

# Package: PSCBS (via r-universe)

October 14, 2024

**Version** 0.67.0

**Depends** R (>= 3.2.0)

**Imports** R.methodsS3 (>= 1.8.2), R.oo (>= 1.25.0), R.utils (>= 2.12.0), R.cache (>= 0.16.0), matrixStats (>= 0.62.0), aroma.light (>= 2.4.0), DNACopy (>= 1.42.0), listenv (>= 0.8.0), future (>= 1.28.0), parallel, graphics, grDevices, stats, utils

**Suggests** R.rsp (>= 0.45.0), R.devices (>= 2.17.1), ggplot2 (>= 3.2.1)

**SuggestsNote** BioC (>= 3.1)

**VignetteBuilder** R.rsp

**Title** Analysis of Parent-Specific DNA Copy Numbers

**Description** Segmentation of allele-specific DNA copy number data and detection of regions with abnormal copy number within each parental chromosome. Both tumor-normal paired and tumor-only analyses are supported.

**License** GPL (>= 2)

**LazyLoad** TRUE

**ByteCompile** TRUE

**biocViews** aCGH, CopyNumberVariants, SNP, Microarray, OneChannel, TwoChannel, Genetics

**URL** <https://github.com/HenrikBengtsson/PSCBS>

**BugReports** <https://github.com/HenrikBengtsson/PSCBS/issues>

**Repository** <https://henrikbengtsson.r-universe.dev>

**RemoteUrl** <https://github.com/HenrikBengtsson/PSCBS>

**RemoteRef** master

**RemoteSha** 6860efd8490cbb1e7cb228422ac00f051ff8483c

## Contents

PSCBS-package . . . . .	2
callSegmentationOutliers . . . . .	3
CBS . . . . .	4
findLargeGaps . . . . .	6
gapsToSegments.data.frame . . . . .	7
NonPairedPSCBS . . . . .	8
PairedPSCBS . . . . .	9
PSCBS . . . . .	11
segmentByCBS . . . . .	12
segmentByNonPairedPSCBS . . . . .	17
segmentByPairedPSCBS . . . . .	20
<b>Index</b>	<b>25</b>

---

PSCBS-package	<i>Package PSCBS</i>
---------------	----------------------

---

### Description

Segmentation of allele-specific DNA copy number data and detection of regions with abnormal copy number within each parental chromosome. Both tumor-normal paired and tumor-only analyses are supported.

This package should be considered to be in an alpha or beta phase. You should expect the API to be changing over time.

### Installation and updates

To install this package, use `install.packages("PSCBS")`.

### To get started

To get started, see:

1. Vignettes '[Parent-specific copy-number segmentation using Paired PSCBS](#)' and '[Total copy-number segmentation using CBS](#)'.
2. `segmentByCBS()` - segments total copy-numbers, or any other unimodal genomic signals, using the CBS method [3,4].
3. `segmentByPairedPSCBS()` - segments allele-specific tumor signal from a tumor *with* a matched normal using the Paired PSCBS method [1,2].
4. `segmentByNonPairedPSCBS()` - segments allele-specific tumor signal from a tumor *without* a matched normal using the Non-Paired PSCBS method adopted from [1,2].

### How to cite

Please use [1] and [2] to cite when using Paired PSCBS, and [3] and [4] when using CBS. When using Non-Paired PSCBS, please cite [1] and [2] as well.

**License**

GPL ( $\geq 2$ ).

**Author(s)**

Henrik Bengtsson

**References**

- [1] A.B. Olshen, H. Bengtsson, P. Neuvial, P.T. Spellman, R.A. Olshen, V.E. Seshan, *Parent-specific copy number in paired tumor-normal studies using circular binary segmentation*, Bioinformatics, 2011
- [2] H. Bengtsson, P. Neuvial and T.P. Speed, *TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays*, BMC Bioinformatics, 2010
- [3] A.B. Olshen, E.S. Venkatraman (aka Venkatraman E. Seshan), R. Lucito and M. Wigler, *Circular binary segmentation for the analysis of array-based DNA copy number data*, Biostatistics, 2004
- [4] E.S. Venkatraman and A.B. Olshen, *A faster circular binary segmentation algorithm for the analysis of array CGH data*, Bioinformatics, 2007

---

callSegmentationOutliers

*Calls/drops single-locus outliers along the genome*

---

**Description**

Calls/drops single-locus outliers along the genome that have a signal that differ significantly from the neighboring loci.

**Usage**

```
## Default S3 method:
callSegmentationOutliers(y, chromosome=0, x=NULL, method="DNACopy::smooth.CNA", ...,
  verbose=FALSE)
## S3 method for class 'data.frame'
callSegmentationOutliers(y, ...)
## Default S3 method:
dropSegmentationOutliers(y, ...)
## S3 method for class 'data.frame'
dropSegmentationOutliers(y, ...)
```

**Arguments**

**y** A [numeric vector](#) of J genomic signals to be segmented.

**chromosome** (Optional) An [integer](#) scalar (or a [vector](#) of length J contain a unique value). Only used for annotation purposes.

x	Optional <a href="#">numeric vector</a> of J genomic locations. If <code>NULL</code> , index locations 1 : J are used.
method	A <a href="#">character</a> string specifying the method used for calling outliers.
...	Additional arguments passed to internal outlier detection method.
verbose	See <a href="#">Verbose</a> .

**Value**

`callSegmentationOutliers()` returns a [logical vector](#) of length J. `dropSegmentationOutliers()` returns an object of the same type as argument `y`, where the signals for which outliers were called have been set to `NA`.

