Package ‘strandCheckR’

April 3, 2026

Type Package

Title Calculate strandness information of a bam file

Version 1.28.0

Maintainer Thu-Hien To <tothuhien@gmail.com>

Description This package aims to quantify and remove putative double strand DNA from a strand-specific RNA sample. There are also options and methods to plot the positive/negative proportions of all sliding windows, which allow users to have an idea of how much the sample was contaminated and the appropriate threshold to be used for filtering.

URL https://github.com/UofABioinformaticsHub/strandCheckR

BugReports https://github.com/UofABioinformaticsHub/strandCheckR/issues

License GPL (>= 2)

LazyData TRUE

Encoding UTF-8

Depends ggplot2 (>= 4.0.0), Rsamtools, S4Vectors

Imports BiocGenerics, dplyr, Seqinfo, GenomicAlignments, GenomicRanges, gridExtra, IRanges, grid, methods, reshape2, rlang, stats, stringr, TxDb.Hsapiens.UCSC.hg38.knownGene, tidyselect

biocViews RNASeq, Alignment, QualityControl, Coverage, ImmunoOncology

RoxygenNote 7.3.3

VignetteBuilder knitr

Suggests BiocStyle, knitr, magrittr, rmarkdown, testthat

git_url https://git.bioconductor.org/packages/strandCheckR

git_branch RELEASE_3_22

git_last_commit 01a9239

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-04-02

Author Thu-Hien To [aut, cre],
Stevie Pederson [aut] (ORCID: <https://orcid.org/0000-0001-8197-3303>)

Contents

`strandCheckR-package` *strandCheckR: Check strandedness of alignments in one or more bam files*

Description

- This package contains functions for checking the strandedness of alignments such as [getStrandFromBamFile()] - The results can easily be visualised using [plotHist()] - Any suspected genomic DNA can also be removed using [filterDNA()], with thresholds for filtering able to be determined manually using [plotWin()]

Author(s)

Thu-Hien To

See Also

Useful links:

- `https://github.com/UofABioinformaticsHub/strandCheckR`
- Report bugs at `https://github.com/UofABioinformaticsHub/strandCheckR/issues`

`.calculateStrandCoverage` *Calculate the strand information based on coverage*

Description

Calculate the coverage coming from '+'/'-' reads in all sliding windows

Usage

```
.calculateStrandCoverage(
  winPosAlignments,
  winNegAlignments,
  winWidth = 1000L,
  winStep = 100L
)
```

Arguments

`winPosAlignments`	a list that has a 'Coverage' field containing coverage coming from positive reads
`winNegAlignments`	a list that has a 'Coverage' field containing coverage coming from negative reads
`winWidth`	the length of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.

Value

a list of two vectors, containing a positive/negative coverage of the input positive/negative windows

---

`.calculateStrandNbReads` *Calculate the strand information based on the number of reads*

Description

Calculate the number of reads coming from '+'/'-' strands in all sliding wndows

Usage

```
.calculateStrandNbReads(winPosAlignments, winNegAlignments)
```

Arguments

`winPosAlignments`	a list that has a 'Win' field that contains information of sliding windows overalapping positive reads
`winNegAlignments`	a a list that has a 'Win' field that contains information of sliding windows overalapping negative reads

Value

a list of two vectors, containing a positive/negative number of reads of the input positive/negative windows

---

`.concatenateAlignments` *Concatenate a list of Alignments into One*

Description

Concatenate a list of Alignments from multiple sequences into a single object

Usage

```
.concatenateAlignments(readInfo, seqInfo)
```

Arguments

`readInfo`	a list returned by `scanBam` function, each element correspond to a sequence, containing the information of strand, starting position, cigar string, and eventually flag, qname
`seqInfo`	a data frame that contains some key information of the sequences

Details

This method take a list of alignments across one or more sequences as output by `scanBam` and concatenates them into a single set of alignments which may include multiple sequences

