Package 'blockseg'

April 18, 2019

Type Package

Title Two Dimensional Change-Points Detection

Version 0.5.2

Date 2019-04-15

Maintainer Vincent Brault <vincent.brault@univ-grenoble-alpes.fr>

Description Segments a matrix in blocks with constant values.
BRAULT V, CHIQUET J. and LEVY-LEDUC C. (2017) <doi:10.1214/17-EJS1270>.

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

URL https://github.com/jchiquet/blockseg (dev version)

Imports Rcpp, Matrix, ggplot2, reshape2, methods, stats, shiny

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

NeedsCompilation yes

Author Julien Chiquet [aut] (<https://orcid.org/0000-0002-3629-3429>),
Vincent Brault [aut, cre] (<https://orcid.org/0000-0003-4228-6687>)

Repository CRAN

Date/Publication 2019-04-18 19:10:03 UTC

R topics documented:

blockseg-package 2
blockSeg 3
blockSeg-class 3
criteria 5
evolution 6
plot,blockSeg-method 7
plot,stab.blockSeg-method 8
predict,blockSeg-method 9

rblockdata 10
stab.blockSeg 11
stab.blockSeg-class 12

Index **13**

`blockseg-package` *blockseg package*

Description

This package is designed to segment a matrix in blocks with constant values.

Features

Package for the segmentation of the rows and columns inducing a grid.

Algorithm

`blockSeg`, `stab.blockSeg`

Technical remarks

Display of the result with `plot,blockSeg-method` and `plot,stab.blockSeg-method` and the evolution with `predict,blockSeg-method` and `evolution,stab.blockSeg-method`.

Author(s)

Julien Chiquet <julien.chiquet@gmail.com>

Vincent Brault <vincent.brault@univ-grenoble-alpes.fr>

References

BRAULT V, CHIQUET J. and LEVY-LEDUC C.Efficient block boundaries estimation in blockwise constant matrices: An application to HiC data, Electronic Journal of Statistics, Volume 11, Number 1 (2017), 1570-1599 <doi:10.1214/17-EJS1270>.

blockSeg blockSeg *fitting procedure*

Description

Produce a block-wise estimation of a matrix.

Usage

```
blockSeg(Y, max.break = floor(min(ncol(Y), nrow(Y))/10 + 1),
  max.var = floor(ncol(Y)^2/2), verbose = TRUE, Beta = FALSE)
```

Arguments

Y	matrix of observations.
max.break	a positive integer less than number of columns and number of rows. By default, floor(min(ncol(Y),nrow(Y))/10+1).
max.var	a positive integer less than number of columns times number of rows. By default, ncol(Y)**2/2.
verbose	logical. To display each step. By default TRUE.
Beta	logical. To save each Beta associated at each lambda. By default FALSE (very heavy in memory space).

Examples

```
 ## model parameters
n <- 100
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
res <- blockSeg(Y, 50)
```

blockSeg-class *Class* blockSeg

Description

Class of object returned by the blockSeg function.

Usage

```
## S4 method for signature 'blockSeg'
print(x, ...)

## S4 method for signature 'blockSeg'
show(object)

getComplexity(object)

## S4 method for signature 'blockSeg'
getComplexity(object)

## S4 method for signature 'blockSeg'
residuals(object, Y)

## S4 method for signature 'blockSeg'
deviance(object, Y)

getBreaks(object)

## S4 method for signature 'blockSeg'
getBreaks(object)

getCompressYhat(object, Y)

## S4 method for signature 'blockSeg'
getCompressYhat(object, Y)
```

Arguments

x	in the print method, a blockSeg object
...	in the print method, additional parameters (ignored)
object	an object with class blockSeg
Y	the original data matrix

Slots

Beta a Matrix object of type dgCMatrix, encoding the solution path of the underlying LARS algorithm. Omitted if the blockSeg function was called with the option Beta=FALSE.

Lambda a numeric vector with the successive values of Lambda, that is, the value of the penalty parameter corresponding to a new event in the path (either a variable activation or deactivation).

RowBreaks a list of vectors, one per step of the LARS algorithm. Each vector contains the breaks currently identified along the ROWS of the 2-dimensional signal at the current step.

ColBreaks a list of vectors, one per step of the LARS algorithm. Each vector contains the breaks currently identified along the COLUMNS of the 2-dimensional signal at the current step.

Actions a list with the successive actions at each step of the LARS algorithm.

See Also

See also plot,blockSeg-method, predict,blockSeg-method and blockSeg.

---

criteria *Penalized criteria based on estimation of degrees of freedom*

Description

Produce a plot or send back the values of some penalized criteria accompanied with the vector(s) of parameters selected accordingly. The default behavior plots the BIC and the AIC (with respective factor $\log(n)$ and 2) yet the user can specify any penalty.

