Package: jointseg (via r-universe)

September 8, 2024

Type Package

Title Joint Segmentation of Multivariate (Copy Number) Signals

Version 1.0.2

Description Methods for fast segmentation of multivariate signals into piecewise constant profiles and for generating realistic copy-number profiles. A typical application is the joint segmentation of total DNA copy numbers and allelic ratios obtained from Single Nucleotide Polymorphism (SNP) microarrays in cancer studies. The methods are described in Pierre-Jean, Rigaill and Neuvial (2015) <doi:10.1093/bib/bbu026>.

License LGPL (>= 2.1)

Depends R (>= 3.1.0)

Imports acnr (>= 0.3.1), matrixStats (>= 0.6.0), DNAcopy

Suggests PSCBS, R.cache, digest, changepoint (>= 1.0.2), knitr, rmarkdown, testthat

VignetteBuilder knitr

URL https://github.com/mpierrejean/jointseg

RoxygenNote 6.1.1

BugReports https://github.com/mpierrejean/jointseg/issues

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

RemoteUrl https://github.com/mpierrejean/jointseg

RemoteRef HEAD

RemoteSha 5b7ab68d331dc0311264b28ceb264e35d96fb0a2

Contents

`binMissingValues` *binMissingValues*

Description

Perform binning in order to remove missing values

Usage

```
binMissingValues(Y, verbose = FALSE)
```

Arguments

`Y`	A numeric matrix
`verbose`	A `logical` value: should extra information be output ? Defaults to `FALSE`.

Details

Some segmentation methods (in particular, GFLars) do not natively handle the situation when some observations have missing values in one or more dimensions. In order to avoid dropping the corresponding observations entirely, `binMissingValues` bins the signal values of the last complete observation before a (range of) observations with missing entries using the `binMeans` function.

In the specific case when the first row has NA values, the first non-missing entry is replicated in order to make smoothing possible. This choice is arbitrary but some arbitrary choice is needed in that case.

Note

Currently this function is only used by doGFLars in order to make it possible to run GFLars segmentation on SNP array data where most markers (on the order of 2/3 to 5/6) have missing values, because of uninformative or missing allelic ratio signals.

The binMissingValues function may be used for other segmentation methods suffering from the same limitation. However, we emphasize that handling missing values natively in the segmentation method would be a better solution.

Currently this function is only used by doGFLars in order to make it possible to run GFLars segmentation on SNP array data where most markers (on the order of 2/3 to 5/6) have missing values, because of uninformative or missing allelic ratio signals. The binMissingValues function may be used for other segmentation methods suffering from the same limitation. However, we emphasize that handling missing values natively in the segmentation method would be a better solution.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Bleakley, K., & Vert, J. P. (2011). The group fused lasso for multiple change-point detection. arXiv preprint arXiv:1106.4199.

Vert, J. P., & Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group LARS. Advances in Neural Information Processing Systems, 23, 2343-2351.

Examples

```
sim <- randomProfile(10, 1, 0.1, 3)
Y <- sim$profile
Y[c(4, 8), 2] <- NA
Y[c(7, 8), 3] <- NA

res <- binMissingValues(Y)

Y <- sim$profile
Y[1:5, 2] <- NA
Yb <- binMissingValues(Y)

Y <- sim$profile
Y[3:5, 2] <- NA
Yb <- binMissingValues(Y)
```

---

doCBS *Run CBS segmentation*

Description

This function is a wrapper for convenient use of the CBS segmentation method by PSSeg. It applies the segment function and reshapes the results

Usage

```
doCBS(y, ..., verbose = FALSE)
```

Arguments

y	A numeric vector, the signal to be segmented
...	Arguments to be passed to segment
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Value

A list with a single element:

bkp breakpoint positions

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

See Also

segment

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)

## generate a synthetic CN profile
K <- 10
len <- 1e4
sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
datS <- sim$profile

## run CBS segmentation
res <- doCBS(datS[["c"]])
getTpFp(res$bkp, sim$bkp, tol=5, relax = -1)   ## true and false positives
plotSeg(datS, breakpoints=list(sim$bkp, res$bkp))
```

`doDynamicProgramming` *Run segmentation by dynamic programming*

Description

High-level function for univariate or multivariate segmentation by dynamic programming

Usage

```
doDynamicProgramming(Y, K, stat = NULL, verbose = FALSE)
```

Arguments

`Y`	A numeric vector or a matrix, the signal to be segmented
`K`	The number of change points to find
`stat`	A vector containing the names or indices of the columns of `Y` to be segmented
`verbose`	A `logical` value: should extra information be output ? Defaults to `FALSE`.

Details

If the signal is uni-dimensional, this function simply uses the segmentation method provided in the `cghseg` package reshapes the results.

If the signal is multi-dimensional, this function applies the `pruneByDP` function and reshapes the results.

Value

`bkp`	A vector of `K` indices for candidate change points
`dpseg`	A list of two elements
	bkp A list of vectors of change point positions for the best model with k change points, for k=1, 2, ... K
	rse A vector of K+1 residual squared errors

Note

This is essentially a wrapper for convenient segmentation by dynamic programming using the `PSSeg` function.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Rigaill, G. (2015). A pruned dynamic programming algorithm to recover the best segmentations with 1 to K_max change-points. Journal de la Societe Francaise de Statistique, 156(4), 180-205.

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)

## generate a synthetic CN profile
K <- 10
len <- 1e4
sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
datS <- sim$profile

## run pruned DPA segmentation
resDP <- doDynamicProgramming(datS[["c"]], K=K)
getTpFp(resDP$bkp, sim$bkp, tol=5, relax = -1)   ## true and false positives
plotSeg(datS, breakpoints=list(sim$bkp, resDP$bkp))

## run 2d dynamic programming segmentation
K <- 2
len <- 1e3
sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
datS <- sim$profile
datS$d <- 2*abs(datS$b-1/2)
datS[which(datS$genotype!=0.5),"d"] <- NA
Y = cbind(datS$c,datS$d)
resDP2d <- doDynamicProgramming(Y, K = K)
```

