Package 'GenomicPlot'

April 20, 2023

Type Package

Title Plot profiles of next generation sequencing data in genomic features

Version 0.1.0

Description Visualization of next generation sequencing (NGS) data is essential for interpreting high-throughput genomics experiment results. 'GenomicPlot' facilitates plotting of NGS data in various formats (bam, bed, wig and bigwig); both coverage and enrichment over input can be computed and displayed with respect to genomic features (such as UTR, CDS, enhancer), and user defined genomic loci or regions. Statistical tests on signal intensity within user defined regions of interest can be performed and represented as boxplots or bar graphs. Parallel processing is used to speed up computation on multicore platforms. In addition to genomic plots which is suitable for displaying of coverage of genomic DNA (such as ChIPseq data), metagenomic (without introns) plots can also be made for RNAseq or CLIPseq data as well.

License GPL-2

Encoding UTF-8

LazyData FALSE

Collate ``DrawingFunctions.R'' ``GenomicPlot.R'' ``HandleDataMatrix.R'' ``HandleFeatures.R'' ``Parallel.R'' ``ReadData.R'' ``Setup.R''

Depends R (>= 4.0.0), GenomicRanges (>= 1.46.1), GenomicFeatures

Imports methods, Rsamtools, parallel, grid, ggplot2 (>= 3.3.5), tidyr, rtracklayer (>= 1.54.0), BiocGenerics,
plyranges (>= 1.14.0), dplyr (>= 1.0.8), cowplot (>= 1.1.1), VennDiagram, ggplotify, GenomeInfoDb, IRanges,
ComplexHeatmap, RCAS (>= 1.20.0), scales (>= 1.2.0), GenomicAlignments (>= 1.30.0), edgeR, forcats,
circlize, viridis, ggsignif (>= 0.6.3), ggsci (>= 2.9), genomation (>= 1.26.0), ggpubr

RoxygenNote 7.2.1

VignetteBuilder knitr

Suggests knitr, rmarkdown, AnnotationHub, AnnotationDbi, R.utils, BSgenome.Hsapiens.UCSC.hg19 (>= 1.4.3), BiocFileCache,
TxDb.Hsapiens.UCSC.hg19.knownGene (>= 3.2.2), GenomicPlotData

R topics documented:

aov_TukeyHSD *Perform one-way ANOVA and post hoc TukeyHSD tests*

Description

This is a helper function for performing one-way ANOVA analysis and post hoc Tukey's Honest Significant Differences tests

Usage

```
aov_TukeyHSD(df, xc = "Group", yc = "Intensity", op = NULL, verbose = FALSE)
```

Arguments

df	a dataframe
xc	a string denoting column name for grouping
yc	a string denoting column name for numeric data to be plotted
op	output prefix for statistical analysis results
verbose	logical, to indicate whether a file should be produced to save the test results

Value

a list of two elements, the first is the p-value of ANOVA test and the second is a matrix of the output of TukeyHSD tests

Note

used in `plot_locus`

Author(s)

Shuye Pu

check_constraints *Check constraints of genomic ranges*

Description

Make sure the coordinates of GRanges are within the boundaries of chromosomes, and trim anything that goes beyond. Also, remove entries whose seqname is not in the seqname of a query GRanges.

Usage

```
check_constraints(gr, genome, queryRle = NULL)
```

Arguments

gr	a GenomicRanges object
genome	genomic version name such as "hg19"
queryRle	a RleList object used as a query against gr

Value

a GRanges object

Author(s)

Shuye Pu

---

draw_boxplot_by_factor

Plot boxplot with two factors

Description

Plot boxplot for data with one or two factors, with p-value significance levels displayed

Usage

```
draw_boxplot_by_factor(
  stat_df,
  xc = "Feature",
  yc = "Intensity",
  fc = xc,
  comp = list(c(1, 2)),
  stats = "wilcox.test",
  Xlab = xc,
  Ylab = yc,
  nf = 1
)
```

Arguments

stat_df	a dataframe with column names c(xc, yc)
xc	a string denoting column name for grouping
yc	a string denoting column name for numeric data to be plotted
fc	a string denoting column name for sub-grouping based on an additional factor
comp	a list of vectors denoting pair-wise comparisons to be performed between groups
stats	the name of pair-wise statistical tests, like t.test or wilcox.test
Xlab	a string for x-axis label
Ylab	a string for y-axis label
nf	a integer normalizing factor for correct count of observations when the data table has two factors, such as those produced by pivot_longer, equals to the number of factors

Value

a ggplot object

Note

used by `plot_locus`, `plot_locus_with_random`, `plot_region`

Author(s)

Shuye Pu

Examples

```
stat_df <- data.frame(Feature=rep(c("A", "B"), c(20, 30)),
Intensity=c(rnorm(20, 2, 0.5), rnorm(30, 3, 0.6)))
p <- draw_boxplot_by_factor(stat_df, xc="Feature", yc="Intensity",
Ylab="Signal Intensity")
p
```

---

draw_boxplot_wo_outlier

Plot boxplot without outliers

Description

Plot boxplot without outliers, with p-value significance levels displayed

Usage

```
draw_boxplot_wo_outlier(
  stat_df,
  xc = "Feature",
  yc = "Intensity",
  fc = xc,
  comp = list(c(1, 2)),
  stats = "wilcox.test",
  Xlab = xc,
  Ylab = yc,
  nf = 1
)
```

Arguments

`stat_df`	a dataframe with column names c(xc, yc)
`xc`	a string denoting column name for grouping
`yc`	a string denoting column name for numeric data to be plotted
`fc`	a string denoting column name for sub-grouping
`comp`	a list of vectors denoting pair-wise comparisons to be performed between groups
`stats`	the name of pair-wise statistical tests, like t.test or wilcox.test
`Xlab`	a string for x-axis label

Ylab	a string for y-axis label
nf	a integer normalizing factor for correct count of observations when the data table has two factors, such as those produced by pivot_longer, equals to the number of factors

Value

a ggplot object

Examples

```
stat_df <- data.frame(Feature=rep(c("A", "B"), c(20, 30)), Intensity=c(rnorm(20, 2),
rnorm(30, 3)))
p <- draw_boxplot_wo_outlier(stat_df, xc="Feature", yc="Intensity",
Ylab="Signal Intensity")
p
```

---

draw_locus_profile *Plot signal profile around genomic loci*

Description

Plot lines with standard error as the error band

Usage

```
draw_locus_profile(
  plot_df,
  xc = "Position",
  yc = "Intensity",
  cn = "Query",
  sn = "Reference",
  Xlab = "Center",
  Ylab = "Signal Intensity",
  shade = FALSE,
  hl = c(0, 0)
)
```

