Package 'mosum'

November 4, 2019

Title Moving Sum Based Procedures for Changes in the Mean

Version 1.2.3

Date 2019-10-21

Description Implementations of MOSUM-based statistical procedures and algorithms for detecting multiple changes in the mean. This comprises the MOSUM procedure for estimating multiple mean changes from Eichinger and Kirch (2018) <doi:10.3150/16-BEJ887> and the multiscale algorithmic extensions from Cho and Kirch (2019) <arXiv:1910.12486>.

Depends R (>= 3.1.2)

License GPL (>= 3)

LazyData true

Imports RColorBrewer, plot3D, Rcpp (>= 0.12.5)

LinkingTo Rcpp

Maintainer Haeran Cho <haeran.cho@bristol.ac.uk>

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Author Alexander Meier [aut],
Haeran Cho [aut, cre],
Claudia Kirch [aut]

Repository CRAN

Date/Publication 2019-11-04 14:30:02 UTC

R topics documented:

`bandwidths.default` *Default choice for the set of multiple bandwidths*

Description

Create bandwidths according to a default function of the sample size

Usage

```
bandwidths.default(n, d.min = 10, G.min = 10, G.max = min(n/2,
  n^(2/3)))
```

Arguments

`n`	integer representing the sample size
`d.min`	integer for the minimal mutual distance of change-points that can be expected
`G.min`	integer for the minimal allowed bandwidth
`G.max`	integer for the maximal allowed bandwidth

Details

Returns an integer vector of bandwidths (G_1,...,G_m), with G_0 = G_1 = max(`G.min`, 2/3*`d.min`), G_j+1 = G_j-1 + G_j (for j = 1, ..., m-1) and m satisfying G_m <= `G.max` while G_m+1 > `G.max`.

Value

an integer vector of bandwidths

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-point Analysis. *To appear in the Journal of Statistical Software*.

H. Cho and C. Kirch (2019) Localised pruning for data segmentation based on multiscale change point procedures. *arXiv preprint arXiv:1910.12486*.

Examples

```
bandwidths.default(1000, 10, 10, 200)
```

`confint.mosum.cpts` *Confidence intervals for change-points*

Description

Generate bootstrap confidence intervals for change-points.

Usage

```
## S3 method for class 'mosum.cpts'
confint(object, parm = "cpts", level = 0.05,
  N_reps = 1000, ...)
```

Arguments

`object`	an object of class `mosum.cpts`
`parm`	specification of which parameters are to be given confidence intervals; `parm = "cpts"` is supported
`level`	numeric value in (0, 1), such that the `100(1-level)%` confidence bootstrap intervals are computed
`N_reps`	number of bootstrap replications
`...`	not in use

Details

See the referenced literature for further details

Value

S3 object of class `cpts.ci`, containing the following fields:

`level, N_reps`	input parameters
`CI`	data frame of five columns, containing the estimated change-points (column `cpts`), the pointwise confidence intervals (columns `pw.left` and `pw.right`) and the uniform confidence intervals (columns `unif.left` and `unif.right`) for the corresponding change-points

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-point Analysis. *To appear in the Journal of Statistical Software*.

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 3, 1), sds = rep(1, 3), seed = 1337)$x
m <- mosum(x, G = 40)
ci <- confint(m, N_reps = 5000)
print(ci$CI)
```

`confint.multiscale.cpts` *Confidence intervals for change-points*

Description

Generate bootstrap confidence intervals for change-points.

Usage