`doGFLars` *Group fused Lars segmentation*

Description

High-level function for multivariate fused Lars (GFLars) segmentation

Usage

```
doGFLars(Y, K, stat = NULL, ..., verbose = FALSE)
```

Arguments

`Y`	A `n*p` signal to be segmented
`K`	The number of change points to find
`stat`	A vector containing the names or indices of the columns of `Y` to be segmented
`...`	Further arguments to be passed to 'segmentByGFLars'
`verbose`	A `logical` value: should extra information be output ? Defaults to `FALSE`.

Details

This function is a wrapper around the lower-level segmentation function segmentByGFLars. It can be run on p-dimensional, piecewise-constant data in order to defined a set of candidate change points. It is recommended to prune this list of candidates using dynamic programming (pruneByDP), combined with a selection of the best number of change points. The jointSeg function provides a convenient wrapper for performing segmentation, pruning and model selection.

For the specific case of DNA copy number data segmentation, see the dedicated wrapper PSSeg.

The default weights $\sqrt{n/(i*(n-i))}$ are calibrated as suggested by Bleakley and Vert (2011). Using this calibration, the first breakpoint maximizes the likelihood ratio test (LRT) statistic.

Value

An object of the same structure as the output of segmentByGFLars

Note

This implementation is derived from the MATLAB code by Vert and Bleakley: http://cbio.ensmp.fr/GFLseg.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Bleakley, K., & Vert, J. P. (2011). The group fused lasso for multiple change-point detection. arXiv preprint arXiv:1106.4199.

Vert, J. P., & Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group LARS. Advances in Neural Information Processing Systems, 23, 2343-2351.

Examples

```
p <- 2
trueK <- 10
sim <- randomProfile(1e4, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- doGFLars(Y, K)
print(res$bkp)
print(sim$bkp)
plotSeg(Y, res$bkp)

## a toy example with missing values
sim <- randomProfile(1e2, 1, 0.1, 2)
Y <- sim$profile
Y[3:50, 2] <- NA

res <- doGFLars(Y, 10, 2, verbose=TRUE)
print(res$bkp)
print(sim$bkp)
```

```
plotSeg(Y, res$bkp)
```

---

doPSCBS *Run Paired PSCBS segmentation*

Description

This function is a wrapper for convenient use of the PSCBS segmentation method by PSSeg. It applies the segmentByPairedPSCBS function and reshapes the results

Usage

```
doPSCBS(Y, ..., verbose = FALSE)
```

Arguments

Y	A matrix of signals to be segmented, containing the following columns **c** total copy numbers **b** allele B fractions (a.k.a. BAF) **genotype** germline genotypes
...	Arguments to be passed to segmentByPairedPSCBS
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Value

A list with a single element:

bkp breakpoint positions

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

See Also

segmentByPairedPSCBS

Examples

```
## Not run:
    ## load known real copy number regions
    affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)

    ## generate a synthetic CN profile
    K <- 10
    len <- 1e4
    sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
```

```
datS <- sim$profile

## run PSCBS segmentation
Y <- as.matrix(subset(datS, select=c(c, b, genotype)))
res <- doPSCBS(Y)
getTpFp(res$bkp, sim$bkp, tol=5, relax = -1)   ## true and false positives
plotSeg(datS, breakpoints=list(sim$bkp, res$bkp))

## End(Not run)
```

---

doPSCN *Run PSCN segmentation (defunct)*

Description

The 'PSCN' package is not maintained anymore and it is not available for R >= 3.0.0. The original 'doPSCN' function has been moved to the directory 'zzz.defunct'. The skeleton of that function is kept for backward compatibility.

Usage

```
doPSCN(Y, alpha = 0.01, platform = c("Illumina", "Affymetrix"),
  verbose = FALSE, ...)
```

Arguments

Y	The signal to be segmented, a matrix containing the following columns: **c** Total copy number (log scale) **b** Allele B fraction (a.k.a. BAF)
alpha	sensitivity level in [0,1] to be passed to PSCN::segmentation.
platform	Specifies form which array platform 'Y' was generated: Illumina or Affymetrix
verbose	A logical value: should extra information be output ? Defaults to FALSE.
...	Further arguments to be passed to PSCN::smoothing

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Chen, H., Xing, H., & Zhang, N. R. (2011). Estimation of parent specific DNA copy number in tumors using high-density genotyping arrays. PLoS computational biology, 7(1), e1001060.