Arguments

plot_df	a dataframe with column names c(xc, yc, cn, "lower", "upper")
xc	a string denoting column name for values on x-axis
yc	a string denoting column name for numeric data to be plotted
cn	a string denoting column name for sample grouping, like 'Query' or 'Reference'
sn	a string denoting column name for the subject of sample grouping, if 'cn' is 'Query', then 'sn' will be 'Reference'
Xlab	a string for x-axis label
Ylab	a string for y-axis label
shade	logical indicating whether to place a shaded rectangle around the loci
hl	a vector of two integers defining upstream and downstream boundaries of the rectangle

Value

a ggplot object

Note

used by `plot_locus`, `plot_locus_with_random`

Author(s)

Shuye Pu

`draw_matrix_heatmap` *Display matrix as a heatmap*

Description

Make a complex heatmap with column annotations

Usage

```
draw_matrix_heatmap(
  fullMatrix,
  dataName = "geneData",
  labels_col = NULL,
  levels_col = NULL,
  ranking = "Sum",
  ranges = NULL,
  verbose = FALSE
)
```

Arguments

`fullMatrix`	a numeric matrix
`dataName`	the nature of the numeric data
`labels_col`	a vector make column annotation
`levels_col`	factor levels for labels_col, specifying the order of labels_col
`ranking`	method for ranking the rows of the input matrix, options are c("Sum", "Max", "Hierarchical", "None")
`ranges`	a numeric vector with two elements, defining custom range for color ramp, default=NULL, i.e. the range is defined automatically based on the range of fullMatrix
`verbose`	logical, whether to output the input matrix for inspection

Author(s)

Shuye Pu

Examples

```
fullMatrix <- matrix(rnorm(10000), ncol=100)
for(i in 1:80){fullMatrix[i,16:75] <- runif(60) + i}
labels_col <- as.character(seq(1:100))
levels_col <- c("start", "center", "end")
names(labels_col) <- rep(levels_col, c(15, 60, 25))

draw_matrix_heatmap(fullMatrix, dataName="test", labels_col, levels_col)
draw_matrix_heatmap(fullMatrix, dataName="test", labels_col, levels_col,
ranking="Hierarchical")
```

---

draw_mean_se_barplot *Plot barplot for mean with standard error bars*

Description

Plot barplot for mean with standard error bars, no p-value significance levels are displayed, but ANOVA p-value is provided as tag and TukeyHSD test are displayed as caption.

Usage

```
draw_mean_se_barplot(
  stat_df,
  xc = "Feature",
  yc = "Intensity",
  comp = list(c(1, 2)),
  Xlab = xc,
  Ylab = yc,
  Ylim = NULL,
  nf = 1
)
```

Arguments

stat_df	a dataframe with column names c(xc, yc)
xc	a string denoting column name for grouping
yc	a string denoting column name for numeric data to be plotted
comp	a list of vectors denoting pair-wise comparisons to be performed between groups
Xlab	a string for x-axis label
Ylab	a string for y-axis label
Ylim	a numeric vector of two elements, defining custom limits of y-axis
nf	a integer normalizing factor for correct count of observations when the data table has two factors, such as those produced by pivot_longer, equals to the number of factors

Value

a ggplot object

Note

used by plot_locus, plot_locus_with_random

Author(s)

Shuye Pu

Examples

```
stat_df <- data.frame(Feature=rep(c("A", "B"), c(20, 30)),
Intensity=c(rnorm(20, 2), rnorm(30, 3)))
p <- draw_mean_se_barplot(stat_df, xc="Feature", yc="Intensity",
Ylab="Signal Intensity")
p
```

---

draw_rank_plot *Plot cumulative sum or quantile over rank*

Description

Plot cumulative sum over rank as line plot, both cumulative sum and rank are scaled between 0 and 1. This is the same as the fingerprint plot of the deepTools. Quantiles can also be used as y-axis, and values can also be used as x-axis. If the curve is skewed toward ends, the x-axis is truncated for better visualization.

Usage

```
draw_rank_plot(
  stat_df,
  xc = "Feature",
  yc = "Intensity",
  Ylab = yc,
  ecdf = TRUE,
  rank = FALSE
)
```

Arguments

stat_df	a dataframe with column names c(xc, yc)
xc	a string denoting column name for grouping
yc	a string denoting column name for numeric data to be plotted
Ylab	a string for y-axis label
ecdf	logical, indicating using quantile instead of cumulative sum as y-axis
rank	logical, indicating using rank of values instead of value itself as x-axis

Value

a ggplot object

Note

used by plot_reference_locus, plot_reference_locus_with_random

Author(s)

Shuye Pu

Examples

```
stat_df <- data.frame(Feature=rep(c("A", "B"), c(20, 30)),
                      Intensity=c(rlnorm(20, 5, 5), rlnorm(30, 1, 5)))
stat_df1 <- data.frame(Feature=rep(c("A", "B"), c(20, 30)),
                      Height=c(rnorm(20, 5, 5), rnorm(30, 1, 5)))

for(e in c(TRUE, FALSE)){
   for (r in c(TRUE, FALSE)){
      print(draw_rank_plot(stat_df, xc="Feature", yc="Intensity",
      Ylab="Signal Intensity", ecdf=e, rank=r))
      print(draw_rank_plot(stat_df1, xc="Feature", yc="Height",
      Ylab="Height", ecdf=e, rank=r))
   }
}
```

---

draw_region_landmark *Plot genomic region landmark indicator*

Description

Plot a gene centered polygon for demarcating gene and its upstream and downstream regions

Usage

```
draw_region_landmark(featureNames, vx, xmax)
```

Arguments

featureNames	a string vector giving names of sub-regions
vx	a vector on integers denoting the x coordinates of start of each sub-region
xmax	an integer denoting the left most boundary

Value

a ggplot object

Note

used by plot_3parts_metagene, plot_5parts_metagene, plot_region

Author(s)

Shuye Pu

draw_region_name *Plot genomic region names*

Description

Plot sub-region labels under the landmark

Usage

```
draw_region_name(featureNames, scaled_bins, xmax)
```

Arguments

featureNames	a string vector giving names of sub-regions
scaled_bins	a vector on integers denoting the length of each sub-region
xmax	an integer denoting the left most boundary

Value

a ggplot object

Note

used by plot_3parts_metagene, plot_5parts_metagene, plot_region

Author(s)

