Package: epiwraps (via r-universe)

June 12, 2026

Type Package

Title epiwraps: Wrappers for plotting and dealing with epigenomics data

Version 0.99.120

Date 2026-06-11

Depends R (>= 4.1.0), SummarizedExperiment, GenomicRanges, EnrichedHeatmap

Imports IRanges, data.table, BiocParallel, GenomeInfoDb, GenomicAlignments, GenomicFeatures, GenomicFiles, rtracklayer, ComplexHeatmap, S4Vectors, grid, Rsamtools, Gviz, ensembldb, AnnotationFilter, matrixStats, circlize, viridisLite, ggplot2, patchwork, edgeR, dplyr, scales, stats, pbapply, Matrix, methods

Suggests knitr, rmarkdown, BiocStyle, magick, grDevices, testthat (>= 3.0.0)

Description A set of wrappers to facilitate working with epigenomics data. Includes in particular wrappers for producing multitrack single-regions plots, enrichment heatmaps, bigwig generation, normalization, etc. The focus was put on a simple yet flexible interface.

License GPL (>= 3)

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

URL https://ethz-ins.github.io/epiwraps/

BugReports http://github.com/ETHZ-INS/epiwraps/issues

biocViews Epigenetics, Sequencing, Normalization, Visualization

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl xz-utils zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-06-12 07:21:06 UTC

RemoteUrl https://github.com/BiocStaging/epiwraps

RemoteRef HEAD

RemoteSha fd46fd6925525eda0ce56221d8ebbc3b6ff5459e

Contents

`addAssayToESE` *addAssayToESE*

Description

Adds an assay of signal matrices to an existing 'EnrichmentSE' object.

Usage

```
addAssayToESE(x, a, name = "normalized", replace = TRUE)
```

Arguments

`x`	An object of class 'EnrichmentSE', as produced by `signal2Matrix`.
`a`	The assay to add, e.g. a list of normalizedMatrix objects
`name`	The assay name under which to add 'a'.
`replace`	Logical, whether to replace any existing assay of the same name (default TRUE). If FALSE and the assay already existed, the new assay name is given a suffix.

Value

'x' with the added/updated assay.

Examples

```
# we first get an EnrichmentSE object:
data(exampleESE)
# then we will create a new assay which is simply sqrt-transformed, and add
# it back in the object
newAssay <- lapply(getSignalMatrices(exampleESE), sqrt)
exampleESE <- addAssayToESE(exampleESE, newAssay, name="sqrt")
```

annotateRegions *annotateRegions*

Description

Annotates a GRanges on the basis of an annotation object (e.g. EnsDb).

Usage

```
annotateRegions(
  regions,
  anno,
  proximal = c(2500, 1000),
  filter = AnnotationFilterList(),
  extra = list(),
  ignore.strand = TRUE,
  ...
)
```

Arguments

regions	A GRanges object
anno	An annotation object, such as an EnsDb) object, a TxDb object, or a GRanges object of a GENCODE-like gtf or the path to such a file.
proximal	The threshold(s) for TSS proximal regions. Multiple values will result in multiple class factor levels.
filter	An AnnotationFilter to filter transcripts. Only used if 'anno' is an EnsDb).
extra	An optional named list of GRanges for additional overlaps. Each list element will create an additional binary metadata column.
ignore.strand	Whether to ignore the strand for the overlap with elements of 'extra' (default TRUE).
...	Passed to overlapsAny for the overlaps with 'extra'.

Value

The sorted 'regions' object with additional annotation columns.

Examples

```
# we first create some regions we want to annotate:
regions <- as(c("chrY:2655742-2655890", "chrY:28730110-28730950"), "GRanges")
# we'll make a lightweight ensembldb annotation for the annotation:
library(ensembldb)
chrY <- system.file("chrY", package="ensembldb")
edb <- EnsDb(makeEnsemblSQLiteFromTables(path=chrY ,dbname=tempfile()))
# we run teh annotation:
```

```
regions <- annotateRegions(regions, edb)
# this adds metadata columns to the regions:
regions
table(regions$class)
```

---

bam2bw *bam2bw: create a coverage bigwig file from an alignment bam file.*

Description

bam2bw: create a coverage bigwig file from an alignment bam file.

Usage

```
bam2bw(
  bamfile,
  output_bw,
  bgbam = NULL,
  paired = NULL,
  binWidth = 20L,
  trim = 0L,
  extend = 0L,
  scaling = TRUE,
  shift = 0L,
  type = c("full", "center", "start", "end", "ends"),
  compType = c("log2ratio", "subtract", "log10ppois", "log10FE"),
  strand = c("*", "+", "-"),
  strandMode = 1,
  log1p = FALSE,
  exclude = NULL,
  includeDuplicates = TRUE,
  includeSecondary = FALSE,
  minMapq = 1L,
  minFragLength = 1L,
  maxFragLength = 5000L,
  keepSeqLvls = NULL,
  splitByChr = 3,
  pseudocount = 1L,
  localBackground = 1L,
  only = NULL,
  zeroCap = TRUE,
  forceSeqlevelsStyle = NULL,
  verbose = TRUE,
  binSummarization = c("mean", "max", "min"),
  ...
)
```

Arguments

bamfile	The path to the signal bam file.
output_bw	The path to the output bigwig file
bgbam	The optional path to the control/input bam file to compute relative signal (see the ‘compType‘ argument).
paired	Logical; whether to consider fragments instead of reads for paired-end data. For spliced (e.g. RNA-seq) data, paired should always be set to FALSE.
binWidth	The window size. A lower value (min 1) means a higher resolution, but larger file size.
trim	The amount by which to trim reads or fragments. Can either be a single integer, which will be removed at both ends of the fragment, or an integer of length=2 indicating the amount to trim at the beginning and end of the read/fragments. This is applied before ‘type‘ and ‘extension‘.
extend	The amount *by* which to extend single-end reads (e.g. fragment length minus read length). If ‘paired=TRUE‘ and ‘type‘ is either ’ends’ or ’center’, then the extension will be applied after taking the (shifted) fragment ends or centers, resulting in ranges of width equal to ‘extend‘.
scaling	Either TRUE (performs Count Per Million scaling), FALSE (no scaling), or a numeric value by which the signal will be divided. If ‘bgbam‘ is given and ‘scaling=TRUE‘, the background will be scaled to the main signal.
shift	Shift (from 3’ to 5’) by which reads/fragments will be shifted. If ‘shift‘ is an integer vector of length 2, the first value will represent the shift for the positive strand, and the second for the negative strand.
type	Type of the coverage to compile. Either full (full read/fragment), start (count read/fragment start locations), end, center, or ’ends’ (both ends of the read/fragment).
compType	The type of relative signal to compute (ignored if ‘bgbam‘ isn’t given). Can either be ’log2ratio’ (log2-foldchange between signals), ’subtract’ (difference), ’log10ppois’ (rounded -log10 poisson p-value), ’log10FE’ (log10 fold-enrichment). For fold-based signals, ‘pseudocount‘ will be added before computing the ratio. By default negative values are capped (see ‘zeroCap‘).
strand	Strand(s) to capture (any by default).
strandMode	The strandMode of the data (whether the strand is given by the first or second mate, which depends on the library prep protocol). See strandMode for more information. This parameter has no effect unless one of the ‘strand‘, ‘extend‘ parameters or a strand-specific ‘shift‘ are used.
log1p	Whether to log-transform (‘log(x+1)‘) the (scaled) signal. Ignored when the signal is relative to a background.
exclude	An optional GRanges of regions for which overlapping reads should be excluded.
includeDuplicates	Logical, whether to include reads flagged as duplicates.
includeSecondary	Logical; whether to include secondary alignments

minMapq	Minimum mapping quality (1 to 255)
minFragLength	Minimum fragment length (ignored if 'paired=FALSE')
maxFragLength	Maximum fragment length (ignored if 'paired=FALSE')
keepSeqLvls	An optional vector of seqLevels (i.e. chromosomes) to include.
splitByChr	Whether to process chromosomes separately, and if so by how many chunks. The should not affect the output, and is simply slightly slower and consumes less memory. Can be a logical value (in which case each chromosome is processed separately), but we instead recommend giving a positive integer indicating the number of chunks.
pseudocount	The count to be added before fold-enrichment calculation between signals. Ignored if 'compType="subtract"'.
localBackground	A (vector of) number of windows around each tile/position for which a local background will be calculated. The background used will be the maximum of the one at the given position or of the mean of each of the local backgrounds.
only	An optional GRanges of regions for which overlapping reads should be included. If set, all other reads are discarded.
zeroCap	Logical; whether to cap values below zero to zero when producing relative signals.
forceSeqlevelsStyle	If specified, forces the use of the specified seqlevel style for the output bigwig. Can take any value accepted by 'seqlevelsStyle'.
verbose	Logical; whether to print progress messages
binSummarization	The method to summarize nucleotides into each bin, either "mean" (default), "min" or "max".
...	Passed to 'ScanBamParam'

Details

* For single-end ChIPseq data, extend reads to the expected fragment size using the 'extend' argument (i.e. setting extend to the read length plus the mean expected fragment size). * For ATAC-seq data, when interested in high-resolution profiling of the Tn5 insertion sites, it is customary to shift reads based on their strand, i.e. using 'shift=c(4L,-5L)' and 'type="start"'. With paired-end data, we recommend instead using both ends with 'type="ends"'. In this case, the shifting should occur from both ends inwards, which is best accomplished with 'trim=4L' (rather than using the 'shift' argument). * The implementation involves reading the reads before processing, which can be quite memory-hungry. To avoid this the files are read by chunks (composed of chromosomes of balanced sizes), controlled by the 'splitByChr' argument. This reduces memory usage but also creates overhead which reduces the speed. In contexts where the file is small or memory isn't a problem, using 'splitByChr=FALSE' will achieve the best speed. Otherwise, increasing 'splitByChr' will decrease the memory footprint. * Consider restricting the read quality using 'includeDuplicates=FALSE' and 'minMapq=20' for example.