See Also

PSSeg

Examples

```
print("The 'PSCN' package is not available for R >= 3.0.0.")
print("See http://cran.r-project.org/web/packages/PSCN/index.html")
```

---

doRBS *Run RBS segmentation*

Description

High-level function for multivariate recursive binary (RBS) segmentation

Usage

```
doRBS(Y, K, stat = NULL, ..., verbose = FALSE)
```

Arguments

Y	A n*p signal to be segmented
K	The number of change points to find
stat	A vector containing the names or indices of the columns of Y to be segmented
...	Further arguments to be passed to 'segmentByRBS'
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Details

This function is a wrapper around the lower-level segmentation function segmentByRBS. It can be run on p-dimensional, piecewise-constant data in order to defined a set of candidate change points. It is recommended to prune this list of candidates using dynamic programming (pruneByDP), combined with a selection of the best number of change points. The jointSeg function provides a convenient wrapper for performing segmentation, pruning and model selection.

Value

An object of the same structure as the output of segmentByRBS

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Gey, S., & Lebarbier, E. (2008). Using CART to Detect Multiple Change Points in the Mean for Large Sample. http://hal.archives-ouvertes.fr/hal-00327146/

See Also

PSSeg, pruneByDP

Examples

```
p <- 2
trueK <- 10
len <- 5e4
sim <- randomProfile(len, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- doRBS(Y, K)
getTpFp(res$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives

cols <- rep(2, K)
cols[1:trueK] <- 3
par(mfrow=c(p,1))
for (ii in 1:p) {
    plot(Y[, ii], pch=19, cex=0.2)
    abline(v=res$bkp[1:trueK], col= cols)
    abline(v=sim$bkp, col=8, lty=2)
}

## NA:s in one dimension at a true breakpoint
jj <- sim$bkp[1]
Y[jj-seq(-10, 10), p] <- NA
res2 <- doRBS(Y, K)
getTpFp(res2$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives

## NA:s in both dimensions at a true breakpoint
Y[jj-seq(-10, 10), ] <- NA
res3 <- doRBS(Y, K)
getTpFp(res3$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives
```

---

estimateSd *Robust standard deviation estimator*

Description

Estimate standard deviation of an unimodal signal with possible changes in mean

Usage

```
estimateSd(y, method = c("Hall", "von Neumann"))
```

Arguments

y	A numeric vector
method	Method used to estimate standard deviation

Details

von Neumann's estimator is proportional to the mean absolute deviation (mad) of the first-order differences of the original signals: mad(diff(y)). By construction this estimator is robust to 1) changes in the mean of the signal (through the use of differences) and 2) outliers (through the use of mad instead of mean).

The proportionality constant $1/\sqrt{2} \times 1/\Phi^{-1}(3/4)$ ensures that the resulting estimator is consistent for the estimation of the standard deviation in the case of Gaussian signals.

Hall's estimator is a weigthed sum of squared elements of y. Let m=3. $sigma^2 = (\sum_{k=1}^{n-m} \sum_{j=1}^{m+1} (\texttt{wei[i]y}[i+k])^2)/(n-m)$

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Von Neumann, J., Kent, R. H., Bellinson, H. R., & Hart, B. T. (1941). The mean square successive difference. The Annals of Mathematical Statistics, 153-162.

Peter Hall, J. W. Kay and D. M. Titterington (1990). Asymptotically Optimal Difference-Based Estimation of Variance in Nonparametric Regression Biometrika,521-528

Examples

```
n <- 1e4
y <- rnorm(n)  ## a signal with no change in mean
estimateSd(y)
estimateSd(y, method="von Neumann")
sd(y)
mad(y)

z <- y + rep(c(0,2), each=n/2)  ## a signal with *a single* change in mean
estimateSd(z)
estimateSd(z, method="von Neumann")
sd(z)
mad(z)

z <- y + rep(c(0,2), each=100)  ## a signal with many changes in mean
estimateSd(z)
estimateSd(z, method="von Neumann")
sd(z)
mad(z)
```

Fpsn *Pruned dynamic programming algorithm*

Description

Low-level API for the pruned dynamic programming algorithm (pDPA)

Usage

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

Arguments

x	A vector of double : the signal to be segmented
Kmax	Max number of segments
mini	Min value for the mean parameter of the segment
maxi	Max value for the mean parameter of the segment

Details

This implementation uses functional pruning and segment neighborhood, and the L2-loss function

Value

A list with a vector containing the position of the change-points

Author(s)

Guillem Rigaill

References

Rigaill, G. (2015). A pruned dynamic programming algorithm to recover the best segmentations with 1 to K_max change-points. Journal de la Societe Francaise de Statistique, 156(4), 180-205.

See Also

doDynamicProgramming for a higher-level function

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)

## generate a synthetic CN profile
K <- 10
len <- 1e4
sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
```

```
datS <- sim$profile

## run pruned DPA segmentation
res <- Fpsn(datS[["c"]], Kmax=2*K+1)

## plot segmentation results for the true number of breakpoints
bkp <- res$t.est[K+1, 1:K]
plotSeg(datS, breakpoints=bkp)
```

`getCopyNumberDataByResampling` *Generate a copy number profile by resampling*

Description

Generate a copy number profile by resampling input data

Usage

