Package: fpopw (via r-universe)

August 21, 2024

Version 1.1

Date 2022-05-30

Title Weighted Segmentation using Functional Pruning and Optimal Partioning

Depends R (>= 3.1.0)

Description Weighted-L2 FPOP Maidstone et al. (2017) <doi:10.1007/s11222-016-9636-3> and pDPA/FPSN Rigaill (2010) <arXiv:1004.0887> algorithm for detecting multiple changepoints in the mean of a vector. Also includes a few model selection functions using Lebarbier (2005) <doi:10.1016/j.sigpro.2004.11.012> and the 'capsushe' package.

License GPL (>= 3)

RoxygenNote 7.2.0

NeedsCompilation yes

Author Guillem Rigaill [aut, cre]

Maintainer Guillem Rigaill <guillem.rigaill@inrae.fr>

Date/Publication 2022-06-05 14:10:03 UTC

Repository https://guillemr.r-universe.dev

RemoteUrl https://github.com/cran/fpopw

RemoteRef HEAD

RemoteSha 32d5b73a3678c5f99562bf9440238ecfafdbb11f

Contents

`compress.data` *compress.data*

Description

compress data and return a weighted profile

Usage

```
compress.data(x)
```

Arguments

x	a numerical vector

Value

a list with the compressed profile x and associated repeat vector vrep

---

`Fpop` *Fpop*

Description

Function to run the Fpop algorithm (Maidstone et al. 2016). It uses functional pruning and optimal partionning. It optimizes the L2-loss for a penalty lambda per change.

Usage

```
Fpop(x, lambda, mini = min(x), maxi = max(x))
```

Arguments

x	a numerical vector to segment
lambda	the penalty per changepoint (see Maidstone et al. 2016)
mini	minimum mean segment value to consider in the optimisation.
maxi	maximum mean segment value to consider in the optimisation.

Value

return a list with a vector t.est containing the position of the change-points, the number of changes K and, the cost J.est.

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
est.sd <- sdDiff(x) ## rough estimate of std-deviation
res <- Fpop(x=x,lambda=2*est.sd^2*log(length(x)))
smt <- getSMT(res)
```

---

fpopw *fpopw: A package to solve the optimal partionning and segment neighborhood problems using a weighted L2-loss.*

Description

The fpopw package provides wrapper to four functionnal pruning functions to to solve the optimal partionning and segment neighborhood problems with the L2-loss: Fpop, Fpop_w, Fpsn, Fpsn_w

fpopw functions

fpopw functions are Fpop, Fpop_w, Fpsn, Fpsn_w, Fpsn_w_nomemory

---

Fpop_w *Fpop_w*

Description

Function to run the Fpop algorithm (Maidstone et al. 2016) with weights. It uses functional pruning and optimal partionning. It optimizes the weighted L2-loss ($w_i(x_i - \mu)2$) for a penalty lambda per change.

Usage

```
Fpop_w(x, w, lambda, mini = min(x), maxi = max(x))
```

Arguments

x	a numerical vector to segment.
w	a numerical vector of weights (values should be larger than 0).
lambda	the penalty per changepoint (see Maidstone et al. 2016).
mini	minimum mean segment value to consider in the optimisation.
maxi	maximum mean segment value to consider in the optimisation.

Value

return a list with a vector t.est containing the position of the change-points, the number of changes K and, the cost J.est.

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
est.sd <- sdDiff(x) ## rough estimate of std-deviation
res <- Fpop_w(x=x, w=rep(1, length(x)), lambda=2*est.sd^2*log(length(x)))
smt <- getSMT(res)
```

Fpsn *Fpsn*

Description

Function to run the pDPA algorithm (Rigaill 2010 and 2015). It uses functional pruning and segment neighborhood. It optimizes the L2-loss for 1 to Kmax changes.

Usage

```
Fpsn(x, Kmax, mini = min(x), maxi = max(x))
```

Arguments

x	a numerical vector to segment
Kmax	max number of segments (segmentations in 1 to Kmax segments are recovered).
mini	minimum mean segment value to consider in the optimisation
maxi	maximum mean segment value to consider in the optimisation

Value

return a list with a matrix t.est containing the change-points of the segmentations in 1 to Kmax changes and, the cost J.est in 1 to Kmax changes.

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
res <- Fpsn(x=x, K=100)
select.res <- select_Fpsn(res, method="givenVariance")
smt <- getSMT(res, select.res)
```

`Fpsn_w` *Fpsn_w*

Description

Function to run the weighted pDPA algorithm (Rigaill 2010 and 2015). It uses functional pruning and segment neighborhood. It optimizes the weighted L2-loss $(w_i(x_i - \mu)2)$ for 1 to Kmax changes.

Usage

```
Fpsn_w(x, w, Kmax, mini = min(x), maxi = max(x))
```

Arguments

`x`	a numerical vector to segment
`w`	a numerical vector of weights (values should be larger than 0).
`Kmax`	max number of segments (segmentations in 1 to Kmax segments are recovered).
`mini`	minimum mean segment value to consider in the optimisation
`maxi`	maximum mean segment value to consider in the optimisation

Value

return a list with a matrix t.est containing the change-points of the segmentations in 1 to Kmax changes and, the costs J.est in 1 to Kmax changes.

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
res <- Fpsn_w(x=x, w=rep(1, length(x)), K=100)
select.res <- select_Fpsn(res, method="givenVariance")
smt <- getSMT(res, select.res)
```

`Fpsn_w_nomemory` *Fpsn_w_nomemory*

Description

Function to run the weighted pDPA algorithm (Rigaill 2010 and 2015) without storing the set of last changes. It only return the cost in 1 to Kmax changes. It uses functional pruning and segment neighborhood. It optimizes the weighted L2-loss ($w_i(x_i - \mu)2$) for 1 to Kmax changes.