**Missing and non-finite values**

Signals as well as genomic positions may contain missing values, i.e. `NA`s or `NaN`s. By definition, these cannot be outliers.

**Author(s)**

Henrik Bengtsson

**See Also**

Internally `smooth.CNA` is utilized to identify the outliers.

---

CBS

*The CBS class*

---

**Description**

A CBS object holds results from the Circular Binary Segmentation (CBS) method for *one* sample for one or more chromosomes.

Package: PSCBS

**Class CBS**

```
list
~~|
~~+--AbstractCBS
~~~~~|
~~~~~+--CBS
```

**Directly known subclasses:**

```
public abstract class CBS
extends AbstractCBS
```

**Usage**

```
CBS(...)
```

**Arguments**

... Arguments passed to the constructor of [AbstractCBS](#).

**Fields and Methods****Methods:**

as	-
c	-
estimateStandardDeviation	-
plotTracks	-
pruneBySdUndo	-
segmentByCBS	-
seqOfSegmentsByDP	-
writeSegments	-

**Methods inherited from AbstractCBS:**

adjustPloidyScale, all.equal, as.data.frame, clearCalls, drawChangePoints, drawKnownSegments, dropChangePoint, dropChangePoints, dropRegion, dropRegions, extractCNs, extractChromosome, extractChromosomes, extractRegions, extractSegments, extractWIG, getChangePoints, getChromosomeOffsets, getChromosomeRanges, getChromosomes, getLocusData, getLocusSignalNames, getMeanEstimators, getSampleName, getSegmentSizes, getSegmentTrackPrefixes, getSegments, mergeThreeSegments, mergeTwoSegments, nbrOfChangePoints, nbrOfChromosomes, nbrOfLoci, nbrOfSegments, normalizeTotalCNs, ploidy, ploidy<-, plotTracks, print, pruneByDP, pruneByHClust, renameChromosomes, report, resegment, resetSegments, sampleCNs, sampleName, sampleName<-, seqOfSegmentsByDP, setLocusData, setMeanEstimators, setPloidy, setSampleName, setSegments, shiftTCN, tileChromosomes, updateMeans, writeWIG

**Methods inherited from list:**

Ops,nonStructure,vector-method, Ops,structure,vector-method, Ops,vector,nonStructure-method, Ops,vector,structure-method, all.equal, as.data.frame, attachLocally, callHooks, coerce,ANY,list-method, relist, type.convert, within

**Difference to DNACopy object**

A CBS object is similar to DNACopy objects with the major difference that a CBS object holds only one sample, whereas a DNACopy object can hold more than one sample.

**See also**

The [segmentByCBS\(\)](#) method returns an object of this class.

**Author(s)**

Henrik Bengtsson

---

findLargeGaps	<i>Identifies gaps of a genome where there exist no observations</i>
---------------	--

---

**Description**

Identifies gaps of a genome where there exist no observations.

**Usage**

```
## Default S3 method:  
findLargeGaps(chromosome=NULL, x, minLength, resolution=1L, ...)
```

**Arguments**

chromosome	(Optional) An <a href="#">integer vector</a> of length J of chromosome indices.
x	A <a href="#">numeric vector</a> of J of genomic locations.
minLength	A positive <a href="#">numeric</a> scalar specifying the minimum length of a gap.
resolution	A non-negative <a href="#">numeric</a> specifying the minimum length unit, which by default equals one nucleotide/base pair.
...	Not used.

**Value**

Returns [data.frame](#) zero or more rows and with columns chromosome (if given), start, stop, and length.

**Author(s)**

Henrik Bengtsson

**See Also**

Use [gapsToSegments\(\)](#) to turn the set of identified gaps into the complementary set of segments such that they can be passed to [segmentByCBS\(\)](#), [segmentByPairedPSCBS\(\)](#) and [segmentByNonPairedPSCBS\(\)](#) via argument `knownSegments`.

---

`gapsToSegments.data.frame`*Gets the genomic segments that are complementary to the gaps*

---

### Description

Gets the genomic segments that are complementary to the gaps, with default chromosome boundaries being `-Inf` and `+Inf`.

### Usage

```
## S3 method for class 'data.frame'  
gapsToSegments(gaps, resolution=1L, minLength=0L, dropGaps=FALSE, ...)
```

### Arguments

<code>gaps</code>	A <code>data.frame</code> with columns <code>chromosome</code> , <code>start</code> , and <code>stop</code> . Any overlapping gaps will throw an error.
<code>resolution</code>	A non-negative <code>numeric</code> specifying the minimum length unit, which by default equals one nucleotide/base pair.
<code>minLength</code>	Minimum length of segments to be kept.
<code>dropGaps</code>	If <code>TRUE</code> , the gaps themselves are not part of the output.
<code>...</code>	Not used.

### Value

Returns `data.frame` of least one row with columns `chromosome` if that argument is given), `start`, `stop` and `length`. The segments are ordered along the genome.

### Author(s)

Henrik Bengtsson

### See Also

[findLargeGaps\(\)](#).

---

NonPairedPSCBS      *The NonPairedPSCBS class*

---

## Description

Package: PSCBS

### Class NonPairedPSCBS

```
list
~~|
~~+--AbstractCBS
~~~~~|
~~~~~+--PSCBS
~~~~~|
~~~~~+--NonPairedPSCBS
```

### Directly known subclasses:

```
public abstract class NonPairedPSCBS
extends PSCBS
```

A NonPairedPSCBS is an object containing the results from the Non-paired PSCBS method.

## Usage

```
NonPairedPSCBS(fit=list(), ...)
```

## Arguments

fit	A <code>list</code> structure containing the Non-paired PSCBS results.
...	Not used.