Value

The bigwig filepath. Alternatively, if ‘output_bw=NA_character_‘, the coverage data is not written to file but returned.

Author(s)

Pierre-Luc Germain

Examples

```
# get an example bam file
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
# create bigwig
bam2bw(bam, "output.bw")
```

`bamChrChunkApply` *bamChrChunkApply*

Description

Runs a function on reads/fragments from chunks of (chromosomes) of an indexed bam file. This is especially used by other functions to avoid loading all alignments into memory, or to parallelize reads processing.

Usage

```
bamChrChunkApply(
  x,
  FUN,
  paired = FALSE,
  keepSeqLvls = NULL,
  nChunks = 4,
  strandMode = 2,
  flgs = scanBamFlag(),
  exclude = NULL,
  mapqFilter = NA_integer_,
  progress = TRUE,
  BPPARAM = NULL,
  ...
)
```

Arguments

`x`	A bam file.
`FUN`	The function to be run, the first argument of which should be a ‘GRanges‘
`paired`	Logical; whether to consider the reads as paired (fragments, rather than reads, will be returned)

keepSeqLvls	An optional vector of seqLevels to keep
nChunks	The number of chunks to use (higher will use less memory but increase overhead)
strandMode	Strandmode for paired data (see readGAlignmentPairs).
flgs	'scanBamFlag' for filtering the reads
exclude	An optional GRanges of regions for which overlapping reads should be excluded.
mapqFilter	Integer of the minimum mapping quality for reads to be included.
progress	Logical; whether to show a progress bar.
BPPARAM	A 'BiocParallel' parameter object for multithreading. Note that if used, memory usage will be high; in this context we recommend a high 'nChunks'.
...	Passed to 'FUN'

Value

A list of whatever 'FUN' returns

Examples

```
# as an example we'll use the function to obtain fragment sizes:
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
fragLen <- bamChrChunkApply(bam, paired=TRUE, FUN=function(x) width(x))
quantile(unlist(fragLen))
```

---

breakStrings *breakStrings*

Description

breaks a string of long-enough words (or vector thereof) into two lines.

Usage

```
breakStrings(x, minSizeForBreak = 20, lb = "\n")
```

Arguments

x	A character (or factor) vector.
minSizeForBreak	The minimum number of characters to be broken into two lines.
lb	The separation character (e.g. line break).

Value

A character vector of length=length(x).

Examples

```
breakStrings("this is too long for practical purposes")
```

`callPeaks` *callPeaks*

Description

This is a native R peak caller loosely based on the general MACS2/3 strategy (Zhang et al., Genome Biology 2008). It can work from BAM files, fragment files, or from coverage tracks. Note that this was developed for teaching purposes, and the results are decent, but might not be state-of-the-art. The function also requires much more memory than MACS (see details).

Usage

```
callPeaks(
  bam,
  ctrl = NULL,
  paired,
  type = c("narrow", "broad"),
  fragLength = 200L,
  globalNullH = FALSE,
  gsize = NULL,
  blacklist = NULL,
  binSize = 10L,
  nChunks = 8L,
  flags = scanBamFlag(isDuplicate = FALSE),
  minPeakCount = 5L,
  minFoldEnr = 1.4,
  pthres = 10^-5,
  maxSize = NULL,
  bgWindow = c(1, 5, 10) * 1000,
  pseudoCount = 0.5,
  useStrand = !paired,
  outFormat = c("narrowPeak", "custom"),
  verbose = TRUE,
  ...
)
```

Arguments

`bam`	A signal bam file (which should be accompanied by an index file). Alternatively, a TABIX-indexed fragment file, or an RleList object.
`ctrl`	An optional path to a control bam file. Alternatively, an RleList object.
`paired`	Logical, whether the reads are paired.
`type`	The type of peaks to identify ('narrow' or 'broad').

fragLength	Fragment length. Ignored if 'paired=TRUE'. If 'useStrand=TRUE' (default), this is only used for the initial candidate region identification, and sizes are adjusted after, so it doesn't need to be very precise.
globalNullH	Logical; whether to use a global expectation, rather than the local background (default), in the absence of a control.
gsize	The mappable genome size. Ignored unless 'globalNullH=TRUE'. Can also be a species acronym in 'hs', 'mm', 'dm', and 'ce'.
blacklist	An optional 'GRanges' of regions to be excluded (or the path to such a file). Since the blacklisted regions are removed from both the signal and control peaks, this also has an important impact on the empirical FDR (when 'ctrl' is given).
binSize	Binsize used to estimate peak shift.
nChunks	The number of processing chunks. Increasing this will reduce memory usage.
flags	An optional scanBamFlag object to filter the reads. By default, reads flagged as optical duplicates are excluded.
minPeakCount	The minimum summit count for a region to be considered. Decreasing this can substantially increase the running time.
minFoldEnr	The minimum fold-enrichment for a region to be considered. Decreasing this can substantially increase the running time.
pthres	The p-value threshold to use.
maxSize	The loose maximum size of a peak. This is the size above which the method will attempt to break up peaks into smaller ones. The default is 5000 for 'type="broad"', and will depend on the estimated fragment size for narrow peak calls.
bgWindow	The windows to consider (in addition to the peak itself) for local background.
pseudoCount	The pseudocount to use when computing logFC.
useStrand	Logical; whether to use strand information to better estimate the peak boundaries with single-end data.
outFormat	The output format ('custom' or 'narrowPeak')
verbose	Logical; whether to output progress messages
...	Passed to bamChrChunkApply

Details

Unless 'globalNullH=TRUE', this function uses MACS' local lambda (defined by 'bgWindow'). A major difference with MACS2/3 is that, rather than using sliding windows, if works on the running list encoding of the coverage(s). As a consequence, significance is estimated based on the peak's maximum coverage, which is very similar for narrow peaks, but very different for broad peaks, which will not produce astronomical p-values as is the case with MACS. Peak sizes will also differ and the distribution tends to be narrower (i.e. fewer small and very large peaks).

Because FDR calculation is so tricky for peak calling, the function uses an uncorrected p-value threshold for its output. If desired, users can further filter based on the FDR. We recommend including a 'blacklist' for FDR estimation.

On a single thread, the function is nearly as fast as MACS2/3, but uses much more memory, chiefly due to the fact that reads are read into memory in large chunks. On a single thread, memory usage

will be roughly the sum of the filesize of your bam files (i.e. IP + input, if using an input) divided by 'nChunks-1'.

The function uses bamChrChunkApply to obtain the coverages, and can accept any argument of that function. This means for instance that the 'mapqFilter' argument can be used to restrict the reads used, or a 'BPPARAM' argument to enable multi-threading (beware of memory consumption!).

To export as a narrowPeak file, see exportNarrowPeaks.

Value

A 'GRanges', by default following the narrowPeak format specification.

Examples

```
# we use the example bam file from the Rsamtools package:
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
peaks <- callPeaks(bam, paired=TRUE)
```

---

clusterSignalMatrices *clusterSignalMatrices: clusters the regions of a (set of) signal matrices.*

Description

clusterSignalMatrices: clusters the regions of a (set of) signal matrices.

Usage

```
clusterSignalMatrices(
  ml,
  k,
  scaleRows = FALSE,
  scaleCols = FALSE,
  use = c("enrich", "full", "max", "center"),
  by = rep(1L, length(ml)),
  assay = 1L,
  trim = c(0.05, 0.95),
  nstart = 3,
  ...
)
```

Arguments

ml	A named list of signal matrices or an EnrichmentSE object as produced by signal2Matrix
k	The number of clusters to generate
scaleRows	Logical; whether to scale rows for clustering

scaleCols	Logical; whether to scale columns (i.e. signals/samples)
use	What values to use for clustering. By default, uses enriched_score. Other options are 'full' (uses the full signal for clustering), 'max' (uses the maximum value in the region), or 'center' (use the value at the center of the region).
by	Optional factor/character/integer vector of the same length as 'ml'. When scaling rows, this can be used to indicate which rows should be scaled together.
assay	Assay to use (ignored unless 'ml' is an ESE object)
trim	Values to trim (applied individually for each signal matrix)
nstart	Number of starts for k-means clustering
...	Passed to 'kmeans'

Value

If 'k' is of length 1, a vector of cluster labels, corresponding to the rows of 'ml'. If 'length(k)>1', a list of two data.frames containing: 1) the cluster labels at the different resolutions, and 2) the variance explained by clusters at each resolution.

Examples

```
data(exampleESE)
rowData(exampleESE)$cluster <- clusterSignalMatrices(exampleESE, 3)
# we could plot the data clustered:
plotEnrichedHeatmaps(exampleESE, row_split="cluster")
# we could also try with different values of k:
cl <- clusterSignalMatrices(exampleESE, 2:5)
cl$varExplained
```

---

colOverlaps *colOverlaps*

Description

Computes pairwise overlap metric between columns

Usage

```
colOverlaps(x, y = NULL, metric = c("overlap", "jaccard", "overlapCoef"))
```

Arguments

x	A logical matrix.
y	An optional nother logical matrix or vector. If NULL (default), overlaps are computed between columns of 'x'.
metric	Either 'overlap', 'jaccard', or 'overlapCoef'.

Value

A matrix of pairwise metric values.

Examples

```
m <- matrix(sample(c(TRUE,FALSE),12,replace=TRUE), nrow=4)
colOverlaps(m, metric="jaccard")
```

`EnrichmentSE-class` *EnrichmentSE class and constructor*

Description

The 'EnrichmentSE' class is a container for epigenomic enrichment data, extending the `RangedSummarizedExperiment` class.