Usage

```
criteria(object, Y, penalty = setNames(c(2, log(length(Y))), c("AIC",
  "BIC")), sigma = NULL, log.scale = TRUE, xvar = "lambda",
  plot = TRUE)

## S4 method for signature 'blockSeg'
criteria(object, Y, penalty = setNames(c(2,
  log(length(Y))), c("AIC", "BIC")), sigma = NULL, log.scale = TRUE,
  xvar = "lambda", plot = TRUE)
```

Arguments

object	output of a fitting procedure of the **blockseg** package (e.g. blockSeg). Must be of class blockSeg.
Y	matrix of observations.
penalty	a vector with as many penalties a desired. The default contains the penalty corresponding to the AIC and the BIC (2 and $\log(n)$). Setting the "names" attribute, as done in the default definition, leads to outputs which are easier to read.
sigma	scalar: an estimate of the residual variance. When available, it is plugged-in the criteria, which may be more relevant. If NULL (the default), it is estimated as usual (see details).
log.scale	logical; indicates if a log-scale should be used when xvar="lambda". Default is TRUE.
xvar	variable to plot on the X-axis: either "df" (the estimated degrees of freedom), "lambda" (λ_1 penalty level) or "fraction" (ℓ_1-norm of the coefficients). Default is set to "lambda".
plot	logical; indicates if the graph should be plotted on call. Default is TRUE.

Value

When `plot` is set to `TRUE`, an invisible **ggplot2** object is returned, which can be plotted via the `print` method. On the other hand, a list with a two data frames containing the criteria and the chosen vector of parameters are returned.

Note

When `sigma` is provided, the criterion takes the form

$$\left\|\mathbf{y} - \mathbf{X}\hat{\beta}\right\|^2 + \text{penalty} \times \frac{\hat{\text{df}}}{n}\,\sigma^2.$$

When it is unknown, it writes

$$\log\left(\left\|\mathbf{y} - \mathbf{X}\hat{\beta}\right\|^2\right) + \text{penalty} \times \hat{\text{df}}.$$

Estimation of the degrees of freedom (for the elastic-net, the LASSO and also bounded regression) are computed by applying and adapting the results of Tibshirani and Taylor (see references below).

References

Ryan Tibshirani and Jonathan Taylor. Degrees of freedom in lasso problems, Annals of Statistics, 40(2) 2012.

See Also

`blockSeg`.

Examples

```
n <- 100
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
res <- blockSeg(Y, 50)
criteria(res, Y, sigma=.5)
```

`evolution` *Plot method for a* `stab.blockSeg` *object*

Description

Produce a plot of two-dimensional segmentation of a `stab.blockSeg` fit.

Usage

```
evolution(x, y, thresholds = 10 * (8:1), postprocessing = list(post =
  TRUE, adjacent = 2), col = "GrayLevel", ask = TRUE)

## S4 method for signature 'stab.blockSeg'
evolution(x, y, thresholds = 10 * (8:1),
  postprocessing = list(post = TRUE, adjacent = 2), col = "GrayLevel",
  ask = TRUE)
```

Arguments

x	an object of class stab.blockSeg.
y	the observations data (or a transformation).
thresholds	the thresholds used (percent the maximum value). By default, thresholds = 10 * (8:1).
postprocessing	the condition if plot used a post-processing (if $post=TRUE) or not. If there is a post-processing, post-processing$adjacent is the maximal distance between two points.
col	colors of the graphics. By default, it is "GrayLevel" to black and white colors. If it is another "character", it is a level blue or red. Else, it is possible to propose a sequence with the color (rgb format).
ask	If TRUE, to hit will be necessary to see next plot.
...	used for S4 compatibility.

See Also

stab.blockSeg.

Examples

```
n <- 100
## model parameters
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
stab.out <- stab.blockSeg(Y, 100, 15)
evolution(stab.out,Y)
```

---

plot,blockSeg-method *Plot method for a blockSeg object*

Description

Produce a plot of two-dimensional segmentation of a blockSeg fit.

Usage

```
## S4 method for signature 'blockSeg'
plot(x, y, lambda = NULL, ask = TRUE,
  col = "GrayLevel", ...)
```

Arguments

x	an object of class blockSeg.
y	used for S4 compatibility.
lambda	parameter used in the LASSO.
ask	If TRUE, to hit will be necessary to see next plot.
col	for the colors of the representations
...	used for S4 compatibility.

Value

a **ggplot2** object which can be plotted via the `print` method.

See Also

blockSeg.

---

plot,stab.blockSeg-method

Plot method for a `stab.blockSeg` *object*

Description

Produce a plot of two-dimensional segmentation of a `stab.blockSeg` fit.

Usage

```
## S4 method for signature 'stab.blockSeg'
plot(x, y, threshold = 40,
  postprocessing = list(post = TRUE, adjacent = 2), col = "GrayLevel",
  shiny = FALSE, ...)
```

Arguments

x	an object of class `stab.blockSeg`.
y	the observations data (or a transformation).
threshold	the threshold used (percent the maximum value).
postprocessing	the condition if plot used a post-processing (if $post=TRUE) or not. If there is a post-processing, post-processing$adjacent is the maximal distance between two points.

col	colors of the graphics. By default, it is "GrayLevel" to black and white colors. If it is another "character", it is a level blue or red. Else, it is possible to propose a sequence with the color (rgb format).
shiny	for a representation with a shiny application. By default shiny=FALSE
...	used for S4 compatibility.