```
## S3 method for class 'multiscale.cpts'
confint(object, parm = "cpts", level = 0.05,
  N_reps = 1000, ...)
```

Arguments

`object`	an object of class `multiscale.cpts`
`parm`	specification of which parameters are to be given confidence intervals; `parm = "cpts"` is supported
`level`	numeric value in (0, 1), such that the `100(1-level)%` confidence bootstrap intervals are computed
`N_reps`	number of bootstrap replications
`...`	not in use

Details

See the referenced literature for further details

Value

S3 object of class `cpts.ci`, containing the following fields:

`level, N_reps`	input parameters
`CI`	data frame of five columns, containing the estimated change-points (column `cpts`), the pointwise confidence intervals (columns `pw.left` and `pw.right`) and the uniform confidence intervals (columns `unif.left` and `unif.right`) for the corresponding change-points

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-point Analysis. *To appear in the Journal of Statistical Software*.

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 3, 1), sds = rep(1, 3), seed = 1337)$x
mlp <-  multiscale.localPrune(x, G = c(8, 15, 30, 70))
ci <- confint(mlp, N_reps = 5000)
print(ci$CI)
```

`mosum` *MOSUM procedure for multiple change-point estimation*

Description

Computes the MOSUM detector, detects (multiple) change-points and estimates their locations.

Usage

```
mosum(x, G, G.right = G, var.est.method = c("mosum", "mosum.min",
  "mosum.max", "custom")[1], var.custom = NULL,
  boundary.extension = TRUE, threshold = c("critical.value",
  "custom")[1], alpha = 0.1, threshold.custom = NULL,
  criterion = c("eta", "epsilon")[1], eta = 0.4, epsilon = 0.2,
  do.confint = FALSE, level = 0.05, N_reps = 1000)
```

Arguments

`x`	input data (a `numeric` vector or an object of classes `ts` and `timeSeries`)
`G`	an integer value for the moving sum bandwidth; `G` should be less than `length(n)/2`. Alternatively, a number between `0` and `0.5` describing the moving sum bandwidth relative to `length(x)` can be given
`G.right`	if `G.right != G`, the asymmetric bandwidth `(G,G.right)` will be used; if `max(G,G.right)/min(G,G.rig` `> 4`, a warning message is generated
`var.est.method`	how the variance is estimated; possible values are • `"mosum"` both-sided MOSUM variance estimator • `"mosum.min"` minimum of the sample variance estimates from the left and right summation windows • `"mosum.max"` maximum of the sample variance estimates from the left and right summation windows • `"custom"` a vector of `length(x)` is to be parsed by the user; use `var.custom` in this case to do so
`var.custom`	a numeric vector (of the same length as `x`) containing local estimates of the variance or long run variance; use iff `var.est.method = "custom"`
`boundary.extension`	a logical value indicating whether the boundary values should be filled-up with CUSUM values

threshold	string indicating which threshold should be used to determine significance. By default, it is chosen from the asymptotic distribution at the given significance level alpha. Alternatively it is possible to parse a user-defined numerical value with threshold.custom
alpha	a numeric value for the significance level with 0 <= alpha <= 1; use iff threshold = "critical.value"
threshold.custom	a numeric value greater than 0 for the threshold of significance; use iff threshold = "custom"
criterion	string indicating how to determine whether each point k at which MOSUM statistic exceeds the threshold is a change-point; possible values are • "eta"there is no larger exceeding in an eta*G environment of k • "epsilon"k is the maximum of its local exceeding environment, which has at least size epsilon*G
eta	a positive numeric value for the minimal mutual distance of changes, relative to moving sum bandwidth (iff criterion = "eta")
epsilon	a numeric value in (0,1] for the minimal size of exceeding environments, relative to moving sum bandwidth (iff criterion = "epsilon")
do.confint	flag indicating whether to compute the confidence intervals for change-points
level	use iff do.confint = TRUE; a numeric value (0 <= level <= 1) with which 100(1-level)% confidence interval is generated
N_reps	use iff do.confint = TRUE; number of bootstrap replicates to be generated

