Package 'strandCheckR'

April 15, 2020

Type Package

Title Calculate strandness information of a bam file

Version 1.4.0

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.

License GPL (>= 2)

LazyData TRUE

Imports dplyr, magrittr, GenomeInfoDb, GenomicAlignments, GenomicRanges, IRanges, Rsamtools, S4Vectors, grid, BiocGenerics, ggplot2, reshape2, stats, gridExtra, TxDb.Hsapiens.UCSC.hg38.knownGene, methods, stringr

biocViews RNASeq, Alignment, QualityControl, Coverage, ImmunoOncology

RoxygenNote 6.0.1

VignetteBuilder knitr

Suggests BiocStyle, knitr, testthat

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

git_branch RELEASE_3_10

git_last_commit c5f3d37

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Thu-Hien To [aut, cre], Steve Pederson [aut]

R topics documented:

`strandCheckR-package` *Quantify and Filter putative double strand DNA from strand-specific RNA bam file*

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.

Details

The package has some following main functions:

- `getWinFromBamFile`: calculate positive/negative proprortion and sum of reads over all sliding windows from a bam file

- `plotHist`: plot histogram of positive proportion of windows calculated from `getWinFromBamFile` method

- `plotWin`: plot positive proportion vs number of reads of windows calculated from `getWinFromBamFile` method

- `filterDNA`: filter a bam file

Author(s)

Thu-Hien To & Steve Pederson

Maintainer: Hien To <hien.to@adelaide.edu.au>

Examples

```
bamfilein <- system.file("extdata","s1.sorted.bam",package = "strandCheckR")
windows <- getWinFromBamFile(bamfilein)
plotWin(windows)
plotHist(windows)
filterDNA(file = bamfilein,destination = "filter.bam")
```

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

Description

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

Usage

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

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 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

checkPairedEnd *Test whether a bam file if 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)
```

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 alignments

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

`filterDNA` *Filter 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, sequences, mapqFilter = 0, paired,
yieldSize = 1e+06, winWidth = 1000, winStep = 100, 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 records whose size is defined by this paramter. It is used to pass to same paramter 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 getWinFromBamFile

See Also

getWinFromBamFile, plotHist, plotWin

Examples

```
file <- system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
filterDNA(file,sequences='10',destination='out.bam')
```

---

getWinFromBamFile *get the strand information of all windows from bam files*

Description

get the number of positive/negative reads of all windows from bam files

Usage

```
getWinFromBamFile(files, sequences, mapqFilter = 0, yieldSize = 1e+06,
winWidth = 1000, winStep = 100, 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	the list of sequences to be read
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 records whose size is defined by this paramter. It is used to pass to same paramter 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.

Value

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

See Also

filterDNA, plotHist, plotWin

Examples

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

---

getWinFromGranges *Get the Sliding Windows from a GRanges object*

Description

Get the positive/negative windows that overlap a GRanges object

Usage

```
getWinFromGranges(x, seqInfo, winWidth = 1000, winStep = 100)
```

Arguments

x	a GRanges object
seqInfo	a data frame that contains some key information of the alignments
winWidth	The width of each window
winStep	The step size for sliding the window

Value

A list of two logical vectors (for positive and negative strand) defining which windows that overlap the given Granges objects

---

getWinFromIRanges *Get the Ranges of Sliding Windows from an IRanges object*

Description

Get the Ranges of Sliding Windows from an IRanges object

Usage

```
getWinFromIRanges(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 each window
winStep	The step size for sliding the window
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 fragment of a read and returns a range containing this list of windows for each read fragment. This allows the total number of read fragments within a window to be calculated simply using coverage.

Value

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

getWinFromReadInfo *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 scanBam function

Usage

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

Arguments

readInfo	a list contains read information returned by scanBam function when read a bam file
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.
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, getWinFromBamFile

getWinOfAlignments *get the window ranges of alignments*

Description

calculate the windows that contain each read fragment

Usage

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

Arguments

readInfo	a list contains the read information of one sequence
strand	the considering strand
winWidth	the window size
winStep	the window winStep
readProp	a read 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

---

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 getWinFromBamFile.
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

getWinFromBamFile, plotHist, plotWin

Examples

```
bamfilein = system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
windows <- getWinFromBamFile(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')
```

---

keptProbaWin *get the probability of begin 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 getWinOfAlignments for positive reads
winNegAlignments	an object returned by getWinOfAlignments 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 least than minCov reads, then it will be rejected regardless 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 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 IRange object defind 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

plotHist *Plot the histogram of positive proportions*

Description

Plot the histogram of positive proportions of the input histogram data frame

Usage

```
plotHist(windows, save = FALSE, file = "hist.pdf", group_by = NULL,
normalize_by = 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 getWinFromBamFile.
save	if TRUE, then the plot will be save into the file given by file parameter
file	the file name to save to plot
group_by	the column names of windows that will be used to group the data
normalize_by	the column names of windows that will be used to normalize the read count or read coverage into proportion
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

getWinFromBamFile, plotWin

Examples

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

plotWin *Plot the Proportion of + Stranded Reads*

Description

Plot the number of reads vs the proportion of '+' stranded reads.

Usage

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

Arguments

windows	data frame containing the strand information of the sliding windows. Windows should be obtained using the function getWinFromBamFile 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
group_by	colnames of windows which will be used to split the plot
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

getWinFromBamFile, plotHist

Examples

```
bamfilein = system.file('extdata','s2.sorted.bam',package = 'strandCheckR')
windows <- getWinFromBamFile(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))
```

`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 `getWinFromBamFile`

Usage

```
summarizeHist(windows, split = c(10, 100, 1000), breaks = 100,
useCoverage = FALSE, group_by = NULL, normalize_by = NULL)
```

Arguments

`windows`	data frame containing the strand information of the sliding windows. Windows can be obtained using the function `getWinFromBamFile`.
`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
`group_by`	the column names of windows that will be used to group the data
`normalize_by`	the column names of windows that will be used to normalize the read count or read coverage into proportion

Value

a dataframe object

See Also

getWinFromBamFile, plotHist, plotWin

Index