Package ‘DEScan2’

March 18, 2024

Type Package
Title Differential Enrichment Scan 2
Version 1.22.0
Date 2023-03-09
Maintainer Dario Righelli <dario.righelli@gmail.com>
Description Integrated peak and differential caller, specifically designed for
broad epigenomic signals.
Encoding UTF-8
License Artistic-2.0
LazyData TRUE
biocViews ImmunoOncology, PeakDetection, Epigenetics, Software,
   Sequencing, Coverage
Depends R (>= 3.5), GenomicRanges
Imports BiocParallel, BiocGenerics, ChIPpeakAnno, data.table,
   DelayedArray, GenomeInfoDb, GenomicAlignments, glue, IRanges,
   plyr, Rcpp (>= 0.12.13), rtracklayer, S4Vectors (>= 0.23.19),
   SummarizedExperiment, tools, utils
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.2.3
Suggests BiocStyle, knitr, rmarkdown, testthat, edgeR, limma, EDASeq,
   RUVSeq, RColorBrewer, statmod
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/DEScan2
git_branch RELEASE_3_18
git_last_commit 9abfd1
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-18
**Author**  Dario Righelli [aut, cre],
John Koberstein [aut],
Bruce Gomes [aut],
Nancy Zhang [aut],
Claudia Angelini [aut],
Lucia Peixoto [aut],
Davide Risso [aut]

---

**R topics documented:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>binnedCoverage</td>
<td>3</td>
</tr>
<tr>
<td>binnedCovOnly</td>
<td>4</td>
</tr>
<tr>
<td>binToChrCoordMatRowNames</td>
<td>4</td>
</tr>
<tr>
<td>computeCoverageMovingWindowOnChr</td>
<td>5</td>
</tr>
<tr>
<td>computeLambdaOnChr</td>
<td>5</td>
</tr>
<tr>
<td>computeZ</td>
<td>6</td>
</tr>
<tr>
<td>constructBedRanges</td>
<td>7</td>
</tr>
<tr>
<td>countFinalRegions</td>
<td>8</td>
</tr>
<tr>
<td>createGranges</td>
<td>9</td>
</tr>
<tr>
<td>cutGRangesPerChromosome</td>
<td>10</td>
</tr>
<tr>
<td>c_get_disjoint_max_win</td>
<td>11</td>
</tr>
<tr>
<td>DEScan2</td>
<td>11</td>
</tr>
<tr>
<td>divideEachSampleByChromosomes</td>
<td>12</td>
</tr>
<tr>
<td>evenRunMean</td>
<td>13</td>
</tr>
<tr>
<td>evenRunSum</td>
<td>13</td>
</tr>
<tr>
<td>finalRegions</td>
<td>14</td>
</tr>
<tr>
<td>findOverlapsOverSamples</td>
<td>15</td>
</tr>
<tr>
<td>findPeaks</td>
<td>16</td>
</tr>
<tr>
<td>fromSamplesToChrsGRangesList</td>
<td>18</td>
</tr>
<tr>
<td>generateDFofSamplesPerChromosomes</td>
<td>18</td>
</tr>
<tr>
<td>get_disjoint_max_win</td>
<td>19</td>
</tr>
<tr>
<td>giveUniqueNamesToPeaksOverSamples</td>
<td>20</td>
</tr>
<tr>
<td>initMergedPeaksNames</td>
<td>20</td>
</tr>
<tr>
<td>keepRelevantChrs</td>
<td>21</td>
</tr>
<tr>
<td>rcpparma_get_disjoint_max_win</td>
<td>21</td>
</tr>
<tr>
<td>readBamAsBed</td>
<td>22</td>
</tr>
<tr>
<td>readBedFile</td>
<td>23</td>
</tr>
<tr>
<td>readFilesAsGRangesList</td>
<td>23</td>
</tr>
<tr>
<td>RleListToRleMatrix</td>
<td>24</td>
</tr>
<tr>
<td>saveGRangesAsBed</td>
<td>25</td>
</tr>
<tr>
<td>saveGRangesAsTsv</td>
<td>26</td>
</tr>
<tr>
<td>setGRGenomeInfo</td>
<td>27</td>
</tr>
</tbody>
</table>

**Index**  28
binnedCoverage

Description

description of the function computes the coverage over a binned chromosome, starting from a per base computed coverage.