Value

the concatenated alignments of the input list

`.getWinInSequence` *Get window data frame with the correct sequence name and position*

Description

Get the correct sequence name and position for each window

Usage

```
.getWinInSequence(Win, seqInfo, winWidth = 1000L, winStep = 100L)
```

Arguments

`Win`	a data frame contains the strand information of every window
`seqInfo`	a data frame that contains some key information of the sequences
`winWidth`	the length of sliding window
`winStep`	the step length to sliding the window

Value

A subset of the input object

`.keptProbaWin` *Get the probability of being kept for each window*

Description

Calculate the keeping probability of each window based on its positive/negative proportion

Usage

```
.keptProbaWin(
  winPosAlignments,
  winNegAlignments,
  winWidth,
  winStep,
  threshold,
  pvalueThreshold,
  errorRate,
  mustKeepWin,
  minCov,
  maxCov,
  getWin,
  useCoverage = FALSE
)
```

Arguments

`winPosAlignments`	an object returned by getWinOverlapEachReadFragment for positive reads
`winNegAlignments`	an object returned by getWinOverlapEachReadFragment for negative reads
`winWidth`	the width of the sliding window, 1000 by default.
`winStep`	the winStep length to sliding the window, 100 by default.
`threshold`	the strand proportion threshold to test whether to keep a window or not.
`pvalueThreshold`	threshold of p-value
`errorRate`	the probability that an RNA read takes the false strand. 0.01 by default
`mustKeepWin`	the windows that must be kept regardless their strand proportion
`minCov`	In the case that `useCoverage=FALSE`, if a window has less than `minCov` reads, then it will be rejected regardless of the strand proportion. For the case that `useCoverage=TRUE`, if a window has max coverage least than `minCov`, then it will be rejected. 0 by default
`maxCov`	In the case that `useCoverage=FALSE`, if a window has more than `maxCov` reads, then it will be kept regardless of the strand proportion. For the case that `useCoverage=TRUE`, if a window has max coverage more than `maxCov`, then it will be kept. If 0 then it doesn't have effect on selecting window. 0 by default.

`getWin`	if TRUE, the function will return a data frame containing the information of all windows. It's FALSE by default.
`useCoverage`	if TRUE, then the strand information in each window corresponds to the sum of coverage coming from positive/negative reads; and not the number of positive/negative reads as default.

Value

A list of 2 numeric-Rle objects containing keeping probability of each +/- alignments. If `getWin=TRUE` then the list contains an additional DataFrame for the number of reads and coverage of the input window +/- alignments

---

`.keptReadFragment` *Calculate the read fragments to be kept*

Description

Calculate the keeping probability of each read fragment based on the keeping probability of the windows containing it. Then get the list of read fragments to be kept.

Usage

```
.keptReadFragment(fragments, keptProbaW)
```

Arguments

`fragments`	an IRanges object defining the starting, ending position of each fragment
`keptProbaW`	an Rle object define the kept probability of each sliding window

Value

an integer vector of read fragment indices to be kept

`.sequenceInfoInPartition` *Calculate the first/last base/read of each sequence within each part of the partition.*

Description

Calculate the first/last base/read of each sequence within each part of the partition.

Usage

```
.sequenceInfoInPartition(seqInfo, winWidth, winStep)
```

Arguments

`seqInfo`	a data frame that contains some key information of the sequences
`winWidth`	the length of sliding window
`winStep`	the step length to sliding the window

Value

Reduced sequence information (data.frame)

`.summarizeHist` *Summarize the histogram of strand proportions from the input windows data frame*

Description

Summarize the histogram of positive proportions from the input windows obtained from the function `getStrandFromBamFile`

Usage

```
.summarizeHist(
  windows,
  split = c(10L, 100L, 1000L),
  breaks = 100L,
  useCoverage = FALSE,
  groupBy = NULL,
  normalizeBy = NULL
)
```

