Package ‘cpss’

October 12, 2022

Title Change-Point Detection by Sample-Splitting Methods

Version 0.0.3

Description Implements multiple change searching algorithms for a variety of frequently considered parametric change-point models. In particular, it integrates a criterion proposed by Zou, Wang and Li (2020) <doi:10.1214/19-AOS1814> to select the number of change-points in a data-driven fashion. Moreover, it also provides interfaces for user-customized change-point models with one's own cost function and parameter estimation routine. It is easy to get started with the cpss.* set of functions by accessing their documentation pages (e.g., ?cpss).

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, magrittr, methods, stats, mvtnorm, Rfast, tibble, dplyr, tidyr, rlang, ggplot2, gridExtra

Suggests MASS

URL `https://github.com/ghwang-nk/cpss`

BugReports `https://github.com/ghwang-nk/cpss/issues`

Depends R (>= 2.10)

Maintainer Guanghui Wang <`ghwang.nk@gmail.com`>

NeedsCompilation yes

Author Guanghui Wang [aut, cre],
Changliang Zou [aut]

Repository CRAN

Date/Publication 2022-08-22 09:00:16 UTC

R topics documented:

`algo` *Generic functions and methods: algo*

Description

Generic functions and methods: algo

Usage

```
algo(x)

algo(x) <- value

## S4 method for signature 'cpss'
algo(x)

## S4 replacement method for signature 'cpss'
algo(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`algo_param_dim` *Generic functions and methods: algo_param_dim*

Description

Generic functions and methods: algo_param_dim

Usage

```
algo_param_dim(x)

algo_param_dim(x) <- value

## S4 method for signature 'cpss'
algo_param_dim(x)

## S4 replacement method for signature 'cpss'
algo_param_dim(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`coef,cpss-method` *coef method*

Description

coef method

Usage

```
## S4 method for signature 'cpss'
coef(object)
```

Arguments

`object`	object from cpss
`cpss`	cpss class

cps *Generic functions and methods: cps*

Description

Generic functions and methods: cps

Usage

```
cps(x)

cps(x) <- value

## S4 method for signature 'cpss'
cps(x)

## S4 replacement method for signature 'cpss'
cps(x) <- value
```

Arguments

x	object from cpss
value	value assigned to x

cpss *cpss: Change-Point Detection by Sample-Splitting Methods*

Description

Implements multiple change searching algorithms for a variety of frequently considered parametric change-point models. In particular, it integrates a criterion proposed by Zou, Wang and Li (2020) doi:10.1214/19-AOS1814 to select the number of change-points in a data-driven fashion. Moreover, it also provides interfaces for user-customized change-point models with one's own cost function and parameter estimation routine.

Getting started

Easy to get started with the cpss.* set of functions by accessing their documentation pages
library(cpss)
?cpss.mean
?cpss.var
?cpss.meanvar
?cpss.glm
?cpss.lm
?cpss.em
?cpss.custom

cpss-class *cpss: an S4 class which collects data and information required for further change-point analyses and summaries*

Description

cpss: an S4 class which collects data and information required for further change-point analyses and summaries

Slots

`dat` ANY.

`mdl` character.

`algo` character.

`algo_param_dim` numeric.

`SC` character.

`ncps` integer.

`pelt_pen` numeric.

`cps` numeric.

`params` list.

`S_vals` numeric.

`SC_vals` matrix.

`call` list.

`update_inputs` list.

cpss.custom *Detecting changes in uers-customized models*

Description

Detecting changes in uers-customized models

Usage

```
cpss.custom(
  dataset,
  n,
  g_subdat,
  g_param,
  g_cost,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2,
  model = NULL,
  g_smry = NULL,
  easy_cost = NULL,
  param.opt = NULL
)
```

Arguments

`dataset` an `ANY` object that could be a vector, matrix, tensor, list, etc.

`n` an integer indicating the sample size of the data `dataset`.

`g_subdat` a customized R function of two arguments `dat` and `indices`, which extracts a subset of data `dat` according to a collection of time indices `indices`. The returned object inherits the class from that of `dataset`. The argument `dat` inherits the class from that of `dataset`, and the argument `indices` is a logical vector with `TRUE`s indicating extracted indices.

`g_param` a customized R function of two arguments `dat` (cf. `dat` of `g\_subdat`) and `param.opt` (cf. `param.opt` of `cpss.custom`), which returns estimated parameters based on the data segment `dat`. It could return a numeric value, vector, matrix, list, etc.