Usage

```
Fpsn_w_nomemory(x, w, Kmax, mini = min(x), maxi = max(x))
```

Arguments

`x`	a numerical vector to segment
`w`	a numerical vector of weights (values should be larger than 0).
`Kmax`	max number of segments (segmentations in 1 to Kmax segments are recovered).
`mini`	minimum mean segment value to consider in the optimisation
`maxi`	maximum mean segment value to consider in the optimisation

Value

return a list with the costs J.est in 1 to Kmax changes.

Examples

```
res <- Fpsn_w_nomemory(x=rnorm(10^4), w=rep(1, 10^4), K=100)
```

`get.change` *get.change*

Description

Function returning changes in a smoothed profile

Usage

```
get.change(smt)
```

Arguments

`smt`	smoothed profile

Value

a vector of changes including n

`getSegSums_` *getSegSums_*

Description

A function to get the segment sums of a vector given some changes including n

Usage

```
getSegSums_(x, tau)
```

Arguments

`x`	data
`tau`	changes (including n)

Value

a vector of the sums

`getSMT` *getSMT*

Description

A function to get the smoothed profile from the output of Fpop, Fpop_w, Fpsn and Fpsn_w

Usage

```
getSMT(res, K = NULL)
```

Arguments

`res`	output of Fpop, Fpop_w, Fpsn or Fpsn_w
`K`	the number of changes (only if Fpsn or Fpsn_w)

Value

a vector of the smoothed profile

---

getSMT_ *getSMT_*

Description

A function to get the smoothed profile from the data, weights and changepoints

Usage

```
getSMT_(x, weights = NULL, tauHat)
```

Arguments

x	data
weights	weights
tauHat	changes (including n)

Value

a vector of the smoothed profile

---

getTau_nomemory *getTau_nomemory*

Description

function to recover changes for a given selected K after fpsn_nomemory

Usage

```
getTau_nomemory(res_fpsn, K_selected)
```

Arguments

res_fpsn	output of the function res_fpsn_nomemory
K_selected	K obtained using select_Fpsn

Value

return a set of changes

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
res <- Fpsn_w_nomemory(x=x, w=rep(1, length(x)), K=100)
select.res <- select_Fpsn(res, method="givenVariance")
tau <- getTau_nomemory(res, select.res)
smt <- getSMT_(res$signal, res$weights, tau)
```

retour_op *retour_op*

Description

Function used internally by Fpop and Fpop_w to do the backtracking and recover the best set of changes from 1 to i

Usage

```
retour_op(path, i)
```

Arguments

path	vector of length n containing the best last changes for any j in $[1, n]$. This vector is computed in the Fpop and Fpop_w using the colibri_op_c or colibri_op_weight_c function.
i	the last position to consider to start the backtracking.

Value

set of optimal changes up to i.

retour_sn *retour_sn*

Description

Function used internally by Fpsn and Fpsn_w to do the backtracking and recover the best set of segmentations in 1 to K changes from 1 to n.

Usage

```
retour_sn(path)
```

Arguments

path	matrix of size (K x n) containing the last optimal changes up to j in k segments with i in $[1, n]$ and k in $[1, K]$. This matrix is computed in the Fpsn or Fpsn_w function using the colibri_sn_c or colibri_sn_weight_c functions.

Value

a matrix of size (K x K) containing the best segmentations in 1 to K segments.

saut *saut*

Description

model selection function taken from S3IB,

Usage

```
saut(Lv, pen, Kseq, n, seuil = sqrt(n)/log(n), biggest = TRUE)
```

Arguments

Lv	likelihood
pen	penalty
Kseq	number of changes
n	number of datapoints
seuil	threshold
biggest	heuristic (biggest jump or slope)

Value

a selected number of chagnes

sdDiff *sdDiff*

Description

Function to estimate the standard deviation

Usage

```
sdDiff(x, method = "MAD")
```

Arguments

x	signal
method	used to estimate the variance : MAD or HALL

Value

return a numeric value

`select_Fpsn` *select_Fpsn*

Description

function to select the number of changepoints after Fpsn or Fpsn_w using the penalty of Lebarbier 2005 given a estimator of the variance

Usage

```
select_Fpsn(
  res_fpsn,
  method = "givenVariance",
  sigma = sdDiff(res_fpsn$signal)
)
```

Arguments

`res_fpsn`	output of Fpsn or Fpsn_w containg the costs in J.est and the segmented signal
`method`	one of (1) "givenVariance" = using the penalty of Lebarbier 2005 given a estimator of the variance, (2) "biggest.S3IB" = biggest=TRUE in saut taken from S3IB, (3) "notbiggest.S3IB" biggest=FALSE in saut taken from S3IB.
`sigma`	variance used of the selection. If NULL use MAD on unweighted data.

Value

return an integer: selected number of changes

Examples

```
x <- c(rnorm(100), rnorm(10^3)+2, rnorm(1000)+1)
res <- Fpsn_w(x=x, w=rep(1, length(x)), K=100)
select.res <- select_Fpsn(res, method="givenVariance")
smt <- getSMT(res, select.res)
```

`uncompress.smt` *decompress.smt*

Description

vector to decompress a compressed smoothed profile (a call to rep)

Usage

```
uncompress.smt(smt.CP, vec.rep)
```

Arguments

`smt.CP`	smoothed and compressed profile
`vec.rep`	weights to use for decompression

Value

a vector to replicate duplicated datapoints

`uncompress.vec` *uncompress.vec*

Description

return a vector to uncompress a profile, segmentation or smt

Usage

```
uncompress.vec(vec.rep)
```

Arguments

`vec.rep`	integer vector with the number of time each point should be repeated

Value

return a vector to uncompress a profile, segmentation or smt

Index