Usage

```
EnrichmentSE(assays, rowRanges = NULL, ...)

## S4 method for signature 'EnrichmentSE,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'EnrichmentSE'
show(object)

## S4 method for signature 'EnrichmentSE'
score(x, ...)
```

Arguments

`assays`	A list of matrices or 'normalizedMatrix' objects.
`rowRanges`	A `GRanges` object.
`...`	Arguments passed to `getSignalMatrices`.
`x`	An object of class 'EnrichmentSE', as produced by `signal2Matrix`.
`i, j`	Indices for subsetting.
`drop`	Logical; whether to drop dimensions.
`object`	An 'EnrichmentSE' object.

Value

An 'EnrichmentSE' object.

Functions

- `x[i`: Subset method for EnrichmentSE
- `show(EnrichmentSE)`: Show method for EnrichmentSE
- `score(EnrichmentSE)`: Access the score (first assay)

`estimateFragSize` *estimateFragSize*

Description

estimateFragSize

Usage

```
estimateFragSize(
  bam,
  ctrl = NULL,
  binSize = 10L,
  mfold = c(10, 50),
  minSummitCount = 8L,
  useSeqLevels = NULL,
  maxSize = 2500L,
  priorLength = 200L,
  blacklist = NULL,
  ret = c("mode", "median", "mean", "tables", "plots", "distances"),
  BPPARAM = SerialParam()
)
```

Arguments

`bam`	The path to one or more bam files
`ctrl`	Optional path to a control bam file (if 'length(bam)>1', the same control will be used for all).
`binSize`	Bin size. The precision of the reported fragment size is necessary lower than this. A higher bin size will improve the summit identification in low-coverage regions. We recommend leaving the default value.
`mfold`	The range of fold-enrichment over the control (if 'ctrl' provided) or of coverages for the identification of regions based on which distance will be estimated.
`minSummitCount`	The minimum read count for a summit to be considered.
`useSeqLevels`	An optional vector of seqLevels in which to conduct the analysis.
`maxSize`	The maximum size of regions to be used
`priorLength`	The prior fragment length (use for read extension to identify enriched regions)
`blacklist`	Optional 'GRanges' of blacklisted regions to be excluded.

ret	The type of return, either a 'table' of pairs of summits and their properties, or a 'plot', or the median/mean/mode of the distance distribution.
BPPARAM	A 'BiocParallel' parameter object for multithreading. Only used if multiple files are given in 'bam'.

Value

By default, the estimated (mode) fragment length(s), but see the 'ret' argument

Examples

```
# get an example bam file
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
suppressWarnings(estimateFragSize(bam))
```

---

exampleDNAme *Example DNAme data on some active gene bodies*

Description

A GRanges object (called 'geneBodies') containing the coordinates of some gene bodies on chr9 of hg38, as well as a GPos object (called 'exampleDNAme') containing methylation percentages at CpGs in those regions. Taken from WG bisulfite sequencing data from A549, from ENCODE accession ID ENCFF948WVD.

Value

a named character vector of length 2.

---

exampleESE *Example EnrichmentSE object*

Description

Small sample signal from ENCODE ChIP-seq for H3K27ac, H3K4me3 and p300, around some p300 binding sites and TSS in mESC.

Value

a named character vector of length 1.

exportNarrowPeaks *exportNarrowPeaks*

Description

exportNarrowPeaks

Usage

```
exportNarrowPeaks(p, file)
```

Arguments

p	A GRanges, as produced by ‘callPeaks‘
file	The path to the file where to save

Value

Invisible file path.

Examples

```
# call some peaks:
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
peaks <- callPeaks(bam, paired=TRUE)
# save them:
filepath <- tempfile(fileext="narrowPeak")
exportNarrowPeaks(peaks, filepath)
```

formatGenomicDist *formatGenomicDist*

Description

formatGenomicDist

Usage

```
formatGenomicDist(
  e,
  allowFraction = TRUE,
  sameUnits = TRUE,
  head0 = TRUE,
  units = c(mb = 1e+06, kb = 1000, bp = 1)
)
```

Arguments

e	An integer vector of genomic sizes/distances to be formatted
allowFraction	Whether to allow decimals; can be either logical or a number of decimals allowed (defaults to TRUE / 1)
sameUnits	Logical; whether all values should get the same unit.
head0	Logical; whether to keep heading zero
units	The units to use.

Value

A character vector.

Examples

```
formatGenomicDist(1235000)
```

---

frag2bw *frag2bw*

Description

Creates a coverage bigwig file from a Tabix-indexed fragment file.

Usage

```
frag2bw(
  tabixFile,
  output_bw,
  binWidth = 20L,
  scaling = TRUE,
  type = c("full", "center", "start", "end", "ends"),
  barcodes = NULL,
  strand = c("*", "+", "-"),
  shift = 0L,
  log1p = FALSE,
  exclude = NULL,
  minFragLength = 1L,
  maxFragLength = 5000L,
  keepSeqLvls = NULL,
  useScore = FALSE,
  forceSeqlevelsStyle = NULL,
  only = NULL,
  format = "bed",
  binSummarization = c("mean", "max", "min"),
  verbose = TRUE
)
```

Arguments

tabixFile	The path to a tabix-indexed bam file, or a TabixFile object.
output_bw	The path to the output bigwig file
binWidth	The window size. A lower value (min 1) means a higher resolution, but larger file size.
scaling	Either TRUE (performs Count Per Million scaling), FALSE (no scaling), or a numeric value by which the signal will be divided. If 'bgbam' is given and 'scaling=TRUE', the background will be scaled to the main signal.
type	Type of the coverage to compile. Either full (full read/fragment), start (count read/fragment start locations), end, center, or 'ends' (both ends of the read/fragment).
barcodes	An optional list of barcodes to use (assuming that the file contains the column)
strand	Strand(s) to capture (any by default).
shift	Shift (from 3' to 5') by which reads/fragments will be shifted. If 'shift' is an integer vector of length 2, the first value will represent the shift for the positive strand, and the second for the negative strand.
log1p	Whether to log-transform ('log(x+1)') the (scaled) signal.
exclude	An optional GRanges of regions for which overlapping reads should be excluded.
minFragLength	Minimum fragment length (ignored if 'paired=FALSE')
maxFragLength	Maximum fragment length (ignored if 'paired=FALSE')
keepSeqLvls	An optional vector of seqLevels (i.e. chromosomes) to include.
useScore	Whether to use the score column (if any) as coverage weights.
forceSeqlevelsStyle	If specified, forces the use of the specified seqlevel style for the output bigwig. Can take any value accepted by 'seqlevelsStyle'.
only	An optional GRanges of regions for which overlapping reads should be included. If set, all other reads are discarded.
format	The format of the fragment file.
binSummarization	The method to summarize nucleotides into each bin, either "mean" (default), "min" or "max".
verbose	Logical; whether to print progress messages

Value

The bigwig filepath. Alternatively, if 'output_bw=NA_character_', the coverage data is not written to file but returned.

Examples

```
# we first create a fake tabix file:
library(GenomicRanges)
library(rtracklayer)
```

```
reads <- GRanges(rep(c("1","2"), c(5,2)),
                 IRanges(5000+10*1:7, width=100))
bedf <- tempfile(fileext=".bed")
rtracklayer::export.bed(reads, bedf)
bedf <- Rsamtools::bgzip(bedf)
Rsamtools::indexTabix(bedf, format="bed")
# convert to bigwig
frag2bw(bedf, tempfile(fileext=".bw"))
```

`fragSizesDist` *fragSizesDist*

Description

fragSizesDist

Usage

```
fragSizesDist(
  x,
  what = 10000,
  flags = scanBamFlag(isProperPair = TRUE),
  BPPARAM = SerialParam(),
  returnPlot = TRUE
)
```

Arguments

`x`	A (named) vector of paths to bam files.
`what`	Either a positive integer (length 1) indicating how many reads to randomly sample, or a character vector (of length 1) indicating which chromosome to read.
`flags`	A 'scanBamFlag' object (see ScanBamParam)
`BPPARAM`	A BiocParallel BPPARAM object for multithreading.
`returnPlot`	Logical; whether to return a plot.

Value

If 'returnPlot=TRUE', returns a ggplot object, otherwise a data.frame of fragment lengths.

Examples

```
# example bam file:
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
fragSizesDist(bam, what=100)
```

getCovStats *getCovStats*

Description

Assembles read distribution statistics from a set of bigwig files based on random windows.

Usage

```
getCovStats(
  x,
  binSize = 1000,
  nbBins = 10000,
  exclude = NULL,
  canonical.chr = TRUE,
  maxCovQuant = 0.999,
  BPPARAM = SerialParam()
)
```

Arguments

x	A (named) vector of paths to bigwig files (all from the same genome)
binSize	The size of bins
nbBins	The approximate number of random bins. More bins gives more accurate readouts but take longer to read and compute.
exclude	Region to exclude
canonical.chr	Logical; whether to restrict the sampling to standard chromosomes.
maxCovQuant	The quantile to use as maximum coverage (default 0.999)
BPPARAM	BioParallel BPPARAM for multithreading across files.

Value

A named list of QC tables

Examples

```
# we use an example bigwig file
bwf <- system.file("extdata/example_atac.bw", package="epiwraps")
# because most of the file is empty, we'll exclude some of the ranges
cs <- getCovStats(bwf, exclude=GRanges("1", IRanges(1, 4300000)))
plotCovStats(cs)
```

getEmpiricalFDR *getEmpiricalFDR*

Description

Computes the FDR as the proportion of negative peaks with a more extreme p-value, eventually extrapolating and preserving the ranking.

