Package 'not'

September 23, 2024

Type Package

Title Narrowest-Over-Threshold Change-Point Detection

Version 1.6

Date 2024-09-23

Depends graphics, stats, splines

Description Provides efficient implementation of the Narrowest-Over-Threshold methodology for detecting an unknown number of change-points occurring at unknown locations in one-dimensional data following 'deterministic signal + noise' model. Currently implemented scenarios are: piecewise-constant signal, piecewise-constant signal with a heavy-tailed noise, piecewise-linear signal, piecewise-quadratic signal, piecewise-constant signal and with piecewise-constant variance of the noise. For details, see Baranowski, Chen and Fryzlewicz (2019) <doi:10.1111/rssb.12322>.

License GPL-2

NeedsCompilation yes

Repository CRAN

Author Rafal Baranowski [aut],
Yining Chen [aut, cre],
Piotr Fryzlewicz [aut]

Maintainer Yining Chen <y.chen101@lse.ac.uk>

Date/Publication 2024-09-23 13:40:28 UTC

Contents

not-package *Narrowest-Over-Threshold Change-Point Detection*

Description

Implements the Narrowest-Over-Threshold approach for general multiple change-point detection in one-dimensional data following 'deterministic signal + noise' model. Scenarios that are currently implemented are: piecewise-constant signal, piecewise-constant signal with a heavy tailed noise, piecewise-linear signal, piecewise-quadratic signal, piecewise-constant signal and with piecewise-constant standard deviation of the noise. The main routines of the package are `not` and `features`.

References

R. Baranowski, Y. Chen, and P. Fryzlewicz (2019). Narrowest-Over-Threshold Change-Point Detection. (`http://stats.lse.ac.uk/fryzlewicz/not/not.pdf`)

aic.penalty *Akaike Information Criterion penalty*

Description

The function evaluates the penalty term for Akaike Information Criterion. This routine is typically not called directly by the user; its name can be passed as an argument to `features`.

Usage

```
aic.penalty(n, n.param, ...)
```

Arguments

`n`	The number of observations.
`n.param`	The number of parameters in the model for which the penalty is evaluated.
`...`	Not in use.

Value

The penalty term $2 \times$ `n.param`.

References

R. Baranowski, Y. Chen, and P. Fryzlewicz (2019). Narrowest-Over-Threshold Change-Point Detection. (`http://stats.lse.ac.uk/fryzlewicz/not/not.pdf`)

Examples

```
#*** a simple example how to use the AIC penalty
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- not(x)
w.cpt <- features(w, penalty="aic")
w.cpt$cpt[[1]]
```

features *Extract locations of features from a 'not' object*

Description

The function applies user-specified stopping criteria to extract change-points from `object` generated by `not`.

Usage

```
features(object, ...)

## Default S3 method:
features(object, method = c("ic", "threshold"),
  penalty = c("sic", "aic", "user"), q.max = 25, penalty.fun, th, ...)
```

Arguments

`object`	An object of 'not' class returned by `not`.
`...`	Further arguments that can be passed to the penalty function.
`method`	A method of choosing the best solution in `object$solution.path`. If `method="ic"`, model minimising a chosen information criterion is selected. If `method="threshold"`, model is selected based on thresholding (see references for more details).
`penalty`	Name of the penalty function to be used if `method="ic"`. If `penalty="user"`, a user-defined penalty function has to be passed via `penalty.fun`.
`q.max`	Maximum number of change-points allowed to be detected. Used only for `method="ic"`.
`penalty.fun`	Used only if `penalty="user"`. A function includes at least the following arguments: sample size `n`, number of parameters used in a model `n.param`, and `...`. For examples of such functions, see `aic.penalty` and `sic.penalty`.
`th`	Used only if `method="threshold"`. A positive real number.