Usage

binnedCoverage(
  bins,
  numvar,
  mcolname,
  covMethod = c("max", "mean", "sum", "min"),
  roundingMethod = c("none", "floor", "ceiling", "round")
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bins</td>
<td>a GRanges object representing a chromosome binned.</td>
</tr>
<tr>
<td>numvar</td>
<td>an RleList representing the per base coverage over the chr.</td>
</tr>
<tr>
<td>mcolname</td>
<td>the name of column where the sum have to be stored.</td>
</tr>
<tr>
<td>covMethod</td>
<td>a method to apply for the computing of the coverate it can be one of &quot;max&quot;, &quot;mean&quot;, &quot;sum&quot;, &quot;min&quot;.&quot;(&quot;max&quot; is default)</td>
</tr>
<tr>
<td>roundingMethod</td>
<td>a method to apply to round the computations it can be one of &quot;none&quot;, &quot;floor&quot;, &quot;ceiling&quot;, &quot;round&quot;. It's useful only when using covMethod=&quot;mean&quot;. (&quot;none&quot; is default)</td>
</tr>
</tbody>
</table>

Value

the bins GRanges with the mcolname attached

Examples

```r
## dividing one chromosome in bins of 50 bp each
seqinfo <- GenomeInfoDb::Seqinfo(genome="mm9")
bins <- GenomicRanges::tileGenome(
  seqlengths=GenomeInfoDb::seqlengths(seqinfo)[1],
  tilewidth=50,
  cut.last.tile.in.chrom=TRUE)
gr <- GenomicRanges::GRanges(seqnames = S4Vectors::Rle("chr1", 100),
ranges=IRanges::IRanges(start = seq(from=10, to=1000, by=10),
end=seq(from=20, to=1010, by = 10)))
cov <- GenomicRanges::coverage(x=gr)
(binnedMaxCovGR <- binnedCoverage(bins, cov, "binned_cov"))
(binnedMeanCovGR <- binnedCoverage(bins, cov, "binned_cov", covMethod="mean")
```
```r
binnedCovOnly <- binnedCoverage(bins, cov, "binned_cov", covMethod="sum")
```

### Description

It’s useful just to coerce the bin coverage to an Rle object.

### Usage

```r
binnedCovOnly(bins, numvar, mcolname)
```

### Arguments

- **bins**: a GRanges object representing a chromosome binned
- **numvar**: an RleList representing the per base coverage over the chr
- **mcolname**: the name of column where the sum have to be stored

### Value

An Rle within the per bin computed coverage.

```r
binToChrCoordMatRowNames
```

### Description

Computes the starting range of the bins for the binMatrix, taking in input the length of the chromosome of the matrix.

### Usage

```r
binToChrCoordMatRowNames(binMatrix, chrLength, binWidth = 50)
```

### Arguments

- **binMatrix**: a matrix where each row represents a bin.
- **chrLength**: the length of the chromosome of the binMatrix.
- **binWidth**: the width of the bin.

### Value

The binMatrix with start range as rownames.
computeCoverageMovingWindowOnChr

Description

computes the coverage on a chromosome with a set of moving windows of dimensions minWinWidth:maxWinWidth

Usage

computeCoverageMovingWindowOnChr(
  chrBedGRanges,
  minWinWidth = 50,
  maxWinWidth = 1000,
  binWidth = 50,
  verbose = TRUE
)

Arguments

  chrBedGRanges  a GRanges to compute the coverage
  minWinWidth    the minimum width of the window to use for the coverage
  maxWinWidth    the maximum width of the window to use for the coverage
  binWidth       the dimension of the bin in base number

Value

  RleList where each element is a window within the Rle of its coverage

computeLambdaOnChr

Description

computes the lambdas on a chromosome for the winVector windows and other two windows (min/maxCompWinWidth) to compare with
Usage

```r
computeLambdaOnChr(
    chrGRanges, 
    winVector = seq_len(20), 
    minChrRleWComp, 
    minCompWinWidth = 5000, 
    maxChrRleWComp, 
    maxCompWinWidth = 10000, 
    verbose = TRUE
)
```

Arguments

- `chrGRanges`: the GRanges representing the reads of the chromosome.
- `winVector`: the of width of the windows used to compute the coverage.
- `minChrRleWComp` and Rle object within coverage of window of width `minCompWinWidth`.
- `minCompWinWidth`: the width of the window used for the coverage of `minChrRleWCom` in bases.
- `maxChrRleWComp` and Rle object within coverage of window of width `maxCompWinWidth`.
- `maxCompWinWidth`: the width of the window used for the coverage of `maxChrRleWCom` in bases.
- `verbose`: verbose flag.
- `binSize`: the size of the bin in bases.