Value

S3 object of class mosum.cpts, which contains the following fields:

x	input data
G.left, G.right	left and right summation bandwidths
var.est.method, var.custom,boundary.extension	input parameters
stat	a series of MOSUM statistic values; the first G and last G.right values are NA iff boundary.extension = FALSE
rollsums	a series of MOSUM detector values; equals stat*sqrt(var.estimation)
var.estimation	the local variance estimated according to var.est.method
threshold, alpha, threshold.custom	input parameters
threshold.value	threshold value of the corresponding MOSUM test
criterion, eta, epsilon	input parameters
cpts	a vector containing the estimated change-point locations

`cpts.info`	data frame containing information about change-point estimators including detection bandwidths, asymptotic p-values for the corresponding MOSUM statistics and (scaled) size of jumps
`do.confint`	input parameter
`ci`	S3 object of class `cpts.ci` containing confidence intervals for change-points iff `do.confint=TRUE`

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-Point Analysis. *To appear in the Journal of Statistical Software*.

B. Eichinger and C. Kirch (2018) A MOSUM procedure for the estimation of multiple random change-points. *Bernoulli*, Volume 24, Number 1, pp. 526-564.

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 5, -2), sds = rep(1, 3), seed = 1234)$x
m <- mosum(x, G = 40)
plot(m)
summary(m)
```

`mosum.criticalValue` *MOSUM asymptotic critical value*

Description

Computes the asymptotic critical value for the MOSUM test.

Usage

```
mosum.criticalValue(n, G.left, G.right, alpha)
```

Arguments

`n`	an integer value for the length of the input data
`G.left, G.right`	integer values for the left and right moving sum bandwidth (G.left, G.right)
`alpha`	a numeric value for the significance level with `0 <= alpha <= 1`

Value

a numeric value for the asymptotic critical value for the MOSUM test

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 5, -2), sds = rep(1, 3), seed = 1234)$x
m <- mosum(x, G = 40)
par(mfrow = c(2, 1))
plot(m$stat, type = "l", xlab = "Time", ylab = "", main = "mosum")
abline(h = mosum.criticalValue(300, 40, 40, .1), col = 4)
abline(v = m$cpts, col = 2)
plot(m, display = "mosum") # identical plot is produced
```

`multiscale.bottomUp` *Multiscale MOSUM algorithm with bottom-up merging*

Description

Multiscale MOSUM procedure with symmetric bandwidths combined with bottom-up bandwidth-based merging.

Usage

```
multiscale.bottomUp(x, G = bandwidths.default(length(x), G.min = max(20,
  ceiling(0.05 * length(x)))), threshold = c("critical.value",
  "custom")[1], alpha = 0.1, threshold.function = NULL, eta = 0.4,
  do.confint = FALSE, level = 0.05, N_reps = 1000, ...)
```

Arguments

`x`	input data (a `numeric` vector or an object of classes `ts` and `timeSeries`)
`G`	a vector of (symmetric) bandwidths, given as either integers less than `length(x)/2`, or numbers between `0` and `0.5` describing the moving sum bandwidths relative to `length(x)`. If the smallest bandwidth is smaller than `min(20,0.05*length(x))` (`0.05` if relative bandwidths are given) and `threshold = "critical.value"`, it generates a warning message
`threshold`	string indicating which threshold should be used to determine significance. By default, it is chosen from the asymptotic distribution at the given significance level `alpha`. Alternatively, it is possible to parse a user-defined function with `threshold.function`
`alpha`	a numeric value for the significance level with `0 <= alpha <= 1`; use iff `threshold = "critical.value"`
`threshold.function`	function object of form `function(G,length(x),alpha)`, to compute a threshold of significance for different bandwidths G; use iff `threshold = "custom"`
`eta`	see mosum
`do.confint`	flag indicating whether to compute the confidence intervals for change-points
`level`	use iff `do.confint = TRUE`; a numeric value (`0 <= level <= 1`) with which `100(1-level)%` confidence interval is generated
`N_reps`	use iff `do.confint = TRUE`; number of bootstrap replicates to be generated
`...`	further arguments to be passed to the mosum calls

Details

See Algorithm 1 in the first referenced paper for a comprehensive description of the procedure and further details.