Shuye Pu

draw_region_profile *Plot signal profile in genomic regions*

Description

Plot lines with standard error as the error band

Usage

```
draw_region_profile(
  plot_df,
  xc = "Position",
  yc = "Intensity",
  cn = "Query",
  sn = "Reference",
  Ylab = "Signal Intensity",
  vx
)
```

Arguments

plot_df	a dataframe with column names c(xc, yc, cn, "lower", "upper")
xc	a string denoting column name for values on x-axis
yc	a string denoting column name for numeric data to be plotted
cn	column name in plot_df for query samples grouping
sn	column name in plot_df for subject name to be shown in the plot title
Ylab	a string for Y-axis label
vx	a vector on integers denoting the x coordinates of start of each sub-region

Value

a ggplot object

Note

used by plot_3parts_metagene, plot_5parts_metagene, plot_region

Author(s)

Shuye Pu

---

draw_stacked_profile *Plot signal profile around start, center, and end of genomic regions*

Description

Plot lines with standard error as the error band, also plots number of regions having non-zero signals

Usage

```
draw_stacked_profile(
  plot_df,
  xc = "Position",
  yc = "Intensity",
  cn = "Query",
  ext = c(0, 0, 0, 0),
  hl = c(0, 0, 0, 0),
  atitle = "title",
  insert = 0,
  Ylab = "Signal Intensity",
  shade = FALSE,
  stack = TRUE
)
```

Arguments

plot_df	a dataframe with column names c(xc, yc, cn, "Interval", "lower", "upper")
xc	a string denoting column name for values on x-axis
yc	a string denoting column name for numeric data to be plotted
cn	a string denoting column name for grouping
ext	a vector of 4 integers denoting upstream and downstream extension around start and end
hl	a vector of 4 integers defining upstream and downstream boundaries of the rectangle for start and end
atitle	a string for the title of the plot
insert	a integer denoting the width of the center region
Ylab	a string for y-axis label
shade	logical, indicating whether to place a shaded rectangle around the point of interest
stack	logical, indicating whether to plot the number of valid (non-zero) data points in each bin

Value

a ggplot object

Note

used by plot_start_end, plot_start_end_with_random

Author(s)

Shuye Pu

---

effective_size *Normalize sample library size to effective size*

Description

This is a helper function for handle_input. edgeR::calcNormFactors function is used to estimate normalizing factors, which is used to multiply library sizes. The function only works for human genome only at present.

Usage

```
effective_size(outlist, outRle, genome = "hg19", nc = 2, verbose = FALSE)
```

Arguments

outlist	a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the library size, 'type' is the input file type, 'weight' is the name of the metadata column
outRle	logical, indicating whether the output is a list of RleList objects or GRanges objects
genome	a string denoting the genome name and version
nc	integer, number of cores for parallel processing
verbose	logical, whether to output additional information

Value

a list object with four elements ('query', 'size', 'type', 'weight'), with the 'size' element modified.

Author(s)

Shuye Pu

---

extract_longest_tx *Extract the longest transcript for each protein-coding genes*

Description

Gene level computations require selecting one transcript per gene to avoid bias by genes with multiple isoforms. In ideal case, the most abundant transcript (principal or canonical isoform) should be chosen. However, the most abundant isoform may vary depending on tissue type or physiological condition, the longest transcript is usually the principal isoform, and alternatively spliced isoforms are not. This method get the longest transcript for each gene. The longest transcript is defined as the isoform that has the longest transcript length. In case of tie, the one with longer CDS is selected. If the lengths of CDS tie again, the transcript with smaller id is selected arbitrarily.

Usage

```
extract_longest_tx(txdb, plot = FALSE)
```

Arguments

txdb	a TxDb object defined in the GenomicFeatures package
plot	logical, indicating whether feature length plots should be generated

Value

a dataframe of transcript information with the following columns: "tx_id tx_name gene_id nexon tx_len cds_len utr5_len utr3_len"

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql",
package="GenomicPlot"))
longestTx <- extract_longest_tx(txdb, plot=FALSE)
```

`filter_by_nonoverlaps_stranded`
Filter GRanges by nonoverlaps in stranded way

Description

This function reports all query GRanges that do not overlaps GRanges in subject. Strand information is used to define overlap.

Usage

```
filter_by_nonoverlaps_stranded(query, subject)
```

Arguments

`query`	a GRanges object
`subject`	a GRanges object

Value

a GRanges object

Author(s)

Shuye Pu

`filter_by_overlaps_stranded`
Filter GRanges by overlaps in a stranded way

Description

This function reports all query GRanges that have overlaps in subject GRanges. Strand information is used to define overlap.

Usage

```
filter_by_overlaps_stranded(query, subject, maxgap = -1L, minoverlap = 0L)
```

Arguments

query	a GRanges object
subject	a GRanges object
maxgap	an integer denoting the distance that define overlap
minoverlap	The minimum amount of overlap between intervals as a single integer greater than 0. If you modify this argument, maxgap must be held fixed.

Value

a GRanges object

Author(s)

Shuye Pu

---

format_genomic_coordinates

Format genomic coordinates in GRanges or GRrangesList as strings used in igv

---

Description

This function takes a GRanges or GRangesList object, and transform each range into a string

Usage

```
format_genomic_coordinates(x)
```

Arguments

x	a GRanges or GRangesList object

Value

a vector of strings in the format of 'chr:start-end(strand)'

Author(s)

Shuye Pu

Examples

```
gr1 <- GenomicRanges::GRanges("chr2", IRanges::IRanges(3, 6))
gr2 <- GenomicRanges::GRanges(c("chr1", "chr1"), IRanges::IRanges(c(7,13), width=3),
 strand=c("+", "-"))
gr3 <- GenomicRanges::GRanges(c("chr1", "chr2"), IRanges::IRanges(c(1, 4), c(3, 9)),
 strand="-")

grl <- GenomicRanges::GRangesList(gr1= gr1, gr2=gr2, gr3=gr3)
grl

out <- format_genomic_coordinates(grl)
```

```
cat(out)
```

`gene2tx` *Translate gene names to transcript ids using a GTF file for a subset of genes*

Description

Given a list of gene names in a file or in a character vector, turn them into a vector of transcript ids.

Usage

