

# Package ‘breakpointR’

May 24, 2024

**Type** Package

**Title** Find breakpoints in Strand-seq data

**Version** 1.22.0

**Date** 2021-11-20

**Author** David Porubsky, Ashley Sanders, Aaron Taudt

**Maintainer** David Porubsky <david.porubsky@gmail.com>

**Description** This package implements functions for finding breakpoints, plotting and export of Strand-seq data.

**Depends** R (>= 3.5), GenomicRanges, cowplot, breakpointRdata

**Imports** methods, utils, grDevices, stats, S4Vectors, GenomeInfoDb (>= 1.12.3), IRanges, Rsamtools, GenomicAlignments, ggplot2, BiocGenerics, gtools, doParallel, foreach

**Suggests** knitr, BiocStyle, testthat

**License** file LICENSE

**LazyLoad** yes

**VignetteBuilder** knitr

**biocViews** Software, Sequencing, DNaseSeq, SingleCell, Coverage

**URL** <https://github.com/daewoooo/BreakPointR>

**RoxygenNote** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/breakpointR>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 9362506

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-24

## Contents

breakpointR-package	2
BreakPoint	3
breakpointr	3
breakpointr2UCSC	5
breakSeekr	6
collapseBins	7
confidenceInterval	8
confidenceInterval.binomial	9
createCompositeFile	10
deltaWCalculator	11
deltaWCalculatorVariousWindows	12
exportRegions	12
genotype.binom	13
genotype.fisher	14
genotyping	15
hotspotter	16
insertchr	17
loadFromFiles	18
plotBreakpoints	18
plotBreakpointsPerChr	19
plotHeatmap	20
ranges2UCSC	20
readBamFileAsGRanges	21
readConfig	22
removeDoubleSCEs	23
removeReadPileupSpikes	23
runBreakpointr	24
summarizeBreaks	26
synchronizeReadDir	27
transCoord	27
writeConfig	28
<b>Index</b>	<b>29</b>

---

breakpointR-package    *Breakpoint detection in Strand-Seq data*

---

### Description

This package implements functions for finding breakpoints, plotting and export of Strand-seq data.

### Details

The main function of this package is [breakpointr](#) and produces several plots and browser files. If you want to have more fine-grained control over the different steps check the vignette [How to use breakpointR](#).

**Author(s)**

David Porubsky, Ashley Sanders, Aaron Taudt

---

BreakPoint

*BreakPoint object*

---

**Description**

The BreakPoint object is output of the function `runBreakpointR` and is basically a list with various entries. The `class()` attribute of this list was set to "BreakPoint". Entries can be accessed with the list operators `'[[']` and `'$'`.

**Value**

fragments	A <a href="#">GRanges-class</a> object with read fragments.
deltas	A <a href="#">GRanges-class</a> object with deltaWs.
breaks	A <a href="#">GRanges-class</a> object containing the breakpoint coordinates.
counts	A <a href="#">GRanges-class</a> object with the regions between breakpoints.
params	A vector with parameters that were used to obtain the results.

**See Also**

`runBreakpointR`

---

breakpointR

*Main function for the [breakpointR](#) package*

---

**Description**

This function is an easy-to-use wrapper to find breakpoints with `runBreakpointR` in parallel, write the results to file, plot results and find hotspots.

**Usage**

```
breakpointR(  
  inputfolder,  
  outputfolder,  
  configfile = NULL,  
  numCPU = 1,  
  reuse.existing.files = FALSE,  
  windowsize = 1e+06,  
  binMethod = "size",  
  multi.sizes = NULL,  
  pairedEndReads = FALSE,
```

```

pair2frgm = FALSE,
chromosomes = NULL,
min.mapq = 10,
filtAlt = FALSE,
genoT = "fisher",
trim = 10,
peakTh = 0.33,
zlim = 3.291,
background = 0.05,
minReads = 10,
maskRegions = NULL,
callHotSpots = FALSE,
conf = 0.99
)

