Package ‘FDRSeg’

September 20, 2017

Type Package

Title FDR-Control in Multiscale Change-Point Segmentation

Version 1.0-3

Date 2017-09-20

Author Housen Li [aut],
Hannes Sieling [aut],
Timo Aspelmeier [cre]

Maintainer Timo Aspelmeier <timo.aspelmeier@mathematik.uni-goettingen.de>


Depends R (>= 2.15.3)

License GPL-3

Imports Rcpp (>= 0.11.5), stepR (>= 1.0-1), stats

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-09-20 08:34:11 UTC

R topics documented:

FDRSeg-package ................................................................. 2
computeFdp ................................................................. 4
dfdrseg ................................................................. 5
evalStepFun ............................................................ 6
fdrseg ................................................................. 7
simulQuantile ............................................................ 9
smuce ............................................................... 10
teethfun ............................................................. 12
v_measure ............................................................. 13

Index 14
Description


Details

Package: FDRSeg
Type: Package
Version: 1.0-3
Date: 2017-09-20
License: The GNU General Public License

Index:

- computeFdp: Compute false discovery proportion (FDP)
- dfdrseg: Piecewise constant regression with D-FDRSeg
- evalStepFun: Evaluate step function
- fdrSeg: Piecewise constant regression with FDRSeg
- simulQuantile: Quantile simulations
- smuce: Piecewise constant regression with SMUCE
- teethfun: Teeth function
- v_measure: Compute V-measure

Author(s)

Housen Li [aut], Hannes Sieling [aut], Timo Aspelmeier [cre]
Maintainer: Timo Aspelmeier <timo.aspelmeier@mathematik.uni-goettingen.de>

References


See Also

- smuceR
- jsmurf
library(stepR)

## (I) Independent Gaussian Data
# simulate data
n <- 300  # number of observations
K <- 20  # number of change-points
u0 <- teethfun(n, K)
set.seed(2)
Y <- rnorm(n, u0, 0.3)

# plot data
plot(Y, pch = 20, col = "grey", ylab = "")
lines(u0, type = "s")

# estimate standard deviation
sd <- sdrobvnorm(Y)

# simulate quantiles
alpha <- 0.1
qs <- simulQuantile(1 - alpha, n, type = "smuce")  # for SMUCE
qfs <- simulQuantile(1 - alpha, n, type = "fdrseg")  # for FDRSeg

# compute estimates
us <- smuce(Y, qs, sd = sd)  # SMUCE
ufs <- fdrseg(Y, qfs, sd = sd)  # FDRSeg

# plot results
lines(evalStepFun(us), type = "s", col = "blue")
lines(evalStepFun(ufs), type = "s", col = "red")
legend("topleft", c("Truth", "SMUCE", "FDRSeg"), lty = c(1, 1, 1), col = c("black", "blue", "red"))

## (II) Dependent Gaussian Data
# simulate data (a continuous time Markov chain)
ts <- 0.1  # sampling time
SNR <- 3  # signal-to-noise ratio
sampling <- 1e4  # sampling rate 10 kHz
over <- 10  # tenfold oversampling
cutoff <- 1e3  # 1 kHz 4-pole Bessel-filter, adjusted for oversampling
simdf <- dfilter("bessel", list(pole=4, cutoff=cutoff/sampling/over))
transRate <- 50
rates <- rbind(c(0, transRate), c(transRate, 0))
set.seed(123)
sim <- contMC(ts>sampling, c(0,SNR), rates, sampling = sampling, family = "gaussKern",
param = list(df=simdf, over=over, sd=1))
Y <- sim$data$y
x <- sim$data$x

# D-FDRseg
convKern <- dfilter("bessel", list(pole=4, cutoff=cutoff/sampling))$kern
alpha <- 0.1
r <- 10  # r could be much larger
Compute false discovery proportion (FDP)

Description

Compute false discovery proportion for estimated change-points, see (Li et al., 2015) for a detailed explanation.

Usage

```
computeFdp(u, eJ)
```

Arguments

- **u**: true signal; a numeric vector
- **eJ**: estimated change-points; a numeric vector

Value

A scalar takes value in [0, 1].