Arguments

`windows`	data frame containing the strand information of the sliding windows. Windows can be obtained using the function `getStrandFromBamFile`.
`split`	an integer vector that specifies how you want to partition the windows based on the coverage. By default `split` = c(10,100,1000), which means that your windows will be partitionned into 4 groups, those have coverage < 10, from 10 to 100, from 100 to 1000, and > 1000
`breaks`	an integer giving the number of bins for the histogram
`useCoverage`	if TRUE then plot the coverage strand information, otherwise plot the number of reads strand information. FALSE by default
`groupBy`	the column names of windows that will be used to group the data
`normalizeBy`	instead of using the raw read count/coverage, we will normalize it to a proportion by dividing it to the total number of read count/coverage of windows that have the same value in the `normalizeBy` columns.

Value

a dataframe object

See Also

getStrandFromBamFile, plotHist, plotWin

`checkPairedEnd` *Test whether a bam file is single-end or paired-end*

Description

Check the first 100000 first reads of the bam file to see whether it is single-end or paired-end

Usage

```
checkPairedEnd(file, yieldSize = 1e+05)
```

Arguments

`file`	the input bam file. Your bamfile should be sorted and have an index file located at the same path as well.
`yieldSize`	the number of reads to be checked, 100000 by default.

Value

return TRUE if the input file is paired end, and FALSE if it is single end

Examples

```
file <- system.file('extdata','s1.sorted.bam',package = 'strandCheckR')
checkPairedEnd(file)
```

`filterDNA` *Filter reads comming from double strand sequences from a bam File*

Description

Filter putative double strand DNA from a strand specific RNA-seq using a window sliding across the genome.

Usage

```
filterDNA(
  file,
  destination,
  statFile = "out.stat",
  sequences,
  mapqFilter = 0,
  paired,
  yieldSize = 1e+06,
  winWidth = 1000L,
  winStep = 100L,
```

```
  readProp = 0.5,
  threshold = 0.7,
  pvalueThreshold = 0.05,
  useCoverage = FALSE,
  mustKeepRanges,
  getWin = FALSE,
  minCov = 0,
  maxCov = 0,
  errorRate = 0.01
)
```

Arguments

`file`	the input bam file to be filterd. Your bamfile should be sorted and have an index file located at the same path.
`destination`	the file path where the filtered output will be written
`statFile`	the file to write the summary of the results
`sequences`	the list of sequences to be filtered
`mapqFilter`	every read that has mapping quality below `mapqFilter` will be removed before any analysis. If missing, the entire bam file will be read.
`paired`	if TRUE then the input bamfile will be considered as paired-end reads. If missing, 100 thousands first reads will be inspected to test if the input bam file in paired-end or single-end.
`yieldSize`	by default is 1e6, i.e. the bam file is read by block of reads whose size is defined by this parameter. It is used to pass to same parameter of the scanBam function.
`winWidth`	the length of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`readProp`	a read is considered to be included in a window if at least `readProp` of it is in the window. Specified as a proportion. 0.5 by default.
`threshold`	the strand proportion threshold to test whether to keep a window or not. 0.7 by default
`pvalueThreshold`	the threshold for the p-value in the test of keeping windows. 0.05 by default
`useCoverage`	if TRUE, then the strand information in each window corresponds to the sum of coverage coming from positive/negative reads; and not the number of positive/negative reads as default.
`mustKeepRanges`	a GRanges object; all reads that map to those ranges will be kept regardless the strand proportion of the windows containing them.
`getWin`	if TRUE, the function will not only filter the bam file but also return a data frame containing the information of all windows of the original and filtered bam file.
`minCov`	if `useCoverage=FALSE`, every window that has less than `minCov` reads will be rejected regardless the strand proportion. If `useCoverage=TRUE`, every window has max coverage least than `minCov` will be rejected. 0 by default
`maxCov`	if `useCoverage=FALSE`, every window that has more than `maxCov` reads will be kept regardless the strand proportion. If `useCoverage=TRUE`, every window with max coverage more than `maxCov` will be kept. If 0 then it doesn't have effect on selecting window. 0 by default.
`errorRate`	the probability that an RNA read takes the false strand. 0.01 by default.