`g_cost` a customized R function of two arguments `dat` (cf. `dat` of `g\_subdat`) and `param`, which returns a numeric value of the associated cost for data segment `dat` with parameters `param`. The argument `param` inherits the class from that of the returned object of `g\_param`.

`algorithm` a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.

`dist_min` an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

`model` a character string indicating the considered change model.

`g_smry` a customized R function of two arguments `dataset` (cf. `dataset` of `cpss.custom`) and `param.opt` (cf. `param.opt` of `cpss.custom`), which calculates the summary statistics that will be used for cost evaluation. The returned object is a list.

`easy_cost` a customized R function of three arguments `data_smry`, `s` and `e`, which evaluates the value of the cost for a date segment form observed time point $s$ to $e$. The argument `data_smry` inherits the class from that of the returned object of `g_smry`.

`param.opt` an `ANY` object specifying additional constant parameters needed for parameter estimation or cost evaluation beyond unknown parameters.

Value

`cpss.custom` returns an object of an S4 class, called `"cpss"`, which collects data and information required for further change-point analyses and summaries.

`dat` data set

`mdl` considered change-point model

`algo` change-point searching algorithm

`algo_param_dim` user-specified upper bound of the number of true change-points if `algorithm = "SN"/"BS"/"WBS"`, or user-specified candidate values of the penalty only if `algorithm = "PELT"`

`SC` model selection criterion

`ncps` estimated number of change-points

`pelt_pen` selected value of the penalty only if `algorithm = "PELT"`

`cps` a vector of estimated locations of change-points

`params` a list object, each member is a list containing estimated parameters in the associated data segment

`S_vals` a numeric vector of candidate model dimensions in terms of a sequence of numbers of change-points or values of the penalty

`SC_vals` a numeric matrix, each column records the values of the criterion based on the validation data split under the corresponding model dimension (`S_vals`), and each row represents a splitting at each time

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500): 1590–1598.

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

Examples

```
library("cpss")
g_subdat_l1 <- function(dat, indices) {
  dat[indices]
}
g_param_l1 <- function(dat, param.opt = NULL) {
  return(median(dat))
}
g_cost_l1 <- function(dat, param) {
  return(sum(abs(dat - param)))
}
res <- cpss.custom(
  dataset = well, n = length(well),
  g_subdat = g_subdat_l1, g_param = g_param_l1, g_cost = g_cost_l1,
  ncps_max = 11
)
summary(res)
plot(well)
abline(v = res@cps, col = "red")
```

cpss.em *Detecting changes in exponential family*

Description

Detecting changes in exponential family

Usage

```
cpss.em(
  dataset,
  family,
  size = NULL,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2
)
```

Arguments

`dataset` a numeric matrix of dimension $n \times d$, where each row represents an observation and each column stands for a variable. A numeric vector is also acceptable for univariate observations.

`family` a character string specifying the underlying distribution. In the current version, detecting changes in binomial (`"binom"`), multinomial (`"multinom"`), Poisson (`"pois"`), exponential (`"exp"`), geometric (`"geom"`), Dirichlet (`"diri"`), gamma (`"gamma"`), beta (`"beta"`), chi-square (`"chisq"`) and inverse gaussian (`"invgauss"`) distributions are supported.

`size` an integer indicating the number of trials only if `family = "binom"` or `family = "multinom"`.

`algorithm` a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.

`dist_min` an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

Value

`cpss.em` returns an object of an S4 class, called `"cpss"`, which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500):1590–1598.

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

See Also

`cpss.meanvar` `cpss.mean` `cpss.var`

Examples

```
library("cpss")
set.seed(666)
n <- 1000
tau <- c(100, 300, 700, 900)
tau_ext <- c(0, tau, n)
theta <- c(1, 0.2, 1, 0.2, 1)
seg_len <- diff(c(0, tau, n))
y <- unlist(lapply(seq(1, length(tau) + 1), function(k) {
  rexp(seg_len[k], theta[k])
}))
res <- cpss.em(
  y, family = "exp", algorithm = "WBS", ncps_max = 10,
  criterion = "MS", times = 10
)
cps(res)
# [1] 100 299 705 901
```

---

cpss.glm *Detecting changes in GLMs*

Description

Detecting changes in GLMs

Usage

```
cpss.glm(
  formula,
  family,
  data = NULL,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2
)
```

Arguments

`formula`	a `formula` object specifying the GLM with change-points.
`family`	a description of the error distribution and link function to be used in the model, which can be a character string naming a family function or a family function.
`data`	an optional data frame containing the variables in the model.