See Also

stab.blockSeg.

stab.blockSeg.

Examples

```
## Not run:
n <- 100
## model parameters
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
stab.out <- stab.blockSeg(Y, 100, 15)
plot(stab.out,Y)

## End(Not run)
```

---

predict,blockSeg-method

Predict method for a blockSeg *object*

Description

Produce a prediction for a vector of lambda parameter and an array of class.

Usage

```
## S4 method for signature 'blockSeg'
predict(object, Y, lambda = NULL)
```

Arguments

object	an object of class blockSeg.
Y	matrix of observations.
lambda	a numeric vector giving the list of λ for which to predict. By default, NULL. If NULL, it is set to the lambdalist slot of object. If this slot is empty, lambda is set to the fusion times detected in the blockSeg function.

See Also

blockSeg.

Examples

```
require(blockseg)
n <- 100
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
res <- blockSeg(Y, 100)
predict(res, Y, lambda=slot(res, "Lambda")[1:3])
```

`rblockdata` *Random generation noisy block-wise matrices*

Description

Function to draw data.

Usage

```
rblockdata(n, mu, sigma, type = c("Eq", "NEq", "NEqbis"))
```

Arguments

`n`	number of rows and columns.
`mu`	symmetric matrix to the means.
`sigma`	variance of the variables.
`type`	represent the spacing between two change-point: "`Eq`" for a homogenous spacing, "`NEq`" for an arithmetic spacing and "`NEqbis`" for a decreasing arithmetic spacing.

Examples

```
## model parameters
n <- 100
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)
```

stab.blockSeg stab.blockSeg *algorithm*

Description

Model selection for the blockSeg algorithm.

Usage

```
stab.blockSeg(Y, nsimu, max.break, max.var = floor(ncol(Y)^2/8),
  random.break = TRUE, sym.break = FALSE, mc.cores = 2,
  verbose = TRUE)
```

Arguments

Y	matrix of observations.
nsimu	a positive integer.
max.break	a positive integer less than number of columns divided by 2 and number of rows divided by 2.
max.var	a positive integer less than number of columns times number of rows. By default, ncol(Y)**2/8.
random.break	logical. To change the position of the first row (resp. column); the rows before this position are moved to the end. By default TRUE.
sym.break	logical. In the case of symmetric matrices, it is possible to accumulate breaks in row and columns to improve the quality of the estimation. By default FALSE. Warning: a check is made on the dimensions of the matrix but not on the fact that it is symmetrical or not; this choice was made for the case where the user would like to have symmetrical breaks even if the matrix is not (not recommended by the authors of the package).
mc.cores	a positive integer giving the number of cores used. If you use windows, the parallelization is impossible. By default, 2
verbose	logical. To display each step. By default TRUE.

Examples

```
 ## model parameters
n <- 100
K <- 5
mu <- suppressWarnings(matrix(rep(c(1,0),ceiling(K**2/2)), K,K))
Y <- rblockdata(n,mu,sigma=.5)$Y
res <- stab.blockSeg(Y, 100, 20)
```

stab.blockSeg-class *Class* stab.blockSeg

Description

Class of object returned by the stab.blockSeg function.

Slots

RowBreaks: a vectors of length the number of rows. Each case contains the number of active variable identified along the stability selection.

ColBreaks: a vectors of length the number of columns. Each case contains the number of active variable identified along the stability selection.

Methods

Specific plotting and predict methods are available and documented (plot,stab.blockSeg-method, evolution,stab.blockSeg-method.

See Also

See also plot,stab.blockSeg-method, evolution,stab.blockSeg-method print,blockSeg-method and stab.blockSeg.

Index

∗Topic **class**
    stab.blockSeg-class, 12

blockSeg, *2*, 3, *4–9*
blockSeg-class, 3
blockseg-package, 2

criteria, 5
criteria,blockSeg-method *(*criteria*)*, 5

deviance,blockSeg-method
        *(*blockSeg-class*)*, 3

evolution, 6
evolution,stab.blockSeg-method
        *(*evolution*)*, 6

getBreaks *(*blockSeg-class*)*, 3
getBreaks,blockSeg-method
        *(*blockSeg-class*)*, 3
getComplexity *(*blockSeg-class*)*, 3
getComplexity,blockSeg-method
        *(*blockSeg-class*)*, 3
getCompressYhat *(*blockSeg-class*)*, 3
getCompressYhat,blockSeg-method
        *(*blockSeg-class*)*, 3

plot,blockSeg-method, 7
plot,stab.blockSeg-method, 8
predict,blockSeg-method, 9
print,blockSeg-method *(*blockSeg-class*)*,
        3
print,stab.blockSeg-method
        *(*stab.blockSeg-class*)*, 12

rblockdata, 10
residuals,blockSeg-method
        *(*blockSeg-class*)*, 3
rgb, *7*, *9*

show,blockSeg-method *(*blockSeg-class*)*, 3
show,stab.blockSeg-method,
        *(*stab.blockSeg-class*)*, 12
stab.blockSeg, *2*, *7*, *9*, 11, *12*
stab.blockSeg-class, 12