```

### Arguments

inputfolder	Folder with BAM files.
outputfolder	Folder to output the results. If it does not exist it will be created.
configfile	A file specifying the parameters of this function (without inputfolder, outputfolder and configfile). Having the parameters in a file can be handy if many samples with the same parameter settings are to be run. If a configfile is specified, it will take priority over the command line parameters.
numCPU	The numbers of CPUs that are used. Should not be more than available on your machine.
reuse.existing.files	A logical indicating whether or not existing files in outputfolder should be reused.
windowSize	The window size used to calculate deltaWs, either number of reads or genomic size depending on binMethod.
binMethod	Method used to calculate optimal number of reads in the window ("size", "reads"). By default binMethod='size'.
multi.sizes	User defined multiplications of the original window size.
pairedEndReads	Set to TRUE if you have paired-end reads in your file.
pair2frgm	Set to TRUE if every paired-end read should be merged into a single fragment.
chromosomes	If only a subset of the chromosomes should be binned, specify them here.
min.mapq	Minimum mapping quality when importing from BAM files.
filtAlt	Set to TRUE if you want to filter out alternative alignments defined in 'XA' tag.
genoT	A method ('fisher' or 'binom') to genotype regions defined by a set of breakpoints.
trim	The amount of outliers in deltaWs removed to calculate the stdev (10 will remove top 10% and bottom 10% of deltaWs).
peakTh	The threshold that the peak deltaWs must pass to be considered a breakpoint (e.g. 0.33 is 1/3 of max(deltaW)).

zlim	The number of stdev that the deltaW must pass the peakTh (ensures only significantly higher peaks are considered).
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
minReads	The minimal number of reads between two breaks required for genotyping.
maskRegions	List of regions to be excluded from the analysis (tab-separated file: chromosomes start end).
callHotSpots	Search for regions of high abundance of breakpoints in single cells.
conf	Desired confidence interval of localized breakpoints.

**Value**

NULL

**Author(s)**

David Porubsky, Aaron Taudt, Ashley Sanders

**Examples**

```
## Not run:
## The following call produces plots and genome browser files for all BAM files in "my-data-folder"
breakpointr(inputfolder="my-data-folder", outputfolder="my-output-folder")
## End(Not run)
```

---

breakpointr2UCSC	<i>Export UCSC browser formatted files</i>
------------------	--

---

**Description**

Write a bedfile or bedgraph from a breakpointR object for upload on to the UCSC Genome browser.

**Usage**

```
breakpointr2UCSC(
  index,
  outputDirectory,
  fragments = NULL,
  deltaWs = NULL,
  breakTrack = NULL,
  confidenceIntervals = NULL,
  breaksGraph = NULL
)
```

**Arguments**

index	A character used to name the bedfile(s).
outputDirectory	Location to write bedfile(s).
fragments	A <a href="#">GRanges-class</a> object with strand and mapq metadata, such as that generated by <a href="#">readBamFileAsGRanges</a>
deltaWs	A <a href="#">GRanges-class</a> object with metadata column "deltaW" generated by <a href="#">deltaWCalculator</a> .
breakTrack	A <a href="#">GRanges-class</a> object with metadata "genoT" (e.g. newBreaks) will write a bedtrack with refined breakpoints.
confidenceIntervals	A <a href="#">GRanges-class</a> object with metadata "genoT" the same length as breakTrack (e.g. confint) will write a bedtrack with breakpoints confidence intervals.
breaksGraph	A <a href="#">GRanges-class</a> object.

**Value**

NULL

**Author(s)**

Ashley Sanders, David Porubsky, Aaron Taudt

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
brkpts <- get(load(exampleFile))
## Write results to BED files
breakpointr2UCSC(index='testfile', outputDirectory=tempdir(), breakTrack=brkpts$breaks)
```

---

breakSeekr

*Find breakpoints from deltaWs*

---

**Description**

Find breakpoints from deltaWs by localizing significant peaks based on z-score calculation.

**Usage**

```
breakSeekr(deltaWs, trim = 10, peakTh = 0.33, zlim = 3.291)
```

**Arguments**

deltaWs	A <a href="#">GRanges-class</a> object with metadata column "deltaW" generated by <a href="#">deltaWCalculator</a> .
trim	The amount of outliers in deltaWs removed to calculate the stdev (10 will remove top 10% and bottom 10% of deltaWs).
peakTh	The threshold that the peak deltaWs must pass to be considered a breakpoint (e.g. 0.33 is 1/3 of max(deltaW)).
zlim	The number of stdev that the deltaW must pass the peakTh (ensures only significantly higher peaks are considered).

**Value**

A [GRanges-class](#) object containing breakpoint coordinates with various metadata columns.

**Author(s)**

David Porubsky, Aaron Taudt, Ashley Sanders

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_bams", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
fragments <- readBamFileAsGRanges(exampleFile, pairedEndReads=FALSE, chromosomes='chr22')
## Calculate deltaW values
dw <- deltaWCalculator(fragments)
## Get significant peaks in deltaW values
breaks <- breakSeekr(dw)
```

---

collapseBins

*Collapse consecutive bins with the same ID value*


---

**Description**

Collapse consecutive bins with the same value defined in 'id.field'.