Details

Denote by $T_1, \ldots, T_N$ the elements on the solution path `object$solution.path`, each representing a set of change-points. When (`method="ic"`), the returned set of change-points is the one that minimises

$$-2\text{log-likelihood}(\texttt{object}, \texttt{cpt} = T_k) + \text{penalty}(\texttt{object\$n}, \texttt{n.param}, ...),$$

over all k such that the number of change-points in T_k is smaller than or equal `q.max`. The log-likelihood is computed using the `logLik` routine, while the penalty function is computed with `sic.penalty` (`penalty="sic"`), `aic.penalty` (`penalty="aic"`) or a user-defined penalty function (`penalty="user"`).

Value

`th`	Value of the threshold used (if `method="threshold"`) or selected on the solution path (if `method="ic"`).
`cpt`	Estimated locations of the change-points.
`ic`	Values of the information criterion minimised in order to find an optimal solution on the path (only if `method="ic"` was used).

References

R. Baranowski, Y. Chen, and P. Fryzlewicz (2019). Narrowest-Over-Threshold Change-Point Detection. (http://stats.lse.ac.uk/fryzlewicz/not/not.pdf)

Examples

```
# **** Piecewisce-constant mean with Gaussian noise.
x <- c(rep(0, 100), rep(1,100)) + rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMean")
# *** choose change-points using default settings
fo <- features(w)
# *** get the change-points
fo$cpt
# *** plot the SIC curve
plot(fo$ic)
```

`logLik.not` *Extract likelihood from a 'not' object*

Description

Calculates the Gaussian log-likelihood for the signal estimated using `predict.not` with the change-points at `cpt`. The type of the signal depends on on the value of `contrast` that has been passed to `not` (see `predict.not`).

Usage

```
## S3 method for class 'not'
logLik(object, cpt, ...)
```

Arguments

`object`	An object of class 'not', returned by `not`.
`cpt`	An integer vector with locations of the change-points. If missing, the `features` is called internally to extract the change-points from `object`.
`...`	Further parameters that can be passed to `predict.not` and `features`.

Examples

```
#' # **** Piecewisce-constant mean with Gaussian noise.
x <- c(rep(0, 100), rep(1,100)) + rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMean")
# *** log-likelihood for the model with the change-point estimated  via 'not'
logLik(w)
# *** log-likelihood for the model with the change-point at 100
logLik(w, cpt=100)
```

`not` *Narrowest-Over-Threshold Change-Point Detection*

Description

Identifies potential locations of the change-points in the data following 'deterministic signal + noise' model (see details below) in a number of different scenarios. The object returned by this routine can be further passed to the `features` function, which finds the final estimate of the change-points based on a chosen stopping criterion. It can be also passed to `plot`, `predict` and `residuals` routines.

Usage

```
not(x, ...)

## Default S3 method:
not(x, M = 10000, method = c("not", "max"),
  contrast = c("pcwsConstMean", "pcwsConstMeanHT", "pcwsLinContMean",
  "pcwsLinMean", "pcwsQuadMean", "pcwsConstMeanVar"),
  rand.intervals = TRUE, parallel = FALSE, augmented = FALSE,
  intervals, ...)
```

Arguments

`x`	A numeric vector with data points.
`...`	Not in use.
`M`	A number of intervals drawn in the procedure.
`method`	Choice of "not" (recommended) and "max". If `method="not"`, the Narrowest-Over-Threshold intervals are used in the algorithm. If `method="max"`, the intervals corresponding to the largest contrast function are used. For an explanation, see the references.

`contrast`	A type of the contrast function used in the NOT algorithm. Choice of `"pcwsConstMean"`, `"pcwsConstMeanHT"`, `"pcwsLinContMean"`, `"pcwsLinMean"`, `"pcwsQuadMean"`, `"pcwsConstMeanVar"`. For the explanation, see details below.
`rand.intervals`	A logical variable. If `rand.intervals=TRUE` intervals used in the procedure are drawn uniformly using the `random.intervals` routine. If `rand.intervals=FALSE`, the intervals need to be passed using the `intervals` argument.
`parallel`	A logical variable. If TRUE some of computations are run in parallel using OpenMP framework. Currently this option is not supported on Windows.
`augmented`	A logical variable. if TRUE, the entire data are considered when the NOT segmentation tree is constructed (see the solution path algorithm in the references).
`intervals`	A 2-column matrix with the intervals considered in the algorithm, with start- and end- points of the intervals in, respectively, the first and the second column. The intervals are used only if `rand.intervals=FALSE`.