Value

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

`x`	input data
`cpts`	estimated change-points
`cpts.info`	data frame containing information about estimated change-points
`pooled.cpts`	set of change-point candidates that have been considered by the algorithm
`G`	bandwidths
`threshold, alpha, threshold.function`	input parameters
`eta`	input parameters
`do.confint`	input parameter
`ci`	object of class `cpts.ci` containing confidence intervals for change-points iff `do.confint = TRUE`

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-point Analysis. *To appear in the Journal of Statistical Software*.

M. Messer et al. (2014) A multiple filter test for the detection of rate changes in renewal processes with varying variance. *The Annals of Applied Statistics*, Volume 8, Number 4, pp. 2027-2067.

Examples

```
x1 <- testData(lengths = c(100, 200, 300, 300),
means = c(0, 1, 2, 2.7), sds = rep(1, 4), seed = 123)$x
mbu1 <- multiscale.bottomUp(x1)
plot(mbu1)
summary(mbu1)

x2 <- testData(model = "mix", seed = 1234)$x
threshold.custom <- function(G, n, alpha) {
mosum.criticalValue(n, G, G, alpha) * log(n/G)^0.1
}
mbu2 <- multiscale.bottomUp(x2, G = 10:40, threshold = "custom",
threshold.function = threshold.custom)
plot(mbu2)
summary(mbu2)
```

`multiscale.localPrune` *Multiscale MOSUM algorithm with localised pruning*

Description

Multiscale MOSUM procedure with (possibly) assymetric bandwidths and localised pruning based on Schwarz criterion.

Usage

```
multiscale.localPrune(x, G = bandwidths.default(length(x)),
  max.unbalance = 4, threshold = c("critical.value", "custom")[1],
  alpha = 0.1, threshold.function = NULL, criterion = c("eta",
  "epsilon")[1], eta = 0.4, epsilon = 0.2, rule = c("pval",
  "jump")[1], penalty = c("log", "polynomial")[1], pen.exp = 1.01,
  do.confint = FALSE, level = 0.05, N_reps = 1000, ...)
```

Arguments

`x`	input data (a `numeric` vector or an object of classes `ts` and `timeSeries`)
`G`	a vector of bandwidths, given as either integers less than `length(x)/2`, or numbers between `0` and `0.5` describing the moving sum bandwidths relative to `length(x)`. Asymmetric bandwidths obtained as the Cartesian product of the set `G` with itself are used for change-point analysis
`max.unbalance`	a numeric value for the maximal ratio between maximal and minimal bandwidths to be used for candidate generation, `1 <= max.unbalance <= Inf`
`threshold`	string indicating which threshold should be used to determine significance. By default, it is chosen from the asymptotic distribution at the significance level `alpha`. Alternatively, it is possible to parse a user-defined function with `threshold.function`
`alpha`	a numeric value for the significance level with `0 <= alpha <= 1`. Use iff `threshold = "critical.value"`
`threshold.function`	function object of form `function(G_l,G_r,length(x),alpha)`, to compute a threshold of significance for different bandwidths `(G_l,G_r)`; use iff `threshold = "custom"`
`criterion`	how to determine whether an exceeding point is a change-point; to be parsed to mosum
`eta, epsilon`	see mosum
`rule`	string for the choice of sorting criterion for change-point candidates in merging step. Possible values are: • `"pval"` smallest p-value • `"jump"` largest (rescaled) jump size
`penalty`	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.confint	flag indicating whether confidence intervals for change-points should be computed
level	use iff do.confint = TRUE; a numeric value (0 <= level <= 1) with which 100(1-level)% confidence interval is generated
N_reps	use iff do.confint = TRUE; number of bootstrap replicates to be generated
...	further arguments to be parsed to mosum calls

Details

See Algorithm 2 in the first referenced paper for a comprehensive description of the procedure and further details.