```
gene2tx(gtfFile, geneList, geneCol = 1)
```

Arguments

`gtfFile`	path to a GTF file
`geneList`	path to a tab-delimited text file with one gene name on each line, or a character vector of gene names (eg. RPRD1B)
`geneCol`	the position of the column that containing gene names in the case that geneList is a file

Value

a vector of transcript ids (eg. ENST00000577222.1)

Author(s)

Shuye Pu

`get_genomic_feature_coordinates` *Extract genomic features from TxDb object*

Description

Extract genomic coordinates and make bed or bed 12 files from a TxDb object for a variety of annotated genomic features. The output of this function is a list. The first element of the list is a GRanges object that provide the start and end information of the feature. The second element is a GRangesList providing information for sub-components. The third element is the name of a bed file. For "utr3", "utr5", "cds" and "transcript", the GRanges object denotes the start and end of the feature in one transcript, and the range is named by the transcript id and may span introns; the GrangesList object is a list of exons comprising each feature and indexed on transcript id. The bed file is in bed12 format. For "exon" and "intron", the GRanges object denotes unnamed ranges of individual exon and intron, and the GrangesList object is a list of exons or introns belonging to one transcript and indexed on transcript id. The bed file is in bed6 format. For "gene", both GRanges object and GRangesList object have the same ranges and names. The bed file is in bed6 format.

Usage

```
get_genomic_feature_coordinates(
  txdb,
  featureName,
  featureSource = NULL,
  export = FALSE,
  longest = FALSE,
  protein_coding = FALSE
)
```

Arguments

txdb	a TxDb object defined in the GenomicFeatures package
featureName	one of the gene feature in c("utr3", "utr5", "cds", "intron", "exon", "transcript", "gene")
featureSource	the name of the gtf/gff3 file or the online database from which txdb is derived, used as name of output file
export	logical, indicating if the bed file should be produced
longest	logical, indicating whether the output should be limited to the longest transcript of each gene
protein_coding	logical, indicating whether to limit to protein_coding genes

Value

a list of three objects, the first is a GRanges object, the second is a GRangesList object, the last is the output file name if export is TRUE

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql",
package="GenomicPlot"))
output <- get_genomic_feature_coordinates(txdb, featureName="cds", featureSource="gencode",
export=FALSE, longest=TRUE, protein_coding=TRUE)
```

---

get_targeted_genes *Get the number of peaks overlapping each feature of all protein-coding genes*

Description

Annotate each peak with genomic features based on overlap, and produce summary statistics for distribution of peaks in features of protein-coding genes.

Usage

```
get_targeted_genes(peak, features, stranded = TRUE)
```

Arguments

peak	a GRanges object defining query ranges
features	a GRangesList object representing genomic features
stranded	logical, indicating whether the overlap should be strand-specific

Value

a list object

Note

used in plot_peak_annotation

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))
f <- get_txdb_features(txdb, dsTSS=100, fiveP=0, threeP=1000)

p <- RCAS::importBed(system.file("extdata", "test_chip_peak_chr19.bed", package="GenomicPlot"))
ann <- get_targeted_genes(peak=p, features=f, stranded=FALSE)

pp <- RCAS::importBed(system.file("extdata", "test_clip_peak_chr19.bed", package="GenomicPlot"))
ann <- get_targeted_genes(peak=pp, features=f, stranded=TRUE)
```

---

get_txdb_features *Get genomic coordinates of features of protein-coding genes*

Description

Get genomic coordinates of promoter, 5'UTR, CDS, 3'UTR, TTS and intron for the longest transcript of protein-coding genes. The range of promoter is defined by fiveP and dsTSS upstream and downstream TSS, respectively, the TTS ranges from the 3' end of the gene to threeP downstream, or the start of a downstream gene, whichever is closer.

Usage

```
get_txdb_features(txdb, fiveP = -1000, dsTSS = 300, threeP = 1000)
```

Arguments

txdb	a TxDb object defined in the GenomicFeatures package
fiveP	extension upstream of the 5' boundary of genes
dsTSS	range of promoter extending downstream of TSS
threeP	extension downstream of the 3' boundary of genes

Value

a GRangesList object

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql",
package="GenomicPlot"))

f <- get_txdb_features(txdb, dsTSS=100, fiveP=-100, threeP=100)
```

---

gr2df *Convert GRanges to dataframe*

Description

Convert GRanges object with metacolumns to dataframe

Usage

```
gr2df(gr)
```

Arguments

gr	a GRanges object

Value

a dataframe

Author(s)

Shuye Pu

Examples

```
gr2 <- GenomicRanges::GRanges(c("chr1", "chr1"), IRanges::IRanges(c(7,13), width=3),
 strand=c("+", "-"))
GenomicRanges::mcols(gr2) <- data.frame(score=c(0.3, 0.9), cat=c(TRUE, FALSE))
df2 <- gr2df(gr2)
```

handle_bam *Handle files in bam format*

Description

This is a function for read NGS reads data in bam format, store the input data in a list of GRanges objects or RleList objects. For paired-end reads, only take the second read in a pair, assuming which is the sense read for strand-specific RNAseq.

Usage

```
handle_bam(inputFile, handleInputParams = NULL, verbose = FALSE)
```

Arguments

inputFile	a string denoting path to the input file
handleInputParams	a list of parameters for handle_input
verbose	logical, whether to output additional information

Details

The reads are filtered using mapq score >= 10 by default, only mapped reads are counted towards library size.

Value

a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the library size, 'type' is the input file type, weight' is the name of the metadata column to be used as weight for coverage calculation

Author(s)

Shuye Pu

handle_bed *Handle files in bed\narrowPeak\broadPeak format*

Description

This is a function for read peaks data in bed format, store the input data in a list of GRanges objects or RleList objects.

Usage

```
handle_bed(inputFile, handleInputParams = NULL, verbose = FALSE)
```

Arguments

inputFile	a string denoting path to the input file
handleInputParams	a list of parameters for handle_input
verbose	logical, whether to output additional information

Value

a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the library size, 'type' is the input file type, 'weight' is the name of the metadata column to be used as weight for coverage calculation

Author(s)

Shuye Pu

---

handle_bw *Handle files in bw|bigwig|bigWig|BigWig|BW|BIGWIG format*

Description

This is a function for read NGS coverage data in bigwig format, store the input data in a list of GRanges objects or RleList objects. The input bw file can be stranded or non-stranded. Library size is calculate as the sum of all coverage.

Usage

```
handle_bw(inputFile, handleInputParams, verbose = FALSE)
```

Arguments

inputFile	a string denoting path to the input file
handleInputParams	a list of parameters for handle_input
verbose	logical, whether to output additional information

Details

For stranded files, forward and reverse strands are stored in separate files, with '+' or 'p' in the forward strand file name and '-' or 'm' in the reverse strand file name.