Value

- an RleList where each element is a window of `winVector`, within an Rle representing the lambda computed for that window.

Descripción

Computes Z-Scores returning the z matrix.

Usage

```r
computeZ( 
    lambdaChrRleList, 
    runWinRleList, 
    chrLength, 
    minCount = 0.1, 
    binSize = 50, 
    verbose = FALSE
)
```
constructBedRanges

Arguments

lambdaChrRleList an RleList of lambda values computed by computeLambdaOnChr function each element of the list is an Rle representing the lambda for the moving window in the list position.

runWinRleList an RleList of coverage values computed by computeCoverageMovingWindowOnChr function each element of the list is an Rle representing the coverage for the moving window in the list position.

chrLength the length of the chr in analysis.

minCount A small constant (usually no larger than one) to be added to the counts prior to the log transformation to avoid problems with log(0).

binSize the size of the bin.

verbose verbose output.

Value

z a matrix of z scores for each window (column) and bin (row), where the rownames represent the starting base of each bin.

Description

Constructs a GRanges object from a bam/bed/bed.zip file in a consistent way.

Usage

constructBedRanges(
  filename,
  filetype = c("bam", "bed", "bed.zip", "narrow", "broad"),
  genomeName = NULL,
  onlyStdChrs = FALSE,
  arePeaks = FALSE,
  verbose = FALSE
)

Arguments

filename the complete file path of a bam?bed file.

filetype the file type bam/bed/bed.zip/narrow/broad.

genomeName the name of the genome used to map the reads (i.e. "mm9"). N.B. if NOT NULL the GRanges Seqinfo will be forced to genomeName Seqinfo (needs Internet access, but strongly suggested!)

onlyStdChrs flag to keep only standard chromosome.

arePeaks flag indicating if the file contains peaks.

verbose flag to obtain verbose output.
Value

a GRanges object.

Examples

```r
files <- list.files(system.file("extdata/bam/", package="DEScan2"), pattern="bam$", full.names=TRUE)
bgr <- constructBedRanges(files[1], filetype="bam", genomeName="mm9", onlyStdChrs=TRUE)
bgr
countFinalRegions

countFinalRegions

Description

count reads falling within the final regions.

Usage

countFinalRegions(
  regionsGRanges,
  readsFilePath = NULL,
  fileType = c("bam", "bed"),
  minCarriers = 2,
  genomeName = NULL,
  onlyStdChrs = FALSE,
  carrierscolname = "k-carriers",
  ignStrandSO = TRUE,
  modeSO = "Union",
  saveFlag = FALSE,
  savePath = "finalRegions",
  verbose = TRUE
)

Arguments

regionsGRanges a GRanges objects representing the peaks to compute the coverage, with a "k-carriers" mcols. (typically generated by finalRegions function).
readsFilepath the filepath of bam or bed files necessary to compute the coverage.
fileType the file type of the input files.
minCarriers minimum number of carriers (samples).
genomeName code name of the genome of reads files (i.e. "mm9").
onlyStdChrs a flag indicating if to keep only the standard chromosomes
createGranges

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>carrierscolname</td>
<td>character describing the name of the column within the carriers number (default is &quot;k-carriers&quot;).</td>
</tr>
<tr>
<td>ignStrandSO</td>
<td>a flag indicating if to ignore the reads strand. (see GenomicAlignments::summarizeOverlaps).</td>
</tr>
<tr>
<td>modeSO</td>
<td>the mode to use, default is &quot;Union&quot;. (see GenomicAlignments::summarizeOverlaps).</td>
</tr>
<tr>
<td>saveFlag</td>
<td>a flag indicating if to save the results.</td>
</tr>
<tr>
<td>savePath</td>
<td>the path where to store the results.</td>
</tr>
<tr>
<td>verbose</td>
<td>verbose output.</td>
</tr>
</tbody>
</table>

Value

A SummarizedExperiment object containing as assays the read counts matrix with regions as rows and samples as columns, and as rowRanges the GRanges object representing the peaks used as rows in the matrix.

Examples

```r
filename <- system.file("extdata/regions/regions.rds", package="DEScan2")
regionsGR <- readRDS(file=filename)
reads.path <- system.file("extdata/bam", package="DEScan2")
finalRegionsSE <- countFinalRegions(regionsGRanges=regionsGR,
                                   readsFilePath=reads.path, fileType="bam", minCarriers=1,
                                   genomeName="mm9", onlyStdChrs=TRUE, ignStrandSO=TRUE, saveFlag=FALSE,
                                   verbose=TRUE)
library("SummarizedExperiment")
assay(finalRegionsSE) ## matrix of counts
rowRanges(finalRegionsSE) ## the GRanges of the input regions
```

createGranges

createGranges

Description

A simplified wrapper function to create a GRanges object.

Usage