```
getCopyNumberDataByResampling(length, nBkp = NA, bkp = NULL,
  regData = NULL, regions = NULL, regAnnot = NULL, minLength = 0,
  regionSize = 0, connex = TRUE)
```

Arguments

`length`	length of the profile
`nBkp`	number of breakpoints. If `NULL`, then argument `bkp` is expected to be provided.
`bkp`	a numeric vector of breakpoint positions that may be used to bypass the breakpoint generation step. Defaults to `NULL`.
`regData`	a data.frame containing copy number data for different types of copy number regions. Columns: **c** Total copy number **b** Allele B fraction (a.k.a. BAF) **region** a character value, annotation label for the region. See Details. **genotype** the (germline) genotype of SNPs. By definition, rows with missing genotypes are interpreted as non-polymorphic loci (a.k.a. copy number probes).
`regions`	a character vector of region labels that may be used to bypass the region label generation step. Defaults to `NULL`.
`regAnnot`	a data.frame containing annotation data for each copy number region. Columns: **region** label of the form (must match `regData[["region"]]`). **freq** frequency (in [0,1]) of this type of region in the genome. If `NULL` (the default), frequencies of regions (0,1), (0,2), (1,1) and (1,2) (the most common alterations) are set to represent 90% of the regions. `sum(regAnnot[["freq"]])` should be 1.

minLength	minimum length of region between breakpoints. Defaults to 0.
regionSize	If regionSize>0, breakpoints are included by pairs, where the distance within pair is set to regionSize. nBkp is then required to be an even number.
connex	If TRUE, any two successive regions are constrained to be connex in the (minor CN, major CN) space. See 'Details'.

Details

This function generates a random copy number profile of length 'length', with 'nBkp' breakpoints randomly chosen. Between two breakpoints, the profile is constant and taken among the different types of regions in regData.

Elements of regData[["region"]] must be of the form "(C1,C2)", where C1 denotes the minor copy number and C2 denotes the major copy number. For example,

(1,1) Normal

(0,1) Hemizygous deletion

(0,0) Homozygous deletion

(1,2) Single copy gain

(0,2) Copy-neutral LOH

(2,2) Balanced two-copy gain

(1,3) Unbalanced two-copy gain

(0,3) Single-copy gain with LOH

If 'connex' is set to TRUE (the default), transitions between copy number regions are constrained in such a way that for any breakpoint, one of the minor and the major copy number does not change. Equivalently, this means that all breakpoints can be seen in both total copy numbers and allelic ratios.

Value

A list with elements

profile the profile (a length by 2 data.frame containing the same fields as the input data regData.

bkp a vector of bkp positions (the last row index before a breakpoint)

regions	a character vector of region labels

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Pierre-Jean, M, Rigaill, G. J. and Neuvial, P. (2015). "Performance Evaluation of DNA Copy Number Segmentation Methods." *Briefings in Bioinformatics*, no. 4: 600-615.

Examples

```
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)
sim <- getCopyNumberDataByResampling(len=1e4, nBkp=5, minLength=100, regData=affyDat)
plotSeg(sim$profile, sim$bkp)

## another run with identical parameters
bkp <- sim$bkp
regions <- sim$regions
sim2 <- getCopyNumberDataByResampling(len=1e4, bkp=bkp, regData=affyDat, regions=regions)
plotSeg(sim2$profile, bkp)

## change tumor fraction but keep same "truth"
affyDatC <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=0.5)
simC <- getCopyNumberDataByResampling(len=1e4, bkp=bkp, regData=affyDatC, regions=regions)
plotSeg(simC$profile, bkp)

## restrict to only normal, single copy gain, and copy-neutral LOH
## with the same bkp
affyDatR <- subset(affyDat, region %in% c("(1,1)", "(0,2)", "(1,2)"))
simR <- getCopyNumberDataByResampling(len=1e4, bkp=bkp, regData=affyDatR)
plotSeg(simR$profile, bkp)

## Same 'truth', on another dataSet
regions <- simR$regions
illuDat <- acnr::loadCnRegionData(dataSet="GSE11976", tumorFraction=1)
sim <- getCopyNumberDataByResampling(len=1e4, bkp=bkp, regData=illuDat, regions=regions)
plotSeg(sim$profile, sim$bkp)
```

---

getTpFp *Calculate the number of true positives and false positives*

Description

Calculate the number of true positives and false positives among candidate breakpoints

Usage

```
getTpFp(candidates, trueBkp, tol, relax = -1)
```

Arguments

candidates	Breakpoints found by the methods
trueBkp	True breakpoints
tol	Tolerance on the position of candidate breakpoints called true
relax	Controls the way multiple breapoints within tolerance area are recorded.
	1 count one true positive if there is at least one breakpoint within tolerance area

0 count one true positive only if there is exactly one breakpoint within tolerance area

-1 count only one true positive if there is exactly one breakpoint within tolerance area; other breakpoints are counted as false positives

Value

A list with elements:

TP The number of true positives

FP The number of false positives

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=0.7)

## generate a synthetic CN profile
K <- 10
len <- 2e4
sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=affyDat)
datS <- sim$profile

## (group-)fused Lasso segmentation
res <- PSSeg(data=datS, K=2*K, method="GFLars", stat="c", profile=TRUE)

## results of the initial (group-)fused lasso segmentation
getTpFp(res$initBkp, sim$bkp, tol=10, relax=-1)
getTpFp(res$initBkp, sim$bkp, tol=10, relax=0)
getTpFp(res$initBkp, sim$bkp, tol=10, relax=1)
plotSeg(datS, breakpoints=list(sim$bkp, res$initBkp))

## results after pruning (group-)fused Lasso candidates by dynamic programming)
getTpFp(res$bestBkp, sim$bkp, tol=10, relax=-1)
getTpFp(res$bestBkp, sim$bkp, tol=10, relax=0)
getTpFp(res$bestBkp, sim$bkp, tol=10, relax=1)
plotSeg(datS, breakpoints=list(sim$bkp, res$bestBkp))
```