Value

a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the estimated library size, 'type' is the input file type, weight' is the name of the metadata column to be used as weight for coverage calculation

Author(s)

Shuye Pu

handle_input *Hand input of NGS data with various formats*

Description

This is a wrapper function for read NGS data in different file formats, store the input data in a list of GRanges objects or RleList objects. File names end in bed|bam|bw|bigwig|bigWig|BigWig|BW|BIGWIG are recognized, and a list of files with mixed formats are allowed.

Usage

```
handle_input(inputFiles, handleInputParams = NULL, verbose = FALSE, nc = 2)
```

Arguments

inputFiles	a vector of strings denoting file names
handleInputParams	a list with the following elements: 'CLIP_reads' logical, indicating if the bam reads should be shifted to the -1 position at the 5' of the reads. 'fix_width' an integer defines how long should the reads should be extended to. 'fix_point' a string in c("start", "end", "center") denoting the anchor point for extension. 'useScore' logical, indicating whether the 'score' column of the bed file should be used in calculation of coverage. 'outRle' logical, indicating whether the output should be RleList objects or GRanges objects. 'norm' logical, indicating whether the output RleList should be normalized to RPM using library sizes. 'genome' a string denoting the genome name and version. 'useSizeFactor' logical, indicating whether the library size should be adjusted with a size factor, using the 'calcNormFactors' function in the edgeR package
verbose	logical, whether to output additional information
nc	integer, number of cores for parallel processing

Details

when 'useScore' is TRUE, the score column of the bed file will be used in the metadata column 'score' of the GRanges object, or the 'Values' field of the RleList object. Otherwise the value 1 will be used instead. When the intended use of the input bed is a reference feature, both 'useScore' and 'outRle' should be set to FALSE.

Value

a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the library size, 'type' is the input file type, 'weight' is the name of the metadata column to be used as weight for coverage calculation

Author(s)

Shuye Pu

Examples

```
queryFiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryFiles) <- "query"

inputFiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputFiles) <- "input"

handleInputParams <- list(CLIP_reads=TRUE, fix_width=0, fix_point="start", norm=TRUE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

out_list <- handle_input(inputFiles=c(queryFiles, inputFiles),
handleInputParams=handleInputParams, verbose=TRUE, nc=2)
```

`handle_wig` *Handle files in wig format*

Description

This is a function for read NGS coverage data in wig format, store the input data in a list of GRanges objects or RleList objects. The input wig file can be stranded or non-stranded. Library size is calculate as the sum of all coverage.

Usage

```
handle_wig(inputFile, handleInputParams, verbose = FALSE)
```

Arguments

`inputFile`	a string denoting path to the input file
`handleInputParams`	a list of parameters for `handle_input`
`verbose`	logical, whether to output additional information

Details

For stranded files, forward and reverse strands are stored in separate files, with '+' or 'p' in the forward strand file name and '-' or 'm' in the reverse strand file name.

Value

a list object with four elements, 'query' is a list GRanges objects or RleList objects, 'size' is the library size, 'type' is the input file type, 'weight' is the name of the metadata column to be used as weight for coverage calculation

Author(s)

Shuye Pu

make_subTxDb_from_GTF *Make TxDb object from a GTF file for a subset of genes*

Description

Make a partial TxDb object given a GTF file and a list of gene names in a file or in a character vector.

Usage

```
make_subTxDb_from_GTF(gtfFile, geneList, geneCol = 1)
```

Arguments

gtfFile	path to a GTF file
geneList	path to a tab-delimited text file with one gene name on each line, or a character vector of gene names
geneCol	the position of the column that containing gene names in the case that geneList is a file

Value

a TxDb object

Author(s)

Shuye Pu

overlap_pair *Plot two-sets Venn diagram*

Description

This is a helper function for Venn diagram plot. A Venn diagram is plotted as output.

Usage

```
overlap_pair(apair, overlap_fun)
```

Arguments

apair	a list of two vectors
overlap_fun	the name of the function that defines overlap, depending on the type of object in the vectors.

Author(s)

Shuye Pu

overlap_quad *Plot four-sets Venn diagram*

Description

This is a helper function for Venn diagram plot. A Venn diagram is plotted as output.

Usage

```
overlap_quad(aquad, overlap_fun)
```

Arguments

aquad	a list of four vectors
overlap_fun	the name of the function that defines overlap

Author(s)

Shuye Pu

overlap_triple *Plot three-sets Venn diagram*

Description

This is a helper function for Venn diagram plot. A Venn diagram is plotted as output.

Usage

```
overlap_triple(atriple, overlap_fun)
```

Arguments

atriple	a list of three vectors
overlap_fun	the name of the function that defines overlap

Author(s)

Shuye Pu

`parallel_binnedAverage` *Parallel execution of binnedAverage*

Description

Function for parallel computation of binnedAverage function in the GenomicRanges package

Usage

```
parallel_binnedAverage(Rle_list, tileBins, nc = 2)
```

Arguments

`Rle_list`	a list of RleList objects.
`tileBins,`	a GRanges object of tiled genome
`nc`	integer, number of cores for parallel processing

Value

a list of numeric vectors

Author(s)

Shuye Pu

`parallel_countOverlaps` *Parallel execution of countOverlaps*

Description

Function for parallel computation of countOverlaps function in the GenomicRanges package

Usage

```
parallel_countOverlaps(grange_list, tileBins, nc = 2, switch = FALSE)
```

Arguments

`grange_list`	a list of GRanges objects.
`tileBins,`	a GRanges object of tiled genome
`nc`	integer, number of cores for parallel processing
`switch,`	logical, switch the order of query and feature

Value

a list of numeric vectors

Author(s)

Shuye Pu

`parallel_scoreMatrixBin` *Parallel execution of scoreMatrixBin on a huge target windows object split into chunks*

Description

Function for parallel computation of scoreMatrixBin. The 'windows' parameter of the scoreMatrixBin method is split into 5 chunks, and scoreMatrixBin is called on each chunk simultaneously to speed up the computation.