References


See Also

- `fdrseg`, `v_measure`

Examples

```
# simulate data
set.seed(2)
u0 <- c(rep(1, 50), rep(0, 50))
Y <- rnorm(100, u0)

# compute FDRSeg
uh <- fdrseg(Y)
```
dfdrseg

plot(Y, pch = 20, col = "grey", xlab = "", ylab = "")
lines(u0, type = "s", col = "blue")
lines(evalStepFun(uh), type = "s", col = "red")
legend("topleft", c("Truth", "FDRSeg"), lty = c(1, 1), col = c("blue", "red"))

# compute false discovery proportion
fdp <- computeFdp(u0, uh$left)
cat("False discovery proportion is ", fdp, "\n")

---

**dfdrseg**

*Piecewise constant regression with D-FDRSeg*

**Description**

Compute the D-FDRSeg estimator for one-dimensional data with dependent Gaussian noises, especially for ion channel recordings, see (Hotz et al., 2013; Li et al., 2015) for further details.

**Usage**

```r
dfdrseg(Y, q, alpha = 0.1, r = round(50/min(alpha, 1-alpha)), convKern, 
   sd = stepR::sdrobnorm(Y, lag=length(convKern)+1))
```

**Arguments**

- `Y` a numeric vector containing the noisy data
- `q` threshold value; a numeric vector of the same length as the data
- `alpha` significance level; if `q` is missing, `q` is chosen as the (1-alpha)-quantile of the null distribution of the multiscale statistic via Monte Carlo simulation, see (Li et al., 2015) for an explanation
- `r` number of Monte Carlo simulations
- `convKern` kernel of the low-pass filter, see (Li et al., 2015)
- `sd` standard deviation of noises

**Value**

A list with components

- `value` function values on each segment of the estimator
- `left` indices of leftmost points within each segment of the estimator
- `n` number of samples

**References**


evalStepFun

Evaluate step function

Description

Transform the return value by `smuce`, `fdrseg`, or `dfdrseg` into a numeric vector.

See Also

`smuce`, `dfdrseg`, `jsmurf`, `simulQuantile`, `sdrobnorm`, `contMC`, `dfilter`, `evalStepFun`

Examples

```r
library(stepR)

# simulate data (a continuous time Markov chain)
ts <- 0.1 # sampling time
SNR <- 3 # signal-to-noise ratio
sampling <- 1e4 # sampling rate 10 kHz
over <- 10 # tenfold oversampling
cutoff <- 1e3 # 1 kHz 4-pole Bessel-filter, adjusted for oversampling
simdf <- dfilter("bessel", list(pole=4, cutoff=cutoff/sampling/over))
transRate <- 50
rates <- rbind(c(0, transRate), c(transRate, 0))
set.seed(123)
sim <- contMC(ts*sampling, c(0,SNR), rates, sampling = sampling, family = "gaussKern",
             param = list(df=simdf, over=over, sd=1))
Y <- sim$data$y
x <- sim$data$x

# D-FDRSeg
library(stepR)
convKern <- dfilter("bessel", list(pole=4, cutoff=cutoff/sampling))$kern
uh <- dfdrseg(Y, convKern = convKern, r = 10) # r could be much larger

# plot results
plot(x, Y, pch = 20, col = "grey", xlab="", ylab="", main = "Simulate Ion Channel Data")
lines(sim$discr, col = "blue")
lines(x, evalStepFun(uh), col = "red")
legend("topleft", c("Truth", "D-FDRSeg"), lty = c(1, 1), col = c("blue", "red"))

# Not run:
# alternatively simulate quantiles first
alpha <- 0.1
q <- simulQuantile(1 - alpha, ts*sampling, type = "dfdrseg", convKern = convKern)

# then compute the estimate
uh <- dfdrseg(Y, q, convKern = convKern)
```

```
Usage

`evalStepFun(stepF)`

Arguments

- `stepF`: a list returned by `smuce`, `fdrseg`, or `dfdrseg` with components:
  - `value`: function values on each segment of the estimator.
  - `left`: indices of leftmost points within each segment of the estimator.
  - `n`: number of samples.

Value

A numeric vector gives function values of `stepF` at sampling locations.

See Also

`smuce`, `fdrseg`, `dfdrseg`

Examples

```r
# simulate data
set.seed(2)
u0 <- c(rep(1, 5), rep(5, 5))
Y <- rnorm(10, u0)