**Usage**

```
collapseBins(gr, id.field = 3)
```

**Arguments**

gr	A <a href="#">GRanges-class</a> object.
id.field	A number of metadata column to use for region merging.

**Value**

A [GRanges-class](#) object.

---

confidenceInterval     *Estimate confidence intervals for breakpoints*

---

### Description

Estimate confidence intervals for breakpoints by going outwards from the breakpoint read by read, and multiplying the probability that the read doesn't belong to the assigned segment.

### Usage

```
confidenceInterval(breaks, fragments, background = 0.05, conf = 0.99)
```

### Arguments

breaks	Genotyped breakpoints as outputted from function <a href="#">GenotypeBreaks</a> .
fragments	Read fragments from function <a href="#">readBamFileAsGRanges</a> .
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
conf	Desired confidence interval of localized breakpoints.

### Value

A [GRanges-class](#) object of breakpoint ranges for a given confidence interval in conf.

### Author(s)

Aaron Taudt, David Porubsky

### Examples

```
## Not run:
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
breakpoint.objects <- get(load(exampleFile))
## Calculate confidence intervals of genotyped breakpoints
confint <- confidenceInterval(breaks=breakpoint.objects$breaks, fragments=breakpoint.objects$fragments, background=0.05, conf=0.99)
## End(Not run)
```



---

`confidenceInterval.binomial`*Estimate confidence intervals for breakpoints*

---

**Description**

Estimate confidence intervals for breakpoints by going outwards from the breakpoint read by read, and performing a binomial test of getting the observed or a more extreme outcome, given that the reads within the confidence interval belong to the other side of the breakpoint.

**Usage**

```
confidenceInterval.binomial(breaks, fragments, background = 0.02, conf = 0.99)
```

**Arguments**

<code>breaks</code>	Genotyped breakpoints as outputted from function <a href="#">GenotypeBreaks</a> .
<code>fragments</code>	Read fragments from function <a href="#">readBamFileAsGRanges</a> .
<code>background</code>	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
<code>conf</code>	Desired confidence interval of localized breakpoints.

**Value**

A [GRanges-class](#) object of breakpoint ranges for a given confidence interval in `conf`.

**Author(s)**

Aaron Taudt, David Porubsky

**Examples**

```
## Not run:
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
breakpoint.objects <- get(load(exampleFile))
## Calculate confidence intervals of genotyped breakpoints
confint <- confidenceInterval.binomial(breakpoint.objects$breaks, breakpoint.objects$fragments, background=0.02)
## End(Not run)
```

---

createCompositeFile    *Create composite Strand-seq file*

---

### Description

This function will move through BAM files in a folder, read in each individual file and go through each chromosome, determine if the chromosome is WW or CC based on WCcutoff, reverse complement all reads in the WW file, append to a new composite file for that chromosome, order the composite file of each chromosome based on position.

### Usage

```
createCompositeFile(  
  file.list,  
  chromosomes = NULL,  
  pairedEndReads = TRUE,  
  pair2frgm = FALSE,  
  min.mapq = 10,  
  filtAlt = FALSE,  
  WC.cutoff = 0.9,  
  genoT = "fisher",  
  background = 0.05  
)
```

### Arguments

file.list	A list of BAM files to process.
chromosomes	If only a subset of the chromosomes should be binned, specify them here.
pairedEndReads	Set to TRUE if you have paired-end reads in your file.
pair2frgm	Set to TRUE if every paired-end read should be merged into a single fragment.
min.mapq	Minimum mapping quality when importing from BAM files.
filtAlt	Set to TRUE if you want to filter out alternative alignments defined in 'XA' tag.
WC.cutoff	Percentage of WW or CC reads to consider chromosome being WW or CC
genoT	A method ('fisher' or 'binom') to genotype regions defined by a set of break-points.
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.

### Value

A [GRanges-class](#) object.