Value

S3 object of class multiscale.cpts, which contains the following fields:

x	input data
cpts	estimated change-points
cpts.info	data frame containing information about estimated change-points
sc	Schwarz criterion values of the estimated change-point set
pooled.cpts	set of change-point candidates that have been considered by the algorithm
G	input parameter
threshold, alpha, threshold.function	input parameters
criterion, eta, epsilon	input parameters
rule, penalty, pen.exp	input parameters
do.confint	input parameter
ci	object of class cpts.ci containing confidence intervals for change-points iff do.confint = TRUE

References

A. Meier, C. Kirch and H. Cho (2019) mosum: A Package for Moving Sums in Change-point Analysis. *To appear in the Journal of Statistical Software*.

H. Cho and C. Kirch (2019) Localised pruning for data segmentation based on multiscale change point procedures. *arXiv preprint arXiv:1910.12486*.

Examples

```
x <- testData(model = "mix", seed = 123)$x
mlp <- multiscale.localPrune(x, G = c(8, 15, 30, 70), do.confint = TRUE)
print(mlp)
summary(mlp)
par(mfcol=c(2, 1), mar = c(2, 4, 2, 2))
plot(mlp, display = "data", shaded = "none")
plot(mlp, display = "significance", shaded = "CI", CI = "unif")
```

`persp3D.multiscaleMosum` *3D Visualisation of multiscale MOSUM statistics*

Description

3D Visualisation of multiscale MOSUM statistics.

Usage

```
persp3D.multiscaleMosum(x, mosum.args = list(),
  threshold = c("critical.value", "custom")[1], alpha = 0.1,
  threshold.function = NULL, pal.name = "YlOrRd", expand = 0.2,
  theta = 120, phi = 20, xlab = "G", ylab = "time",
  zlab = "MOSUM", ticktype = "detailed", NAcol = "#800000FF", ...)
```

Arguments

`x`	a numeric input data vector
`mosum.args`	a named list containing further arguments to be parsed to the respective `mosum` function calls, see mosum; the bandwidths are chosen by the function and should not be given as an argument in `mosum.args`
`threshold`	string indicating which threshold should be used for normalisation of MOSUM statistics computed with different bandwidths. By default, it is chosen from the asymptotic distribution at the given significance level `alpha`. Alternatively it is possible to parse a user-defined numerical value with `threshold.custom`; see also Details.
`alpha`	a numeric value for the significance level with `0 <= alpha <= 1`; use iff `threshold = "critical.value"`
`threshold.function`	function object of form `function(G)`, to compute a threshold of significance for different bandwidths G; use iff `threshold='custom'`
`pal.name`	a string containing the name of the ColorBrewer palette to be used; sequential palettes are recommended. See `RColorBrewer::brewer.pal.info` for details
`expand`	expansion factor applied to the z coordinates
`theta`	azimuthal angle defining the viewing direction

`phi`	colatitude angle defining the viewing direction
`xlab, ylab, zlab, ticktype`	graphical parameters
`NAcol`	coloring parameter
`...`	further arguments to be passed to function call of persp3D

Details

The visualisation is based on persp3D. MOSUM statistics computed with different bandwidths are rescaled for making them visually comparable. Rescaling is done either by dividing by their respective critical value at the significance level `alpha` (iff `threshold = "critical.value"`) or by a custom value given by `threshold.function` (iff `threshold = "custom"`). By default, `clim` argument of persp3D is given so that the three lightest (for sequential palettes) hues indicate insignificance of the corresponding MOSUM statistics, while darker hues indicate the presence of significant changes.