---

jointSeg *Joint segmentation of multivariate signals*

Description

Joint segmentation of multivariate signals in two steps:

1. first-pass segmentation. By default, a fast, greedy approach is used (see `method`).
2. pruning of the candidate change points obtained by dynamic programming

Usage

```
jointSeg(Y, method = "RBS", stat = NULL, dpStat = stat,
  segFUN = NULL, jitter = NULL,
  modelSelectionMethod = ifelse(match(method, c("DynamicProgramming",
  "RBS", "GFLars"), nomatch = 0) > 0, "Lebarbier", "none"),
  modelSelectionOnDP = (match(method, c("DynamicProgramming", "RBS",
  "GFLars"), nomatch = 0) > 0), ..., profile = FALSE, verbose = FALSE)
```

Arguments

Y	The signal to be segmented (a matrix or a numeric vector)
method	A character value, the type of segmentation method used. May be one of: **"RBS"** Recursive Binary Segmentation (the default), see segmentByRBS as described in Gey and Lebarbier (2005) **"GFLars"** Group fused LARS as described in Bleakley and Vert (2011). **"DP"** Dynamic Programming (Bellman, 1956). For univariate signals the pruned DP of Rigaill et al (2010) is used. **"other"** The segmentation method is passed as a function using argument segFUN (see examples in directory otherMethods of the jointseg package).
stat	A vector containing the names or indices of the columns of Y to be segmented
dpStat	A vector containing the names or indices of the columns of Y to be segmented at the second round of segmentation. Defaults to the value of argument stat.
segFUN	The segmentation function to be used when method is set to other. Not used otherwise.
jitter	Uncertainty on breakpoint position after initial segmentation. Defaults to NULL. See Details.
modelSelectionMethod	A character value, the name of the method used to perform model selection.
modelSelectionOnDP	A logical value. If TRUE (the default), model selection is performed on segmentation after dynamic programming; else model selection is performed on initial segmentation. Only applies to methods "DP", "RBS" and "GFLars".
...	Further arguments to be passed to the lower-level segmentation method determined by argument method.
profile	A logical value: trace time and memory usage ?
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Details

If modelSelectionOnDP is set to FALSE, then model selection is run on the sets of the form bkp[1:k] for $1 \leq k \leq length(bkp)$, where bkp is the set of breakpoints identified by the initial segmentation. In particular, this implies that the candidate breakpoints in bkp are sorted by order of appearance and not by position.

If jitter is not NULL, it should be a vector of integer indices. The set of candidate breakpoints passed on to dynamic programming is augmented by all indices distant from an element of jitter

from one of the candidates. For example, if `jitter==c(-1, 0, 1)` and the initial set of breakpoints is `c(1,5)` then dynamic programming is run on `c(1,2,4,5,6)`.

If the return value of the initial segmentation has an element named `dpseg`, then initial segmentation results are not pruned by dynamic programming.

References

Bleakley, K., & Vert, J. P. (2011). The group fused lasso for multiple change-point detection. arXiv preprint arXiv:1106.4199.

Vert, J. P., & Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group LARS. Advances in Neural Information Processing Systems, 23, 2343-2351.

Gey, S., & Lebarbier, E. (2008). Using CART to Detect Multiple Change Points in the Mean for Large Sample. http://hal.archives-ouvertes.fr/hal-00327146/

Rigaill, G. (2010). Pruned dynamic programming for optimal multiple change-point detection. arXiv preprint arXiv:1004.0887.

See Also

`pruneByDP`

Examples

```
## A two-dimensional signal
p <- 2
trueK <- 10
len <- 1e4
sim <- randomProfile(len, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- jointSeg(Y, method="RBS", K=K)
bkp <- res$bestBkp
getTpFp(bkp, sim$bkp, tol=5, relax = -1)   ## true and false positives
plotSeg(Y, list(sim$bkp, res$bestBkp), col=1)

## Now we add some NA:s in one dimension
jj <- sim$bkp[1]
Y[jj-seq(-10,10), p] <- NA
res2 <- jointSeg(Y, method="RBS", K=K, verbose=TRUE)
bkp <- res2$bestBkp
getTpFp(res2$bestBkp, sim$bkp, tol=5, relax = -1)   ## true and false positives
```

`mapPositionsBack` *mapPositionsBack*

Description

Map breakpoint positions back to original space

Usage

```
mapPositionsBack(pos)
```

Arguments

pos	A sorted list of position (or indices)

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

---

modelSelection *Model selection*

Description

Select the best number of breakpoints

Usage

```
modelSelection(rse, n, c = 2.5, lambdas = NULL, method = c("Birge",
  "Lebarbier"))
```

Arguments

rse	RSE as output by pruneByDP
n	Length of the profile
c	Parameter for the model selection
lambdas	A list of candidate values for the calibration of the penalty
method	Method to calibrate the constant in the penalty for model selection

Details

This function is not intended to be called directly, but implicitly through jointSeg or PSSeg.

