Package: wbs (via r-universe)

August 25, 2024

Type Package

Title Wild Binary Segmentation for Multiple Change-Point Detection

Version 1.4

Date 2019-04-17

Author Rafal Baranowski and Piotr Fryzlewicz

Maintainer Rafal Baranowski <package_maintenance@rbaranowski.com>

Depends graphics

Description Provides efficient implementation of the Wild Binary Segmentation and Binary Segmentation algorithms for estimation of the number and locations of multiple change-points in the piecewise constant function plus Gaussian noise model.

License GPL-2

NeedsCompilation yes

Repository CRAN

RoxygenNote 6.1.1

Date/Publication 2019-05-14 21:40:03 UTC

Contents

wbs-package 2
bic.penalty 3
changepoints 3
fixed.intervals 5
mbic.penalty 6
means.between.cpt 7
plot.sbs 8
plot.wbs 9
print.sbs 9
print.wbs 10
random.intervals 10
sbs 11
ssic.penalty 12
wbs 13

Index 15

`wbs-package` *Wild Binary Segmentation for multiple change-point detection*

Description

The package implements Wild Binary Segmentation, a technique for consistent estimation of the number and locations of multiple change-points in data. It also provides a fast implementation of the standard Binary Segmentation algorithm.

Details

The main routines of the package are `wbs`, `sbs` and `changepoints`.

References

P. Fryzlewicz (2014), Wild Binary Segmentation for multiple change-point detection. Annals of Statistics, to appear. (http://stats.lse.ac.uk/fryzlewicz/wbs/wbs.pdf)

Examples

```
#an example in which standard Binary Segmentation fails to detect change points
x <- rnorm(300)+ c(rep(0,130),rep(-1,20),rep(1,20),rep(0,130))

s <- sbs(x)
w <- wbs(x)

s.cpt <- changepoints(s)
s.cpt

w.cpt <- changepoints(w)
w.cpt
# in this example, both algorithms work well
x <- rnorm(300) + c(rep(1,50),rep(0,250))

s <- sbs(x)
w <- wbs(x)

s.cpt <- changepoints(s)
s.cpt

w.cpt <- changepoints(w)
w.cpt
```

`bic.penalty` *Bayesian Information Criterion penalty term*

Description

The function evaluates the penalty term for the standard Bayesian Information Criterion applied to the change-point detection problem. This routine is typically not called directly by the user; its name can be passed as an argument to `changepoints`.

Usage

```
bic.penalty(n, cpt)
```

Arguments

`n`	the number of observations
`cpt`	a vector with localisations of change-points

Value

the penalty term $k \log(n)$ where k denotes the number of elements in `cpt`

Examples

```
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- wbs(x)
w.cpt <- changepoints(w,penalty="bic.penalty")
w.cpt$cpt.ic
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- wbs(x)
w.cpt <- changepoints(w,penalty="bic.penalty")
w.cpt$cpt.ic
```

`changepoints` *Change-points detected by WBS or BS*

Description

The function applies user-specified stopping criteria to extract change-points from `object` generated by `wbs` or `sbs`. For `object` of class 'sbs', the function returns change-points whose corresponding test statistic exceeds threshold given in `th`. For `object` of class 'wbs', the change-points can be also detected using information criteria with penalties specified in `penalty`.

Usage

```
changepoints(object, ...)

## S3 method for class 'sbs'
changepoints(object, th = NULL, th.const = 1.3,
  Kmax = NULL, ...)

## S3 method for class 'wbs'
changepoints(object, th = NULL, th.const = 1.3,
  Kmax = 50, penalty = c("ssic.penalty", "bic.penalty",
  "mbic.penalty"), ...)
```

Arguments

`object`	an object of 'wbs' or 'sbs' class returned by, respectively, `wbs` and `sbs` functions
`...`	further arguments that may be passed to the penalty functions
`th`	a vector of positive scalars
`th.const`	a vector of positive scalars
`Kmax`	a maximum number of change-points to be detected
`penalty`	a character vector with names of penalty functions used

Details

For the change-point detection based on thresholding (`object` of class 'sbs' or 'wbs'), the user can either specify the thresholds in `th` directly, determine the maximum number `Kmax` of change-points to be detected, or let `th` depend on `th.const`.