Details

The data points provided in `x` are assumed to follow

$$Y_t = f_t + \sigma_t \varepsilon_t,$$

for $t = 1, \ldots, n$, where n is the number of observations in `x`, the signal f_t and the standard deviation σ_t are non-stochastic with structural breaks at unknown locations in time t. Currently, thefollowing scenarios for f_t and σ_t are implemented:

- Piecewise-constant signal with a Gaussian noise and constant standard deviation.
  Use `contrast="pcwsConstMean"` here.
- Piecewise-constant mean with a heavy-tailed noise and constant standard deviation.
  Use `contrast="pcwsConstMeanHT"` here.
- Piecewise-linear continuous signal with Gaussian noise and constant standard deviation.
  Use `contrast="pcwsLinContMean"` here.
- Piecewise-linear signal with Gaussian noise and constant standard deviation.
  Use `contrast="pcwsLinMean"` here.
- Piecewise-quadratic signal with Gaussian noise and constant standard deviation.
  Use `contrast="pcwsQuadMean"` here.
- Piecewise-constant signal and piecewise-constant standard deviation of the Gaussian noise.
  Use `contrast="pcwsConstMeanVar"` here.

Value

An object of class "not", which contains the following fields:

`x`	The input vector.
`n`	The length of `x`.
`contrast`	A scenario for the change-points.

`contrasts`	A 5-column matrix with the values of the contrast function, where 's' and 'e' denote start- end points of the intervals in which change-points candidates 'arg.max' have been found; 'length' shows the length of the intervals drawn, column 'max.contrast' contains corresponding value of the contrast statistic.
`solution.path`	A list with the solution path of the NOT algorithm (see the references) containing three fields of the same length: `cpt` - a list with consecutive solutions, i.e. s the sets of change-point candidates, `th` - a vector of thresholds corresponding to the solutions, `n.cpt` - a vector with the number of change-points for each solution.

References

R. Baranowski, Y. Chen, and P. Fryzlewicz (2019). Narrowest-Over-Threshold Change-Point Detection. (http://stats.lse.ac.uk/fryzlewicz/not/not.pdf)

Examples

```
# **** Piecewisce-constant mean with Gaussian noise.
# *** signal
pcws.const.sig <- c(rep(0, 100), rep(1,100))
# *** data vector
x <- pcws.const.sig + rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMean")
# *** some examples of how the w object can be used
plot(w)
plot(residuals(w))
plot(predict(w))
# *** this is how to extract the change-points
fo <- features(w)
fo$cpt

# **** Piecewisce-constant mean with a heavy-tailed noise.
# *** data vector, signal the same as in the previous example, but heavy tails
x <- pcws.const.sig + rt(100, 3)
# *** identify potential locations of the change-points,
# using a contrast taylored to heavy-tailed data
w <- not(x, contrast = "pcwsConstMeanHT")
plot(w)

# **** Piecewisce-constant mean and piecewise-constant variance
# *** signal's standard deviation
pcws.const.sd <- c(rep(2, 50), rep(1,150))
# *** data vector with pcws-const mean and variance
x <- pcws.const.sig + pcws.const.sd * rnorm(100)
# *** identify potential locations of the change-points in this model
w <- not(x, contrast = "pcwsConstMeanVar")
# *** extracting locations of the change-points
fo <- features(w)
fo$cpt

# **** Piecewisce-linear coninuous mean
# *** signal with a change in slope
pcws.lin.cont.sig <- cumsum(c(rep(-1/50, 100), rep(1/50,100)))
# *** data vector
x <- pcws.lin.cont.sig +  rnorm(100)
# *** identify potential locations of the change-points in the slope coefficient
w <- not(x, contrast = "pcwsLinContMean")
# *** ploting the results
plot(w)
# *** location(s) of the change-points
fo <- features(w)
fo$cpt

# **** Piecewisce-linear mean with jumps
# *** signal with a change in slope and jumpe
pcws.lin.sig <- pcws.lin.cont.sig + pcws.const.sig
# *** data vector
x <- pcws.lin.sig +  rnorm(100)
# *** identify potential locations of the change-points in the slope coefficient and the intercept
w <- not(x, contrast = "pcwsLinMean")
# *** ploting the results
plot(w)
# *** location(s) of the change-points
fo <- features(w)
fo$cpt

# **** Piecewisce-quadratic mean with jumps
# *** Piecewise-quadratic signal
pcws.quad.sig <- 2*c((1:50)^2 /1000, rep(2, 100), 1:50 / 50 )
# *** data vector
x <- pcws.quad.sig +  rnorm(100)
# *** identify potential locations of the change-points in the slope coefficient and the intercept
w <- not(x, contrast = "pcwsQuadMean")
# *** ploting the results
plot(w)
# *** location(s) of the change-points
fo <- features(w)
fo$cpt
```