Value

A list with elements

kbest the best number of breakpoints

lambda A numerical value, the result of an internal model selection function

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Lebarbier, E. (2005). Detecting multiple change-points in the mean of Gaussian process by model selection. Signal processing, 85(4), 717-736

Birg\'e, L. (2001). Gaussian model selection. J.Eur Math. Soc, 3(3):203-268

See Also

jointSeg

PSSeg

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)
sim <- getCopyNumberDataByResampling(1e4, 5, minLength=100, regData=affyDat)
Y <- as.matrix(sim$profile[, "c"])

## Find candidate breakpoints
K <- 50
resRBS <- segmentByRBS(Y, K=K)
## Prune candidate breakpoints
resDP <- pruneByDP(Y, candCP=resRBS$bkp)
selectedModel <- modelSelection(rse=resDP$rse, n=nrow(Y), method="Lebarbier")
str(selectedModel)

## breakpoints of the best model
bestBkp <- resDP$bkp[[selectedModel$kbest]]
print(bestBkp)

## truth
print(sim$bkp)

## Note that all of the above can be done directly using 'PSSeg'
res <- PSSeg(sim$profile, method="RBS", stat="c", K=K)
##  stopifnot(identical(res$bestBkp, bestBkp))
```

---

plotSeg *Plot signal and breakpoints with segment-level signal estimates*

Description

Plot signal and breakpoints with segment-level signal estimates

Usage

```
plotSeg(dat, breakpoints = NULL, regNames = NULL,
  exclNames = c("genotype", "region", "bT", "bN", "cellularity"),
  ylabs = colnames(dat), ylims = NULL, binExclPattern = "^b[N|T]*$",
  col = "#33333333", pch = 19, cex = 0.3)
```

Arguments

dat	A matrix or data frame whose rows correspond to loci sorted along the genome, or a numeric vector.
breakpoints	A vector of breakpoints positions, or a list of such vectors.
regNames	Region labels, a vector of length length(breakpoints)+1 (if breakpoints is a vector) or of length length(breakpoints[[1]])+1 (if breakpoints is a list).
exclNames	A vector of column names corresponding to columns that should not be plotted.
ylabs	A vector of 'y' labels (column names or indices) that should be plotted.
ylims	An optional $2 * d$ matrix with ylim values for each of the d dimensions to be plotted.
binExclPattern	A vector of column names or indices in colnames(dat) for which segment-level signal estimates should *not* be drawn.
col	Color of plotting symbol, see par
pch	Plotting symbol, see par
cex	Magnification factor for plotting symbol, see par

Details

Argument 'binCols' is mainly used to avoid calculating mean levels for allelic ratios, which would not make sense as they are typically multimodal.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

Examples

```
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=1)
sim <- getCopyNumberDataByResampling(1e4, 5, minLength=100, regData=affyDat)
dat <- sim$profile
res <- PSSeg(dat, method="RBS", stat=c("c", "d"), K=50)
bkpList <- list(true=sim$bkp, est=res$bestSeg)
plotSeg(dat, breakpoints=bkpList)
```

---

pruneByDP *Exact segmentation of a multivariate signal using dynamic programming.*

Description

Exact segmentation of a multivariate signal using dynamic programming.

Usage

```
pruneByDP(Y, candCP = 1:(nrow(Y) - 1), K = length(candCP),
  allowNA = TRUE, verbose = FALSE)
```

Arguments

Y	A n*p signal to be segmented
candCP	A vector of candidate change point positions (defaults to 1:(n-1))
K	The maximum number of change points to find
allowNA	A boolean value specifying whether missing values should be allowed or not.
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Details

This function retrieves the maximum likelihood solution of the gaussian homoscedastic change model into segments, for $K \in 1 \dots length(candCP)$. The dynamic programming algorithm used is quadratic in time. For signals containing more than 1000 points, we recommend using a first pass segmentation (see segmentByRBS) to find a smaller number of candidates, and to run pruneByDP on these candidates only, as initially suggested by Gey and Lebarbier (2008). These two steps can be performed using jointSeg for generic multivariate signals, and using PSSeg for copy number signals from SNP array data.

if allowNA, the calulation of the cost of removing candidate breakpoints between i and j for i<j tolerates missing values. Method !allowNA is maintained in order to check consistency with the original dynamic programming in the absence of NA:s.

Value

A list with elements:

bkpList	A list of vectors of change point positions for the best model with k change points, for k=1, 2, ... K
rse	A vector of K+1 residual squared errors
V	V[i,j] is the best RSE for segmenting intervals 1 to j

Note

This implementation is derived from the MATLAB code by Vert and Bleakley: http://cbio.ensmp.fr/GFLseg.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Bellman, R. (1961). On the approximation of curves by line segments using dynamic programming. Communications of the ACM, 4(6), 284.

Gey, S., & Lebarbier, E. (2008). Using CART to Detect Multiple Change Points in the Mean for Large Sample. http://hal.archives-ouvertes.fr/hal-00327146/

See Also

jointSeg, PSSeg

Examples

```
p <- 2
trueK <- 10
sim <- randomProfile(1e4, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- segmentByRBS(Y, K)
resP <- pruneByDP(Y, res$bkp)

##   Note that all of the above can be dmethod=="other"one directly using 'jointSeg'
resJ <- jointSeg(sim$profile, method="RBS", K=K)
stopifnot(identical(resP$bkpList, resJ$dpBkp))