## Fields and Methods

### Methods:

*No methods defined.*

### Methods inherited from PSCBS:

as.data.frame, c, drawChangePoints, extractChromosomes, extractWIG, getLocusData, getLocusSignalNames, getSegmentTrackPrefixes, isLocallyPhased, isSegmentSplitter, normalizeTotalCNs, writeSegments

### Methods inherited from AbstractCBS:

adjustPloidyScale, all.equal, as.data.frame, clearCalls, drawChangePoints, drawKnownSegments, dropChangePoint, dropChangePoints, dropRegion, dropRegions, extractCNs, extractChromosome,



extractChromosomes, extractRegions, extractSegments, extractWIG, getChangePoints, getChromosomeOffsets, getChromosomeRanges, getChromosomes, getLocusData, getLocusSignalNames, getMeanEstimators, getSampleName, getSegmentSizes, getSegmentTrackPrefixes, getSegments, mergeThreeSegments, mergeTwoSegments, nbrOfChangePoints, nbrOfChromosomes, nbrOfLoci, nbrOfSegments, normalizeTotalCNs, ploidy, ploidy<-, plotTracks, print, pruneByDP, pruneByHClust, renameChromosomes, report, resegment, resetSegments, sampleCNs, sampleName, sampleName<-, seqOfSegmentsByDP, setLocusData, setMeanEstimators, setPloidy, setSampleName, setSegments, shiftTCN, tileChromosomes, updateMeans, writeWIG

#### Methods inherited from list:

Ops,nonStructure,vector-method, Ops,structure,vector-method, Ops,vector,nonStructure-method, Ops,vector,structure-method, all.equal, as.data.frame, attachLocally, callHooks, coerce,ANY,list-method, relist, type.convert, within

#### Author(s)

Henrik Bengtsson

#### See Also

The [segmentByNonPairedPSCBS\(\)](#) method returns an object of this class.

---

PairedPSCBS

*The PairedPSCBS class*

---

#### Description

Package: PSCBS

**Class PairedPSCBS**

```
list
~~|
~~+---AbstractCBS
~~~~~|
~~~~~+---PSCBS
~~~~~|
~~~~~+---PairedPSCBS
```

#### Directly known subclasses:

```
public abstract class PairedPSCBS
extends PSCBS
```

A PairedPSCBS is an object containing the results from the Paired PSCBS method.

**Usage**

```
PairedPSCBS(fit=list(), ...)
```

**Arguments**

<code>fit</code>	A <code>list</code> structure containing the Paired PSCBS results.
<code>...</code>	Not used.

**Fields and Methods****Methods:**

<code>callAB</code>	-
<code>callCopyNeutral</code>	-
<code>callGNL</code>	-
<code>callGNLByTCNofAB</code>	-
<code>callGainNeutralLoss</code>	-
<code>callLOH</code>	-
<code>callNTCN</code>	-
<code>callROH</code>	-
<code>estimateDeltaAB</code>	-
<code>estimateDeltaLOH</code>	-
<code>estimateKappa</code>	-
<code>extractCNS</code>	-
<code>hasBootstrapSummaries</code>	-
<code>plotTracks</code>	-
<code>segmentByNonPairedPSCBS</code>	-
<code>segmentByPairedPSCBS</code>	-
<code>seqOfSegmentsByDP</code>	-

**Methods inherited from PSCBS:**

`as.data.frame`, `c`, `drawChangePoints`, `extractChromosomes`, `extractWIG`, `getLocusData`, `getLocusSignalNames`, `getSegmentTrackPrefixes`, `isLocallyPhased`, `isSegmentSplitter`, `normalizeTotalCNS`, `writeSegments`

**Methods inherited from AbstractCBS:**

`adjustPloidyScale`, `all.equal`, `as.data.frame`, `clearCalls`, `drawChangePoints`, `drawKnownSegments`, `dropChangePoint`, `dropChangePoints`, `dropRegion`, `dropRegions`, `extractCNS`, `extractChromosome`, `extractChromosomes`, `extractRegions`, `extractSegments`, `extractWIG`, `getChangePoints`, `getChromosomeOffsets`, `getChromosomeRanges`, `getChromosomes`, `getLocusData`, `getLocusSignalNames`, `getMeanEstimators`, `getSampleName`, `getSegmentSizes`, `getSegmentTrackPrefixes`, `getSegments`, `mergeThreeSegments`, `mergeTwoSegments`, `nbrOfChangePoints`, `nbrOfChromosomes`, `nbrOfLoci`, `nbrOfSegments`, `normalizeTotalCNS`, `ploidy`, `ploidy<-`, `plotTracks`, `print`, `pruneByDP`, `pruneByHClust`, `renameChromosomes`, `report`, `resegment`, `resetSegments`, `sampleCNS`, `sampleName`, `sampleName<-`, `seqOfSegmentsByDP`, `setLocusData`, `setMeanEstimators`, `setPloidy`, `setSampleName`, `setSegments`, `shiftTCN`, `tileChromosomes`, `updateMeans`, `writeWIG`

**Methods inherited from list:**

Ops,nonStructure,vector-method, Ops,structure,vector-method, Ops,vector,nonStructure-method, Ops,vector,structure-method, all.equal, as.data.frame, attachLocally, callHooks, coerce,ANY,list-method, relist, type.convert, within

**Author(s)**

Henrik Bengtsson

**See Also**

The [segmentByPairedPSCBS\(\)](#) method returns an object of this class.