Usage

```
parallel_scoreMatrixBin(
  queryRegions,
  windowRs,
  bin_num,
  bin_op,
  weight_col,
  stranded,
  nc = 2
)
```

Arguments

`queryRegions,`	a RleList object or Granges object providing input for the 'target' parameter of the scoreMatrixBin method
`windowRs,`	a single GRangesList object.
`bin_num,`	number of bins the windows should be divided into
`bin_op,`	operation on the signals in a bin, a string in c("mean", "max", "min", "median", "sum") is accepted.
`weight_col,`	if the queryRegions is a GRanges object, a numeric column in meta data part can be used as weights.
`stranded,`	logical, indicating if the strand of the windows should be considered to determine upstream and downstream
`nc,`	an integer denoting the number of cores requested, 2 is the default number that is allowed by CRAN but 5 gives best trade-off between speed and space

Value

a numeric matrix

Author(s)

Shuye Pu

`plot_3parts_metagene` *Plot promoter, gene body and TTS*

Description

Plot reads or peak Coverage/base/gene of samples in the query files around genes. The upstream and downstream windows flanking genes can be given separately, the parameter 'meta' controls if gene or metagene plots are generated. If Input files are provided, ratio over Input is computed and displayed as well.

Usage

```
plot_3parts_metagene(
  queryFiles,
  gFeatures,
  inputFiles = NULL,
  scale = FALSE,
  verbose = FALSE,
  Ylab = "Coverage/base/gene",
  handleInputParams = NULL,
  smooth = FALSE,
  stranded = TRUE,
  outPrefix = NULL,
  heatmap = FALSE,
  rmOutlier = FALSE,
  heatRange = NULL,
  transform = NA,
  nc = 2
)
```

Arguments

`queryFiles`	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
`gFeatures`	genomic features as output of the function 'prepare_3parts_genomic_features'
`inputFiles`	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
`scale`	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
`verbose`	logical, whether to output additional information (data used for plotting or statistical test results)
`Ylab`	a string for y-axis label
`handleInputParams`	a list of parameters for `handle_input`
`smooth`	logical, indicating whether the line should smoothed with a spline smoothing algorithm
`stranded`	logical, indicating whether the strand of the feature should be considered
`outPrefix`	a string specifying output file prefix for plots (outPrefix.pdf)

heatmap	logical, indicating whether a heatmap of the score matrix should be generated
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
heatRange	a numerical vector of two elements, defining range for heatmap color ramp generation
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
nc	integer, number of cores for parallel processing

Value

a dataframe containing the data used for plotting

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql",
                              package="GenomicPlot"))

queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"

inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"

gfeatures <- prepare_3parts_genomic_features(txdb, featureName="transcript", meta=TRUE,
nbins=100, fiveP=-1000, threeP=1000, longest=TRUE, protein_coding=TRUE, verbose=FALSE)

handleInputParams <- list(CLIP_reads=FALSE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

df <- plot_3parts_metagene(queryFiles=queryfiles, gFeatures=gfeatures, inputFiles
                           =inputfiles, scale=FALSE, verbose=TRUE,
                           Ylab="Coverage/base/gene", handleInputParams
                           =handleInputParams, smooth=TRUE, stranded=TRUE,
                          outPrefix=NULL, heatmap=TRUE, rmOutlier=FALSE, heatRange=NULL,
                           transform=NA, nc=2)
```

---

plot_5parts_metagene *Plot promoter, 5'UTR, CDS, 3'UTR and TTS*

Description

Plot reads or peak Coverage/base/gene of samples in the query files around genes. The upstream and downstream windows flanking genes can be given separately, metagene plots are generated with 5'UTR, CDS and 3'UTR segments. The length of each segments are prorated according to the median length of each segments. If Input files are provided, ratio over Input is computed and displayed as well.

Usage

```
plot_5parts_metagene(
  queryFiles,
  gFeatures_list,
  inputFiles = NULL,
  handleInputParams = NULL,
  verbose = FALSE,
  transform = NA,
  smooth = FALSE,
  scale = FALSE,
  stranded = TRUE,
  outPrefix = NULL,
  heatmap = FALSE,
  heatRange = NULL,
  rmOutlier = FALSE,
  Ylab = "Coverage/base/gene",
  nc = 2
)
```

Arguments

queryFiles	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
gFeatures_list	a list of genomic features as output of the function 'prepare_5parts_genomic_features'
inputFiles	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
handleInputParams	a list of parameters for handle_input
verbose	logical, indicating whether to output additional information (data used for plotting or statistical test results)
transform	logical, whether to log2 transform the matrix
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
stranded	logical, indicating whether the strand of the feature should be considered
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)
heatmap	logical, indicating whether a heatmap of the score matrix should be generated
heatRange	a numerical vector of two elements, defining range for heatmap color ramp generation
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
Ylab	a string for y-axis label
nc	integer, number of cores for parallel processing

Value

a dataframe containing the data used for plotting

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))

queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"

inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"

gfeatures <- prepare_5parts_genomic_features(txdb, meta=TRUE, nbins=100, fiveP=-1000,
threeP=1000, longest=TRUE, verbose=FALSE)

handleInputParams <- list(CLIP_reads=FALSE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

df <- plot_5parts_metagene(queryFiles=queryfiles, gFeatures=list("metagene"=gfeatures),
inputFiles=inputfiles, scale=FALSE, verbose=TRUE, Ylab="Coverage/base/gene",
handleInputParams=handleInputParams, smooth=TRUE,  stranded=TRUE, outPrefix=NULL, heatmap=TRUE,
 rmOutlier=FALSE, heatRange=NULL, transform=NA, nc=2)
```

---

plot_bam_correlation *Plot correlation of bam files*

Description

plot correlation in reads coverage distributions along the genome for bam files

Usage

```
plot_bam_correlation(
  bamfiles,
  binSize = 1e+06,
  outPrefix = NULL,
  handleInputParams = NULL,
  verbose = FALSE,
  nc = 2
)
```

Arguments

bamfiles	a named vector of strings denoting file names
binSize	an integer denoting the tile width for tiling the genome, default 1000000
outPrefix	a string denoting output file name in pdf format
handleInputParams	a list of parameters for handle_input
verbose	logical, indicating whether to output additional information
nc	integer, number of cores for parallel processing

Examples

```
queryFiles <- c(system.file("extdata", "chip_treat_chr19.bam", package="GenomicPlot"),
                system.file("extdata", "treat_chr19.bam", package="GenomicPlot"))
names(queryFiles) <- c("chip_query", "clip_query")

inputFiles <- c(system.file("extdata", "chip_input_chr19.bam", package="GenomicPlot"),
                 system.file("extdata", "input_chr19.bam", package="GenomicPlot"))
names(inputFiles) <- c("chip_input", "clip_input")

handleInputParams <- list(CLIP_reads=FALSE, fix_width=0, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=FALSE, useSizeFactor=FALSE, genome="hg19")

plot_bam_correlation(bamfiles=c(queryFiles, inputFiles), binSize=10000, outPrefix=NULL,
   handleInputParams=handleInputParams, nc=2)
```

`plot_locus` *Plot signal around custom genomic loci*

Description

Plot reads or peak Coverage/base/gene of samples in the query files around reference locus (start, end or center of a genomic region) defined in the centerFiles. The upstream and downstream windows flanking loci can be given separately, a smaller window can be defined to allow statistical comparisons between samples for the same reference, or between references for a given sample. If Input files are provided, ratio over Input is computed and displayed as well.