When `Kmax` is given, the function automatically sets `th` to the lowest threshold such that the number of detected change-points is lower or equal than `Kmax`. Note that for the BS algorithm it might be not possible to find the threshold such that exactly `Kmax` change-points are found.

When `th` and `Kmax` are omitted, the threshold value is set to

$$th = sigma \times th.const\sqrt{2\log(n)},$$

where sigma is the Median Absolute Deviation estimate of the noise level and n is the number of elements in `x`.

For the change-point detection based on information criteria (`object` of class 'wbs' only), the user can specify both the maximum number of change-points (`Kmax`) and a type of the penalty used. Parameter `penalty` should contain a list of characters with names of the functions of at least two arguments (`n` and `cpt`). For each penalty given, the following information criterion is minimized over candidate sets of change-points `cpt`:

$$\frac{n}{2}\log\hat{\sigma}_k^2 + penalty(n, cpt),$$

where k denotes the number of elements in cpt, $\hat{\sigma}_k$ is the corresponding maximum likelihood estimator of the residual variance.

Value

sigma	Median Absolute Deviation estimate of the noise level
th	a vector of thresholds
no.cpt.th	the number of change-points detected for each value of th
cpt.th	a list with the change-points detected for each value of th
Kmax	a maximum number of change-points detected
ic.curve	a list with values of the chosen information criteria
no.cpt.ic	the number of change-points detected for each information criterion considered
cpt.ic	a list with the change-points detected for each information criterion considered

Examples

```
#we generates  gaussian noise + Poisson process signal with 10 jumps on average
set.seed(10)
N <- rpois(1,10)
true.cpt <- sample(1000,N)
m1 <- matrix(rep(1:1000,N),1000,N,byrow=FALSE)
m2 <- matrix(rep(true.cpt,1000),1000,N,byrow=TRUE)
x <- rnorm(1000) + apply(m1>=m2,1,sum)

# we apply  the BS and WBS algorithms with default values for their parameters

s <- sbs(x)
w <- wbs(x)

s.cpt <- changepoints(s)
s.cpt

w.cpt <- changepoints(w)
w.cpt

#we can use different stopping criteria, invoking sbs/wbs functions is not necessary

s.cpt <- changepoints(s,th.const=c(1,1.3))
s.cpt
w.cpt <- changepoints(w,th.const=c(1,1.3))
w.cpt
```

---

fixed.intervals *Fixed intervals*

Description

The function generates approximately M intervals with endpoints in 1,2,...,n, without random drawing. This routine can be used inside wbs function and is typically not called directly by the user.

Usage

```
fixed.intervals(n, M)
```

Arguments

n	a number of endpoints to choose from
M	a number of intervals to generate

Details

Function finds the minimal m such that $M \leq \frac{m(m-1)}{2}$. Then it generates m approximately equally-spaced positive integers lower than n and returns all possible intervals consisting of any two of these points.

Value

a 2-column matrix with start (first column) and end (second column) points of an interval in each row

See Also

random.intervals wbs

Examples

```
fixed.intervals(10,100)
```

---

mbic.penalty *Modified Bayes Information Criterion penalty term*

Description

The function evaluates the penalty term for the Modified Bayes Information Criterion proposed in N. Zhang and D. Siegmund (2007). This routine is typically not called directly by the user; its name can be passed as an argument to changepoints.

Usage

```
mbic.penalty(n, cpt)
```

Arguments

n	the number of observations
cpt	a vector with localisations of change-points

Value

the penalty term

$$\frac{3}{2}k\log(n) + \frac{1}{2}\sum_{i=1}^{k+1}\log\frac{l_i}{n},$$

where k denotes the number of elements in `cpt` and l_i are the lengths of the intervals between changepoints in `cpt`

References

N. Zhang and D. Siegmund (2007), A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data, Biometrics.

Examples

```
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- wbs(x)
w.cpt <- changepoints(w,penalty="mbic.penalty")
w.cpt$cpt.ic
```

`means.between.cpt` *Means between change-points*

Description

The function finds the average of the input vector `x` between change-points given in `cpt`.