Value

see persp3D

Examples

```
## Not run:
# If you run the example be aware that this may take some time
print("example may take some time to run")

x <- testData(model = "blocks", seed = 1234)$x
persp3D.multiscaleMosum(x, mosum.args = list(boundary.extension = FALSE))

## End(Not run)
```

---

plot.mosum.cpts *Plotting the output from MOSUM procedure*

Description

Plotting method for S3 objects of class `mosum.cpts`

Usage

```
## S3 method for class 'mosum.cpts'
plot(x, display = c("data", "mosum")[1],
  cpts.col = "red", critical.value.col = "blue", xlab = "Time", ...)
```

Arguments

x	a mosum.cpts object
display	which to be plotted against the change-point estimators; possible values are • "data"input time series is plotted along with the estimated piecewise constant signal • "mosum"scaled MOSUM detector values are plotted
cpts.col	a specification for the color of the vertical lines at the change-point estimators, see par
critical.value.col	a specification for the color of the horizontal line indicating the critical value, see par; use iff display = "mosum"
xlab	graphical parameter
...	additional graphical arguments, see plot and abline

Details

The location of each change-point estimator is plotted as a vertical line against the input time series and the estimated piecewise constant signal (display = "data") or MOSUM detector values (display = "mosum").

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 5, -2), sds = rep(1, 3), seed = 1234)$x
m <- mosum(x, G = 40)
par(mfrow = c(2, 1), mar = c(2.5, 2.5, 2.5, .5))
plot(m, display = "data")
plot(m, display = "mosum")
```

---

plot.multiscale.cpts *Plotting the output from multiscale MOSUM procedure*

Description

Plotting method for S3 objects of class "multiscale.cpts".

Usage

```
## S3 method for class 'multiscale.cpts'
plot(x, display = c("data", "significance")[1],
  shaded = c("CI", "bandwidth", "none")[1], level = 0.05,
  N_reps = 1000, CI = c("pw", "unif")[1], xlab = "Time", ...)
```

Arguments

x	a multiscale.cpts object
display	which to be plotted against the estimated change-point locations; possible values are • "data"input time series is plotted along with the estimated piecewise constant signal • "significance"one minus the p-values associated with the detection of change-point estimators are represented as the height of vertical lines indicating their locations
shaded	string indicating which to display as shaded areas surrounding the estimated change-point locations. Poissble values are • "bandwidth"respective detection intervals are plotted • "CI"bootstrap confidence intervals are plotted • "none"none is plotted
level, N_reps	argument to be parsed to confint.multiscale.cpts; use iff shaded = "CI".
CI	string indicating whether pointwise (CI = "pw") or uniform (CI = "unif") confidence intervals are to be plotted; use iff shaded = "CI"
xlab	graphical parameter
...	not in use

Details

The locations of change-point estimators are plotted against the input time series and the estimated piecewise constant signal (display = "data"), or the significance of each estimator is represented by the corresponding 1-p.value derived from the asymptotic distribution of MOSUM test statistic (display = "significance"). It also produces the rectangles representing the detection intervals (if shaded = "bandwidth") or bootstrap confidence intervals of the corresponding change-points (if shaded = "CI") around their locations.

Examples

```
x <- testData(model = "blocks", seed = 1234)$x
mlp <- multiscale.localPrune(x)
par(mfrow = c(2, 1))
plot(mlp, display = "data", shaded = "bandwidth")
plot(mlp, display = "significance", shaded = "CI")
```

---

print.mosum.cpts *Change-points estimated by MOSUM procedure*

Description

Print method for objects of class mosum.cpts

Usage

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

Arguments

x	a mosum.cpts object
...	not in use

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 5, -2), sds = rep(1, 3), seed = 1234)$x
m <- mosum(x, G = 40)
print(m)
```

---

print.multiscale.cpts *Change-points estimated by multiscale MOSUM procedure*

Description

Print method for objects of class multiscale.cpts

Usage

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

Arguments

x	a multiscale.cpts object
...	not in use

Examples

```
x <- testData(model = "mix", seed = 12345)$x
mlp <- multiscale.localPrune(x)
print(mlp)
```

summary.mosum.cpts *Summary of change-points estimated by MOSUM procedure*

Description

Summary method for objects of class `mosum.cpts`

Usage

```
## S3 method for class 'mosum.cpts'
summary(object, ...)
```

Arguments

`object`	a `mosum.cpts` object
`...`	not in use

Details

Provide information about each estimated change-point, including the bandwidths used for its estimation, associated p-value and (scaled) jump size; if `object$do.confint=TRUE`, end points of the pointwise and uniform confidence intervals are also provided.