---

PSCBS

*The PSCBS class*

---

**Description**

Package: PSCBS

**Class PSCBS**

```
list
~~|
~~+--AbstractCBS
~~~~~|
~~~~~+--PSCBS
```

**Directly known subclasses:**

[NonPairedPSCBS](#), [PairedPSCBS](#)

```
public abstract class PSCBS
extends AbstractCBS
```

A PSCBS is an object containing results from parent-specific copy-number (PSCN) segmentation.

**Usage**

```
PSCBS(fit=list(), ...)
```

**Arguments**

```
fit          A list structure containing the PSCN segmentation results.
...          Not used.
```

## Fields and Methods

### Methods:

```

c -
isLocallyPhased -
normalizeTotalCNs -
writeSegments -

```

### Methods inherited from AbstractCBS:

adjustPloidyScale, all.equal, as.data.frame, clearCalls, drawChangePoints, drawKnownSegments, dropChangePoint, dropChangePoints, dropRegion, dropRegions, extractCNs, extractChromosome, extractChromosomes, extractRegions, extractSegments, extractWIG, getChangePoints, getChromosomeOffsets, getChromosomeRanges, getChromosomes, getLocusData, getLocusSignalNames, getMeanEstimators, getSampleName, getSegmentSizes, getSegmentTrackPrefixes, getSegments, mergeThreeSegments, mergeTwoSegments, nbrOfChangePoints, nbrOfChromosomes, nbrOfLoci, nbrOfSegments, normalizeTotalCNs, ploidy, ploidy<-, plotTracks, print, pruneByDP, pruneByHClust, renameChromosomes, report, resegment, resetSegments, sampleCNs, sampleName, sampleName<-, seqOfSegmentsByDP, setLocusData, setMeanEstimators, setPloidy, setSampleName, setSegments, shiftTCN, tileChromosomes, updateMeans, writeWIG

### Methods inherited from list:

Ops,nonStructure,vector-method, Ops,structure,vector-method, Ops,vector,nonStructure-method, Ops,vector,structure-method, all.equal, as.data.frame, attachLocally, callHooks, coerce,ANY,list-method, relist, type.convert, within

### Author(s)

Henrik Bengtsson

### See Also

[PairedPSCBS](#).

---

segmentByCBS

*Segment genomic signals using the CBS method*

---

### Description

Segment genomic signals using the CBS method of the **DNAcopy** package. This is a convenient low-level wrapper for the `DNAcopy::segment()` method. It is intended to be applied to a sample at the time. For more details on the Circular Binary Segmentation (CBS) method see [1,2].

**Usage**

```
## Default S3 method:
segmentByCBS(y, chromosome=0L, x=NULL, index=seq_along(y), w=NULL, undo=0,
  avg=c("mean", "median"), ..., joinSegments=TRUE, knownSegments=NULL, seed=NULL,
  verbose=FALSE)
```

**Arguments**

y	A <a href="#">numeric vector</a> of J genomic signals to be segmented.
chromosome	Optional <a href="#">numeric vector</a> of length J, specifying the chromosome of each loci. If a scalar, it is expanded to a vector of length J.
x	Optional <a href="#">numeric vector</a> of J genomic locations. If <code>NULL</code> , index locations 1:J are used.
index	An optional <a href="#">integer vector</a> of length J specifying the genomewide indices of the loci.
w	Optional <a href="#">numeric vector</a> in [0,1] of J weights.
undo	A non-negative <a href="#">numeric</a> . If greater than zero, then arguments <code>undo.splits="sundo"</code> and <code>undo.SD=undo</code> are passed to <code>DNAcopy::segment()</code> . In the special case when <code>undo</code> is <code>+Inf</code> , the segmentation result will not contain any changepoints (in addition to what is specified by argument <code>knownSegments</code> ).
avg	A <a href="#">character</a> string specifying how to calculating segment mean levels <i>after</i> change points have been identified.
...	Additional arguments passed to the <code>DNAcopy::segment()</code> segmentation function.
joinSegments	If <code>TRUE</code> , there are no gaps between neighboring segments. If <code>FALSE</code> , the boundaries of a segment are defined by the support that the loci in the segments provides, i.e. there exist a locus at each end point of each segment. This also means that there is a gap between any neighboring segments, unless the change point is in the middle of multiple loci with the same position. The latter is what <code>DNAcopy::segment()</code> returns.
knownSegments	Optional <a href="#">data.frame</a> specifying <i>non-overlapping</i> known segments. These segments must not share loci. See <a href="#">findLargeGaps()</a> and <a href="#">gapsToSegments()</a> .
seed	An (optional) <a href="#">integer</a> specifying the random seed to be set before calling the segmentation method. The random seed is set to its original state when exiting. If <code>NULL</code> , it is not set.
verbose	See <a href="#">Verbose</a> .

**Details**

Internally [segment](#) of `DNAcopy` is used to segment the signals. This segmentation method support weighted segmentation.

**Value**

Returns a [CBS](#) object.

## Reproducibility

The `DNACopy::segment()` implementation of CBS uses approximation through random sampling for some estimates. Because of this, repeated calls using the same signals may result in slightly different results, unless the random seed is set/fixed.

## Missing and non-finite values

Signals may contain missing values (`NA` or `NaN`), but not infinite values (`+/-Inf`). Loci with missing-value signals are preserved and kept in the result.

Likewise, genomic positions may contain missing values. However, if they do, such loci are silently excluded before performing the segmentation, and are not kept in the results. The mapping between the input locus-level data and ditto of the result can be inferred from the `index` column of the locus-level data of the result.

None of the input data may have infinite values, i.e. `-Inf` or `+Inf`. If so, an informative error is thrown.