# compute the SMUCE estimate
uh <- smuce(Y)

# print results
# step function returned by smuce
print(uh)
# vector returned by evalStepFun
print(evalStepFun(uh))
```

---

**fdrseg**

Piecewise constant regression with FDRSeg

Description

Compute the FDRSeg estimator for one-dimensional data with i.i.d. Gaussian noises.

Usage

```r
fdrseg(Y, q, alpha = 0.1, r = round(50/min(alpha, 1-alpha)), sd = stepR::sdrobnorm(Y))
```
Arguments

Y  a numeric vector containing the noisy data
q  threshold value; a numeric vector of the same length as the data
alpha  significance level; if q is missing, q is chosen as the (1-alpha)-quantile of the null distribution of the multiscale statistic via Monte Carlo simulation, see (Li et al., 2015) for an explanation
r  number of Monte Carlo simulations
sd  standard deviation of noises

Value

A list with components

value  function values on each segment of the estimator
left  indices of leftmost points within each segment of the estimator
n  number of samples

References


See Also

smuce, dfdrseg, simulquantile, sdrobnorm, evalstepfun, computeFdp, v_measure

Examples

# simulate data
set.seed(123)
u0 <- c(rep(1, 50), rep(5, 50))
Y <- rnorm(100, u0)

# compute the estimate (q is automatically simulated)
# it might take a while due to simulating quantiles and will
# be faster for later calls on signals of the same length
uh <- fdrseg(Y)

# plot result
plot(Y, pch = 20, col = "grey", ylab = "", main = expression(alpha*" = 0.1"))
lines(u0, type = "s", col = "blue")
lines(evalStepFun(uh), type = "s", col = "red")
legend("topleft", c("Truth", "FDRSeg"), lty = c(1, 1), col = c("blue", "red"))

# other choice of alpha
uh <- fdrseg(Y, alpha = 0.05)
# plot result
plot(Y, pch = 20, col = "grey", ylab = "", main = expression(alpha*" = 0.05"))
simulQuantile

Quantile simulations

Description

Simulate the quantiles of multiscale statistics for SMUCE, FDRSeg, and D-FDRSeg under null hypothesis.

Usage

```r
simulQuantile(alpha, n, r = round(50/min(alpha, 1-alpha)),
    type = c("smuce","fdrseg","dfdrseg"), convKern, pos = .GlobalEnv)
```

Arguments

- **alpha**: a scalar with values in \([0, 1]\); the alpha-quantile of the null distribution of the multiscale statistic for SMUCE, FDRSeg, or D-FDRSeg via Monte Carlo simulation, see (Frick et al., 2014; Hotz et al., 2013; Li et al., 2015) for an explanation
- **n**: number of observations
- **r**: number of Monte Carlo simulations
- **type**: either "smuce" simulate quantile for SMUCE, "fdrseg" simulate quantiles for FDRSeg, or "dfdrseg" simulate quantiles for D-FDRSeg
- **convKern**: convolution kernel, only needed when type is "dfdrseg"
- **pos**: environment for saving the simulations for possible later usage

Value

A scalar value if type is chosen as "smuce"; a numeric vector of length n if type is chosen as "fdrseg" or "dfdrseg".
References


See Also

smuce, fdrseg, dfdrseg

Examples

```r
library(stepR)

# simulate quantiles for independent Gaussian noises
qs <- simuQuantile(0.9, 100, type = "smuce")
qfs <- simuQuantile(0.9, 100, type = "fdrseg")

# plot result
yrng <- range(qs, qfs)
plot(qfs, pch = 20, ylim = yrng, xlab = "n", ylab = "")
abline(h = qs)

# simulate quantiles for dependent Gaussian noises
convKern <- dfilter("bessel")$kern  # create digital filters
qdfs <- simuQuantile(0.9, 100, type = "dfdrseg", convKern = convKern)
plot(qdfs, pch = 20, xlab = "n", ylab = "")
```

---

smuce  
*Piecewise constant regression with SMUCE*

Description

Compute the SMUCE estimator for one-dimensional data with i.i.d. Gaussian noises.

Usage