plot.not *Plot a 'not' object*

Description

Plots the input vector used to generate 'not' object x with the signal fitted with predict.not.

Usage

```
## S3 method for class 'not'
plot(x, ...)
```

Arguments

`x`	An object of class 'not', returned by `not`.
`...`	Further parameters which may be passed to `predict.not` and `features`.

See Also

`predict.not` `not` `features`

Examples

```
# **** Piecewisce-constant mean with Gaussian noise.
x <- c(rep(0, 100), rep(1,100)) + rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMean")
# *** when 'cpt' is omitted, 'features' function is used internally
# to choose change-points locations
plot(w)
# *** estimate and plot the signal specifying the location of the change-point
plot(w, cpt=100)
```

`predict.not` *Estimate signal for a 'not' object.*

Description

Estimates signal in `object$x` with change-points at `cpt`. The type of the signal depends on on the value of `contrast` that has been passed to `not` (see details below).

Usage

```
## S3 method for class 'not'
predict(object, cpt, ...)
```

Arguments

`object`	An object of class 'not', returned by `not`.
`cpt`	An integer vector with locations of the change-points. If missing, the `features` is called internally to extract the change-points from `object`.
`...`	Further parameters that can be passed to `predict.not` and `features`.

Details

The data points provided in `object$x` are assumed to follow

$$Y_t = f_t + \sigma_t \varepsilon_t,$$

for $t = 1, \ldots, n$, where n is the number of observations in `object$x`, the signal f_t and the standard deviation σ_t are non-stochastic with change-points at locations given in `cpt` and ε_t is a white-noise.

Denote by $\tau_1, \dots, \tau_q$ the elements in `cpt` and set $\tau_0 = 0$ and $\tau_{q+1} = T$. Depending on the value of `contrast` that has been passed to `not` to construct `object`, the returned value is calculated as follows.

- For `contrast="pcwsConstantMean"` and `contrast="pcwsConstantMeanHT"`, in each segment $(\tau_j + 1, \tau_{j+1})$, f_t for $t \in (\tau_j + 1, \tau_{j+1})$ is approximated by the mean of Y_t calculated over $t \in (\tau_j + 1, \tau_{j+1})$.
- For `contrast="pcwsLinContMean"`, f_t is approximated by the linear spline fit with knots at $\tau_1, \dots, \tau_q$ minimising the l2 distance between the fit and the data.
- For `contrast="pcwsLinMean"` in each segment $(\tau_j + 1, \tau_{j+1})$, the signal f_t for $t \in (\tau_j + 1, \tau_{j+1})$ is approximated by the line $\alpha_j + \beta_j t$, where the regression coefficients are found using the least squares method.
- For `contrast="pcwsQuad"`, the signal f_t for $t \in (\tau_j + 1, \tau_{j+1})$ is approximated by the curve $\alpha_j + \beta_j t + \gamma_j t^2$, where the regression coefficients are found using the least squares method.
- For `contrast="pcwsConstMeanVar"`, in each segment $(\tau_j + 1, \tau_{j+1})$, f_t and σ_t for $t \in (\tau_j + 1, \tau_{j+1})$ are approximated by, respectively, the mean and the standard deviation of Y_t, both calculated over $t \in (\tau_j + 1, \tau_{j+1})$.

