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>

Description Estimate step functions via multiscale inference with controlled false discovery rate (FDR). For details see H. Li, A. Munk and H. Sieling (2016) <doi:10.1214/16-EJS1131>.

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 *FDR-Control in Multiscale Change-Point Segmentation*

Description

Estimate step functions via multiscale inference with controlled false discovery rate (FDR). For details see H. Li, A. Munk and H. Sieling (2016) <doi:10.1214/16-EJS1131>.

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

Frick, K., Munk, A., and Sieling, H. (2014). Multiscale Change-Point Inference. *J. R. Statist. Soc. B, with discussion and rejoinder by the authors*, 76:495–580.

Hotz, T., Schuette, O. M., Sieling, H., Polupanow, T., Diederichsen, U., Steinem, C., and Munk, A. (2013). Idealizing ion channel recordings by a jump segmentation multiresolution filter. *IEEE Transactions on Nanobioscience*, 12(4), 376–86.

Li, H., Munk, A., and Sieling, H. (2015). FDR-control in multiscale change-point segmentation. arXiv:1412.5844.

See Also

smuceR, jsmurf

Examples

```
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 <- sdrobnorm(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
```

```
qdfs     <- simulQuantile(1 - alpha, ts*sampling, r, "dfdrseg", convKern)
udfs     <- dfdrseg(Y, qdfs, convKern = convKern)

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

---

computeFdp *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

Li, H., Munk, A., and Sieling, H. (2015). FDR-control in multiscale change-point segmentation. arXiv:1412.5844.

See Also

fdrseg, v_measure

Examples

```
# simulate data
set.seed(2)
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 false discovery proportion
fdp <- computeFdp(u0, uh$left)
cat("False discovery propostion 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

```
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`	numer 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

Hotz, T., Schuette, O. M., Sieling, H., Polupanow, T., Diederichsen, U., Steinem, C., and Munk, A. (2013). Idealizing ion channel recordings by a jump segmentation multiresolution filter. *IEEE Transactions on Nanobioscience*, 12(4), 376-86.

Li, H., Munk, A., and Sieling, H. (2015). FDR-control in multiscale change-point segmentation. arXiv:1412.5844.

See Also

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

Examples

```
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)
## End(Not run)
```

---

evalStepFun *Evaluate step function*

Description

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

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

```
# 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

```
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	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

References

Li, H., Munk, A., and Sieling, H. (2015). FDR-control in multiscale change-point segmentation. arXiv:1412.5844.

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"))
```

```
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"))

## Not run:
# alternatively simulate quantiles first
alpha <- 0.1
q     <- simulQuantile(1 - alpha, 100, type = "fdrseg")

# then compute the estimate
uh <- fdrseg(Y, q)
## End(Not run)
```

---

simulQuantile *Quantile simulations*

Description

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

Usage

```
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	numer of Monte Carlo simulations
type	"smuce" simulate quantile for SMUCE "fdrseg" simulate quantiles for FDRSeg "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

Frick, K., Munk, A., and Sieling, H. (2014). Multiscale Change-Point Inference. *J. R. Statist. Soc. B, with discussion and rejoinder by the authors*, 76:495–580.

Hotz, T., Schuette, O. M., Sieling, H., Polupanow, T., Diederichsen, U., Steinem, C., and Munk, A. (2013). Idealizing ion channel recordings by a jump segmentation multiresolution filter. *IEEE Transactions on Nanobioscience*, 12(4):376–86.

Li, H., Munk, A., and Sieling, H. (2015). FDR-control in multiscale change-point segmentation. arXiv:1412.5844.

See Also

smuce, fdrseg, dfdrseg

Examples

```
library(stepR)

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

```
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 smuceR in R package stepR (CRAN) for data with i.i.d. Gaussian noises. The detailed algorithm is described in (Seiling, 2013).

References

Frick, K., Munk, A., and Sieling, H. (2014). Multiscale Change-Point Inference. *J. R. Statist. Soc. B, with discussion and rejoinder by the authors*, 76:495–580.

Seiling, H. (2013). Statistical Multiscale Segmentation: Inference, Algorithms and Applications. PhD thesis, University of Goettingen, Germany.

See Also

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

Examples

```
# 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")
```

```
legend("topleft", c("Truth", "SMUCE"), lty = c(1, 1), col = c("blue", "red"))

## Not run:
# alternatively simulate quantiles first
alpha <- 0.1
q     <- simulQuantile(1 - alpha, 100, type = "smuce")

# then compute the estimate
uh <- smuce(Y, q)
## End(Not run)
```

---

teethfun *Teeth function*

Description

Creat 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

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *Ann. Statist.*, 42(6): 2243–1572.

Examples

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

# plot
plot(u, type = "s")
```

v_measure *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_measure(sig, est, beta = 1)
```

Arguments

`sig`	true signal; a numeric vector
`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

Rosenberg, A., and Hirschberg, J. (2007). V-measure: a conditional entropy-based external cluster evaluation measures. *Proc. Conf. Empirical Methods Natural Lang. Process.*, (June):410–420.

See Also

`computeFdp`, `smuce`, `fdrseg`, `evalStepFun`

Examples

```
# 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