```r
createGranges(chrSeqInfo, starts, widths, mcolname = NULL, mcolvalues = NULL)
```

Arguments

- **chrSeqInfo**: a seqinfo object.
- **starts**: the start ranges.
- **widths**: the width of each range.
- **mcolname**: the name for the mcol attribute.
- **mcolvalues**: the values for the mcol attribute.
cutGRangesPerChromosome

**Value**

a GRanges object.

**Examples**

```r
chrSeqInfo <- GenomeInfoDb::Seqinfo(genome="mm9")["chr1"]
starts=sample(seq_len(100), 10)
widths=starts+10;
mcolname <- "z-score";
mcolvalues <- sample(seq_len(100), 10)
chrGR <- createGranges(chrSeqInfo=chrSeqInfo, starts=starts, widths=widths,
mcolname=mcolname, mcolvalues=mcolvalues)
```

**Description**

takes in input a GRanges object, producing a LIST of GRanges, one for each chromosome.

**Usage**

cutGRangesPerChromosome(GRanges)

**Arguments**

- **GRanges**
a GRanges object.

**Value**
a named list of GRanges, one for each chromosome.

**Examples**

```r
library("GenomicRanges")
gr <- GRanges(
  seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
ranges=IRanges(1:10, end=10),
strand=Rle(strand(c("-", "*", "+", "+", "+")), c(1, 2, 2, 3, 2)),
seqlengths=c(chr1=11, chr2=12, chr3=13))
(grchrlst <- cutGRangesPerChromosome(gr))
```
Description

just a wrapper for the C function. Useful to modify indexes and colnames.

Usage

c_get_disjoint_max_win(
  z0,
  sigwin = 10,
  nmax = 9999999,
  zthresh = 10,
  verbose = FALSE
)

Arguments

z0 the z matrix.
sigwin the sigwin.
nmax the nmax.
zthresh peaks lower than this value will not be kept.
verbose verbose flag.

Value

a matrix

DEScan2

Description

integrated peak and differential caller, specifically designed for broad epigenomic signals.

Author(s)

some authors
divideEachSampleByChromosomes

Description

taken in input a grangeslist of samples, generate a list of samples where each element has a GRangesList each element of the GRangesList represents a single chromosome.

Usage

divideEachSampleByChromosomes(samplesGRangesList)

Arguments

samplesGRangesList

a GRangesList of samples.

Value

list of samples where each element is a list of chromosomes and each of these elements is a GRanges.

Examples

library("GenomicRanges")
gr1 <- GRanges(
  seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
  ranges=IRanges(1:10, end=10),
  strand=Rle(strand(c("-", "+", "+", "+-")), c(1, 2, 2, 3, 2)),
  seqlengths=c(chr1=11, chr2=12, chr3=13))
gr2 <- GRanges(
  seqnames=Rle(c("chr1", "chr4", "chr1", "chr3"), c(1, 3, 2, 4)),
  ranges=IRanges(1:10, end=10),
  strand=Rle(strand(c("-", "+", "-", "+-")), c(1, 2, 2, 3, 2)),
  seqlengths=c(chr1=11, chr4=12, chr3=13))
sgrl <- GRangesList(gr1, gr2)
names(sgrl) <- c("samp1", "samp2")
(sampChrGr1 <- divideEachSampleByChromosomes(sgrl))
evenRunMean

Description
this function computes a running mean over x with a window width k (modified from S4Vectors package to work on even k, see evenRunSum).

Usage
evenRunMean(x, k, endrule = c("drop", "constant"), na.rm = FALSE)

Arguments
x an Rle object, typically a coverage object.
k window dimension for the running sum over x.
endrule refer to S4Vectors::runMean.
na.rm refer to S4Vectors::runMean.

Value
an Rle within the running mean over x with a win of length k.

evenRunSum

Description
this function computes a running sum over x with a window width k (modified from S4Vectors package to work on even k, in such a case it adds a length at the end of the output Rle).