## Author(s)

Henrik Bengtsson

## References

- [1] A.B. Olshen, E.S. Venkatraman (aka Venkatraman E. Seshan), R. Lucito and M. Wigler, *Circular binary segmentation for the analysis of array-based DNA copy number data*, Biostatistics, 2004
- [2] E.S. Venkatraman and A.B. Olshen, *A faster circular binary segmentation algorithm for the analysis of array CGH data*, Bioinformatics, 2007

## See Also

To segment allele-specific tumor copy-number signals from a tumor *with* a matched normal, see `segmentByPairedPSCBS()`. For the same *without* a matched normal, see `segmentByNonPairedPSCBS()`.

It is also possible to prune change points after segmentation (with identical results) using `pruneBySdUndo()`.

## Examples

```
# -----
# Simulating copy-number data
# -----
set.seed(0xBEEF)

# Number of loci
J <- 1000

mu <- double(J)
mu[200:300] <- mu[200:300] + 1
mu[350:400] <- NA # centromere
mu[650:800] <- mu[650:800] - 1
eps <- rnorm(J, sd=1/2)
```

```

y <- mu + eps
x <- sort(runif(length(y), max=length(y))) * 1e5
w <- runif(J)
w[650:800] <- 0.001

# -----
# Segmentation
# -----
fit <- segmentByCBS(y, x=x)
print(fit)
plotTracks(fit)

xlab <- "Position (Mb)"
ylim <- c(-3,3)
xMb <- x/1e6
plot(xMb,y, pch=20, col="#aaaaaa", xlab=xlab, ylim=ylim)
drawLevels(fit, col="red", lwd=2, xScale=1e-6)

# -----
# TESTS
# -----
fit <- segmentByCBS(y, x=x, seed=0xBEEF)
print(fit)
##   id chromosome      start      end nbrOfLoci   mean
## 1  y           0  55167.82 20774251    201  0.0164
## 2  y           0 20774250.85 29320105     99  1.0474
## 3  y           0 29320104.86 65874675    349 -0.0227
## 4  y           0 65874675.06 81348129    151 -1.0813
## 5  y           0 81348129.20 99910827    200 -0.0612

# Test #1: Reverse the ordering and segment
fitR <- segmentByCBS(rev(y), x=rev(x), seed=0xBEEF)
# Sanity check
stopifnot(all.equal(getSegments(fitR), getSegments(fit)))
# Sanity check
stopifnot(all.equal(rev(getLocusData(fitR)$index), getLocusData(fit)$index))

# Test #2: Reverse, but preserve ordering of 'data' object
fitRP <- segmentByCBS(rev(y), x=rev(x), preserveOrder=TRUE)
stopifnot(all.equal(getSegments(fitRP), getSegments(fit)))

# (Test #3: Change points inbetween data points at the same locus)
x[650:654] <- x[649]
fitC <- segmentByCBS(rev(y), x=rev(x), preserveOrder=TRUE, seed=0xBEEF)

# Test #4: Allow for some missing values in signals
y[450] <- NA

```

```

fitD <- segmentByCBS(y, x=x, seed=0xBEEF)

# Test #5: Allow for some missing genomic annotations
x[495] <- NA
fitE <- segmentByCBS(y, x=x, seed=0xBEEF)

# Test #6: Undo all change points found
fitF <- segmentByCBS(y, x=x, undo=Inf, seed=0xBEEF)
print(fitF)
stopifnot(nbrOfSegments(fitF) == 1L)

# -----
# MISC.
# -----
# Emulate a centromere
x[650:699] <- NA
fit <- segmentByCBS(y, x=x, seed=0xBEEF)
xMb <- x/1e6
plot(xMb,y, pch=20, col="#aaaaaa", xlab=xlab, ylim=ylim)
drawLevels(fit, col="red", lwd=2, xScale=1e-6)

fitC <- segmentByCBS(y, x=x, joinSegments=FALSE, seed=0xBEEF)
drawLevels(fitC, col="blue", lwd=2, xScale=1e-6)

# -----
# Multiple chromosomes
# -----
# Appending CBS results
fit1 <- segmentByCBS(y, chromosome=1, x=x)
fit2 <- segmentByCBS(y, chromosome=2, x=x)
fit <- c(fit1, fit2)
print(fit)
plotTracks(fit, subset=NULL, lwd=2, Clim=c(-3,3))

# Segmenting multiple chromosomes at once
chromosomeWG <- rep(1:2, each=J)
xWG <- rep(x, times=2)
yWG <- rep(y, times=2)
fitWG <- segmentByCBS(yWG, chromosome=chromosomeWG, x=xWG)
print(fitWG)
plotTracks(fitWG, subset=NULL, lwd=2, Clim=c(-3,3))

# Assert same results
fit$data[, "index"] <- getLocusData(fitWG)[, "index"] # Ignore 'index'
stopifnot(all.equal(getLocusData(fitWG), getLocusData(fit)))
stopifnot(all.equal(getSegments(fitWG), getSegments(fit)))

```



---

```
segmentByNonPairedPSCBS
```

*Segment total copy numbers and allele B fractions using the Non-paired PSCBS method*

---

## Description

Segment total copy numbers and allele B fractions using the Non-paired PSCBS method [1]. This method does not requires matched normals. This is a low-level segmentation method. It is intended to be applied to one tumor sample at the time.