Usage

```
getEmpiricalFDR(log10p, pneg, n = length(log10p) * 10)
```

Arguments

log10p	-log10 p-values of candidate peaks
pneg	-log10 p-values of negative peaks
n	The number of hypotheses (used when the empirical FDR is zero)

Value

A data.frame with the empirical FDR and a smoothed -log10(FDR)

getNormFactors *getNormFactors : estimate normalization factors from genomic signal files*

Description

Estimates normalization factors for a set of samples (i.e. bam/bigwig files).

Usage

```
getNormFactors(
  x,
 method = c("background", "SES", "enriched", "top", "MAnorm", "S3norm", "2cLinear"),
  wsize = 10L,
  nwind = 20000L,
  peaks = NULL,
  trim = 0.02,
  useSeqLevels = NULL,
  paired = NULL,
  ...,
  verbose = TRUE
)

bwNormFactors(x, ...)
```

Arguments

x	A vector of paths to bigwig files, to bam files, or alternatively a list of coverages in RleList format. (Mixed formats are not supported)
method	Normalization method (see details below).
wsize	The size of the random windows. If any of the bigwig files records point events (e.g. insertions) at high resolution (e.g. nucleotide), use a lot (e.g. >10k) of small windows (e.g. 'wsize=10'), as per default settings. Otherwise the process can be lightened by using fewer bigger windows.
nwind	The number of random windows
peaks	A list of peaks (GRanges) for each experiment in 'x', or a vector of paths to such files, or a single GRanges of unified peaks to use (e.g. for top/MAnorm).
trim	Amount of trimming when calculating means.
useSeqLevels	An optional vector of seqLevels (i.e. chromosomes) to include.
paired	Logical; whether the reads are paired-end. Ignored unless 'x' are paths to bam files.
...	Passed to internal functions.
verbose	Logical; whether to print progress messages.

Details

The function computes per-sample (bigwig or bam file) normalization factors using one of the following methods: * The 'background' or 'SES' normalization method (they are synonyms here) (Diaz et al., Stat Appl Gen et Mol Biol, 2012) assumes that the background noise should on average be the same across experiments, an assumption that works well in practice when there are not very large differences in signal-to-noise ratio. The implementation uses the trimmed mean number of reads in random windows with non-zero counts. * The 'MAnorm' approach (Shao et al., Genome Biology 2012) assumes that regions that are commonly enriched (i.e. common peaks) in two experiments should on average have the same signal in the two experiments. If the data is strictly positive (e.g. counts or CPMs), TMM normalization (Robinson et al., Genome Biology 2010) is then employed on the common peaks. When done on more than two samples, the sample with the highest number of overlaps with other samples is used as a reference. * The 'enriched' approach assumes that enriched regions are on average similarly enriched across samples. Contrarily to 'MAnorm', these regions do not need to be in common across samples/experiments. Note that trimming via the 'trim' parameter is performed before calculating means. * The 'top' approach assumes that the maximum enrichment (after trimming according to the 'trim' parameter) in peaks is the same across samples/experiments. * The 'S3norm' (Xiang et al., NAR 2020) and '2cLinear' methods try to normalize both enrichment and background simultaneously. S3norm does this in a log-linear fashion (as in the publication), while '2cLinear' does it on the original scale.

Several of the methods rely on random regions to estimate background levels. This means that they will not be entirely deterministic, although normally quite reproducible with experiments that are not very sparse. For fully reproducible results, be sure to set the random seed.

When the data is very sparse (e.g. low sequencing depth, or compiling single nucleotide hits rather than fragment coverage), it might be necessary to increase the 'wsize' and 'nwind' to get more robust estimates.

Value

A vector of normalization factors, or for the 'S3norm' and '2cLinear' methods, a numeric matrix with a number of rows equal to the length of 'x', and two columns indicating the alpha and beta terms.

A vector of normalization factors or, for methods 'S3norm' and '2cLinear', a matrix of per-sample normalization parameters.

Functions

- `bwNormFactors()`: deprecated in favor of getNormFactors

Examples

```
# we get an example bigwig file, and use it twice:
bw <- system.file("extdata/example_atac.bw", package="epiwraps")
bw <- c(sample1=bw, sample2=bw)
# we indicate to use only chr1, because it's the only one in the example file
getNormFactors(bw, useSeqLevels="1")
```

`getSignalMatrices` *getSignalMatrices*

Description

Extracts a list of signal matrices from an EnrichmentSE object.

Usage

```
getSignalMatrices(x, assay = 1L)
```

Arguments

`x`	An object of class 'EnrichmentSE', as produced by `signal2Matrix`.
`assay`	The assay to extract (defaults to the first assay).

Value

A list of normalizedMatrix objects.

Examples

```
# we first get an EnrichmentSE object:
data(exampleESE)
# then we can extract the list of signal matrices:
sm <- getSignalMatrices(exampleESE)
```

ggSignalTracks *ggSignalTracks: Plot genomic signal tracks with ggplot2*

Description

ggSignalTracks: Plot genomic signal tracks with ggplot2

Usage

```
ggSignalTracks(
  tracks,
  region,
  ensdb = NULL,
  colors = "darkblue",
  transcripts = c("full", "collapsed", "none"),
  aggregation = c("mean+heatmap", "mean", "heatmap", "heatmap+mean"),
  extend = 0,
  showSE = FALSE,
  nbins = 1000,
  heatmap.palette = c("white", "blue", "black"),
  binSummFn = c("mean", "max"),
  sameLimits = TRUE,
  gene_label = "symbol",
  trans = c("none", "sqrt", "log1p"),
  gene_color = "black",
  baseTextSize = 9,
  xAxis = TRUE,
  coverage.linewidth = 0.2,
  verbose = TRUE
)
```

Arguments

tracks	A named list or named character vector, where each element is the path to one or multiple bigwig files (nesting indicates grouping).
region	The region to plot, provided either as a GRanges or character. If 'ensdb' is given, 'region' can also be a gene name, which will be looked up.
ensdb	An optional EnsDb object form which to grab transcripts. A 'TxDb' object should also be supported.
colors	A vector of colors for each element of 'tracks' (nested elements have the same colors) which will be use to color the coverage profiles. Colors will be recycled if necessary.
transcripts	Either 'full' (full transcripts plotted), 'collapsed' (to genes), or 'none'.
aggregation	How to aggregate/show nested tracks. Either 'mean', 'heatmap' (no aggregation), or 'mean+heatmap' (both, default).

extend	A numeric value indicating how much to extend beyond the 'region'. If greater than 1, this indicates the number of nucleotides to add in both directions. If smaller than or equal to 1, this indicates the proportion of the region to add on both sides. Default 0.
showSE	Logical; whether to show the standard error on the coverage tracks of aggregated data.
nbins	The number of bins in which to divide the region.
heatmap.palette	A character vector specifying the colors for the heatmap. If of length 1, is assumed to indicate the RColorBrewer palette to use. If of length>1, the colors will be used with scale_fill_gradientn.
binSummFn	How to summarize date within a display bin. Either 'mean' (default) or 'max'.
sameLimits	Logical; should the tracks have the same y-axis limits (and same color scale for heatmaps)?
gene_label	What labels to print for genes. Either "symbol", "gene_id", "tx_name", or NULL.
trans	Optional transformation of the data, either 'none' (default), 'sqrt', or 'log1p'.
gene_color	The genes' color.
baseTextSize	The base plotting text size.
xAxis	Logical; whether to plot the xAxis in the bottom panel.
coverage.linewidth	Line width of the coverage plots (above ribbons).
verbose	Logical; whether to print progress messages.

Value

A list of ggplot objects.

Examples

```
# we create dummy data
bw1 <- tempfile(fileext=".bw")
cov1 <- GRanges("chr1", IRanges(1L+round(c(500+rnorm(15, sd=20),
                                           1000*runif(20))), width=30))
bw2 <- tempfile(fileext=".bw")
cov2 <- GRanges("chr1", IRanges(1L+abs(round(c(490+rnorm(15, sd=30),
                                               1000*runif(20)))), width=30))
seqlengths(cov1) <- seqlengths(cov2) <- c("chr1"=1500)
rtracklayer::export.bw(coverage(cov1), bw1)
rtracklayer::export.bw(coverage(cov2), bw2)
# then we create the ggplots, and plot them:
pl <- ggSignalTracks(list(group=c(rep1=bw1, rep2=bw2)), region="chr1:1-1030",
                     aggregation="heatmap+mean")
patchwork::wrap_plots(pl, ncol=1, heights=c(2,1))
```

`importBedlike` *importBedlike*

Description

Imports a bed-like file as a GRanges object. Uses ‘rtracklayer‘ import functions if possible, and falls back onto an import that’s not format-committed otherwise.

Usage

```
importBedlike(x, ...)
```

Arguments

`x`	The path to a bed or bed-like file (can be gzipped)
`...`	passed to `fread`

Value

A ‘GRanges‘ object

Examples

```
# example bed file:
filepath <- system.file("extdata/example_peaks.bed",
                        package="epiwraps")
b <- importBedlike(filepath)
```

`inject` *Inject (insert) values at positions in a vector*

Description

Inject (insert) values at positions in a vector

Usage

```
inject(what, inWhat, at)
```

Arguments

`what`	The values to inject
`inWhat`	The vector in which to inject them
`at`	The positions in the vector *after* which to inject the values.

Value

A vector of same mode at ‘inWhat‘, and of length equal to the sum of the lengths of ‘what‘ and ‘inWhat‘.

Examples

```
inWhat <- 1:10
inject(c(21,22,23), inWhat, at=as.integer(c(0,5,10)))
```

`meltSignals` *meltSignals*

Description

Aggregates and melts a list of signal matrices, for plotting (with ggplot).