Usage

```
plot_locus(
  queryFiles,
  centerFiles,
  txdb = NULL,
  ext = c(-100, 100),
  hl = c(0, 0),
  shade = TRUE,
  smooth = FALSE,
  handleInputParams = NULL,
  verbose = FALSE,
  binSize = 10,
  refPoint = "center",
  Xlab = "Center",
  Ylab = "Coverage/base/gene",
  inputFiles = NULL,
  stranded = TRUE,
  heatmap = TRUE,
  scale = FALSE,
  outPrefix = NULL,
  rmOutlier = FALSE,
  transform = NA,
  statsMethod = "wilcox.test",
```

```
  heatRange = NULL,
  nc = 2
)
```

Arguments

queryFiles	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
centerFiles	a named vector of reference file names or genomic features in c("utr3", "utr5", "cds", "intron", "exon", "transcript", "gene"). The file should be in .bed format only
txdb	a TxDb object defined in the GenomicFeatures package. Default NULL, needed only when genomic features are used in the place of centerFiles.
ext	a vector of two integers defining upstream and downstream boundaries of the plot window, flanking the reference locus
hl	a vector of two integers defining upstream and downstream boundaries of the highlight window, flanking the reference locus
shade	logical indicating whether to place a shaded rectangle around the point of interest
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
handleInputParams	a list of parameters for handle_input
verbose	logical, indicating whether to output additional information (data used for plotting or statistical test results)
binSize	an integer defines bin size for intensity calculation
refPoint	a string in c("start", "center", "end")
Xlab	a string denotes the label on x-axis
Ylab	a string for y-axis label
inputFiles	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
stranded	logical, indicating whether the strand of the feature should be considered
heatmap	logical, indicating whether a heatmap of the score matrix should be generated
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
statsMethod	a string in c("wilcox.test", "t.test"), for pair-wise group comparisons
heatRange	a numerical vector of two elements, defining range for heatmap color ramp generation
nc	integer, number of cores for parallel processing

Value

a list of two dataframes containing the data used for plotting and for statistical testing

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))

queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"

inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"

centerfiles <- system.file("extdata", "test_clip_peak_chr19.bed", package="GenomicPlot")
names(centerfiles) <- "clipPeak"

handleInputParams <- list(CLIP_reads=TRUE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

df <- plot_locus(queryFiles=queryfiles, centerFiles=c(centerfiles, "intron"), txdb=txdb,
ext=c(-200,200), hl=c(-20, 20), shade=TRUE, smooth=FALSE, handleInputParams=handleInputParams,
verbose=TRUE, binSize=10, refPoint="center", Xlab="Center", Ylab="Coverage/base/gene",
inputFiles=inputfiles, stranded=TRUE, heatmap=TRUE, scale=FALSE, outPrefix=NULL,
rmOutlier=FALSE, transform=NA, statsMethod="wilcox.test", heatRange=c(0, 0.3), nc=2)
```

---

plot_locus_with_random

Plot signal around custom genomic loci and random loci for comparison

---

Description

Plot reads or peak Coverage/base/gene of samples in the query files around reference locus defined in the centerFiles. The upstream and downstream windows flanking loci can be given separately, a smaller window can be defined to allow statistical comparisons between reference and random loci. The loci are further divided into sub-groups that are overlapping with c("5'UTR", "CDS", "3'UTR"), "unrestricted" means all loci regardless of overlapping.

Usage

```
plot_locus_with_random(
  queryFiles,
  centerFiles,
  txdb,
  ext = c(-200, 200),
  hl = c(-100, 100),
  shade = FALSE,
  handleInputParams = NULL,
```

```
  verbose = FALSE,
  smooth = FALSE,
  transform = NA,
  binSize = 10,
  refPoint = "center",
  Xlab = "Center",
  Ylab = "Coverage/base/gene",
  inputFiles = NULL,
  stranded = TRUE,
  scale = FALSE,
  outPrefix = NULL,
  rmOutlier = FALSE,
  n_random = 1,
  statsMethod = "wilcox.test",
  nc = 2
)
```

Arguments

queryFiles	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
centerFiles	a vector of reference file names. The file should be .bed format only
txdb	a TxDb object defined in the GenomicFeatures package
ext	a vector of two integers defining upstream and downstream boundaries of the plot window, flanking the reference locus
hl	a vector of two integers defining upstream and downstream boundaries of the highlight window, flanking the reference locus
shade	logical indicating whether to place a shaded rectangle around the point of interest
handleInputParams	a list of parameters for handle_input
verbose	logical, indicating whether to output additional information (data used for plotting or statistical test results)
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
binSize	an integer defines bin size for intensity calculation
refPoint	a string in c("start", "center", "end")
Xlab	a string denotes the label on x-axis
Ylab	a string for y-axis label
inputFiles	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
stranded	logical, indicating whether the strand of the feature should be considered
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)

rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
n_random	an integer denotes the number of randomization should be formed
statsMethod	a string in c("wilcox.test", "t.test"), for pair-wise groups comparisons
nc	integer, number of cores for parallel processing

Value

a dataframe containing the data used for plotting

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))

queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"

inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"

centerfiles <- system.file("extdata", "test_clip_peak_chr19.bed", package="GenomicPlot")
names(centerfiles) <- "clipPeak"

handleInputParams <- list(CLIP_reads=TRUE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

df <- plot_locus_with_random(queryFiles=queryfiles, centerFiles=c(centerfiles), txdb=txdb,
ext=c(-200,200), hl=c(-20, 20), shade=TRUE, smooth=TRUE, handleInputParams=handleInputParams,
verbose=TRUE, binSize=10, refPoint="center", Xlab="Center", Ylab="Coverage/base/gene",
inputFiles=inputfiles, stranded=TRUE, scale=FALSE, outPrefix=NULL, rmOutlier=FALSE,
transform=NA, statsMethod="wilcox.test", nc=2)
```

---

plot_overlap_bed *Plot Venn diagrams depicting overlap of genomic regions*

Description

This function takes a list of bed file names, and produce a Venn diagram

Usage

```
plot_overlap_bed(
  bedList,
  outPrefix = NULL,
  handleInputParams = NULL,
  pairOnly = TRUE,
  stranded = TRUE,
  verbose = FALSE
)
```