## Usage

```
## Default S3 method:
segmentByNonPairedPSCBS(CT, betaT, ..., flavor=c("tcn", "tcn&dh", "tcn,dh",
  "sqrt(tcn),dh", "sqrt(tcn)&dh"), tauA=NA, tauB=1 - tauA, verbose=FALSE)
```

## Arguments

CT	A <a href="#">numeric vector</a> of J tumor total copy number (TCN) ratios in $[0, +\text{Inf}]$ (due to noise, small negative values are also allowed). The TCN ratios are typically scaled such that copy-neutral diploid loci have a mean of two.
betaT	A <a href="#">numeric vector</a> of J tumor allele B fractions (BAFs) in $[0, 1]$ (due to noise, values may be slightly outside as well) or <a href="#">NA</a> for non-polymorphic loci.
...	Additional arguments passed to <a href="#">segmentByPairedPSCBS()</a> .
flavor	A <a href="#">character</a> specifying what type of segmentation and calling algorithm to be used.
tauA, tauB	Lower and upper thresholds ( $\text{tauA} < \text{tauB}$ for calling SNPs heterozygous based on the tumor allele B fractions (betaT). If <a href="#">NA</a> , then they are estimates from data.
verbose	See <a href="#">Verbose</a> .

## Details

Internally [segmentByPairedPSCBS\(\)](#) is used for segmentation. This segmentation method does *not* support weights.

## Value

Returns the segmentation results as a [NonPairedPSCBS](#) object.

## Reproducibility

The "DNAcopy::segment" implementation of CBS uses approximation through random sampling for some estimates. Because of this, repeated calls using the same signals may result in slightly different results, unless the random seed is set/fixed.

### Whole-genome segmentation is preferred

Although it is possible to segment each chromosome independently using Paired PSCBS, we strongly recommend to segment whole-genome (TCN,BAF) data at once. The reason for this is that downstream CN-state calling methods, such as the AB and the LOH callers, performs much better on whole-genome data. In fact, they may fail to provide valid calls if done chromosome by chromosome.

### Missing and non-finite values

The total copy number signals as well as any optional positions must not contain missing values, i.e. NAs or NaNs. If there are any, an informative error is thrown. Allele B fractions may contain missing values, because such are interpreted as representing non-polymorphic loci.

None of the input signals may have infinite values, i.e. -Inf or +Inf. If so, an informative error is thrown.

### Non-Paired PSCBS with known genotypes

If allele B fractions for the matched normal (betaN) are not available, but genotypes (muN) are, then it is possible to run Paired PSCBS. See [segmentByPairedPSCBS\(\)](#) for details.

### Author(s)

Henrik Bengtsson

### References

- [1] A.B. Olshen, H. Bengtsson, P. Neuvial, P.T. Spellman, R.A. Olshen, V.E. Seshan, *Parent-specific copy number in paired tumor-normal studies using circular binary segmentation*, Bioinformatics, 2011
- [2] H. Bengtsson, P. Neuvial and T.P. Speed, *TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays*, BMC Bioinformatics, 2010

### See Also

To segment paired tumor-normal total copy numbers and allele B fractions, see [segmentByPairedPSCBS\(\)](#).

To segment total copy numbers, or any other unimodal signals, see [segmentByCBS\(\)](#).

### Examples

```
verbose <- R.utils::Arguments$getVerbose(-10*interactive(), timestamp=TRUE)

# -----
# Load SNP microarray data
# -----
data <- PSCBS::exampleData("paired.chr01")
str(data)
```

```

# -----
# Paired PSCBS segmentation
# -----
# Drop single-locus outliers
dataS <- dropSegmentationOutliers(data)

# Speed up example by segmenting fewer loci
dataS <- dataS[seq(from=1, to=nrow(data), by=20),]

str(dataS)

R.oo::attachLocally(dataS)

# Non-Paired PSCBS segmentation
fit <- segmentByNonPairedPSCBS(CT, betaT=betaT,
                              chromosome=chromosome, x=x,
                              seed=0xBEEF, verbose=verbose)

print(fit)

# -----
# Bootstrap segment level estimates
# (used by the AB caller, which, if skipped here,
# will do it automatically)
# -----
fit <- bootstrapTCNandDHByRegion(fit, B=100, verbose=verbose)
print(fit)

# -----
# Calling segments in allelic balance (AB)
# NOTE: Ideally, this should be done on whole-genome data
# -----
# Explicitly estimate the threshold in DH for calling AB
# (which be done by default by the caller, if skipped here)
deltaAB <- estimateDeltaAB(fit, flavor="qq(DH)", verbose=verbose)
print(deltaAB)

fit <- callAB(fit, delta=deltaAB, verbose=verbose)
print(fit)

# Even if not explicitly specified, the estimated
# threshold parameter is returned by the caller
stopifnot(fit$params$deltaAB == deltaAB)

# -----
# Calling segments in loss-of-heterozygosity (LOH)
# NOTE: Ideally, this should be done on whole-genome data
# -----
# Explicitly estimate the threshold in C1 for calling LOH

```

```

# (which be done by default by the caller, if skipped here)
deltaLOH <- estimateDeltaLOH(fit, flavor="minC1|nonAB", verbose=verbose)
print(deltaLOH)

fit <- callLOH(fit, delta=deltaLOH, verbose=verbose)
print(fit)
plotTracks(fit)

# Even if not explicitly specified, the estimated
# threshold parameter is returned by the caller
stopifnot(fit$params$deltaLOH == deltaLOH)

```

---

segmentByPairedPSCBS *Segment total copy numbers and allele B fractions using the Paired PSCBS method*

---

## Description

Segment total copy numbers and allele B fractions using the Paired PSCBS method [1]. This method requires matched normals. This is a low-level segmentation method. It is intended to be applied to one tumor-normal sample at the time.