Usage
evenRunSum(x, k, endrule = c("drop", "constant"), na.rm = FALSE)

Arguments
x an Rle object, typically a coverage object.
k window dimension for the running sum over x.
endrule refer to S4Vectors::runSum.
na.rm refer to S4Vectors::runSum.

Value
an Rle within the running sum over x with a win of length k.
Description

Align peaks to form common regions then filter regions for presence in multiple replicates taking in input a GRangesList where each element is a sample of called peaks.

Usage

finalRegions(
  peakSamplesGRangesList,  # named GRangesList where each element is a sample of called peaks. A score mcols values is needed for each GRanges. The scorecolname param can be used as reference name for the score. (tipically returned by findPeaks function).
  zThreshold = 20,            # a minimum threshold for the z score. All peaks lesser than this value will be ignored.
  minCarriers = 2,            # a threshold of minimum samples (carriers) for overlapped regions.
  saveFlag = TRUE,            # a flag for saving results in a tsv file.
  outputFolder = "overlappedPeaks",    # the directory name to store the bed file.
  verbose = FALSE,            # verbose output.
  scorecolname = "z-score",  # character describing the name of the column within the peaks score.
  coverageFlag = FALSE,      # boolean indicating if to compute the scores in a coverage mode (sum of the reads of merged peak) or in a score mode (a normalized score across the merged peaks)
  BPPARAM = BiocParallel::bpparam()  # object of class bpparamClass that specifies the back-end to be used for computations. See bpparam for details.
)

Arguments

peakSamplesGRangesList

zThreshold

minCarriers

saveFlag

outputFolder

verbose

scorecolname

coverageFlag

BPPARAM

Value

a GRanges of selected overlapping peaks with z-score, n-peaks, k-carriers as mcols object.
findOverlapsOverSamples

Examples
peak.path <- system.file("extdata/peaks/RData/peaksGRL_all_files.rds", package="DEScan2")
grl <- readRDS(peak.path)
grl

regionsGR <- finalRegions(peakSamplesGRangesList=grl, zThreshold=1, minCarriers=3, saveFlag=FALSE, verbose=TRUE)

findOverlapsOverSamples

Description
given in input a GRangeList where each element is a sample computes the coverage extending a both direction window of prefixed length.

Usage
findOverlapsOverSamples(
samplePeaksGRangelist, extendRegions = 200, minOverlap = 0L, maxGap = -1L, zThresh = 10, verbose = FALSE, scorecolname = "z-score", coverageFlag = FALSE)

Arguments

samplePeaksGRangelist
given a granges list of samples finds the overlapping regions between them.

extendRegions
the number of bases to extend each region at its start and end.

minOverlap
the minimum overlap each peak needs to have. (see ChipPeakAnno::findOverlapsOfPeaks)

maxGap
the maximum gap admissible between the peaks. (see ChipPeakAnno::findOverlapsOfPeaks)

zThresh
a threshold value on z-score/scorecolname

verbose
verbose flag

scorecolname
character describing the name of the column within the peaks score.

coverageFlag
boolean indicating if to compute the scores in a coverage mode (sum of the reads of merged peak) or in a score mode (a normalized score across the merged peaks)
Value

a GRanges of peaks overlapped and unique between samples.

Examples