`algorithm` a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.

`dist_min` an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

Value

`cpss.glm` returns an object of an S4 class, called "cpss", which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500):1590–1598.

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

See Also

`cpss.lm`

Examples

```
library("cpss")
set.seed(666)
n <- 200
size <- rpois(n, 20 - 1) + 1
tau <- c(75, 100, 175)
tau_ext <- c(0, tau, n)
be <- list(c(0, 0.5), c(0, -0.5), c(0.5, -0.5), c(-0.5, -0.5))
seg_len <- diff(c(0, tau, n))
x <- rnorm(n)
eta <- lapply(seq(1, length(tau) + 1), function(k) {
  be[[k]][1] + be[[k]][2] * x[(tau_ext[k] + 1):tau_ext[k + 1]]
})
```

```
eta <- do.call(c, eta)
p <- 1 / (1 + exp(-eta))
y <- rbinom(n, size = size, prob = p)

pelt_pen_val <- (log(n))^seq(0.5, 2, by = 0.1)
res <- cpss.glm(
  formula = cbind(y, size - y) ~ x, family = binomial(),
  algorithm = "PELT", pelt_pen_val = pelt_pen_val, ncps_max = 10
)
summary(res)
# 75  105  175
coef(res)
# [1,] 0.02540872  0.08389551  0.5284425 -0.4980768
# [2,] 0.57222684 -0.45430385 -0.5203319 -0.4581678
```

cpss.lm *Detecting changes in linear models*

Description

Detecting changes in linear models

Usage

```
cpss.lm(
  formula,
  data = NULL,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2
)
```

Arguments

`formula`	a `formula` object specifying the GLM with change-points.
`data`	an optional data frame containing the variables in the model.
`algorithm`	a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.
`dist_min`	an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

Value

`cpss.lm` returns an object of an S4 class, called "`cpss`", which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500):1590–1598.

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

See Also

`cpss.glm`

Examples

```
library("cpss")
set.seed(666)
n <- 400
tau <- c(80, 200, 300)
tau_ext <- c(0, tau, n)
be <- list(c(0, 1), c(1, 0.5), c(0, 1), c(-1, 0.5))
seg_len <- diff(c(0, tau, n))
x <- rnorm(n)
mu <- lapply(seq(1, length(tau) + 1), function(k) {
  be[[k]][1] + be[[k]][2] * x[(tau_ext[k] + 1):tau_ext[k + 1]]
})
mu <- do.call(c, mu)
sig <- unlist(lapply(seq(1, length(tau) + 1), function(k) {
  rep(be[[k]][2], seg_len[k])
}))
y <- rnorm(n, mu, sig)
res <- cpss.lm(
  formula = y ~ x,
  algorithm = "BS", ncps_max = 10
)
summary(res)
# 80  202  291
coef(res)
# $coef
#             [,1]      [,2]        [,3]       [,4]
# [1,] -0.00188792 1.0457718 -0.03963209 -0.9444813
# [2,]  0.91061557 0.6291965  1.20694409  0.4410036
#
# $sigma
# [1] 0.8732233 0.4753216 0.9566516 0.4782329
```

cpss.mean *Detecting changes in mean*

Description

Detecting changes in mean

Usage

```
cpss.mean(
  dataset,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2,
  Sigma = NULL
)
```

Arguments

`dataset` a numeric matrix of dimension $n \times d$, where each row represents an observation and each column stands for a variable. A numeric vector is also acceptable for univariate observations.

`algorithm` a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.

`dist_min` an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

`Sigma` if a numeric matrix (or constant) is supplied, it will be taken as the value of the common covariance (or variance). By default it is `NULL`, and the covariance is estimated by

$$\widehat{\Sigma} = \frac{1}{2(n-1)} \sum_{i=1}^{n-1} (Y_i - Y_{i+1})(Y_i - Y_{i+1})';$$

Value

`cpss.mean` returns an object of an S4 class, called "cpss", which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500): 1590–1598.