### Author(s)

Ashley Sanders, David Porubsky

---

deltaWCalculator	<i>Calculate deltaWs</i>
------------------	--------------------------

---

## Description

This function will calculate deltaWs from a [GRanges-class](#) object with read fragments.

## Usage

```
deltaWCalculator(frags, reads.per.window = 100)
```

## Arguments

`frags`            A [GRanges-class](#) with read fragments (see [readBamFileAsGRanges](#)).

`reads.per.window`  
                  Number of reads in each dynamic window.

## Value

The input frags with additional meta-data columns.

## Author(s)

Aaron Taudt

## See Also

[readBamFileAsGRanges](#)

## Examples

```
## Get an example file
exampleFolder <- system.file("extdata", "example_bams", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
fragments <- readBamFileAsGRanges(exampleFile, pairedEndReads=FALSE, chromosomes='chr22')
## Calculate deltaW values
dw <- deltaWCalculator(fragments)
```

---

`deltaWCalculatorVariousWindows`*Calculate deltaWs using various window sizes*

---

**Description**

This function will calculate deltaWs from a [GRanges-class](#) object with read fragments.

**Usage**

```
deltaWCalculatorVariousWindows(  
  frags,  
  reads.per.window = 100,  
  multi.sizes = c(2, 4, 6)  
)
```

**Arguments**

`frags`            A [GRanges-class](#) with read fragments (see [readBamFileAsGRanges](#)).

`reads.per.window`    Number of reads in each dynamic window.

`multi.sizes`        User defined multiplications of the original window size.

**Value**

The input frags with additional meta-data columns.

**Author(s)**

David Porubsky

**See Also**

`deltaWCalculator`

---

`exportRegions`*Function to print WC regions after breakpointR analysis*

---

**Description**

Function to print WC regions after breakpointR analysis

**Usage**

```
exportRegions(
  datapath,
  file = NULL,
  collapseInversions = FALSE,
  collapseRegionSize = 5e+06,
  minRegionSize = 5e+06,
  state = "wc"
)
```

**Arguments**

datapath	A path to that
file	A filename to print exported regions to.
collapseInversions	Set to TRUE if you want to collapse putative inverted regions.
collapseRegionSize	Upper range of what sized regions should be collapsed.
minRegionSize	Minimal size of the region to be reported.
state	A genotype of the regions to be exported ('ww', 'cc' or 'wc').

**Value**

A data.frame object containing all regions with user defined 'state'.

**Author(s)**

David Porubsky

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
## To export regions genotyped as 'wc'
wc.regions <- exportRegions(datapath=exampleFolder, collapseInversions=FALSE, minRegionSize=5000000, state='wc')
```

---

genotype.binom	<i>Assign states to any given region using binomial test.</i>
----------------	---

---

**Description**

Assign states to any given region using binomial test.

**Usage**

```
genotype.binom(wReads, cReads, background = 0.05, minReads = 10, log = FALSE)
```

**Arguments**

wReads	Number of Watson reads.
cReads	Number of Crick reads.
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
minReads	The minimal number of reads between two breaks required for genotyping.
log	Set to TRUE if you want to calculate probability in log space.

**Value**

A list with the \$bestFit and \$pval.

**Author(s)**

David Porubsky

**Examples**

```
## Get Crick and Watson read counts
## Crick read count
cReads <- 30
## Watson read count
wReads <- 5
genotype.binom(cReads = cReads, wReads = wReads, background = 0.05, minReads = 10, log = TRUE)
```

---

genotype.fisher      *Assign states to any given region using Fisher Exact Test.*

---

**Description**

Assign states to any given region using Fisher Exact Test.

**Usage**

```
genotype.fisher(cReads, wReads, roiReads, background = 0.05, minReads = 10)
```

**Arguments**

cReads	Number of Crick reads.
wReads	Number of Watson reads.
roiReads	Total number of Crick and Watson reads.
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
minReads	The minimal number of reads between two breaks required for genotyping.

**Value**

A list with the \$bestFit and \$pval.

**Author(s)**

David Porubsky, Aaron Taudt

**Examples**

```
## Get Crick and Watson read counts
## Crick read count
cReads <- 30
## Watson read count
wReads <- 5
genotype.fisher(cReads = cReads, wReads = wReads, roiReads = cReads + wReads, background = 0.05, minReads = 10)
```

---

genotyping

*Set of functions to genotype regions in between localized breakpoints*


---

**Description**

Each defined region is given one of the three states ('ww', 'cc' or 'wc') Consecutive regions with the same state are collapsed

**Usage**