Value

A vector wit the estimated signal or a two-column matrix with the estimated estimated signal and standard deviation if `contrast="pcwsConstMeanVar"` was used to construct `object`.

See Also

`not`

Examples

```
# **** Piecewisce-constant mean with Gaussian noise.
x <- c(rep(0, 100), rep(1,100)) + rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMean")
# *** when 'cpt' is omitted, 'features' function is used internally
# to choose change-points locations
signal.est <- predict(w)
# *** estimate the signal specifying the location of the change-point
signal.est.known.cpt <- predict(w, cpt=100)
# *** pass arguments of the 'features' function through 'predict'.
signal.est.aic <- predict(w, penalty.type="aic")

# **** Piecewisce-constant mean and variance with Gaussian noise.
x <- c(rep(0, 100), rep(1,100)) + c(rep(2, 100), rep(1,100)) * rnorm(100)
# *** identify potential locations of the change-points
w <- not(x, contrast = "pcwsConstMeanVar")
# *** here signal is two-dimensional
signal.est <- predict(w)
```

`random.intervals` *Generate random intervals*

Description

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

Usage

```
random.intervals(n, M, min.length = 1, max.length = n, ...)
```

Arguments

`n`	a number of endpoints to choose from
`M`	a number of intervals to generate
`min.length`	an integer specifying minimum interval length
`max.length`	an integer specifying maximum interval length
`...`	not in use

Value

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

See Also

`not`

Examples

```
#*** draw 100 intervals with the endpoints in 1,...,100
intervals <- random.intervals(50, 100)
```

`residuals.not` *Extract residuals from a 'not' object*

Description

Returns a difference between `x` in `object` and the estimated signal with change-points at `cpt`. Type of the signal depends on the value of `contrast` that has been passed to `not` in order to construct `object` (see details of `predict.not`).

Usage

```
## S3 method for class 'not'
residuals(object, cpt, type = c("raw", "standardised"),
  ...)
```

Arguments

`object`	An object of class 'not', returned by `not`.
`cpt`	An integer vector with locations of the change-points. If missing, the `features` is called internally to extract the change-points from `object`.
`type`	Choice of "raw" and "standardised".
`...`	Further parameters that can be passed to `predict.not` and `features`.

Value

If `type="raw"`, the difference between the data and the estimated signal. If `type="standardised"`, the difference between the data and the estimated signal, divided by the estimated standard deviation.

Examples

```
pcws.const.sig <- c(rep(0, 100), rep(1,100))
x <- pcws.const.sig + rnorm(100)
w <- not(x, contrast = "pcwsConstMean")
# *** plot residuals obtained via fitting piecewise-constant function with estimated change-points
plot(residuals(w))
# *** plot residuals with obtained via fitting piecewise-constant function with true change-point
plot(residuals(w, cpt=100))
# *** plot standardised residuals
plot(residuals(w, type="standardised"))
```

`sic.penalty` *Schwarz Information Criterion penalty*

Description

The function evaluates the penalty term for Schwarz Information Criterion. If `alpha` is greater than 1, the strengthen SIC proposed proposed in Fryzlewicz (2014) is calculated. This routine is typically not called directly by the user; its name can be passed as an argument to `features`.

Usage

```
sic.penalty(n, n.param, alpha = 1, ...)
```

Arguments

`n`	The number of observations.
`n.param`	The number of parameters in the model for which the penalty is evaluated.
`alpha`	A scalar greater or equal than one.
`...`	Not in use.

Value

the penalty term `n.param` $\times (\log(n))^{\texttt{alpha}}$.

References

R. Baranowski, Y. Chen, and P. Fryzlewicz (2019). Narrowest-Over-Threshold Change-Point Detection. (http://stats.lse.ac.uk/fryzlewicz/not/not.pdf)

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

Examples

```
#*** a simple example how to use the AIC penalty
x <- rnorm(300) + c(rep(1,50),rep(0,250))
w <- not(x)
w.cpt <- features(w, penalty="sic")
w.cpt$cpt[[1]]
```

Index