- `type` The model type used, inherited from the given `cptpath.object`
- `model.selection` The model selection method used to return the final change-point or change-point-type feature estimators object, here its value is `"ic"`
- `no.of.cpt` The number of estimated features in the mean of the vector `cptpath.object$x` based on the given `type` of the model
- `cpts` The locations of estimated features in the mean of the vector `cptpath.object$x`. These are the end-points of the corresponding constant-mean or constant-slope intervals
- `est` An estimate of the mean of the vector `cptpath.object$x`; for piecewise-constant signals, the values are the sample means of the data (replicated a suitable number of times) between each pair of consecutive detected change-points; for piecewise-linear but discontinuous signals, the values are based on the estimated linear trend between each pair of consecutive detected change of slopes; for piecewise-linear and continuous signals, it is similar to the previous case but with the continuity constraint enforced, which envolves solving a global least squares problem.

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"),
  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*` the value suggested in Fryzlewicz (2020)) is used, see `th.const` in `model.sdll`

Details

Further information can be found in Cho and Kirch (2022).

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 (2022) Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. *Annals of the Institute of Statistical Mathematics*, 74(4), 653–684.

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)) * .5
model.lp(sol.not(x))
```

---

model.sdll — *Estimating change-points in the piecewise-constant or piecewise-linear 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 or piecewise-linear 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. The `cptpath.object$type` variable decides the model type: piecewise-constant (`type == "const"`), piecewise-linear and continuous (`type == "lin.cont"`) or piecewise-linear and discontinuous (`type == "lin.discont"`). In the piecewise-constant model, SDLL model selection should work well when `cptpath.object` is an object returned by the `sol.wbs2` routine. In the piecewise-linear model (whether continuous or not), the output of `sol.idetect` should be supplied as `cptpath.object`. Note that the field `cptpath.object$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 in the piecewise-constant model is the Median Absolute Deviation of the vector `diff(cptpath.object$x)/sqrt(2)`, tuned to the Gaussian distribution. In the piecewise-linear models, `diff(cptpath.object$x, differences = 2)/sqrt(6)` is used by default. 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`. 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 contrast 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 contrasts 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.

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, 49, 1027–1070.

Value

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

- `solution.path` The solution path method used to obtain `cptpath.object`
- `type` The model type used, inherited from the given `cptpath.object`
- `model.selection` 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
- `cpts` The locations of estimated change-points
- `est` An estimate of the mean of the vector `cptpath.object$x`

References

P. Fryzlewicz (2020). Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 49, 1027–1070.

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 or piecewise-linear mean of a noisy data sequence via thresholding*

Description

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

Usage

```
model.thresh(cptpath.object, sigma = NULL, th.const = NULL)
```

Arguments

cptpath.object
: A solution-path object, returned by a sol.[name] routine. The cptpath.object$type variable decides the model type: piecewise-constant (type == "const"), piecewise-linear and continuous (type == "lin.cont") or piecewise-linear and discontinuous (type == "lin.discont"). In the piecewise-linear model (whether continuous or not), the output of sol.idetect_seq or sol.not should be supplied as cptpath.object. Note that the field cptpath.object$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 in the piecewise-constant model is the Median Absolute Deviation of the vector diff(cptpath.object$x)/sqrt(2), tuned to the Gaussian distribution. In the piecewise-linear models, diff(cptpath.object$x, differences = 2)/sqrt(6) is used by default. Note that model.thresh works particularly well when the noise is i.i.d. Gaussian.

th.const
: A positive real number used to define the threshold for the detection process. The default used in the piecewise-constant model is 1.15, while in the piecewise-linear model, the value is taken equal to 1.4.

Value

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

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

type
: The model type used, inherited from the given 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

cpts
: The locations of estimated change-points

est
: An estimate of the mean of the vector cptpath.object$x

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.thresh(sol.idetect_seq(x))
```

plot.breakfast.cpts *Change-points estimated by the "breakfast" routine*

Description

Plot method for objects of class `breakfast.cpts`

Usage

```
## 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

```
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 the "breakfast" routine*

Description

Print method for objects of class `breakfast.cpts`

Usage

```
## 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

```
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

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

Arguments

`x`	a `cptmodel` object
`...`	current not in use

Examples

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

`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, or in its linear trend, 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

```
sol.idetect(
  x,
  type = "const",
  thr_ic_cons = 0.9,
  thr_ic_lin = 1.25,
  points = 3
)
```

Arguments

x	A numeric vector containing the data to be processed.
type	The model type considered. type = "const", type = "lin.cont", type = "lin.discont" mean, respectively, that the signal (mean of x) is piecewise constant, piecewise linear and continuous, and piecewise linear but not necessarily continuous. If not given, the default is type = "const"
thr_ic_cons	A positive real number with default value equal to 0.9. It is used to create the solution path for the piecewise-constant model. The lower the value, the longer the solution path.
thr_ic_lin	A positive real number with default value 1.25. Used to create the solution path if type == "lin.cont" or type == "lin.discont"
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 (2022), Metrika, https://doi.org/10.1007/s00184-021-00821-6.

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
type	The input parameter type
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 (2022). Detecting multiple generalized change-points by isolating single ones. *Metrika*, https://doi.org/10.1007/s00184-021-00821-6.