Details

filterDNA reads a bam file containing strand specific RNA reads, and filter reads coming from putative double strand DNA. Using a window sliding across the genome, we calculate the positive/negative proportion of reads in each window. We then use logistic regression to estimate the strand proportion of reads in each window, and calculate the p-value when comparing that to a given threshold. Let π be the strand proportion of reads in a window.

Null hypothesis for positive window: $\pi \leq threshold$.

Null hypothesis for negative window: $\pi \geq 1 - threshold$.

Only windows with p-value <= `pvalueThreshold` are kept. For a kept positive window, each positive read in this window is kept with the probability (P-M)/P where P be the number of positive reads, and M be the number of negative reads. That is because those M negative reads are supposed to come from double-strand DNA, then there should be also M postive reads among the P positive reads come from double-strand DNA. In other words, there are only (P-M) positive reads come from RNA. Each negative read is kept with the probability equalling the rate that an RNA read of your sample has wrong strand, which is `errorRate`. Similar for kept negative windows.

Since each alignment can be belonged to several windows, then the probability of keeping an alignment is the maximum probability defined by all windows that contain it.

Value

if `getWin` is TRUE: a DataFrame object which could also be obtained by the function `getStrandFromBamFile`

See Also

`getStrandFromBamFile`, `plotHist`, `plotWin`

Examples

```
file <- system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
out_bam <- tempfile(fileext = ".bam")
out_log <- tempfile(fileext = ".log")
filterDNA(file, sequences = '10', destination = out_bam, statFile = out_log)
```

`getStrandFromBamFile` *Get the strand information of all windows from bam files*

Description

Get the number of positive/negative reads/coverage of all slding windows from the bam input files

Usage

```
getStrandFromBamFile(
  files,
  sequences,
  mapqFilter = 0,
  yieldSize = 1e+06,
  winWidth = 1000L,
  winStep = 100L,
```

```
  readProp = 0.5,
  paired
)
```

Arguments

`files`	the input bam files. Your bamfiles should be sorted and have their index files located at the same path.
`sequences`	character vector used to restrict analysed alignments to a subset of chromosomes (i.e. sequences) within the provided bam file. These correspond to chromosomes/scaffolds of the reference genome to which the reads were mapped. If absent, the whole bam file will be read. NB: This must match the chromosomes as defined in your reference genome. If the reference chromosomes were specified using the 'chr' prefix, ensure the supplied vector matches this specification.
`mapqFilter`	every read that has mapping quality below `mapqFilter` will be removed before any analysis.
`yieldSize`	by default is 1e6, i.e. the bam file is read by block of reads whose size is defined by this parameter. It is used to pass to same parameter of the scanBam function.
`winWidth`	the width of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`readProp`	A read is considered to be included in a window if at least `readProp` of it is in the window. Specified as a proportion. 0.5 by default.
`paired`	if TRUE then the input bamfile will be considered as paired-end reads. If missing, 100 thousands first reads will be inspected to test if the input bam file in paired-end or single-end.

Details

This function moves along the specified chromosomes (i.e. sequences) using a sliding window approach, and counts the number of reads in each window which align to the +/- strands of the reference genome. As well as the number of reads, the total coverage for each strand is also returned for each window, representing the total number of bases covered.

Average coverage for the entire window can be simply calculated by dividing the total coverage by the window size.

Value

a DataFrame object containing the number of positive/negative reads and coverage of each window sliding across the bam file. The returned DataFrame has 10 columns:

Type: can be either SE if the input file contains single-end reads, or R1/R2 if the input file contains paired-end reads.

Seq: the reference sequence (chromosome/scaffold) that the reads were mapped to.