## check consistency when no NA
resP2 <- pruneByDP(Y, res$bkp, allowNA=FALSE)
max(abs(resP$rse-resP2$rse))

plotSeg(Y, list(resP$bkp[[trueK]], sim$bkp), col=1)
```

PSSeg *Parent-Specific copy number segmentation*

Description

This function splits (bivariate) copy number signals into parent-specific (PS) segments using recursive binary segmentation

Usage

```
PSSeg(data, method, stat = NULL, dropOutliers = TRUE,
  rankTransform = FALSE, ..., profile = FALSE, verbose = FALSE)
```

Arguments

data	Data frame containing the following columns: **c:** Total copy number (logged or non-logged) **b:** Allele B fraction **genotype:** (germline) genotype of the SNP, coded as 0 for AA, 1/2 for AB, 1 for BB These data are assumed to be ordered by genome position.
method	**"RBS"** Recursive Binary Segmentation, see doRBS **"GFLars"** Group fused LARS as described in Bleakley and Vert (2011). **"DP"** Univariate pruned dynamic programming Rigaill et al (2010) or bivariate dynamic programming **"PSCBS"** Parent-specific copy number in paired tumor-normal studies using circular binary segmentation by Olshen A. et al (2011)

	"other" The segmentation method is passed as a function using argument `segFUN` (see examples in directory `otherMethods`).
`stat`	A vector containing the names or indices of the columns of `Y` to be segmented
`dropOutliers`	If TRUE, outliers are droped by using DNAcopy package
`rankTransform`	If TRUE, data are replaced by their ranks before segmentation
`...`	Further arguments to be passed to `jointSeg`
`profile`	Trace time and memory usage ?
`verbose`	A `logical` value: should extra information be output ? Defaults to `FALSE`.

Details

Before segmentation, the decrease in heterozygosity `d=2|b-1/2|` defined in Bengtsson et al, 2010 is calculated from the input data. `d` is only defined for heterozygous SNPs, that is, SNPs for which `data$genotype==1/2`. `d` may be seen as a "mirrored" version of allelic ratios (`b`): it converts them to a piecewise-constant signals by taking advantage of the bimodality of `b` for heterozygous SNPs. The rationale for this transformation is that allelic ratios (`b`) are only informative for heterozygous SNPs (see e.g. Staaf et al, 2008).

Before segmentation, the outliers in the copy number signal are droped according the method explained by Venkatraman, E. S. and Olshen, A. B., 2007.

The resulting data are then segmented using the `jointSeg` function, which combines an initial segmentation according to argument `method` and pruning of candidate change points by dynamic programming (skipped when the initial segmentation *is* dynamic programming).

If argument `stat` is not provided, then dynamic programming is run on the two dimensional statistic `"(c,d)"`.

If argument `stat` is provided, then dynamic programming is run on `stat`; in this case we implicitly assume that `stat` is a piecewise-constant signal.

Value

A list with elements

bestBkp Best set of breakpoints after dynamic programming

initBkp Results of the initial segmentation, using 'doNnn', where 'Nnn' corresponds to argument `method`

dpBkpList Results of dynamic programming, a list of vectors of breakpoint positions for the best model with k breakpoints for k=1, 2, ... K where `K=length(initBkp)`

prof a `matrix` providing time usage (in seconds) and memory usage (in Mb) for the main steps of the program. Only defined if argument `profile` is set to `TRUE`

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Bengtsson, H., Neuvial, P., & Speed, T. P. (2010). TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays. BMC bioinformatics, 11(1), 245.

Staaf, J., Lindgren, D., Vallon-Christersson, et al. (2008). Segmentation-based detection of allelic imbalance and loss-of-heterozygosity in cancer cells using whole genome SNP arrays. Genome Biol, 9(9), R136.

Pierre-Jean, M, Rigaill, G. J. and Neuvial, P. (2015). "Performance Evaluation of DNA Copy Number Segmentation Methods." *Briefings in Bioinformatics*, no. 4: 600-615.

See Also

jointSeg

Examples

```
## load known real copy number regions
affyDat <- acnr::loadCnRegionData(dataSet="GSE29172", tumorFraction=0.5)

## generate a synthetic CN profile
K <- 10
len <- 1e4
sim <- getCopyNumberDataByResampling(len, K, regData=affyDat)
datS <- sim$profile

## run binary segmentation (+ dynamic programming)
resRBS <- PSSeg(data=datS, method="RBS", stat=c("c", "d"), K=2*K, profile=TRUE)
resRBS$prof

getTpFp(resRBS$bestBkp, sim$bkp, tol=5)
plotSeg(datS, breakpoints=list(sim$bkp, resRBS$bestBkp))
```

---

randomProfile *Generate a random multi-dimensional profile with breakpoints and noise*

Description

Generate a random multi-dimensional profile with breakpoints and noise

Usage

```
randomProfile(length, nBkp, noiseLevel, dim, minLength = 0)
```

Arguments

length	length of the profile
nBkp	number of breakpoints
noiseLevel	variance of the signal between two breakpoints
dim	dimension of the profile
minLength	minimum length of region between breakpoints by default minLength = 0

Details

Generate a random profile (vector) of length length, with nBkp breakpoints randomly chosen. Between two breakpoints, the profile is constant, uniformly chosen between 0 and 1, and a Gaussian noice of variance noiseLevel is added.

Value

a list with elements

profile the profile (a length by dim matrix)

bkp the list of breakpoints positions (the last position at the left of a breakpoint)

Note

This implementation is derived from the MATLAB code by Vert and Bleakley: http://cbio.ensmp.fr/GFLseg.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

Examples