```r
(peaks.file <- system.file("extdata/peaks/RData/peaksGRL_all_files.rds", package="DEScan2"))
peaksGRLFiles <- readRDS(peaks.file)
(overlPeaks <- findOverlapsOverSamples(peaksGRLFiles))
```

Description

This function calls peaks from bed or bam inputs using a variable window scan with a poisson model using the surrounding maxCompWinWidth (10kb) as background.

Usage

```r
findPeaks(
  files,
  filetype = c("bam", "bed"),
  genomeName = NULL,
  binSize = 50,
  minWin = 50,
  maxWin = 1000,
  zthresh = 10,
  minCount = 0.1,
  minCompWinWidth = 5000,
  maxCompWinWidth = 10000,
  outputFolder = "Peaks",
  save = TRUE,
  force = TRUE,
  verbose = FALSE,
  sigwin = 10,
  onlyStdChrs = TRUE,
  chr = NULL,
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

- **files**: Character vector containing paths of files to be analyzed.
- **filetype**: Character, either "bam" or "bed" indicating format of input file.
- **genomeName**: the code of the genome to use as reference for the input files. (cfr. constructBedRanges function parameters)
findPeaks

binSize  Integer size in bases of the minimum window for scanning. 50 is the default.
minWin   Integer indicating the minimum window size in bases notation.
maxWin   Integer indicating the maximum window size in bases notation.
zthresh  Cuttoff value for z-scores. Only windows with greater z-scores will be kept, default is 10.
minCount A small constant (usually no larger than one) to be added to the counts prior to the log transformation to avoid problems with log(0).
minCompWinWidth minimum bases width of a comparing window for Z-score.
maxCompWinWidth  maximum bases width of a comparing window for Z-score.
outputFolder A string. Name of the folder to save the Peaks (optional) if the directory doesn’t exist, it will be created. (Default is "Peaks")
save   Boolean, if TRUE files will be saved in a "./Peaks/chr*" directory created (if not already present) in the current working directory.
force   a boolean flag indicating if to force output overwriting.
verbose if to show additional messages
sigwin  an integer value used to compute the length of the signal of a peak (default value is 10).
onlyStdChrs a flag to work only with standard chromosomes. (cfr. constructBedRanges function parameters).
chr      if not NULL, a character like "chr#" indicating the chromosomes to use.
BPPARAM object of class bpparamClass that specifies the back-end to be used for computations. See bpparam for details.

Value

A GRangesList where each element is a sample. Each GRanges represents the founded peaks and attached the z-score of the peak as mcols.

Examples

```r
bam.files <- list.files(system.file("extdata/bam", package = "DEScan2"),
   full.names = TRUE)
peaks <- findPeaks(files=bam.files[1], filetype="bam",
   genomeName="mm9",
   binSize=50, minWin=50, maxWin=1000,
   zthresh=5, minCount=0.1, sigwin=10,
   minCompWinWidth=5000, maxCompWinWidth=10000,
   save=FALSE,
   onlyStdChrs=TRUE,
   chr=NULL,
   verbose=FALSE)
head(peaks)
```
fromSamplesToChrsGRangesList

Description

converts a GRangesList organized per samples to a GRangesList organized per Chromosomes where each element is a GRangesList of samples.

Usage

fromSamplesToChrsGRangesList(samplesGRangesList)

Arguments

samplesGRangesList

a GRangesList of samples. Typically generated by findPeaks function.

Value

A GRangesList of chromosomes where each element is a GRanges list of samples.

Examples

library("GenomicRanges")
gr1 <- GRanges(
    seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges=IRanges(1:10, end=10),
    strand=Rle(strand(c("-", "+", "+", "-")), c(1, 2, 2, 3)),
    seqlengths=c(chr1=11, chr2=12, chr3=13))
gr2 <- GRanges(
    seqnames=Rle(c("chr1", "chr4", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges=IRanges(1:10, end=10),
    strand=Rle(strand(c("-", "+", "+", "-")), c(1, 2, 2, 3)),
    seqlengths=c(chr1=11, chr4=12, chr3=13))
sgrl <- GRangesList(gr1, gr2)
names(sgrl) <- c("samp1", "samp2")
(chrGr1SampGr <- fromSamplesToChrsGRangesList(sgrl))

generateDFofSamplesPerChromosomes

description

Description

generates a dataframe where each row is a sample (1st col) and a string with its chromosomes separated by ";" (2nd col) (useful to fromSamplesToChromosomesGRangesList function).
**get_disjoint_max_win**

**Usage**

```r
generateDFofSamplesPerChromosomes(samplesChrGRList)
```

**Arguments**

`samplesChrGRList`  
a GRangesList of samples each divided by chromosome.

**Value**

a dataframe where each row is a sample (1st col) and a string with its chromosomes separated by ";" (2nd col).

---

**get_disjoint_max_win**

**Description**

find significant z score windows keeping the max value without intersections

**Usage**

```r
get_disjoint_max_win(
  z0,
  sigwin = 20,
  nmax = Inf,
  zthresh = -Inf,
  two_sided = FALSE,
  verbose = FALSE
)
```

**Arguments**

`z0`  
Matrix containing z scores with bins as rows and windows size as columns.

`sigwin`  
Integer indicating how many bins per fragment.

`nmax`  
Integer indicating the maximum number of windows to return.

`zthresh`  
Integer indicating the minimum z-score considered significant.

`two_sided`  
not used argument.

`verbose`  
verbose flag.

**Value**

a matrix of integer containing founded peaks
**Description**

given a GRangesList of samples assigns unique names to the peaks of each sample.

**Usage**
giveUniqueNamesToPeaksOverSamples(samplePeaksGRangelist)

**Arguments**
- samplePeaksGRangelist
  a GRangesList of peaks, one GRanges for each sample.

**Value**
a GRangesList of samples within renamed peaks for each element.

**Description**
given a GRanges of merged peaks assigns them new names.

**Usage**
initMergedPeaksNames(mergedGRanges)

**Arguments**
- mergedGRanges
  A GRanges object. (Typically Generated in findOverlapsOverSamples function)

**Value**
a granges of renamed peaks.
**keepRelevantChrs**

Description

subselect a list of GRanges created with cutGRangesPerChromosome returning only the relevant chromosomes GRanges.