Usage

```
meltSignals(ml, fun = NULL, splitBy = NULL, trim = 0.98, assay = 1L)
```

Arguments

`ml`	A named list of signal matrices or an EnrichmentSE object as produced by `signal2Matrix`
`fun`	An optional custom aggregation function (or named list thereof).
`splitBy`	A vector of values (factor or character of length equal to ‘nrow(ml)‘) by which to split the aggregation. Can also be the name of a column of ‘rowData(ml)‘.
`trim`	The quantile above which to trim values. If a numeric vector of length 2, will be used as lower and upper quantiles beyond which to trim.
`assay`	Assay to use (ignored unless ‘ml‘ is an ESE object), defaults to the first assay.

Value

A data.frame.

Examples

```
# we first get an EnrichmentSE object:
data(exampleESE)
# we extract the means per position:
d <- meltSignals(exampleESE)
head(d)
## we could then plot for instance using ggplot:
# ggplot(d, aes(position, mean, colour=sample)) + geom_line(linewidth=1.2)
```

mergeSignalMatrices *mergeSignalMatrices: aggregates two or more signal matrices.*

Description

mergeSignalMatrices: aggregates two or more signal matrices.

Usage

```
mergeSignalMatrices(ml, aggregation = c("mean", "sum", "median"), assay = 1L)
```

Arguments

ml	A named list of signal matrices or an EnrichmentSE object as produced by signal2Matrix
aggregation	The method to aggregate matrices
assay	Assay to use (ignored unless 'ml' is an ESE object), defaults to the first assay.

Value

A single 'normalizedMatrix' object.

Examples

```
# we first get an EnrichmentSE object:
data(exampleESE)
# we merge the two tracks:
merged <- mergeSignalMatrices(exampleESE)
# we could then plot the merge (not run):
# plotEnrichedHeatmaps(merged)
```

ml2ESE *Creates an EnrichmentSE from a list of normalizedMatrix objects*

Description

Creates an EnrichmentSE from a list of normalizedMatrix objects

Usage

```
ml2ESE(ml, rowRanges, assayName = "input", addScore = FALSE, ...)
```

Arguments

ml	A named list of normalizedMatrix objects with corresponding rows.
rowRanges	An optional GRanges object corresponding to the rows of each object of 'ml'.
assayName	The name of the assay, defaults to 'input'
addScore	Logical; whether to add an enriched_score assay.
...	Passed to the SummarizedExperiment constructor.

Value

An 'EnrichedSE' object, inheriting from a RangedSummarizedExperiment.

Examples

```
# for an example we first need a list of signal matrices. To this end,
# we first fetch the path to the example bigwig file:
bw <- system.file("extdata/example_atac.bw", package="epiwraps")
# we load example regions:
regions <- rtracklayer::import(system.file("extdata/example_peaks.bed",
                                           package="epiwraps"))
# we obtain the matrix of the signal around the regions, indicating that we
# want the output as a list of signal matrices:
m <- signal2Matrix(bw, regions, ret="list")
# we can then transform this into an EnrichmentSE object:
m <- ml2ESE(m, rowRanges=regions)
```

---

peakCountsFromBAM *peakCountsFromBAM*

Description

Creates a SummarizedExperiment of fragment (or insertion) counts from bam files that overlap given regions.

Usage

```
peakCountsFromBAM(
  bam_files,
  regions,
  paired,
  extend = 0L,
  shift = 0L,
  type = c("full", "center", "start", "end", "ends"),
  ov.type = "any",
  maxgap = -1L,
  minoverlap = 1L,
  ignore.strand = TRUE,
```

```
  strandMode = 1,
  includeDuplicates = TRUE,
  includeSecondary = FALSE,
  minMapq = 1L,
  minFragLength = 1L,
  maxFragLength = 5000L,
  splitByChr = NULL,
  randomAcc = FALSE,
  getMedianFragLength = FALSE,
  BPPARAM = SerialParam(),
  verbose = TRUE
)
```

Arguments

bam_files	A vector of paths to the bam files.
regions	A 'GRanges' of regions in which to counts.
paired	Logical; whether the data is paired (assumed unpaired by default). Use 'paired="auto"' for automatic detection using the first bam file.
extend	The amount *by* which to extend single-end reads (e.g. fragment length minus read length). If 'paired=TRUE' and 'type' is either 'ends' or 'center', then the extension will be applied after taking the (shifted) fragment ends or centers, resulting in ranges of width equal to 'extend'.
shift	Shift (from 3' to 5') by which reads/fragments will be shifted. If 'shift' is an integer vector of length 2, the first value will represent the shift for the positive strand, and the second for the negative strand.
type	Type of the coverage to compile. Either full (full read/fragment), start (count read/fragment start locations), end, center, or 'ends' (both ends of the read/fragment).
ov.type	Overlap type. See the 'type' argument of link[GenomicRanges]{countOverlaps}.
maxgap	Maximum gap allowed for overlaps (see the corresponding argument of link[GenomicRanges]{countOv
minoverlap	Minimum overlap (see the corresponding argument of link[GenomicRanges]{countOverlaps}).
ignore.strand	Logical; whether to ignore strand for the purpose of counting overlaps (default TRUE).
strandMode	The strandMode of the data (whether the strand is given by the first or second mate, which depends on the library prep protocol). See strandMode for more information. This parameter has no effect unless one of the 'strand', 'extend' parameters or a strand-specific 'shift' are used.
includeDuplicates	Logical, whether to include reads flagged as duplicates.
includeSecondary	Logical; whether to include secondary alignments
minMapq	Minimum mapping quality (1 to 255)
minFragLength	Minimum fragment length (ignored if 'paired=FALSE')
maxFragLength	Maximum fragment length (ignored if 'paired=FALSE')

splitByChr — Whether to process chromosomes separately, and if so by how many chunks. The should not affect the output, and is simply slightly slower and consumes less memory. Can be a logical value (in which case each chromosome is processed separately), but we instead recommend giving a positive integer indicating the number of chunks.

randomAcc — Logical, whether to use random access. This is disabled by default because the overhead of random access to a lot of regions is typically worse than reading the entire file. However, if you need to get counts in few regions, enabling this will be faster. Note however that when using random access, the output object will not contain depth information.

getMedianFragLength — Logical; whether to compile the median fragment length per region. This is slightly slower. The log10-transformed, (weighted mean across samples of the) median fragment length per region is stored in 'rowData(results)$flbias'.

BPPARAM — BiocParallel params for multithreading. Note that multithreading can lead to high memory usage.

verbose — Logical; whether to print progress messages

Value

A RangedSummarizedExperiment with a 'counts' assay.

Examples

```
# get an example bam file
bam <- system.file("extdata", "ex1.bam", package="Rsamtools")
# create regions of interest
peaks <- GRanges(c("seq1","seq1","seq2"), IRanges(c(400,900,500), width=100))
peakCountsFromBAM(bam, peaks, paired=FALSE)
```

---

peakCountsFromFrags *peakCountsFromFrags*

Description

Creates a SummarizedExperiment of cell-level or pseudo-bulk-level fragment (or insertion) counts from a tabix-indexed fragment file and a set of regions of interest.

Usage

```
peakCountsFromFrags(
  fragfile,
  regions,
  barcodes = NULL,
  insertions = FALSE,
  minFragLength = 1L,
  maxFragLength = 5000L,
```

```
  ov.type = "any",
  maxgap = -1L,
  minoverlap = 1L,
  ignore.strand = TRUE,
  ...
)
```

Arguments

fragfile	A path to the tabix-indexed fragment file.
regions	A GRanges of regions in which to count overlaps.
barcodes	An optional character vector of cell barcodes to include. If provided, only these barcodes will be considered, if 'NULL', all barcodes included in the file are used. If 'barcodes' is a named vector, the names will be considered to represent the cell barcode, the values to represent the pseudo-bulk sample in which to include the respective barcodes, and pseudo-bulk counts will be returned.
insertions	Logical; if TRUE, the ends of the fragments (insertions) are counted instead of the entire fragment. This means each fragment can contribute up to two counts. Default FALSE.
minFragLength	Minimum fragment length to be considered. Default 1.
maxFragLength	Maximum fragment length to be considered. Default 5000.
ov.type	Overlap type. See the 'type' argument of link[GenomicRanges]{countOverlaps}.
maxgap	Maximum gap allowed for overlaps (see the corresponding argument of link[GenomicRanges]{countOv
minoverlap	Minimum overlap (see the corresponding argument of link[GenomicRanges]{countOverlaps}).
ignore.strand	Logical; whether to ignore strand for the purpose of counting overlaps (default TRUE).
...	Passed to tabixChrApply.

Value

A RangedSummarizedExperiment with a 'counts' assay. The mean fragment length per region is also stored in the 'rowData' of the object.

Examples

```
# generate dummy regions and save them to a temp file:
frags <- tempfile(fileext = ".tsv")
d <- data.frame(chr=rep(letters[1:2], each=10), start=rep(100*(1:10),2))
d$end <- d$start + 15L
d$cell <- paste0("barcode",sample.int(3, nrow(d), replace=TRUE))
write.table(d, frags, col.names=FALSE, row.names=FALSE, sep="\t", quote=FALSE)
# tabix-index it
frags <- Rsamtools::bgzip(frags)
Rsamtools::indexTabix(frags, format = "bed")
# we create regions of interest:
regions <- GRanges(c("a","b"), IRanges(400,width=300))
# we get the counts:
```

```
se <- peakCountsFromFrags(frags, regions)
se
# we could also get pseudobulk counts by passing a barcode map:
bcmap <- setNames(c("PB1","PB1","PB2"),paste0("barcode",1:3))
pb <- peakCountsFromFrags(frags, regions, barcodes=bcmap)
pb
```

---

plotCorFromCovStats *plotCorFromCovStats*

Description

plotCorFromCovStats

Usage

```
plotCorFromCovStats(
  qc,
  method = c("pearson", "spearman"),
  col = NULL,
  na_col = "white",
  column_title = NULL,
  ...
)
```

Arguments

qc	A list of coverage statistics, as produced by getCovStats.
method	The correlation metrics to include
col	Optional heatmap colors
na_col	Color for the diagonal; passed to Heatmap.
column_title	Column title (if NULL, uses the metric)
...	Passed to Heatmap.