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 arranges all possible change-points in the mean of the input vector, or in its linear trend, in the order of importance, via the Isolate-Detect (ID) method. It is developed to be used with the thresholding model selection rule.

Usage

```
sol.idetect_seq(x, type = "const", points = 4)
```

Arguments

`x`	A numeric vector containing the data to be processed
`type`	The model type considered. `type = "const"`, `type = "lin.cont"`, `type = "lin.discont"` mean, respectively, that the signal (mean of `x`) is piecewise constant, piecewise linear and continuous, and piecewise linear but not necessarily continuous. If not given, the default is `type = "const"`
`points`	A positive integer with default value equal to 4. 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 (2022), Metrika, https://doi.org/10.1007/s00184-021-00821-6.

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, arranged in decreasing order of change-point importance
`solution.set`	Empty list
`x`	Input vector `x`
`type`	The input parameter `type`

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 (2022). Detecting multiple generalized change-points by isolating single ones. *Metrika*, https://doi.org/10.1007/s00184-021-00821-6.

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 features (e.g. change in the mean, change in the slope, etc) of the input vector in the order of importance, via the Narrowest-Over-Threshold (NOT) method.

Usage

```
sol.not(x, type = "const", M = 10000, systematic.intervals = TRUE, seed = NULL)
```

Arguments

x — A numeric vector containing the data to be processed

type — The model type considered. `type = "const"` means the signals are the piecewise constant, `type = "lin.cont"` means the signals are the piecewise linear and continuous, and `type = "lin.discont"` means the signals are the piecewise linear but not necessarily continuous. If not given, the default is `type = "const"`

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

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`

`type`
: The model type used, which is given in the input. If not given, the default is `type="const"`

`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.idetect_seq`, `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, type = "const", p = 0.01)
```

Arguments

`x`	A numeric vector containing the data to be processed
`type`	The model type considered. `type = "const"` means piecewise-constant; this is the only type currently supported in `sol.tguh`
`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`
`type`	Input parameter `type`
`p`	Input parameter `p`
`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, 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-in-mean features of the input vector in the order of importance, via the Wild Binary Segmentation (WBS) method.

Usage

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

Arguments

`x`	A numeric vector containing the data to be processed
`type`	The model type considered. Currently `type = "const"` is the only accepted value. This assumes that the mean of the input vector is piecewise-constant.
`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 set

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`
- `type`: The input parameter `type`
- `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.

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.idetect_seq`, `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, type = "const", M = 1000, systematic.intervals = TRUE, seed = NULL)
```

Arguments

- `x` — A numeric vector containing the data to be processed.
- `type` — The model type considered. `type = "const"` means piecewise-constant; this is the only type currently supported in `sol.wbs2`
- `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.
- `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 set

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, 49, 1027-1070.

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`
- `type` — Input parameter `type`
- `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*, 49, 1027-1070.

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)
```

---

sol.wcm — *Solution path generation via the Wild Contrast Maximisation 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.wcm(
  x,
  type = "const",
  M = 100,
  min.d = NULL,
  Q = floor(log(length(x))^1.9),
  max.iter = 5
)
```

Arguments

`x`	A numeric vector containing the data to be processed.
`type`	The type of change-point models fitted to the data; currently the class of piece-wise constant signals (`type = "const"`) is supported.
`M`	The maximum number of data sub-samples drawn at each recursive stage of the algorithm. The default is `M = 100`.
`min.d`	The minimum distance between candidate change-point estimators; if `min.d = NULL`, it is set to be `max(20, 10 + ceiling(log(length(x))^1.1)`.
`Q`	The maximum number of allowable change-points. The default is `Q = floor(log(length(x))^1.9)`.
`max.iter`	The maximum number of candidate change-point models considered; if a model with the number of change-point estimators exceeding `Q` is required to generate the sequence of required candidate models, this argument is ignored. The default is `max.iter = 5`.

Details

The Wild Contrast Maximisation (WCM) algorithm generates a nested sequence of candidate models by identifying large gaps in the solution path generated by WBS2, which aids the model selection step in the presence of large random fluctuations due to serial dependence. See Cho and Fryzlewicz (2023) for further 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; this is not used by `model.gsa`

`solution.set`
: A list of candidate change-point models. Each model contains possible change-points in the mean of `x`; this is used by `model.gsa`

`x`
: Input vector `x`

`type`
: The type of the change-point model considered, which has value "const" here

`M`
: Input parameter `M`

`cands`
: Matrix of dimensions `Q` 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 "wcm" here

References

H. Cho & P. Fryzlewicz (2024) Multiple change point detection under serial dependence: Wild contrast maximisation and gappy Schwarz algorithm. *Journal of Time Series Analysis*, 45(3): 479–494.

See Also

`model.gsa`

Examples

```
set.seed(111)
f <- rep(c(0, 5, 2, 8, 1, -2), c(100, 200, 200, 50, 200, 250))
x <- f + arima.sim(list(ar = c(.75, -.5), ma = c(.8, .7, .6, .5, .4, .3)), n = length(f), sd = 1)
sol.wcm(x)$solution.set
```

Index