Usage

```r
keepRelevantChrs(chrGRangesList, chr = NULL)
```

Arguments

- `chrGRangesList` where each element is a chromosome, typically created with cutGRangesPerChromosome.
- `chr` a character vector of chromosomes names of the form "chr#".

Value

the input chrGRangesList with only the relevant chromosomes.

Examples

```r
library("GenomicRanges")
gr1 <- GRanges(
  seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
  ranges=IRanges(1:10, end=10),
  strand=Rle(strand(c("-", "+", "*", "+", ":-")), c(1, 2, 2, 3, 2)),
  seqlengths=c(chr1=11, chr2=12, chr3=13))
grlc <- cutGRangesPerChromosome(gr1)
(grlChr <- keepRelevantChrs(grlc, c("chr1", "chr3")))
```

---

**rcpparma_get_disjoint_max_win**

rcpparma_get_disjoint_max_win Computes the disjoint max_win matrix.

Description

rcpparma_get_disjoint_max_win Computes the disjoint max_win matrix.
Usage

rcpparma_get_disjoint_max_win(
    z0,
    sigwin = 10L,
    zthresh = 10,
    nmax = 9999999L,
    verbose = TRUE
)

Arguments

z0 a matrix.
sigwin sigwin.
zthresh zthresh.
nmax nmax.
verbose verbose.

Value

a matrix of three columns (bin_idx, win_idx, z_val) idxs in C style.

Description

read a bam file into a bed like format. forcing UCSC format for chromosomes names.

Usage

readBamAsBed(file)

Arguments

file Character indicating path to bam file.

Value

GRanges object.

Examples

files <- list.files(system.file("extdata/bam", package="DEScan2"),
    full.names=TRUE)
gr <- readBamAsBed(files[1])
**Description**

read a bed file into a GenomicRanges like format. forcing UCSC format for chromosomes names.

**Usage**

```
readBedFile(filename, arePeaks = FALSE)
```

**Arguments**

- `filename` the bed filename.
- `arePeaks` a flag indicating if the bed file represents peaks.

**Value**

GRanges object

**Examples**

```r
bedFile <- list.files(system.file("extdata/bed",package="DEScan2"),
                      full.names=TRUE)
gr <- readBedFile(bedFile)
```

---

**Description**

Takes in input the path of bam/bed files to process and stores them in a GRangesList object, named with filePath/filenames. (for lazy people)

**Usage**

```
readFilesAsGRangesList(
    filePath,
    fileType = c("bam", "bed", "bed.zip", "narrow", "broad"),
    genomeName = NULL,
    onlyStdChrs = TRUE,
    arePeaks = TRUE,
    verbose = TRUE
)
```
### RleListToRleMatrix

**Description**

A wrapper to create a RleMatrix from a RleList object.

**Usage**

```r
class(RleListToRleMatrix(RleList, dimnames = NULL))
```

**Arguments**

- `RleList`: An RleList object with all elements of the same length.
- `dimnames`: The names for dimensions of RleMatrix (see DelayedArray pkg).

**Value**

A RleMatrix from the DelayedArray package.
Examples

```r
library("DelayedArray")
lengths <- c(3, 1, 2)
values <- c(15, 5, 20)
el1 <- S4Vectors::Rle(values=values, lengths=lengths)
el2 <- S4Vectors::Rle(values=sort(values), lengths=lengths)

rleList <- IRanges::RleList(el1, el2)
names(rleList) <- c("one", "two")
(rleMat <- RleListToRleMatrix(rleList))
```

Description

save a GRanges object as bed file.

Usage

```r
saveGRangesAsBed(
  GRanges,
  filepath = tempdir(),
  filename = tempfile(),
  force = FALSE,
  verbose = FALSE
)
```

Arguments

- `GRanges` the GRanges object.
- `filepath` the path to store the files.
- `filename` the name to give to the files.
- `force` force overwriting.
- `verbose` verbose output flag.

Value

none
Examples

```r
library("GenomicRanges")
gr <- GRanges(
    seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges=IRanges(1:10, end=10),
    strand=Rle(strand(c("-", "+", "+", "+", "-")), c(1, 2, 2, 3, 2)),
    seqlengths=c(chr1=11, chr2=12, chr3=13))
saveGRangesAsBed(GRanges=gr, filepath=tempdir(), filename=tempfile(),
    verbose=TRUE)
```

Description

save a GRanges object as tsv file.