Value

A ‘Heatmap‘ or ‘HeatmapList‘ object ready to be plotted.

plotCovStats *plotCovStats*

Description

Plots coverage statistics, such as as fingerprint plot.

Usage

```
plotCovStats(qc)
```

Arguments

qc	A list of coverage statistics, as produced by getCovStats.

Value

A grid object to be plotted.

Examples

```
# we use an example bigwig file
bwf <- system.file("extdata/example_atac.bw", package="epiwraps")
# because most of the file is empty, we'll exclude some of the ranges
cs <- getCovStats(bwf, exclude=GRanges("1", IRanges(1, 4300000)))
plotCovStats(cs)
```

plotEnrichedHeatmaps *plotEnrichedHeatmaps: Plots heatmaps of signals around a set of regions*

Description

Plots enrichment heatmaps from the output of signal2Matrix (i.e. an EnrichmentSE object or a list of signal matrices). This is a convenience wrapper around EnrichedHeatmap.

Usage

```
plotEnrichedHeatmaps(
  ml,
  trim = c(0.02, 0.98),
  assay = 1L,
  colors = NULL,
  scale_title = "density",
  column_title = NULL,
  multiScale = NULL,
```

```
  column_title_gp = gpar(fontsize = 11),
  row_order = NULL,
  cluster_rows = FALSE,
  row_split = NULL,
  axis_name = NULL,
  minRowVal = 0,
  scale_rows = FALSE,
  top_annotation = TRUE,
  left_annotation = NULL,
  right_annotation = NULL,
  mean_color = "red",
  mean_scale_side = NULL,
  mean_trim = TRUE,
  show_heatmap_legend = TRUE,
  use_raster = NULL,
  ...
)
```

Arguments

ml	A named matrix list as produced by signal2Matrix, or an 'EnrichmentSE' object.
trim	The quantile above which to trim values for the colorscale. If a numeric vector of length 2, will be used as lower and upper quantiles beyond which to trim.
assay	Assay to use (ignored unless 'ml' is an ESE object)
colors	The heatmap colors to use, a vector of at least two colors between which to interpolate. Can also be a list of such color scales, with as many slots as there are tracks in 'ml'. If a list of single colors, a color scale from white to that color will be used for each track. Defaults to the 'inferno' viridis palette.
scale_title	The title of the scale. Ignored if 'multiScale=TRUE'.
column_title	The title above the heatmap. If NULL (default), sample (i.e. track) names will be used.
multiScale	Logical; whether to use a different scale for each track. Defaults to TRUE is 'colors' is a list, otherwise FALSE.
column_title_gp	Graphic parameters of the column titles (see gpar)
row_order	Optional order of the rows.
cluster_rows	Logical; whether to cluster rows. Alternatively, 'cluster_rows="sort"' will sort rows using the angle on an MDS based on the enriched_score of the signals (can be very long to compute on large matrices). 'cluster_rows=FALSE' (default) results in the traditional sorting by decreasing 'enriched_score'.
row_split	Variable according to which the rows should be split. This should either be the name of a column of 'rowData(ml)', or a factor/ character vector of length equal to the number of regions in 'ml'.
axis_name	A vector of length 3 giving the labels to put respectively on the left, center and right of the x axis of each heatmap.

minRowVal	Minimum value a row should have to be included
scale_rows	Whether to scale rows, either FALSE (default), 'local' (scales each matrix separately) or 'global'.
top_annotation	Either a logical indicating whether or not to plot the summary profile at the top of each heatmap, a named list of parameters to be passed to 'anno_enrich', or a HeatmapAnnotation-class object that will be passed to EnrichedHeatmap. Additionally, if 'ml' is a 'ESE' object, 'top_annotation' can be a vector of colData column names.
left_annotation	Passed to EnrichedHeatmap
right_annotation	Passed to EnrichedHeatmap
mean_color	Color of the mean signal line in the top annotation. If 'row_split' is used, 'mean_color' can be a named vector indicating the colors for each cluster. Can also be a 'gpar' object.
mean_scale_side	The side on which to show the y-axis scale of the mean plots. Either "both" (default), "left", "right", or "none".
mean_trim	Logical; whether to apply the trimming also to the mean plot.
show_heatmap_legend	Logical, whether to show the heatmap legend
use_raster	Logical; whether to render the heatmap body as a raster image. Turned on by default if any of the matrix dimensions is greater than 1500.
...	Passed to EnrichedHeatmap

Details

When plotting large matrices, the heatmap body will be rasterized to keep its memory footprint decent. For this to work well, make sure the 'magick' package is installed. Depending on your settings, if the heatmap is very big you might hit the preset limits of 'magick' base rasterization, which could result in an error such as 'Image must have at least 1 frame to write a bitmap'. In such cases, you might have to degrade to a lower-quality rasterization using the additional arguments 'raster_by_magick=FALSE, raster_device="CairoJPEG"' . Alternatively, the best solution is to increase the size and memory limits in the 'policy.xml' configuration of the underlying (non-R) imagemagick library.

Value

A HeatmapList object

Examples

```
# we first load an example EnrichmentSE, as produced by signal2Matrix:
data(exampleESE)
plotEnrichedHeatmaps(exampleESE)
# we could also just plot one with:
# plotEnrichedHeatmaps(exampleESEm[,1])
```

```
# or change the aesthetics, e.g.:
plotEnrichedHeatmaps(exampleESE, trim=0.98, scale_title="RPKM",
                     colors=c("white","darkred"), row_title="My regions")
# any argument accepted by `EnrichedHeatmap` (and hence by
# `ComplexHeatmap::Heatmap`) can be used.
```

---

plotSignalTracks *plotSignalTracks*

Description

A wrapper around 'Gviz' for quick plotting of genomic signals in a single region.

Usage

```
plotSignalTracks(
  files = list(),
  region,
  ensdb = NULL,
  colors = "darkblue",
  type = "histogram",
  genomeAxis = 0.3,
  extend = 0.15,
  aggregation = c("mean", "median", "sum", "max", "min", "heatmap", "overlay",
    "heatmap+mean"),
  transcripts = c("collapsed", "full", "coding", "none"),
 genes.params = list(col.line = "grey40", col = NULL, fill = "#000000", rotation.title =
    0),
  align.params = list(color = NULL),
  tracks.params = list(),
  extraTracks = list(),
  background.title = "white",
  col.axis = "grey40",
  bed.rotation.title = 0,
  col.title = "black",
  cex.title = 0.65,
  overlay.alpha = 100,
  normFactors = NULL,
  ...
)
```

Arguments

files	A named list or vector of paths to signal files (e.g. bigwig/bam, but also bed files). If a list, list elements will be overlaid or aggregated (depending on the 'aggregation' argument). Formats accepted by DataTrack's 'range' argument are also accepted. Can also include 'GRanges ' objects (which will be plotted

	as AnnotationTrack) or objects inheriting the GdObject-class class (i.e. any 'Gviz' track object).
region	A genomic region, either as a 'GRanges' object or as a string (i.e. 'region="chr5:10000-12000'). Alternatively, if 'ensdb' is provided, a gene name can be given, and the gene's coordinates will be used as region.
ensdb	An optional EnsDb object form which to grab transcripts.
colors	Signal color(s); will be recycled for elements of 'files'
type	Signal plot type(s); will be recycled for elements of 'files'. This is ignored for bed-like files, which are shown as AnnotationTrack. See the 'type' options of DataTrack. In addition to these options, the type 'alignments' can be given for bam files, which will display them as AlignmentsTrack.
genomeAxis	Whether to plot a genome axis. Alternatively, a numeric scalar between 0 and 1 can be given, in which case a scale will be plotted of this relative size.
extend	Either an integer or vector of two integers indicating the number of base pairs by which to extent on either side. If 'extend'<=1, this will be interpreted as a fraction of the plotted region.
aggregation	Method for aggregation data tracks, one of: 'mean' (default), 'median', 'max', 'overlay', 'heatmap', or 'heatmap+mean'. The latter will create a mean plot of type 'type' followed by a heatmap.
transcripts	Whether to show transcripts (reguires 'ensdb') as "full", "collapsed" (default), "coding" (only coding transcripts) or "none". Alternatively, can be a custom GeneRegionTrack object.
genes.params	Named list of parameters passed to GeneRegionTrack.
align.params	Named list of parameters passed to AlignmentsTrack. Only used for plotting bam files with 'type="alignments"'.
tracks.params	Named list of parameters passed to DataTrack.
extraTracks	List of extra custom tracks to be plotted.
background.title	The background color of the track titles.
col.axis	The color of the axes.
bed.rotation.title	Rotation for track titles of bed files.
col.title	The color of the track titles.
cex.title	Expension factor for the font size of the track titles.
overlay.alpha	Transparency (0 to 250) when overlaying tracks.
normFactors	Optional normalization factors to apply before plotting.
...	Passed to plotTracks.

Value

A list of GenomeGraph tracks to be plotted.

Examples

```
# fetch path to example bigwig file:
(bw <- system.file("extdata/example_rna.bw", package="epiwraps"))
plotSignalTracks(list(track1=bw), region="8:22165140-22212326")
# if we had an EnsDb object loaded, we could just input a gene instead of
# coordinates, and the transcript models would automatically show (not run):
# plotSignalTracks(list(track1=bw), region="BMP1", ensdb=ensdb)
# show all transcript variants:
# plotSignalTracks(list(tracks=bw), region="BMP1", ensdb=ensdb,
#                  transcripts="full")
```

`reduceRleLists` *reduceRleLists*

Description

reduceRleLists

Usage

```
reduceRleLists(res, fn = "+")
```

Arguments

`res`	A list of RleList objects
`fn`	The function to use to combine them

Value

A combined RleList object.