```r
smuce(Y, q, alpha = 0.1, r = round(50/min(alpha, 1-alpha)), sd = stepR::sdrobnorm(Y))
```

Arguments

- **Y**  
a numeric vector containing the noisy data
- **q**  
threshold value; a scalar number
- **alpha**  
significance level; if q is missing, q is chosen as the (1-alpha)-quantile of the null distribution of the multiscale statistic via Monte Carlo simulation, see (Frick et al., 2014) for an explanation
$r$ numer of Monte Carlo simulations
$sd$ standard deviation of noises

**Value**

A list with components

- **value**: function values on each segment of the estimator
- **left**: indices of leftmost points within each segment of the estimator
- **n**: number of samples

**Note**

This is an efficient implementation of function `smuce` in R package stepR (CRAN) for data with i.i.d. Gaussian noises. The detailed algorithm is described in (Seiling, 2013).

**References**


**See Also**

`fdrseg`, `dfdrseg`, `simulQuantile`, `sdrobnorm`, `evalStepFun`, `computeFdp`, `v_measure`

**Examples**

```r
# simulate data
set.seed(2)
u0 <- c(rep(1, 50), rep(5, 50))
Y <- rnorm(100, u0)

# compute the estimate (q is automatically simulated)
uh <- smuce(Y)

# plot result
plot(Y, pch = 20, col = "grey", ylab = "", main = expression(alpha*" = 0.1"))
lines(u0, type = "s", col = "blue")
lines(evalStepFun(uh), type = "s", col = "red")
legend("topleft", c("Truth", "SMUCE"), lty = c(1, 1), col = c("blue", "red"))

# other choice of alpha
uh <- smuce(Y, alpha = 0.05)
# plot result
plot(Y, pch = 20, col = "grey", ylab = "", main = expression(alpha*" = 0.05"))
lines(u0, type = "s", col = "blue")
lines(evalStepFun(uh), type = "s", col = "red")
```

teethfun

Description

Create the teeth function with specified lengths and number of change-points.

Usage

teethfun(n, k, h = 3)

Arguments

  n  length of the vector (values of the teeth function)
  k  number of change-points
  h  height of the jump

Value

A numeric vector gives values of the teeth function.

References


Examples

  # create teeth function
  u <- teethfun(100, 6)

  # plot
  plot(u, type = "s")
Compute V-measure

Description

Compute V-measure, a segmentation evaluation measure, which is based upon two criteria for clustering usefulness, homogeneity and completeness.

Usage

\[ v\_\text{measure}(\text{sig}, \text{est}, \beta = 1) \]

Arguments

- \text{sig} true signal; a numeric vector
- \text{est} estimator; a numeric vector
- \beta parameter in definition of V-measure, see (Rosenberg and Hirschberg, 2007) for details

Value

A scalar takes value in \([0, 1]\), with a larger value indicating higher accuracy.

References


See Also

\text{computeFdp}, \text{smuce}, \text{fdrseg}, \text{evalStepFun}

Examples

```r
# simulate data
u0 <- c(rep(1, 50), rep(5, 50))
Y <- rnorm(100, u0)

# compute FDRSeg
uh <- fdrseg(Y)

plot(Y, pch = 20, col = "grey", xlab = "", ylab = ""
lines(u0, type = "s", col = "blue")
lines(evalStepFun(uh), type = "s", col = "red")
legend("topleft", c("Truth", "FDRSeg"), lty = c(1, 1), col = c("blue", "red"))

# compute V-measure
vm <- v\_measure(u0, evalStepFun(uh))
print(vm)
```
Index

*Topic **nonparametric**
  - computeFdp, 4
  - dfdrseg, 5
  - evalStepFun, 6
  - fdrseg, 7
  - FDRSeg-package, 2
  - simulQuantile, 9
  - smuce, 10
  - teethfun, 12
  - v_measure, 13
*Topic **package**
  - FDRSeg-package, 2

  - computeFdp, 4, 8, 11, 13
  - contMC, 6

  - dfdrseg, 5, 6–8, 10, 11
  - dfilter, 6

  - evalStepFun, 6, 6, 8, 11, 13

  - FDRSeg (FDRSeg-package), 2
  - fdrseg, 4, 6, 7, 7, 10, 11, 13
  - FDRSeg-package, 2

  - jsmurf, 2, 6

  - sdrobnorm, 6, 8, 11
  - simulQuantile, 6, 8, 9, 11
  - smuce, 6–8, 10, 10, 13
  - smuceR, 2

  - teethfun, 12

  - v_measure, 4, 8, 11, 13