Usage

```r
saveGRangesAsTsv(
    GRanges,        # the GRanges object.
    filepath = tempdir(),    # the path to store the files.
    filename = tempfile(),    # the name to give to the files.
    col.names = NA,    # a logical value indicating whether the column names are to be written in the file, or a character vector indicating the column names, or NA for writing column names for writing a TAB for the column names of the row names, default is NA (see write.table).
    row.names = TRUE,    # a logical value indicating whether the row names are to be written in the file, or a character vector indicating the row names (see write.table).
    sep = "\t",    # the column separator character (default is \"\t\")
    force = FALSE,    # force overwriting.
    verbose = FALSE)    # verbose output flag.
```

Arguments

- `GRanges`: the GRanges object.
- `filepath`: the path to store the files.
- `filename`: the name to give to the files.
- `col.names`: a logical value indicating whether the column names are to be written in the file, or a character vector indicating the column names, or NA for writing column names for writing a TAB for the column name of the row names, default is NA (see write.table).
- `row.names`: a logical value indicating whether the row names are to be written in the file, or a character vector indicating the row names (see write.table).
- `sep`: the column separator character (default is \"\t\")
- `force`: force overwriting.
- `verbose`: verbose output flag.
setGRGenomeInfo

Value

none

Examples

gr <- GRanges(
    seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges=IRanges(1:10, end=10),
    strand=Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)),
    seqlengths=c(chr1=11, chr2=12, chr3=13))
saveGRangesAsTsv(gr, verbose=TRUE)

setGRGenomeInfo

setGRGenomeInfo given a genome code (i.e. "mm9","mm10","hg19","hg38") retrieve the SeqInfo of that genome and assigns it to the input GRanges. Finally filters out those Infos not necessary to the GRanges.

Description

setGRGenomeInfo given a genome code (i.e. "mm9","mm10","hg19","hg38") retrieve the SeqInfo of that genome and assigns it to the input GRanges. Finally filters out those Infos not necessary to the GRanges.

Usage

setGRGenomeInfo(GRanges, genomeName = NULL, verbose = FALSE)

Arguments

GRanges a GRanges object.

genomeName a genome code (i.e. "mm9")

verbose verbose output

Value

a GRanges object with the seqinfo of the genome code

Examples

library("GenomicRanges")
gr <- GRanges(
    seqnames=Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges=IRanges(1:10, end=10),
    strand=Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)),
    seqlengths=c(chr1=11, chr2=12, chr3=13))
mm9gr <- setGRGenomeInfo(GRanges=gr, genomeName="mm9", verbose=TRUE)
Index

* internal.
  computeZ, 6
* internal
  binnedCovOnly, 4
  binToChrCoordMatRowNames, 4
  c_get_disjoint_max_win, 11
  computeCoverageMovingWindowOnChr, 5
  computeLambdaOnChr, 5
evenRunMean, 13
evenRunSum, 13
generateDFofSamplesPerChromosomes, 18
get_disjoint_max_win, 19
giveUniqueNamesToPeaksOverSamples, 20
initMergedPeaksNames, 20
rcpparma_get_disjoint_max_win, 21
binnedCoverage, 3
binnedCovOnly, 4
binToChrCoordMatRowNames, 4
bpparam, 14, 17
c_get_disjoint_max_win, 11
computeCoverageMovingWindowOnChr, 5
computeLambdaOnChr, 5
computeZ, 6
constructBedRanges, 7
countFinalRegions, 8
createGranges, 9
cutGRangesPerChromosome, 10
DEScan2, 11
divideEachSampleByChromosomes, 12
evenRunMean, 13
evenRunSum, 13
finalRegions, 14
findOverlapsOverSamples, 15
findPeaks, 16
fromSamplesToChrsGRangesList, 18
generateDFofSamplesPerChromosomes, 18
get_disjoint_max_win, 19
giveUniqueNamesToPeaksOverSamples, 20
initMergedPeaksNames, 20
keepRelevantChrs, 21
rcpparma_get_disjoint_max_win, 21
readBamAsBed, 22
readBedFile, 23
readFilesAsGRangesList, 23
RleListToRleMatrix, 24
saveGRangesAsBed, 25
saveGRangesAsTsv, 26
setGRGenomeInfo, 27
write.table, 26