```
len <- 1e4
nBkp <- 10
noiseLevel <- 1
dim <- 2

sim <- randomProfile(len, nBkp, noiseLevel, dim)
res <- doGFLars(sim$profile, K=5*nBkp)
str(res)
```

retour_sn *Extract endpoint matrix from DP result*

Description

Extract endpoint matrix from DP result

Usage

```
retour_sn(path)
```

Arguments

path	the path vector of the "colibri_sn_R_c C" function

segmentByGFLars *Group fused Lars segmentation (low-level)*

Description

Low-level function for multivariate fused Lars segmentation (GFLars)

Usage

```
segmentByGFLars(Y, K, weights = defaultWeights(nrow(Y)),
  epsilon = 1e-09, verbose = FALSE)
```

Arguments

Y	A n*p matrix of signals to be segmented
K	The number of change points to find
weights	A (n-1)*1 vector of weights for the weigthed group fused Lasso penalty. See Details.
epsilon	Values smaller than epsilon are considered null. Defaults to 1e-9.
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Details

This function recrusively looks for the best candidate change point according to group-fused LARS. This is a low-level function. It is generally advised to use the wrapper doGFLars which also works on data frames, has a convenient argument stat, and includes a basic workaround for handling missing values.

See also jointSeg for combining group fused LARS segmentation with pruning by dynamic programming (pruneByDP).

See PSSeg for segmenting genomic signals from SNP arrays.

The default weights $\sqrt{n/(i*(n-i))}$ are calibrated as suggested by Bleakley and Vert (2011). Using this calibration, the first breakpoint maximizes the likelihood ratio test (LRT) statistic.

Value

A list with elements:

bkp A vector of k candidate change-point positions

lambda The estimated lambda values for each change-point

mean A vector of length p, the mean signal per column

value A i x p matrix of change-point values for the first i change-points

c $\hat{c}$, a n-1 x K matrix

Note

This implementation is derived from the MATLAB code by Vert and Bleakley: http://cbio.ensmp.fr/GFLseg.

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Bleakley, K., & Vert, J. P. (2011). The group fused lasso for multiple change-point detection. arXiv preprint arXiv:1106.4199.

Vert, J. P., & Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group LARS. Advances in Neural Information Processing Systems, 23, 2343-2351.

See Also

PSSeg, jointSeg, doGFLars, pruneByDP

Examples

```
p <- 2
trueK <- 10
sim <- randomProfile(1e4, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- segmentByGFLars(Y, K)
print(res$bkp)
print(sim$bkp)
plotSeg(Y, res$bkp)
```

segmentByRBS *Recursive Binary Segmentation (low-level)*

Description

Low-level function for multivariate Recursive Binary Segmentation (RBS)

Usage

```
segmentByRBS(Y, K, minRegionSize = 2, verbose = FALSE)
```

Arguments

Y	A n*p signal to be segmented
K	The number of change points to find
minRegionSize	Regions with less than minRegionSize are not split
verbose	A logical value: should extra information be output ? Defaults to FALSE.

Details

This function recrusively looks for the best candidate change point according to binary segmentation. This is the low-level function. It is generally advised to use the wrapper doRBS which also works on data frames and has a convenient argument stat.

See jointSeg for combining recursive binary segmentation with pruning by dynamic programming (pruneByDP).

See PSSeg for segmenting genomic signals from SNP arrays.

Each dimension of the original signal is scaled before segmentation, using estimateSd.

Value

A list with elements:

bkp	A vector of K estimated breakpoint positions, sorted by order of appearance
rse	the residual squared error (RSE) for the successive segmentations
gain	The gain provided by each breakpoints in terms of difference between RSE

Author(s)

Morgane Pierre-Jean and Pierre Neuvial

References

Gey, S., & Lebarbier, E. (2008). Using CART to Detect Multiple Change Points in the Mean for Large Sample. http://hal.archives-ouvertes.fr/hal-00327146/

See Also

PSSeg, jointSeg, doRBS, pruneByDP

Examples

```
p <- 2
trueK <- 10
len <- 1e4
sim <- randomProfile(len, trueK, 1, p)
Y <- sim$profile
K <- 2*trueK
res <- segmentByRBS(Y, K)
getTpFp(res$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives

cols <- rep(2, K)
cols[1:trueK] <- 3
par(mfrow=c(p,1))
for (ii in 1:p) {
    plot(Y[, ii], pch=19, cex=0.2)
    abline(v=res$bkp[1:trueK], col= cols)
    abline(v=sim$bkp, col=8, lty=2)
}

## NA:s in one dimension at a true breakpoint
jj <- sim$bkp[1]
Y[jj-seq(-10, 10), p] <- NA
res2 <- segmentByRBS(Y, K)
getTpFp(res2$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives

## NA:s in both dimensions at a true breakpoint
Y[jj-seq(-10, 10), ] <- NA
res3 <- segmentByRBS(Y, K)
getTpFp(res3$bkp, sim$bkp, tol=10, relax = -1)   ## true and false positives
```

Index