```
GenotypeBreaks(
  breaks,
  fragments,
  background = 0.05,
  minReads = 10,
  genoT = "fisher"
)
```

**Arguments**

breaks	A <a href="#">GRanges-class</a> object with breakpoint coordinates.
fragments	A <a href="#">GRanges-class</a> object with read fragments.
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
minReads	The minimal number of reads between two breaks required for genotyping.
genoT	A method ('fisher' or 'binom') to genotype regions defined by a set of breakpoints.

## Details

Function `GenotypeBreaks` exports states of each region defined by breakpoints. Function `genotype.fisher` assigns states to each region based on expected counts of Watson and Crick reads. Function `genotype.binom` assigns states to each region based on expected counts of Watson and Crick reads.

## Value

A `GRanges-class` object with genotyped breakpoint coordinates.

## Functions

- `GenotypeBreaks()`: Genotypes breakpoint defined regions.

## Author(s)

David Porubsky, Ashley Sanders, Aaron Taudt

## Examples

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
breakpoint.objects <- get(load(exampleFile))
## Genotype regions between breakpoints
gbreaks <- GenotypeBreaks(breaks=breakpoint.objects$breaks, fragments=breakpoint.objects$fragments)
```

---

hotspotter

*Find hotspots of genomic events*

---

## Description

Find hotspots of genomic events by using kernel [density](#) estimation.

## Usage

```
hotspotter(gr.list, bw, pval = 1e-08)
```

## Arguments

<code>gr.list</code>	A list or <code>GRangesList-class</code> with <code>GRanges-class</code> object containing the coordinates of the genomic events.
<code>bw</code>	Bandwidth used for kernel density estimation (see <a href="#">density</a> ).
<code>pval</code>	P-value cutoff for hotspots.



## Details

The hotspotter uses [density](#) to perform a KDE. A p-value is calculated by comparing the density profile of the genomic events with the density profile of a randomly subsampled set of genomic events. Due to this random sampling, the result can vary for each function call, most likely for hotspots whose p-value is close to the specified pval.

## Value

A [GRanges-class](#) object containing coordinates of hotspots with p-values.

## Author(s)

Aaron Taudt

## Examples

```
## Get example BreakPoint objects
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFiles <- list.files(exampleFolder, full.names=TRUE)
breakpoint.objects <- loadFromFiles(exampleFiles)
## Extract breakpoint coordinates
breaks <- lapply(breakpoint.objects, '[[', 'breaks')
## Get hotspot coordinates
hotspots <- hotspotter(gr.list=breaks, bw=1e6)
```

---

insertchr

*Insert chromosome for in case it's missing*

---

## Description

Add two columns with transformed genomic coordinates to the [GRanges-class](#) object. This is useful for making genomewide plots.

## Usage

```
insertchr(gr)
```

## Arguments

`gr` A [GRanges-class](#) object.

## Value

The input [GRanges-class](#) object with an additional metadata column containing chromosome name with 'chr'.

---

loadFromFiles	<i>Load <b>breakpointR</b> objects from file</i>
---------------	--

---

**Description**

Wrapper to load **breakpointR** objects from file and check the class of the loaded objects.

**Usage**

```
loadFromFiles(files, check.class = c("GRanges", "BreakPoint"))
```

**Arguments**

files	A list of <b>GRanges-class</b> or <b>BreakPoint</b> objects or a vector of files that contain such objects.
check.class	Any combination of c('GRanges', 'BreakPoint'). If any of the loaded objects does not belong to the specified class, an error is thrown.

**Value**

A list of **GRanges-class** or **BreakPoint** objects.

**Examples**

```
## Get some files that you want to load
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFiles <- list.files(exampleFolder, full.names=TRUE)
## Load the processed data
breakpoint.objects <- loadFromFiles(exampleFiles)
```

---

plotBreakpoints	<i>Plotting genome-wide ideograms <b>breakpointR</b></i>
-----------------	--

---

**Description**

This function will create genome-wide ideograms from a **BreakPoint** object.

**Usage**

```
plotBreakpoints(files2plot, file = NULL)
```

**Arguments**

files2plot	A list of files that contains <b>BreakPoint</b> objects or a single <b>BreakPoint</b> object.
file	Name of the file to plot to.