Examples

```
x <- testData(lengths = rep(100, 3), means = c(0, 5, -2), sds = rep(1, 3), seed = 1234)$x
m <- mosum(x, G = 40, do.confint = TRUE)
summary(m)
```

summary.multiscale.cpts *Summary of change-points estimated by multiscale MOSUM procedure*

Description

Summary method for objects of class `multiscale.cpts`

Usage

```
## S3 method for class 'multiscale.cpts'
summary(object, ...)
```

Arguments

`object`	a `multiscale.cpts` object
`...`	not in use

Details

Provide information about each estimated change-point, including the bandwidths used for its detection, associated p-value and (scaled) jump size; if `object$do.confint=TRUE`, end points of the pointwise and uniform confidence intervals are also provided.

Examples

```
x <- testData(model = "mix", seed = 12345)$x
mlp <- multiscale.localPrune(x, do.confint = TRUE)
summary(mlp)
```

`testData` *Test data with piecewise constant mean*

Description

Generate piecewise stationary time series with independent innovations and change-points in the mean.

Usage

```
testData(model = c("custom", "blocks", "fms", "mix", "stairs10",
  "teeth10")[1], lengths = NULL, means = NULL, sds = NULL,
  rand.gen = rnorm, seed = NULL, ...)
```

Arguments

`model`	a string indicating from which model a realisation is to be generated; possible values are "custom" (for user-specified model using `lengths`, `means` and `sds`), and "blocks", "fms", "mix", "stairs10", "teeth10" (for the referenced test signals)
`lengths`	use iff `model = "custom"`; an integer vector for the lengths of the piecewise stationary segments
`means`	use iff `model = "custom"`; a numeric vector for the means of the piecewise stationary segments
`sds`	use iff `model = "custom"`; a numeric vector for the deviation scaling of the piecewise stationary segments. The values are multiplied to the outcome of `rand.gen`, coinciding with the standard deviation in the case of standard normal innovations (`rand.gen = rnorm`)
`rand.gen`	optional; a function to generate the noise/innovations
`seed`	optional; if a seed value is provided (`!is.null(seed)`), then `set.seed(seed)` is called beforehand)
`...`	further arguments to be parsed to `rand.gen`

Details

See Appendix B in the reference for details about the test signals.

Value

a list containing the following entries:

- x a numeric vector containing a realisation of the piecewise time series model, given as signal + noise
- mu mean vector of piecewise stationary time series model
- sigma scaling vector of piecewise stationary time series model
- cpts a vector of change-points in the piecewise stationary time series model

References

P. Fryzlewicz (2014) Wild Binary Segmentation for Multiple Change-Point Detection. *The Annals of Statistics*, Volume 42, Number 6, pp. 2243-2281.

Examples

```
# visualise estimated changepoints by solid vertical lines
# and true changepoints by broken vertical lines
td <- testData(lengths = c(50, 50, 200, 300, 300), means = c(0, 1, 2, 3, 2.3),
sds = rep(1, 5), seed = 123)
mbu <- multiscale.bottomUp(td$x)
plot(mbu, display = "data")
abline(v = td$cpts, col = 2, lwd = 2, lty = 2)

# visualise estimated piecewise constant signal by solid line
# and true signal by broken line
td <- testData("blocks", seed = 123)
mlp <- multiscale.localPrune(td$x)
plot(mlp, display = "data")
lines(td$mu, col = 2, lwd = 2, lty = 2)
```

Index