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

See Also

`cpss.meanvar` `cpss.var`

Examples

```
library("cpss")
set.seed(666)
n <- 2048
tau <- c(205, 267, 308, 472, 512, 820, 902, 1332, 1557, 1598, 1659)
seg_len <- diff(c(0, tau, n))
mu <- rep(c(0, 14.64, -3.66, 7.32, -7.32, 10.98, -4.39, 3.29, 19.03, 7.68, 15.37, 0), seg_len)
ep <- 7 * rnorm(n)
y <- mu + ep

res <- cpss.mean(y, algorithm = "SN", ncps_max = 20)
summary(res)
# 205  267  307  471  512  820  897  1332  1557  1601  1659
plot(res, type = "scatter")
plot(res, type = "path")
out <- update(res, dim_update = 12)
out@cps
# 205  267  307  471  512  820  897 1332 1557 1601 1659 1769
# coef(out)
```

cpss.meanvar *Detecting changes in mean and (co)variance*

Description

Detecting changes in mean and (co)variance

Usage

```
cpss.meanvar(
  dataset,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2
)
```

Arguments

`dataset`	a numeric matrix of dimension $n \times d$, where each row represents an observation and each column stands for a variable. A numeric vector is also acceptable for univariate observations.
`algorithm`	a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.
`dist_min`	an integer specifying minimum searching distance (length of feasible segments).
`ncps_max`	an integer specifying an upper bound of the number of true change-points.
`pelt_pen_val`	a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.
`pelt_K`	a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).
`wbs_nintervals`	an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

Value

`cpss.meanvar` returns an object of an S4 class, called "cpss", which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500):1590–1598. Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. The Annals of Statistics, 42(6): 2243–2281.

See Also

`cpss.mean` `cpss.var`

Examples

```
library("cpss")
if (!requireNamespace("MASS", quietly = TRUE)) {
  stop("Please install the package \"MASS\".")
}
set.seed(666)
n <- 1000
tau <- c(200, 400, 600, 800)
mu <- list(rep(0, 2), rep(1, 2), rep(1, 2), rep(0, 2), rep(0, 2))
Sigma <- list(diag(2), diag(2), matrix(c(1,-1,-1, 4), 2), matrix(c(1, 0.5, 0.5, 1), 2), diag(2))
seg_len <- diff(c(0, tau, n))
y <- lapply(seq(1, length(tau) + 1), function(k) {
  MASS::mvrnorm(n = seg_len[k], mu = mu[[k]], Sigma = Sigma[[k]])
})
y <- do.call(rbind, y)
res <- cpss.meanvar(y, algorithm = "BS", dist_min = 20)
cps(res)
# [1] 211 402 598 804
plot(res, type = "coef")
```

`cpss.var` *Detecting changes in (co)variance*

Description

Detecting changes in (co)variance

Usage

```
cpss.var(
  dataset,
  algorithm = "BS",
  dist_min = floor(log(n)),
  ncps_max = ceiling(n^0.4),
  pelt_pen_val = NULL,
  pelt_K = 0,
  wbs_nintervals = 500,
  criterion = "CV",
  times = 2,
  mu = NULL
)
```

Arguments

`dataset` a numeric matrix of dimension $n \times d$, where each row represents an observation and each column stands for a variable. A numeric vector is also acceptable for univariate observations.

`algorithm` a character string specifying the change-point searching algorithm, one of the following choices: "SN" (segment neighborhood), "BS" (binary segmentation), "WBS" (wild binary segmentation) and "PELT" (pruned exact linear time) algorithms.

`dist_min` an integer specifying minimum searching distance (length of feasible segments).

`ncps_max` an integer specifying an upper bound of the number of true change-points.

`pelt_pen_val` a numeric vector specifying candidate values of the penalty only if `algorithm = "PELT"`.

`pelt_K` a numeric value for pruning adjustment only if `algorithm = "PELT"`. It is usually taken to be 0 if the negative log-likelihood is used as a cost, see Killick et al. (2012).

`wbs_nintervals` an integer specifying the number of random intervals drawn only if `algorithm = "WBS"`, see Fryzlewicz (2014).

`criterion` a character string specifying the model selection criterion, "CV" ("cross-validation") or "MS" ("multiple-splitting").

`times` an integer specifying how many times of sample-splitting should be performed; It should be 2 if `criterion = "CV"`.

`mu` If a numeric vector or constant is supplied, it will be taken as the value of the common mean. By default it is `NULL`, and the mean is estimated by the sample mean.

Value

`cpss.var` returns an object of an S4 class, called "cpss", which collects data and information required for further change-point analyses and summaries. See `cpss.custom`.

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association, 107(500): 1590–1598.
Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. The Annals of Statistics, 42(6): 2243–2281.