## Usage

```

## Default S3 method:
segmentByPairedPSCBS(CT, thetaT=NULL, thetaN=NULL, betaT=NULL, betaN=NULL, muN=NULL,
  rho=NULL, chromosome=0, x=NULL, alphaTCN=0.009, alphaDH=0.001, undoTCN=0, undoDH=0,
  ..., avgTCN=c("mean", "median"), avgDH=c("mean", "median"),
  flavor=c("tcn&dh", "tcn,dh", "sqrt(tcn),dh", "sqrt(tcn)&dh", "tcn"), tbn=is.null(rho),
  joinSegments=TRUE, knownSegments=NULL, dropMissingCT=TRUE, seed=NULL, verbose=FALSE,
  preserveScale=FALSE)

```

## Arguments

CT	A <a href="#">numeric vector</a> of J tumor total copy number (TCN) ratios in $[0, +\text{Inf}]$ (due to noise, small negative values are also allowed). The TCN ratios are typically scaled such that copy-neutral diploid loci have a mean of two.
thetaT, thetaN	(alternative) As an alternative to specifying tumor TCN <i>ratios</i> relative to the match normal by argument CT, one may specify total tumor and normal signals separately, in which case the TCN ratios CT are calculated as $CT = 2 * \theta T / \theta N$ .
betaT	A <a href="#">numeric vector</a> of J tumor allele B fractions (BAFs) in $[0,1]$ (due to noise, values may be slightly outside as well) or <a href="#">NA</a> for non-polymorphic loci.
betaN	A <a href="#">numeric vector</a> of J matched normal BAFs in $[0,1]$ (due to noise, values may be slightly outside as well) or <a href="#">NA</a> for non-polymorphic loci.
muN	An optional <a href="#">numeric vector</a> of J genotype calls in $\{0, 1/2, 1\}$ for AA, AB, and BB, respectively, and <a href="#">NA</a> for non-polymorphic loci. If not given, they are estimated from the normal BAFs using <a href="#">callNaiveGenotypes</a> as described in [2].

rho	(alternative to betaT and betaN/muN) A <b>numeric vector</b> of J decrease-of-heterozygosity signals (DHs) in [0,1] (due to noise, values may be slightly larger than one as well). By definition, DH should be <b>NA</b> for homozygous loci and for non-polymorphic loci.
chromosome	(Optional) An <b>integer</b> scalar (or a <b>vector</b> of length J), which can be used to specify which chromosome each locus belongs to in case multiple chromosomes are segments. This argument is also used for annotation purposes.
x	Optional <b>numeric vector</b> of J genomic locations. If <b>NULL</b> , index locations 1 : J are used.
alphaTCN, alphaDH	The significance levels for segmenting total copy numbers (TCNs) and decrease-in-heterozygosity signals (DHs), respectively.
undoTCN, undoDH	Non-negative <b>numeric</b> s. If greater than 0, then a cleanup of segmentations post segmentation is done. See argument undo of <code>segmentByCBS()</code> for more details.
avgTCN, avgDH	A <b>character</b> string specifying how to calculating segment mean levels <i>after</i> change points have been identified.
...	Additional arguments passed to <code>segmentByCBS()</code> .
flavor	A <b>character</b> specifying what type of segmentation and calling algorithm to be used.
tbn	If <b>TRUE</b> , betaT is normalized before segmentation using the TumorBoost method [2], otherwise not.
joinSegments	If <b>TRUE</b> , there are no gaps between neighboring segments. If <b>FALSE</b> , the boundaries of a segment are defined by the support that the loci in the segments provides, i.e. there exist a locus at each end point of each segment. This also means that there is a gap between any neighboring segments, unless the change point is in the middle of multiple loci with the same position. The latter is what <code>DNAcopy::segment()</code> returns.
knownSegments	Optional <b>data.frame</b> specifying <i>non-overlapping</i> known segments. These segments must not share loci. See <code>findLargeGaps()</code> and <code>gapsToSegments()</code> .
dropMissingCT	If <b>TRUE</b> , loci for which 'CT' is missing are dropped, otherwise not.
seed	An (optional) <b>integer</b> specifying the random seed to be set before calling the segmentation method. The random seed is set to its original state when exiting. If <b>NULL</b> , it is not set.
verbose	See <b>Verbose</b> .
preserveScale	<i>Defunct - gives an error is specified.</i>

## Details

Internally `segmentByCBS()` is used for segmentation. The Paired PSCBS segmentation method does *not* support weights.

## Value

Returns the segmentation results as a `PairedPSCBS` object.

## Reproducibility

The "DNAcopy::segment" implementation of CBS uses approximation through random sampling for some estimates. Because of this, repeated calls using the same signals may result in slightly different results, unless the random seed is set/fixed.

## Whole-genome segmentation is preferred

Although it is possible to segment each chromosome independently using Paired PSCBS, we strongly recommend to segment whole-genome (TCN,BAF) data at once. The reason for this is that downstream CN-state calling methods, such as the AB and the LOH callers, performs much better on whole-genome data. In fact, they may fail to provide valid calls if done chromosome by chromosome.

## Missing and non-finite values

The total copy number signals as well as any optional positions must not contain missing values, i.e. `NA`s or `NaN`s. If there are any, an informative error is thrown. Allele B fractions may contain missing values, because such are interpreted as representing non-polymorphic loci.

None of the input signals may have infinite values, i.e. `-Inf` or `+Inf`. If so, an informative error is thrown.

## Paired PSCBS with only genotypes

If allele B fractions for the matched normal (`betaN`) are not available, but genotypes (`muN`) are, then it is possible to run a version of Paired PSCBS where TumorBoost normalization of the tumor allele B fractions is skipped. In order for this to work, argument `tbn` must be set to `FALSE`.