Start: the start position of the sliding window.

End: the end position of the sliding window.

NbPos/NbNeg: number of positive/negative reads that overlap the sliding window.

CovPos/CovNeg: number of bases coming from positive/negative reads that were mapped in the sliding window.

MaxCoverage: the maximum coverage within the sliding window.

File: the name of the input file.

See Also

`filterDNA`, `plotHist`, `plotWin`

Examples

```
file <- system.file('extdata','s1.sorted.bam',package = 'strandCheckR')
win <- getStrandFromBamFile(file,sequences='10')
win
```

`getStrandFromReadInfo` *Get the strand information of all windows from read information*

Description

Get the number of positive/negative reads of all windows from read information obtained from [Rsamtools::scanBam()]

Usage

```
getStrandFromReadInfo(
  readInfo,
  winWidth = 1000L,
  winStep = 100L,
  readProp = 0.5,
  subset = NULL
)
```

Arguments

`readInfo`	a list containing read information returned by [Rsamtools::scanBam()].
`winWidth`	the length of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`readProp`	A read is considered to be included in a window if at least `readProp` of it is in the window. Specified as a proportion. 0.5 by default.
`subset`	an integer vector specifying the subset of reads to consider

Value

a DataFrame object containing the number of positive/negative reads and coverage of each window sliding .

See Also

`filterDNA`, `getStrandFromBamFile`

Examples

```
library(Rsamtools)
file <- system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
readInfo <- scanBam(file, param =
ScanBamParam(what = c("pos","cigar","strand")))
getStrandFromReadInfo(readInfo[[1]],1000,100,0.5)
```

`getWinOverlapEachIRange` *Get the ranges of sliding windows that overlap each range of an IRanges object.*

Description

Get the ranges of sliding windows that overlap each range of an IRanges object.

Usage

```
getWinOverlapEachIRange(
  x,
  winWidth = 1000L,
  winStep = 100L,
  readProp = 0.5,
  maxWin = Inf
)
```

Arguments

`x`	an IRanges object containing the start and end position of each read fragment.
`winWidth`	the width of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`readProp`	A read is considered to be included in a window if at least `readProp` of it is in the window. Specified as a proportion.
`maxWin`	The maximum window ID

Details

This finds the windows that overlap each range of the input IRanges object. Each range corresponds to a read fragment. This allows the total number of read fragments within a window to be calculated simply using [IRanges::coverage()].

Value

An IRanges object containing the index of the windows overlapping each read fragment

Examples

```
library(IRanges)
x <- IRanges(start=round(runif(100,1000,10000)),width=100)
getWinOverlapEachIRange(x)
```

`getWinOverlapEachReadFragment` *Get the window ranges that overlap each read fragment*

Description

Calculate the window ranges that overlap each read fragment

Usage

```
getWinOverlapEachReadFragment(
  readInfo,
  strand,
  winWidth,
  winStep,
  readProp,
  useCoverage = FALSE,
  subset = NULL
)
```

Arguments

`readInfo`	a list contains the read information
`strand`	the considering strand
`winWidth`	the width of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`readProp`	a read fragment is considered to be included in a window if and only if at least `readProp` percent of it is in the window.
`useCoverage`	either base on coverage or number of reads
`subset`	if we consider only a subset of the input reads

Value

If `useCoverage=FALSE`: an IRanges object which contains the range of sliding windows that overlap each read fragment. If `useCoverage=TRUE`: a list of two objects, the first one is the later IRanges object, the second one is an integer-Rle object which contains the coverage of the input `readInfo`

Examples

```
library(Rsamtools)
file <- system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
readInfo <- scanBam(file, param =
ScanBamParam(what = c("pos","cigar","strand")))
getWinOverlapEachReadFragment(readInfo[[1]],"+",1000,100,0.5)
```

`getWinOverlapGRanges` *Get the sliding windows that overlap a GRanges object*

Description

Get the sliding windows that overlap a GRanges object.