Examples

```
list_of_rlelists <- list(
  RleList(A=c(0,0,1,0,0), B=c(0,0,0,0,1)),
  RleList(A=c(1,1,1,0,0), C=c(0,1,1,1,1)) )
reduceRleLists(list_of_rlelists)
```

`reduceWithResplit` *Merge regions, re-splitting large merges using local overlap minima*

Description

Merge regions, re-splitting large merges using local overlap minima

Usage

```
reduceWithResplit(
  peaks,
  softMaxSize = 500L,
  relTroughDepth = 1/3,
  minTroughDepth = 2L,
  minTroughWidth = 1L,
  minDistFromBoundary = 150L,
  minPeakSize = 100L,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`peaks`	A list of `GRanges-class`, or a `GRanges-class` containing overlapping regions.
`softMaxSize`	The (merged) peak size below which re-splitting will be attempted
`relTroughDepth`	The minimum depth of local minima, as a fraction of the maximum. E.g. with a maxima of 12 peaks, the default of 1/4 would require the minima to be below or equal to 9.
`minTroughDepth`	The absolute minimum depth of local minima, in number of peaks below the maxima.
`minTroughWidth`	The minimum width of the local minima.
`minDistFromBoundary`	The minimum distance of the local minima from the peak border.
`minPeakSize`	The minimum final peak size.
`BPPARAM`	BiocParallel Param object for multithreading. If set, chromosomes are split into threads.

Details

This is an alternative to something like `reduce(unlist(GRangesList(peaks)))`, which stitches overlapping regions together and can result in large regions that can be problematic for some applications. The function tries to break those large regions into composing by using the coverage by the original (un-merged) regions. See the example below for an illustration. The procedure first reduces 'peaks', then identifies reduced regions whose width is above a certain threshold ('softMaxSize'). For those regions, a coverage by the original peaks is computed to identify local minima ('troughs') in the coverage that could divide the region into sub-regions of desirable lengths. 'relThroughDepth'

determines the minimum depth of the trough (i.e. decrease) as a fraction of the maximum coverage in the region, while 'minTroughDepth' determines the absolute minimum depth. Note that the algorithm iterates through regions one by one and as such is quite slow, hence multithreading is recommended for large sets of regions.

Value

A reduced 'GRanges' of non-overlapping peaks.

Examples

```
# consider the following example set of regions:
gr <- GRanges("1", IRanges(c(100,120,140,390,410,430,120),
                           width=rep(c(200,520),c(6,1))))
plotSignalTracks(list(regions=gr, "# overlapping regions"=coverage(gr),
                      reduced=reduce(gr)), region=reduce(gr))
# if we are interested in having smaller regions, clearly it would seem
# sensible here to cut roughly in the middle, since we have two distinct
# groups of regions that are only joined by a single region

(redGr <- reduceWithResplit(gr, softMaxSize=100))
plotSignalTracks(list("# overlapping regions"=coverage(gr),
                      reduced=reduce(gr), "reduced\n\\w resplit"=redGr),
                      region=reduce(gr))
```

`regionCAT` *regionCAT*

Description

Computes/plots the 'concordance at the top' (CAT) of two lists of genomic regions.

Usage

```
regionCAT(
  regions1,
  regions2,
  start = 5L,
  concord.type = c("both", "inTop", "inAll"),
  returnData = FALSE,
  ignore.strand = TRUE
)
```

Arguments

`regions1, regions2`
: A GRanges object with a 'score' metadata column according to which the regions will be ranked (descending).

start	The rank at which to start plotting (removes large variations at the beginning when very few regions are considered)
concord.type	Concordance type to plot, either 'inTop', 'inAll', or 'both' (see details). Ignored if 'returnData=TRUE'.
returnData	Logical; whether to return the data instead of plotting.
ignore.strand	Logical; whether to ignore the strand for computing overlap (default TRUE)

Details

The two concordance types are as follows: * 'inTop' indicates the proportion of the top X regions that are in the top X in both lists. * 'all' indicates the proportion of the top X regions that are anywhere in the other list (since this relationship is asymmetrical, the mean of both two directions is used).

Value

A ggplot object, or a data.frame if 'returnData=TRUE'.

Examples

```
# we create two GRanges with scores, which have similar high-score peaks but
# the rest random:
gr1 <- GRanges("seq1", IRanges(runif(20,1,2000), width=20),
               score=20:1)
gr2 <- GRanges("seq1", c(head(ranges(gr1),5),
                         IRanges(runif(15,1,2000), width=20)),
               score=c(20:16, sample.int(15)))
regionCAT(gr1,gr2)
```

`regionOverlaps` *regionOverlaps*

Description

A wrapper for visualizing pairwise-wise overlaps across multiple sets of genomic ranges.

Usage

```
regionOverlaps(
  listOfRegions,
  mode = c("reduced", "pairwise"),
  ignore.strand = TRUE,
  cluster = length(listOfRegions) > 2,
  colorBy = c("overlapCoef", "jaccard"),
  ...,
  returnValues = FALSE,
  color = viridisLite::plasma(100),
  number_color = "black"
)
```

Arguments

listOfRegions	A named list of two or more (non-empty) 'GRanges'
mode	Either 'reduced' or 'pairwise'. 'reduced' first uses 'reduce' to get a set of reference regions which are, based on overlap, contained or not in the different sets. It is thus symmetrical. 'pairwise' does pairwise overlap between the sets of regions; it is asymmetrical and slower to compute.
ignore.strand	Logical; whether to ignore strand for overlaps (default TRUE).
cluster	Logical; whether to cluster rows/columns
colorBy	Whether to color by 'overlapCoef' (default), or by 'jaccard' index.
...	Passed to pheatmap
returnValues	Logical; whether to return the matrix of requested values instead of plotting.
color	Heatmap colorscale
number_color	Values color

Value

A 'Heatmap' showing the overlap coefficient as colors, and the overlap size as values. Alternatively, if 'returnValues=TRUE', a matrix of the requested values.

Examples

```
# random list of GRanges:
grl <- lapply(c(A=10,B=20,C=30), FUN=function(x){
  GRanges("seq1", IRanges(runif(x,1,1000), width=20))
})
regionOverlaps(grl)
```

---

regionsToUpset *regionsToUpset*

Description

Prepares sets of regions for UpSet overlap representation.

Usage

```
regionsToUpset(
  x,
  reference = c("reduce", "disjoin"),
  returnList = FALSE,
  ignore.strand = FALSE,
  maxgap = -1L,
  minoverlap = 0L,
  ...
)
```

Arguments

x	A named list of genomic ranges (or paths to bed files)
reference	The method for creating the reference windows ('reduce' or 'disjoin'). Alternatively, a 'GRanges' object of reference windows.
returnList	Logical; whether to return the list of regions instead of plotting.
ignore.strand	Logical; whether to ignore strand for overlaps (default FALSE).
maxgap	Max gap between regions to consider an overlap.
minoverlap	Minimum number of overlapping bases.
...	Further arguments specifying how the overlaps are done, passed to findOverlaps-methods).

Value

A data.frame of set inclusions which can be directly input to make_comb_mat, and then UpSet.

Examples

```
# random list of GRanges:
grl <- lapply(c(A=10,B=20,C=30), FUN=function(x){
  GRanges("seq1", IRanges(runif(x,1,1000), width=20))
})
input_for_upset <- regionsToUpset(grl)
# we would then plot the data with:
ComplexHeatmap::UpSet(make_comb_mat(input_for_upset))
```

---

renormalizeBorders *renormalizeSignalMatrices*

Description

Renormalizes a list of signal matrices or an EnrichmentSE object.

Usage

```
renormalizeBorders(ml, trim = NULL, assay = "input", nWindows = NULL)

renormalizeSignalMatrices(
  ml,
  method = c("border", "top", "manual"),
  trim = NULL,
  fromAssay = "input",
  toAssay = NULL,
  nWindows = NULL,
  scaleFactors = NULL
)
```

Arguments

ml	A named matrix list or EnrichmentSE object as produced by signal2Matrix.
trim	Quantiles trimmed at each extreme before calculating normalization factors.
assay	The name of the assay to use as input.
nWindows	Number of border windows/bins to use for border normalization.
method	Either "border" or "top" (see details below).
fromAssay	Assay to use (ignored unless 'ml' is an EnrichmentSE object), defaults to the first assay.
toAssay	Assay in which to store the normalized data (ignored unless 'ml' is an EnrichmentSE object). By default an assay name will be set based on the normalization method used.
scaleFactors	A numeric vector of same length as 'ml', indicating the scaling factors by which to multiply each matrix. Alternatively, a numeric matrix with a number of rows equal to the length of 'ml', and two columns indicating the alpha and beta arguments of a s3norm normalization. Ignored unless 'method="manual"'.

Details

* 'method="border"' works on the assumption that the left/right borders of the matrices represent background signal which should be equal across samples. As a result, it will work only if 1) the left/right borders of the matrices are sufficiently far from the signal (e.g. peaks) to be chiefly noise, and 2) the signal-to-noise ratio is comparable across tracks/samples. * 'method="top"' instead works on the assumption that the highest signal should be the same across tracks/samples. By default, extreme values are trimmed before establishing either kind of normalization factor. The proportion trimmed can be set using the 'trim' argument, and is by default 10 * 'method="manual"' enables the use of independently computed normalization factors, for instance obtained through getNormFactors.

Value

Either a renormalized list of signal matrices or, if 'ml' was an 'EnrichmentSE' object, the same object with an additional normalized assay automatically put at the front.

Functions

- renormalizeBorders(): deprecated > renormalizeSignalMatrices

Examples