## Author(s)

Henrik Bengtsson

## References

- [1] A.B. Olshen, H. Bengtsson, P. Neuvial, P.T. Spellman, R.A. Olshen, V.E. Seshan, *Parent-specific copy number in paired tumor-normal studies using circular binary segmentation*, Bioinformatics, 2011
- [2] H. Bengtsson, P. Neuvial and T.P. Speed, *TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays*, BMC Bioinformatics, 2010

## See Also

Internally, `callNaiveGenotypes` is used to call naive genotypes, `normalizeTumorBoost` is used for TumorBoost normalization, and `segmentByCBS()` is used to segment TCN and DH separately.

To segment tumor total copy numbers and allele B fractions *without* a matched normal, see `segmentByNonPairedPSCBS()`.

To segment total copy-numbers, or any other unimodal signals, see `segmentByCBS()`.

**Examples**

```

verbose <- R.utils::Arguments$getVerbose(-10*interactive(), timestamp=TRUE)

# -----
# Load SNP microarray data
# -----
data <- PSCBS::exampleData("paired.chr01")
str(data)

# -----
# Paired PSCBS segmentation
# -----
# Drop single-locus outliers
dataS <- dropSegmentationOutliers(data)

# Speed up example by segmenting fewer loci
dataS <- dataS[seq(from=1, to=nrow(data), by=10),]

str(dataS)

R.oo::attachLocally(dataS)

# Paired PSCBS segmentation
fit <- segmentByPairedPSCBS(CT, betaT=betaT, betaN=betaN,
                           chromosome=chromosome, x=x,
                           seed=0xBEEF, verbose=verbose)

print(fit)

# Plot results
plotTracks(fit)

# -----
# Bootstrap segment level estimates
# (used by the AB caller, which, if skipped here,
# will do it automatically)
# -----
fit <- bootstrapTCNandDHByRegion(fit, B=100, verbose=verbose)
print(fit)
plotTracks(fit)

# -----
# Calling segments in allelic balance (AB)
# NOTE: Ideally, this should be done on whole-genome data
# -----
# Explicitly estimate the threshold in DH for calling AB
# (which be done by default by the caller, if skipped here)
deltaAB <- estimateDeltaAB(fit, flavor="qq(DH)", verbose=verbose)
print(deltaAB)
## [1] 0.1657131

```

```
fit <- callAB(fit, delta=deltaAB, verbose=verbose)
print(fit)
plotTracks(fit)

# Even if not explicitly specified, the estimated
# threshold parameter is returned by the caller
stopifnot(fit$params$deltaAB == deltaAB)

# - - - - -
# Calling segments in loss-of-heterozygosity (LOH)
# NOTE: Ideally, this should be done on whole-genome data
# - - - - -
# Explicitly estimate the threshold in C1 for calling LOH
# (which be done by default by the caller, if skipped here)
deltaLOH <- estimateDeltaLOH(fit, flavor="minC1|nonAB", verbose=verbose)
print(deltaLOH)
## [1] 0.625175

fit <- callLOH(fit, delta=deltaLOH, verbose=verbose)
print(fit)
plotTracks(fit)

# Even if not explicitly specified, the estimated
# threshold parameter is returned by the caller
stopifnot(fit$params$deltaLOH == deltaLOH)
```



# Index

- \* **IO**
  - callSegmentationOutliers, 3
  - findLargeGaps, 6
  - gapsToSegments.data.frame, 7
  - segmentByCBS, 12
  - segmentByNonPairedPSCBS, 17
  - segmentByPairedPSCBS, 20
- \* **classes**
  - CBS, 4
  - NonPairedPSCBS, 8
  - PairedPSCBS, 9
  - PSCBS, 11
- \* **methods**
  - callSegmentationOutliers, 3
  - gapsToSegments.data.frame, 7
- \* **package**
  - PSCBS-package, 2
- AbstractCBS, 4, 5, 8, 9, 11
- callNaiveGenotypes, 20, 22
- callSegmentationOutliers, 3
- CBS, 4, 13
- character, 4, 13, 17, 21
- data.frame, 6, 7, 13, 21
- dropSegmentationOutliers
  - (callSegmentationOutliers), 3
- FALSE, 13, 21, 22
- findLargeGaps, 6, 7, 13, 21
- gapsToSegments, 6, 13, 21
- gapsToSegments
  - (gapsToSegments.data.frame), 7
- gapsToSegments.data.frame, 7
- Inf, 13, 14, 17, 18, 20, 22
- integer, 3, 6, 13, 21
- list, 8, 10, 11
- logical, 4
- NA, 4, 14, 17, 18, 20–22
- NaN, 4, 14, 18, 22
- NonPairedPSCBS, 8, 11, 17
- normalizeTumorBoost, 22
- NULL, 4, 13, 21
- numeric, 3, 4, 6, 7, 13, 17, 20, 21
- PairedPSCBS, 9, 11, 12, 21
- pruneBySdUndo, 14
- PSCBS, 8, 9, 11
- PSCBS-package, 2
- segment, 13
- segmentByCBS, 2, 5, 6, 12, 18, 21, 22
- segmentByNonPairedPSCBS, 2, 6, 9, 14, 17, 22
- segmentByPairedPSCBS, 2, 6, 11, 14, 17, 18, 20
- smooth.CNA, 4
- TRUE, 7, 13, 21
- vector, 3, 4, 6, 13, 17, 20, 21
- Verbose, 4, 13, 17, 21