Usage

```
getWinOverlapGRanges(
  x,
  seqInfo,
  winWidth = 1000L,
  winStep = 100L,
  nbOverlapBases = 1
)
```

Arguments

`x`	a GRanges object, which defines the coordinates of the ranges in the reference genome that all reads mapped to those ranges must be kept by the filtering method `filterDNA`.
`seqInfo`	a data frame that contains some key information of the sequences
`winWidth`	the length of the sliding window, 1000 by default.
`winStep`	the step length to sliding the window, 100 by default.
`nbOverlapBases`	a window is considered to overlap with a range of `x` if it overlaps with at least `nbOverlapBases` bases.

Details

This finds the windows that overlaps the positive/negative strand of a GRanges object. The GRanges object, which is `mustKeepRanges` in the `filterDNA` method, defines the coordinates of the ranges in the reference genome that all reads mapped to those ranges must be kept by the filtering method `filterDNA`. This method makes use of the method `getWinOverlapEachIRange` by pretending each given range as the range of a read. Since the widths of `x` are not necessarily the same (as normal read lengths), we use `nbOverlapBases` to specify the minimum number of bases that a window should overlap with a range of `x`, instead of using proprotion as `readProp` in `getWinOverlapEachIRange`.

Value

A list of two logical vectors (for positive and negative strand) defining which windows that overlap with the given GRanges object.

Examples

```
library(GenomicRanges)
x <- GRanges(seqnames = "10",ranges = IRanges(start = c(10000,15000),
end=c(20000,30000)),strand = c("+","-"))
seqInfo <- data.frame("Sequence"=10,"FirstBaseInPart"=1)
getWinOverlapGRanges(x,seqInfo)
seqInfo <- data.frame("Sequence"=10,"FirstBaseInPart"=10000000)
```

```
getWinOverlapGRanges(x,seqInfo)
```

`intersectWithFeature` *Intersect the windows data frame with an annotation data frame*

Description

Intersect the windows with an annotation data frame to get features that overlap with each window

Usage

```
intersectWithFeature(
  windows,
  annotation,
  getFeatureInfo = FALSE,
  overlapCol = "OverlapFeature",
  mcolsAnnot,
  collapse,
  ...
)
```

Arguments

`windows`	data frame containing the strand information of the sliding windows. Windows can be obtained using the function `getStrandFromBamFile`.
`annotation`	a Grange object that you want to intersect with your windows. It can have mcols which contains the information or features that could be able to integrate to the input windows
`getFeatureInfo`	whether to get the information of features in the mcols of annotation data or not. If FALSE the return windows will have an additional column indicating whether a window overlaps with any range of the annotion data. If TRUE the return windows will contain the information of features that overlap each window
`overlapCol`	the columnn name of the return windows indicating whether a window overlaps with any range of the annotion data.
`mcolsAnnot`	the column names of the mcols of the annotation data that you want to get information
`collapse`	character which is used collapse multiple features that overlap with a same window into a string. If missing then we don't collapse them.
`...`	used to pass parameters to GenomicRanges::findOverlaps

Value

the input windows DataFrame with some additional columns

See Also

`getStrandFromBamFile`, `plotHist`, `plotWin`

Examples

```
bamfilein = system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
windows <- getStrandFromBamFile(file = bamfilein)
#add chr before chromosome names to be consistent with the annotation
windows$Seq <- paste0('chr',windows$Seq)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
annot <- transcripts(TxDb.Hsapiens.UCSC.hg38.knownGene)
# get the transcript names that overlap with each window
windows <- intersectWithFeature(windows,annot,mcolsAnnot='tx_name')
# just want to know whether there's any transcript that
# overlaps with each window
windows <- intersectWithFeature(windows,annot,overlapCol='OverlapTranscript')
plotHist(windows,facets = 'OverlapTranscript')
plotWin(windows,facets = 'OverlapTranscript')
```