```
# we first get an EnrichmentSE object:
data(exampleESE)
# we normalize them
exampleESE <- renormalizeSignalMatrices(exampleESE)
# see the `vignette("multiRegionPlot")` for more info on normalization.
```

resizeMatrix *resize a numeric matrix to given dimensions*

Description

resize a numeric matrix to given dimensions

Usage

```
resizeMatrix(mat, ndim = dim(mat), method = c("mean", "max", "min"))
```

Arguments

mat	A numeric matrix
ndim	The desired output dimensions
method	Whether to use normal interpolation (‘method="mean"‘, the default), or the max or min of the overlapping grid cells.

Details

For most cased this is based on Vyha’s implementation (taken from https://stackoverflow.com/a/23429527), but adding the possibility to replace normal interpolation with the max/min of the overlapping grid cells. This works well when the desired dimensions are at least half of the input ones. When the desired dimensions are smaller, a different binning method is used, first applied on columns and then on rows.

Value

A numeric matrix of dimensions ‘ndim‘

Examples

```
# generate a dummy matrix
m <- outer(1:50, 1:50, FUN = \(x,y) sqrt(x*y))
m2 <- resizeMatrix(m, c(10,10))
```

showTrackInfo *showTrackInfo*

Description

Provide some information about the relative signal ranges of each track.

Usage

```
showTrackInfo(x, assay = "input", doPrint = TRUE)
```

Arguments

x	A named list of signal matrices or an EnrichmentSE object as produced by signal2Matrix
assay	The assay to use, defaults to the input assay.
doPrint	Logical; whether to print the information.

Value

An invisible list of captions.

Examples

```
data(exampleESE)
showTrackInfo(exampleESE)
```

---

signal2Matrix *signal2Matrix: reads the signals in/around a set of genomic regions.*

Description

Reads the signals from bam/bigwig files within and/or around (the centers of) a set of regions, and outputs an EnrichmentSE object.

Usage

```
signal2Matrix(
  filepaths,
  regions,
  extend = 2000,
  w = NULL,
  scaling = TRUE,
  scaledBins = round(max(extend)/w),
  type = c("center", "scaled"),
  binMethod = c("mean", "max", "min"),
  BPPARAM = 1L,
  ret = c("EnrichmentSE", "list"),
  verbose = TRUE,
  ...
)
```

Arguments

filepaths	A named vector of filepaths (e.g. to bigwig files; bam files are also supported, but with limited functionalities). Can also be a named list including a combination of paths to such files and 'GRanges' object. For 'GRanges' objects, the 'score' column will be used (absolute coverage mode).

regions	A 'GRanges' of the regions/positions around which to plot, or the path to a bed file of such regions. If 'type="scaled"', 'regions' can also be a 'GRangesList', in which case the coverage of the subregions will be stiched together (e.g. for plotting exonic signal over transcripts).
extend	Number of basepair to extend on either side of the regions. Must be a multiple of 'w'. Can also be an integer of length 2, indicating the extension upstream and downstream.
w	Bin width in number of nucleotides. Defaults to a width producing 200 bins over the whole extended range.
scaling	Logical; whether to scale to library size when reading from BAM files.
scaledBins	The number of bins for the scale region (ignored if 'type="center"')
type	Either 'center' (plots fixed-size region around the centers of 'regions') or 'scaled' (scales the signal in 'regions' and plot surroundings)
binMethod	Whether to compute the 'max' (default), 'mean' or 'min' per bin.
BPPARAM	A BiocParallelParam object, or the number of threads to use to read and prepare the data. Note that the rate-limiting process is reading from disk, so unless you have an unusually fast disk, using multi-threading is actually likely to slow down rather than speed up the process.
ret	The type of output to return, either an "EnrichmentSE" object (default), or a simple list of signal matrices ("list").
verbose	Logical; whether to print processing information
...	Passed to normalizeToMatrix or as.normalizedMatrix, or to bam2bw when reading bam files. For example, this can be used to pass arguments to 'normalizeToMatrix' such as 'smooth=TRUE'.

Value

A list of 'normalizeToMatrix' objects

Examples

```
# we fetch the path to the example bigwig file:
(bw <- system.file("extdata/example_atac.bw", package="epiwraps"))
# we load example regions:
regions <- rtracklayer::import(system.file("extdata/example_peaks.bed",
                                           package="epiwraps"))
length(regions)
# we obtain the matrix of the signal around the regions:
m <- signal2Matrix(bw, regions)
# we can plot it with:
plotEnrichedHeatmaps(m)
# we could also take a broader range around the center of the regions, and
# use bigger bins:
m <- signal2Matrix(bw, regions, extend=2000, w=20)
plotEnrichedHeatmaps(m)
```

signalsAcrossSamples *signalsAcrossSamples*

Description

Obtain a matrix of score/coverages across a region for a list of BigWig files.

Usage

```
signalsAcrossSamples(files, region, ignore.strand = TRUE)
```

Arguments

files	A named list of paths to biwgig files or of ‘GRanges‘ objects with a ‘score‘ column.
region	The region of interest, either given as a string (in the "chr:start-end" format) or as a ‘GRanges‘ of length 1.
ignore.strand	Logical; whether to merge scores from the two strands given stranded objects.

Value

A disjoined ‘GRanges object‘ with the scores as metadata columns.

tabixChrApply *tabixChrApply*

Description

Runs a function on reads/fragments from each chromosomes of a Tabix-indexed fragment file. This is especially used by other functions to avoid loading all alignments into memory, or to parallelize reads processing.

Usage

```
tabixChrApply(
  x,
  fn,
  keepSeqLvls = NULL,
  exclude = NULL,
  only = NULL,
  BPPARAM = NULL,
  progress = TRUE,
  ...
)
```

Arguments

x	The path to a tabix-indexed bam file, or a TabixFile object.
fn	The function to be run, the first argument of which should be a 'GRanges'
keepSeqLvls	An optional vector of seqLevels to keep
exclude	An optional GRanges of regions for which overlapping reads should be excluded.
only	An optional GRanges of regions for which overlapping reads should be included. If set, all other reads are discarded.
BPPARAM	A 'BiocParallel' parameter object for multithreading. Note that if used, memory usage will be high; in this context we recommend a high 'nChunks'.
progress	Logical; whether to show a progress bar.
...	Passed to 'fn'

Value

A list of whatever 'fn' returns

Examples

```
# generate dummy regions and save them to a temp file:
frags <- tempfile(fileext = ".tsv")
d <- data.frame(chr=rep(letters[1:2], each=10), start=rep(100*(1:10),2))
d$end <- d$start + 15L
write.table(d, frags, col.names=FALSE, row.names=FALSE, sep="\t")
# tabix-index it
frags <- Rsamtools::bgzip(frags)
Rsamtools::indexTabix(frags, format = "bed")
# now we can do something chunk-wise, e.g. extract coverage:
res <- tabixChrApply(frags, fn=coverage)
# aggregate the chunk results into an RleList object:
reduceRleLists(res)
```

---

tileRle *tileRle*

Description

Creates an Rle of fixed-with bins from a continuous numeric Rle

Usage

```
tileRle(x, bs = 10L, method = c("max", "min", "mean"), roundSummary = FALSE)
```

Arguments

x	A numeric ‘Rle‘ (or ‘RleList‘)
bs	A positive integer specifying the bin size
method	The method for summarizing bins
roundSummary	Logical; whether to round bins with summarized coverage (default FALSE)

Value

An object of same class and length as ‘x‘

Examples

```
# creating a dummy coverage and visualizing it:
cov <- Rle(rpois(100,0.5))
plot(cov, type="l", col="lightgrey")
# summarizing to tiles of width 5 (by default using maximum)
cov2 <- tileRle(cov, bs=5L)
lines(cov2, col="red")
```

TSSenrichment *TSSenrichment*

Description

A common quality metric for ATAC-seq data (see [definition](https://www.encodeproject.org/data-standards/terms/#enrichment)). The interpretation of the enrichment score depends on the annotation (the score tends to increase when only fewer, more common TSS are used), but according to ENCODE guidelines anything below 5 is of concerning quality, while a score >8 is ideal.

Usage

```
TSSenrichment(tracks, ensdb, useSeqLevels = NULL)
```

Arguments

tracks	A (named) vector of paths to bigwig files.
ensdb	An ‘ensembldb‘ object. Alternatively, a GRanges object of regions centered around TSS.
useSeqLevels	Optional seqlevels to use. If NULL, all are used.

Value

A list with the slots ‘score‘ (numeric vector of TSS enrichment scores per sample) and ‘data‘ (per bin enrichment, for plotting)

Examples

```
# we first fetch the path to the example bigwig file:
bw <- system.file("extdata/example_atac.bw", package="epiwraps")
## normally, we would load an ensembldb object using AnnotationHub. For the
## purpose of this example, we'll pretend that the following set of regions
## represent TSS:
tss <- system.file("extdata/example_peaks.bed", package="epiwraps")
tss <- rtracklayer::import(tss)
en <- TSSenrichment(bw, tss)
en$score
## you can also plot using something like this:
## ggplot(en$data, aes(position, enrichment, colour=sample)) + geom_line()
```

`views2Matrix` *views2Matrix*

Description

converts a RleViews or RleViewsList with views of the same width to a matrix, setting out-of-bounds regions to NA (or 'padVal').

Usage

```
views2Matrix(v, padVal = NA_integer_)
```

Arguments

v	A 'RleViews' or 'RleViewsList' object with views of the same width.
padVal	The value to assign to out-of-bound regions.

Value

A numeric matrix.

Examples

```
# we create an example RleViews with out-of-bound regions:
library(IRanges)
co <- Rle(values=c(0,1,0), lengths=c(100,50,20))
v <- Views(co, c(25,150),c(50,175))
# convert to matrix
views2Matrix(v)
```

Index