**Value**

A list with [ggplot](#) objects.

**Author(s)**

David Porubsky, Aaron Taudt, Ashley Sanders

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Plot the file
plotBreakpoints(files2plot=exampleFile)
```

---

plotBreakpointsPerChr *Plotting chromosome specific ideograms* [breakpointR](#)

---

**Description**

This function will create chromosome specific enome-wide ideograms from a [BreakPoint](#) object.

**Usage**

```
plotBreakpointsPerChr(files2plot, plotspath = NULL, chromosomes = NULL)
```

**Arguments**

files2plot	A list of files that contains <a href="#">BreakPoint</a> objects or a single <a href="#">BreakPoint</a> object.
plotspath	Directory to store plots.
chromosomes	Set specific chromosome(s) to be plotted.

**Value**

A list with [ggplot](#) objects.

**Author(s)**

David Porubsky

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFiles <- list.files(exampleFolder, full.names=TRUE)
## Plot results
plotBreakpointsPerChr(exampleFiles, chromosomes='chr7')
```

---

plotHeatmap	<i>Genome wide heatmap of template inheritance states</i>
-------------	---

---

**Description**

Plot a genome-wide heatmap of template inheritance states from a [BreakPoint](#) object.

**Usage**

```
plotHeatmap(files2plot, file = NULL, hotspots = NULL)
```

**Arguments**

files2plot	A list of files that contains <a href="#">BreakPoint</a> objects or a single <a href="#">BreakPoint</a> object.
file	Name of the file to plot to.
hotspots	A <a href="#">GRanges-class</a> object with locations of breakpoint hotspots.

**Value**

A [ggplot](#) object.

**Author(s)**

David Porubsky, Aaron Taudt, Ashley Sanders

**Examples**

```
## Get example BreakPoint objects to plot
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFiles <- list.files(exampleFolder, full.names=TRUE)
breakpoint.objects <- loadFromFiles(exampleFiles)
## Plot the heatmap
plotHeatmap(breakpoint.objects)
```

---

ranges2UCSC	<i>Generates a bedfile from an input GRanges file</i>
-------------	---

---

**Description**

Write a bedfile from Breakpoint.R files for upload on to UCSC Genome browser

**Usage**

```
ranges2UCSC(gr, outputDirectory = ".", index = "bedFile", colorRGB = "0,0,0")
```

**Arguments**

gr	A <a href="#">GRanges-class</a> object with genomic ranges to be exported into UCSC format.
outputDirectory	Location to write bedfile(s).
index	A character used to name the bedfile(s).
colorRGB	An RGB color to be used for submitted ranges.

**Value**

NULL

**Author(s)**

David Porubsky

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
counts <- get(load(exampleFile))[['counts']]
## Export 'wc' states into a UCSC formatted file
ranges2UCSC(gr=counts[counts$states == 'wc'], index='testfile', outputDirectory=tempdir())
```

---

readBamFileAsGRanges *Import BAM file into GRanges*

---

**Description**

Import aligned reads from a BAM file into a [GRanges-class](#) object.

**Usage**

```
readBamFileAsGRanges(
  file,
  bamindex = file,
  chromosomes = NULL,
  pairedEndReads = FALSE,
  min.mapq = 10,
  remove.duplicate.reads = TRUE,
  pair2frgm = FALSE,
  filtAlt = FALSE
)
```

**Arguments**

file	Bamfile with aligned reads.
bamindex	Bam-index file with or without the .bai ending. If this file does not exist it will be created and a warning is issued.
chromosomes	If only a subset of the chromosomes should be binned, specify them here.
pairedEndReads	Set to TRUE if you have paired-end reads in your file.
min.mapq	Minimum mapping quality when importing from BAM files.
remove.duplicate.reads	A logical indicating whether or not duplicate reads should be kept.
pair2frgm	Set to TRUE if every paired-end read should be merged into a single fragment.
filtAlt	Set to TRUE if you want to filter out alternative alignments defined in 'XA' tag.

**Value**

A `GRanges-class` object.

**Author(s)**

David Porubsky, Aaron Taudt, Ashley Sanders

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_bams", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Load the file
fragments <- readBamFileAsGRanges(exampleFile, pairedEndReads=FALSE, chromosomes='chr22')
```

---

readConfig

*Read breakpointR configuration file*

---

**Description**

Read an breakpointR configuration file into a list structure. The configuration file has to be specified in INI format. R expressions can be used and will be evaluated.