`plotHist` *Plot the histogram of positive proportions*

Description

Plot the histogram of positive proportions of the input data frame coming from `getStrandFromBamFile`

Usage

```
plotHist(
  windows,
  save = FALSE,
  file = "hist.pdf",
  groupBy = NULL,
  normalizeBy = NULL,
  split = c(10, 100, 1000),
  breaks = 100,
  useCoverage = FALSE,
  heatmap = FALSE,
  ...
)
```

Arguments

`windows`	data frame containing the strand information of the sliding windows. Windows can be obtained using the function `getStrandFromBamFile`.
`save`	if TRUE, then the plot will be save into the file given by `file` parameter
`file`	the file name to save to plot
`groupBy`	the columns that will be used to split the data.
`normalizeBy`	instead of using the raw read count/coverage, we will normalize it to a proportion by dividing it to the total number of read count/coverage of windows that have the same value in the `normalizeBy` columns.

`split`	an integer vector that specifies how you want to partition the windows based on the coverage. By default `split` = c(10,100,1000), which means that your windows will be partitionned into 4 groups, those have coverage < 10, from 10 to 100, from 100 to 1000, and > 1000
`breaks`	an integer giving the number of bins for the histogram
`useCoverage`	if TRUE then plot the coverage strand information, otherwise plot the number of reads strand information. FALSE by default
`heatmap`	if TRUE, then use heat map to plot the histogram, otherwise use barplot. FALSE by default.
`...`	used to pass parameters to facet_wrap

Value

If `heatmap=FALSE`: a ggplot object

See Also

`getStrandFromBamFile`, `plotWin`

Examples

```
bamfilein = system.file('extdata','s1.sorted.bam',package = 'strandCheckR')
win  <- getStrandFromBamFile(file = bamfilein,sequences='10')
plotHist(win)
```

`plotWin` *Plot the number of reads vs the proportion of '+' stranded reads.*

Description

Plot the number of reads vs the proportion of '+' stranded reads of all windows from the input data frame.

Usage

```
plotWin(
  windows,
  split = c(10, 100, 1000),
  threshold = c(0.6, 0.7, 0.8, 0.9),
  save = FALSE,
  file = "win.pdf",
  groupBy = NULL,
  useCoverage = FALSE,
  ...
)
```

Arguments

`windows`	data frame containing the strand information of the sliding windows. Windows should be obtained using the function `getStrandFromBamFile` to ensure the correct data structure.
`split`	an integer vector that specifies how you want to partition the windows based on coverage. By default `split` = c(10,100,1000), partition windows into 4 groups based on these values.
`threshold`	a `numeric` vector between 0.5 & 1 that specifies which threshold lines to draw on the plot. The positive windows above the threshold line (or negative windows below the threshold line) will be kept when using `filterDNA`.
`save`	if TRUE, then the plot will be save into the file given by `file` parameter
`file`	the file name to save to plot
`groupBy`	the column that will be used to split the data (which will be used in the facets method of ggplot2).
`useCoverage`	if TRUE then plot the coverage strand information, otherwise plot the number of reads strand information. FALSE by default
`...`	used to pass parameters to facet_wrap during plotting

Details

This function will plot the proportion of '+' stranded reads for each window, against the number of reads in each window. The threshold lines indicate the hypothetical boundary where windows will contain reads to kept or discarded using the filtering methods of `filterDNA`. Any plot can be easily modified using standard ggplot2 syntax (see Examples)

Value

The plot will be returned as a standard ggplot2 object

See Also

`getStrandFromBamFile`, `plotHist`

Examples

```
bamfilein = system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
windows <- getStrandFromBamFile(file = bamfilein,sequences = '10')
plotWin(windows)

# Change point colour using ggplot2
library(ggplot2)
plotWin(windows) +
scale_colour_manual(values = rgb(seq(0, 1, length.out = 4), 0, 0))
```

Index