Usage

```
means.between.cpt(x, cpt = NULL, ...)
```

Arguments

`x`	a vector
`cpt`	a vector of integers with localisations of change-points
`...`	further arguments passed to `mean` method

Value

a vector of the same length as `x`, piecewise constant and equal to the mean between change-points given in `cpt`

Examples

```
x <- rnorm(100)+c(rep(-1,50),rep(1,50))
cpt <- 50
means.between.cpt(x,cpt)
w <- wbs(x)
cpt <- changepoints(w)
means.between.cpt(x,cpt=cpt$cpt.ic$sbic)
```

---

plot.sbs *Plot for an 'sbs' object*

Description

Plots the input vector used to generate 'sbs' object `x` with fitted piecewise constant function, equal to the mean between change-points specified in `cpt`.

Usage

```
## S3 method for class 'sbs'
plot(x, cpt, ...)
```

Arguments

`x`	an object of class 'sbs', returned by `sbs`
`cpt`	a vector of integers with localisations of change-points
`...`	other parameters which may be passed to `plot` and `changepoints`

Details

When `cpt` is omitted, the function automatically finds change-points using `changepoints` function with a default value of the threshold.

See Also

`sbs` `changepoints`

plot.wbs — *Plot for a 'wbs' object*

Description

Plots the input vector used to generate 'wbs' object `x` with fitted piecewise constant function, equal to the mean between change-points specified in `cpt`.

Usage

```
## S3 method for class 'wbs'
plot(x, cpt, ...)
```

Arguments

`x`	an object of class 'wbs', returned by `wbs`
`cpt`	a vector of integers with localisations of change-points
`...`	other parameters which may be passed to `plot` and `changepoints`

Details

When `cpt` is omitted, the function automatically finds change-points using `changepoints` function with strengthened Schwarz Information Criterion as a stopping criterion for the WBS algorithm.

See Also

`wbs` `changepoints` `ssic.penalty`

print.sbs — *Print for an 'sbs' object*

Description

Print for an 'sbs' object

Usage

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

Arguments

`x`	an object of class 'sbs'
`...`	further arguments passed to `print` method

See Also

`sbs`

`print.wbs` *Print for a 'wbs' object*

Description

Print for a 'wbs' object

Usage

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

Arguments

`x`	an object of class 'wbs'
`...`	further arguments passed to `print` method

See Also

`wbs`

`random.intervals` *Random intervals*

Description

The function generates `M` intervals, whose endpoints are are drawn uniformly without replacements from 1,2,..., `n`. This routine can be used inside `wbs` function and is typically not called directly by the user.

Usage

```
random.intervals(n, M)
```

Arguments

`n`	a number of endpoints to choose from
`M`	a number of intervals to generate

Value

a `M` by 2 matrix with start (first column) and end (second column) points of an interval in each row

See Also

`fixed.intervals` `wbs`

Examples

```
random.intervals(10,100)
```

---

sbs   *Change-point detection via standard Binary Segmentation*

Description

The function applies the Binary Segmentation algorithm to identify potential locations of the change-points in the mean of the input vector `x`. The object returned by this routine can be further passed to the `changepoints` function, which finds the final estimate of the change-points based on thresholding.