**Usage**

```
readConfig(configfile)
```

**Arguments**

configfile	Path to the configuration file
------------	--------------------------------

**Value**

A list with one entry for each element in configfile.

**Author(s)**

Aaron Taudt

---

removeDoubleSCEs	<i>Process double SCE chromosomes: with internal WC region.</i>
------------------	---

---

**Description**

This function will take from a double SCE chromosome only WW or CC region (Longer region is taken).

**Usage**

```
removeDoubleSCEs(gr, collapseWidth = 5e+06)
```

**Arguments**

`gr` A [GRanges-class](#) object.  
`collapseWidth` A segment size to be collapsed with neighbouring segments.

**Value**

The input [GRanges-class](#) object with only WW or CC region retained.

---

removeReadPileupSpikes	<i>Remove large spikes in short reads coverage</i>
------------------------	--

---

**Description**

This function takes a [GRanges-class](#) object of aligned short reads and removes pockets of reads that are stacked on top of each other based on the maximum number of reads allowed to pileup in 'max.pileup' parameter.

**Usage**

```
removeReadPileupSpikes(gr = NULL, max.pileup = 30)
```

**Arguments**

`gr` A [GRanges-class](#) object.  
`max.pileup` A maximum number of reads overlapping each other to be kept.

**Value**

A [GRanges-class](#) object.

**Author(s)**

David Porubsky

**Examples**

```
## Get some files that you want to load
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
infile <- list.files(exampleFolder, full.names=TRUE)[1]
## Read in the reads
breakP.obj <- get(load(infile))
frags <- breakP.obj$fragments
## Remove read spikes
frags <- removeReadPileupSpikes(gr=frags)
```

---

`runBreakpointR`*Find breakpoints in Strand-seq data*

---

**Description**

Find breakpoints in Strand-seq data. See section Details on how breakpoints are located.

**Usage**

```
runBreakpointR(
  bamfile,
  ID = basename(bamfile),
  pairedEndReads = TRUE,
  chromosomes = NULL,
  windowSize = 1e+06,
  binMethod = "size",
  multi.sizes = NULL,
  trim = 10,
  peakTh = 0.33,
  zlim = 3.291,
  background = 0.05,
  min.mapq = 10,
  pair2frgm = FALSE,
  filtAlt = FALSE,
  genoT = "fisher",
  minReads = 20,
  maskRegions = NULL,
  conf = 0.99
)
```



**Arguments**

bamfile	A file with aligned reads in BAM format.
ID	A character string that will serve as identifier in downstream functions.
pairedEndReads	Set to TRUE if you have paired-end reads in your file.
chromosomes	If only a subset of the chromosomes should be binned, specify them here.
windowSize	The window size used to calculate deltaWs, either number of reads or genomic size depending on binMethod.
binMethod	Method used to calculate optimal number of reads in the window ("size", "reads"). By default binMethod='size'.
multi.sizes	User defined multiplications of the original window size.
trim	The amount of outliers in deltaWs removed to calculate the stdev (10 will remove top 10% and bottom 10% of deltaWs).
peakTh	The threshold that the peak deltaWs must pass to be considered a breakpoint (e.g. 0.33 is 1/3 of max(deltaW)).
zlim	The number of stdev that the deltaW must pass the peakTh (ensures only significantly higher peaks are considered).
background	The percent (e.g. 0.05 = 5%) of background reads allowed for WW or CC genotype calls.
min.mapq	Minimum mapping quality when importing from BAM files.
pair2frgm	Set to TRUE if every paired-end read should be merged into a single fragment.
filtAlt	Set to TRUE if you want to filter out alternative alignments defined in 'XA' tag.
genoT	A method ('fisher' or 'binom') to genotype regions defined by a set of breakpoints.
minReads	The minimal number of reads between two breaks required for genotyping.
maskRegions	List of regions to be excluded from the analysis in <a href="#">GRanges-class</a> object.
conf	Desired confidence interval of localized breakpoints.

**Details**

Breakpoints are located in the following way:

1. calculate deltaWs chromosome-by-chromosome
2. localize breaks that pass zlim above the threshold
3. genotype both sides of breaks to confirm whether strand state changes
4. write a file of `_reads`, `_deltaWs` and `_breaks` in a chr fold -> can upload on to UCSC Genome browser
5. write a file for each index with all chromosomes included -> can upload on to UCSC Genome browser

**Value**

A [BreakPoint](#) object.