See Also

cpss.meanvar cpss.mean

Examples

```
library("cpss")
if (!requireNamespace("MASS", quietly = TRUE)) {
  stop("Please install the package \"MASS\".")
}
set.seed(666)
n <- 1000
tau <- c(200, 500, 750)
mu <- list(rep(0, 2), rep(0, 2), rep(0, 2), rep(0, 2))
Sigma <- list(diag(2), matrix(c(1, 0, 0, 4), 2), matrix(c(1, -0.5, -0.5, 4), 2), diag(2))
seg_len <- diff(c(0, tau, n))
y <- lapply(seq(1, length(tau) + 1), function(k) {
  MASS::mvrnorm(n = seg_len[k], mu = mu[[k]], Sigma = Sigma[[k]])
})
y <- do.call(rbind, y)
res <- cpss.var(y, algorithm = "BS", dist_min = 20)
cps(res)
# [1] 215 515 751
```

dat *Generic functions and methods: dat*

Description

Generic functions and methods: dat

Usage

```
dat(x)

dat(x) <- value

## S4 method for signature 'cpss'
dat(x)

## S4 replacement method for signature 'cpss'
dat(x) <- value
```

Arguments

x	object from cpss
value	value assigned to x
cpss	cpss class

mdl *Generic functions and methods: mdl*

Description

Generic functions and methods: mdl

Usage

```
mdl(x)

mdl(x) <- value

## S4 method for signature 'cpss'
mdl(x)

## S4 replacement method for signature 'cpss'
mdl(x) <- value
```

Arguments

x	object from cpss
value	value assigned to x

ncps *Generic functions and methods: ncps*

Description

Generic functions and methods: ncps

Usage

```
ncps(x)

ncps(x) <- value

## S4 method for signature 'cpss'
ncps(x)

## S4 replacement method for signature 'cpss'
ncps(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`params` *Generic functions and methods: params*

Description

Generic functions and methods: params

Usage

```
params(x)

params(x) <- value

## S4 method for signature 'cpss'
params(x)

## S4 replacement method for signature 'cpss'
params(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`pelt_pen` *Generic functions and methods: pelt_pen*

Description

Generic functions and methods: pelt_pen

Usage

```
pelt_pen(x)

pelt_pen(x) <- value

## S4 method for signature 'cpss'
pelt_pen(x)

## S4 replacement method for signature 'cpss'
pelt_pen(x) <- value
```

Arguments

x	object from cpss
value	value assigned to x

plot,cpss-method *plot method*

Description

plot method

Usage

```
## S4 method for signature 'cpss'
plot(obj, type, x = c(), y = c(), ...)
```

Arguments

obj	object from cpss
type	type of visualization
x	x
y	y
...	...
cpss	cpss class

SC *Generic functions and methods: SC*

Description

Generic functions and methods: SC

Usage

```
SC(x)

SC(x) <- value

## S4 method for signature 'cpss'
SC(x)

## S4 replacement method for signature 'cpss'
SC(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`SC_vals` *Generic functions and methods: SC_vals*

Description

Generic functions and methods: SC_vals

Usage

```
SC_vals(x)

SC_vals(x) <- value

## S4 method for signature 'cpss'
SC_vals(x)

## S4 replacement method for signature 'cpss'
SC_vals(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`summary,cpss-method` *summary method*

Description

summary method

Usage

```
## S4 method for signature 'cpss'
summary(object)
```

Arguments

`object`	object from cpss
`cpss`	cpss class

S_vals *Generic functions and methods: S_vals*

Description

Generic functions and methods: S_vals

Usage

```
S_vals(x)

S_vals(x) <- value

## S4 method for signature 'cpss'
S_vals(x)

## S4 replacement method for signature 'cpss'
S_vals(x) <- value
```

Arguments

x	object from cpss
value	value assigned to x

update,cpss-method *update method*

Description

update method

Usage

```
## S4 method for signature 'cpss'
update(object, dim_update)
```

Arguments

object	object from cpss
dim_update	model dimension to update
cpss	cpss class

`update_inputs` *Generic functions and methods: update_inputs*

Description

Generic functions and methods: update_inputs

Usage

```
update_inputs(x)

update_inputs(x) <- value

## S4 method for signature 'cpss'
update_inputs(x)

## S4 replacement method for signature 'cpss'
update_inputs(x) <- value
```

Arguments

`x`	object from cpss
`value`	value assigned to x

`well` *Well-log data*

Description

Measurements of the nuclear magnetic response of underground rocks.

Usage

```
well
```

Format

A vector of 4,050 measurements:

well Measurements.

Source

doi:10.1111/14679868.00421

Index