Usage

```
sbs(x, ...)

## Default S3 method:
sbs(x, ...)
```

Arguments

`x`	a numeric vector
`...`	not in use

Value

an object of class "sbs", which contains the following fields

`x`	the vector provided
`n`	the length of `x`
`res`	a 6-column matrix with results, where 's' and 'e' denote start- end points of the intervals in which change-points candidates 'cpt' have been found; column 'CUSUM' contains corresponding value of CUSUM statistic; 'min.th' is the smallest threshold value for which given change-point candidate would be not added to the set of estimated change-points; the last column is the scale at which the change-point has been found

Examples

```
x <- rnorm(300) + c(rep(1,50),rep(0,250))
s <- sbs(x)
s.cpt <- changepoints(s)
s.cpt
th <- c(s.cpt$th,0.7*s.cpt$th)
s.cpt <- changepoints(s,th=th)
s.cpt
```

`ssic.penalty` *Strengthened Schwarz Information Criterion penalty term*

Description

The function evaluates the penalty term for the strengthened Schwarz Information Criterion proposed in P. Fryzlewicz (2014). This routine is typically not called directly by the user; its name can be passed as an argument to `changepoints`.

Usage

```
ssic.penalty(n, cpt, alpha = 1.01, ssic.type = c("log", "power"))
```

Arguments

`n`	the number of observations
`cpt`	a vector with localisations of change-points
`alpha`	a scalar greater than one
`ssic.type`	a string ("log" or "power")

Value

the penalty term $k(\log(n))^{alpha}$ for `ssic.penalty="log"` or kn^{alpha} for `ssic.penalty="power"`, where k denotes the number of elements in `cpt`

References

P. Fryzlewicz (2014), Wild Binary Segmentation for multiple change-point detection. Annals of Statistics, to appear. (http://stats.lse.ac.uk/fryzlewicz/wbs/wbs.pdf)

Examples

```
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- wbs(x)
w.cpt <- changepoints(w,penalty="ssic.penalty")
w.cpt$cpt.ic
```

wbs *Change-point detection via Wild Binary Segmentation*

Description

The function applies the Wild Binary Segmentation algorithm to identify potential locations of the change-points in the mean of the input vector `x`. The object returned by this routine can be further passed to the `changepoints` function, which finds the final estimate of the change-points based on chosen stopping criteria.

Usage

```
wbs(x, ...)

## Default S3 method:
wbs(x, M = 5000, rand.intervals = TRUE,
  integrated = TRUE, ...)
```

Arguments

`x`	a numeric vector
`...`	not in use
`M`	a number of intervals used in the WBS algorithm
`rand.intervals`	a logical variable; if `rand.intervals=TRUE` intervals used in the procedure are random, thus the output of the algorithm may slightly vary from run to run; for `rand.intervals=FALSE` the intervals used depend on `M` and the length of `x` only, hence the output is always the same for given input parameters
`integrated`	a logical variable indicating the version of Wild Binary Segmentation algorithm used; when `integrated=TRUE`, augmented version of WBS is launched, which combines WBS and BS into one

Value

an object of class "wbs", which contains the following fields

`x`	the input vector provided
`n`	the length of `x`
`M`	the number of intervals used
`rand.intervals`	a logical variable indicating type of intervals
`integrated`	a logical variable indicating type of WBS procedure
`res`	a 6-column matrix with results, where 's' and 'e' denote start- end points of the intervals in which change-points candidates 'cpt' have been found; column 'CUSUM' contains corresponding value of CUSUM statistic; 'min.th' is the smallest threshold value for which given change-point candidate would be not added to the set of estimated change-points; the last column is the scale at which the change-point has been found

Examples

```
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- wbs(x)
plot(w)
w.cpt <- changepoints(w)
w.cpt
th <- c(w.cpt$th,0.7*w.cpt$th)
w.cpt <- changepoints(w,th=th)
w.cpt$cpt.th
```

Index

∗ **math**
    wbs-package, 2
∗ **models**
    wbs-package, 2
∗ **ts**
    wbs-package, 2

bic.penalty, 3

changepoints, *2, 3*, 3, *6*, *8, 9*, *11–13*

fixed.intervals, 5, *10*

mbic.penalty, 6
means.between.cpt, 7

plot.sbs, 8
plot.wbs, 9
print.sbs, 9
print.wbs, 10

random.intervals, *6*, 10

sbs, *2–4*, *8, 9*, 11
ssic.penalty, *9*, 12

wbs, *2–6*, *9, 10*, 13
wbs-package, 2