**Author(s)**

David Porubsky, Ashley Sanders, Aaron Taudt

**Examples**

```
## Get an example file
exampleFolder <- system.file("extdata", "example_bams", package="breakpointRdata")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
## Run breakpointR
brkpts <- runBreakpointR(exampleFile, chromosomes='chr22', pairedEndReads=FALSE)
```

---

summarizeBreaks	<i>Compile breakpoint summary table</i>
-----------------	---

---

**Description**

This function will calculate deltaWs from a [GRanges-class](#) object with read fragments.

**Usage**

```
summarizeBreaks(breakpoints)
```

**Arguments**

breakpoints     A list containing breakpoints stored in [GRanges-class](#) object.

**Value**

A data.frame of compiled breakpoints together with confidence intervals.

**Author(s)**

David Porubsky

**Examples**

```
## Get some files that you want to load
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
file <- list.files(exampleFolder, full.names=TRUE)[1]
breakpoints <- get(load(file))[c('breaks', 'confint')]
summarizeBreaks(breakpoints)
```

---

synchronizeReadDir      *Synchronize Strand-seq read directionality*

---

**Description**

This function aims to synchronize strand directionality of reads that fall into WW and CC regions.

**Usage**

```
synchronizeReadDir(files2sync, collapseWidth = 5e+06)
```

**Arguments**

files2sync      A list of files that contains [BreakPoint](#) objects.  
collapseWidth      A segment size to be collapsed with neighbouring segments.

**Value**

A [GRanges-class](#) object that reads synchronized by directionality.

**Author(s)**

David Porubsky

**Examples**

```
## Get some files that you want to load
exampleFolder <- system.file("extdata", "example_results", package="breakpointRdata")
files2sync <- list.files(exampleFolder, full.names=TRUE)[1]
synchronizeReadDir(files2sync=files2sync)
```

---

transCoord      *Transform genomic coordinates*

---

**Description**

Add two columns with transformed genomic coordinates to the [GRanges-class](#) object. This is useful for making genomewide plots.

**Usage**

```
transCoord(gr)
```

**Arguments**

gr      A [GRanges-class](#) object.

**Value**

The input `GRanges-class` with two additional metadata columns 'start.genome' and 'end.genome'.

---

writeConfig	<i>Write breakpointR configuration file</i>
-------------	---

---

**Description**

Write an breakpointR configuration file from a list structure.

**Usage**

```
writeConfig(config, configfile)
```

**Arguments**

config	A list structure with parameter values. Each entry will be written in one line.
configfile	Filename of the outputfile.

**Value**

NULL

**Author(s)**

Aaron Taudt

# Index

BreakPoint, [3](#), [18–20](#), [25](#), [27](#)  
breakpointR, [3](#), [18](#), [19](#)  
breakpointR (breakpointR-package), [2](#)  
breakpointtr, [2](#), [3](#)  
breakpointR-package, [2](#)  
breakpointtr2UCSC, [5](#)  
breakSeekr, [6](#)

collapseBins, [7](#)  
confidenceInterval, [8](#)  
confidenceInterval.binomial, [9](#)  
createCompositeFile, [10](#)

deltaWCalculator, [6](#), [7](#), [11](#)  
deltaWCalculatorVariousWindows, [12](#)  
density, [16](#), [17](#)

exportRegions, [12](#)

genotype.binom, [13](#)  
genotype.fisher, [14](#)  
GenotypeBreaks, [8](#), [9](#)  
GenotypeBreaks (genotyping), [15](#)  
genotyping, [15](#)  
ggplot, [19](#), [20](#)

hotspotter, [16](#)

insertchr, [17](#)

loadFromFiles, [18](#)

plotBreakpoints, [18](#)  
plotBreakpointsPerChr, [19](#)  
plotHeatmap, [20](#)

ranges2UCSC, [20](#)  
readBamFileAsGRanges, [6](#), [8](#), [9](#), [11](#), [12](#), [21](#)  
readConfig, [22](#)  
removeDoubleSCEs, [23](#)  
removeReadPileupSpikes, [23](#)

runBreakpointtr, [3](#), [24](#)  
summarizeBreaks, [26](#)  
synchronizeReadDir, [27](#)  
transCoord, [27](#)  
writeConfig, [28](#)