Arguments

bedList	a named list of bed files, with length = 2, 3 or 4
outPrefix	a string for plot file name
handleInputParams	a list of parameters for handle_input
pairOnly	logical, indicating whether only pair-wise overlap is desired
stranded	logical, indicating whether the feature is stranded. For nonstranded feature, only "*" is accepted as strand
verbose	logical, indicating whether to output additional information

Author(s)

Shuye Pu

Examples

```
queryFiles <- c(system.file("extdata", "test_chip_peak_chr19.narrowPeak", package="GenomicPlot"),
system.file("extdata", "test_chip_peak_chr19.bed", package="GenomicPlot"),
system.file("extdata", "test_clip_peak_chr19.bed", package="GenomicPlot"))
names(queryFiles) <- c("narrowPeak", "summitPeak", "clipPeak")

handleBedParams <- list(fix_width=100, fix_point="center", useScore=FALSE, outRle=FALSE,
                        CLIP_reads=FALSE, norm=FALSE, useSizeFactor=FALSE, genome="hg19")

plot_overlap_bed(bedList=queryFiles, handleInputParams=handleBedParams, pairOnly=FALSE,
stranded=FALSE)
```

---

plot_overlap_genes *Plot Venn diagrams depicting overlap of gene lists*

Description

This function takes a list of (at most 3) tab-delimited file names, and produce a Venn diagram

Usage

```
plot_overlap_genes(fileList, columnList, pairOnly = TRUE, outPrefix = NULL)
```

Arguments

fileList,	a named list of tab-dlimited files
columnList	a vector of integers denoting the columns that have gene names in the list of files
pairOnly,	logical, indicating whether only pair-wise overlap is desired
outPrefix,	a string for plot file name

Value

a list of vectors of gene names

Author(s)

Shuye Pu

`plot_peak_annotation` *Annotate peaks with genomic features and genes*

Description

Produce a table of transcripts targeted by peaks, and generate plots for target gene types, and peak distribution in genomic features

Usage

```
plot_peak_annotation(
  peakFile,
  gtfFile,
  handleInputParams = NULL,
  fiveP = -1000,
  dsTSS = 300,
  threeP = 1000,
  simple = FALSE,
  outPrefix = NULL,
  verbose = FALSE
)
```

Arguments

`peakFile`	a string denoting the peak file name, only .bed format is allowed
`gtfFile`	path to a gene annotation gtf file with gene_biotype field
`handleInputParams`	a list of parameters for `handle_input`
`fiveP`	extension out of the 5' boundary of genes for defining promoter: fiveP TSS + dsTSS
`dsTSS`	extension downstream of TSS for defining promoter: fiveP TSS + dsTSS
`threeP`	extension out of the 3' boundary of genes for defining termination region: -0 TTS + threeP
`simple`	logical, indicating whether 5'UTR and 3'UTR are annotated in the gtffile
`outPrefix`	a string denoting output file name in pdf format
`verbose,`	logical, to indicate whether to write the annotation results to a file

Value

a list of two dataframes, 'annotation' is the annotation per peak, 'stat' is the summary stats for pie chart

Author(s)

Shuye Pu

Examples

```
gtfFile <- system.file("extdata", "gencode.v19.annotation_chr19.gtf", package="GenomicPlot")

centerFile <- system.file("extdata", "test_chip_peak_chr19.bed", package="GenomicPlot")
names(centerFile) <- c("summitPeak")

handleBedparams <- list(fix_width=0, fix_point="center", useScore=FALSE, outRle=FALSE,
CLIP_reads=FALSE, norm=FALSE, useSizeFactor=FALSE, genome="hg19")

plot_peak_annotation(peakFile=centerFile, gtfFile=gtfFile, handleInputParams=handleBedparams,
fiveP=-2000, dsTSS=200, threeP=2000, simple=FALSE)
```

`plot_region` *Plot signal inside as well as around custom genomic regions*

Description

Plot reads or peak Coverage/base/gene of samples in the query files inside regions defined in the centerFiles. The upstream and downstream flanking windows can be given separately. If Input files are provided, ratio over Input is computed and displayed as well.

Usage

```
plot_region(
  queryFiles,
  centerFiles,
  txdb = NULL,
  regionName = "region",
  inputFiles = NULL,
  nbins = 100,
  handleInputParams = NULL,
  verbose = FALSE,
  scale = FALSE,
  heatmap = FALSE,
  fiveP = -1000,
  threeP = 1000,
  smooth = FALSE,
  stranded = TRUE,
  transform = NA,
  outPrefix = NULL,
  rmOutlier = FALSE,
  heatRange = NULL,
  Ylab = "Coverage/base/gene",
  statsMethod = "wilcox.test",
  nc = 2
)
```

Arguments

`queryFiles`	a named vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed

centerFiles	a named vector of reference file names or genomic features in c("utr3", "utr5", "cds", "intron", "exon", "transcript", "gene"). The file should be in .bed format only
txdb	a TxDb object defined in the GenomicFeatures package. Default NULL, needed only when genomic features are used in the place of centerFiles.
regionName	a string specifying the name of the center region in the plots
inputFiles	a named vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
nbins	an integer defines the total number of bins
handleInputParams	a list of parameters for handle_input
verbose	logical, indicating whether to output additional information (data used for plotting or statistical test results)
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
heatmap	logical, indicating whether a heatmap of the score matrix should be generated
fiveP	an integer, indicating extension out or inside of the 5' boundary of gene by negative or positive number
threeP	an integer, indicating extension out or inside of the 5' boundary of gene by positive or negative number
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
stranded	logical, indicating whether the strand of the feature should be considered
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
heatRange	a numerical vector of two elements, defining range for heatmap color ramp generation
Ylab	a string for y-axis label
statsMethod	a string in c("wilcox.test", "t.test"), for pair-wise group comparisons
nc	integer, number of cores for parallel processing

Value

a dataframe containing the data used for plotting

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))
queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"
inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"
centerfiles <- system.file("extdata", "test_chip_peak_chr19.narrowPeak", package="GenomicPlot")
names(centerfiles) <- "narrowPeak"
op <- NULL
handleInputParams <- list(CLIP_reads=FALSE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

plot_region(queryFiles=queryfiles, centerFiles=centerfiles, txdb=NULL, regionName="region",
inputFiles=inputfiles, nbins=100, handleInputParams=handleInputParams, verbose=TRUE,
scale=FALSE, heatmap=TRUE, fiveP=-1000, threeP=1000, smooth=TRUE, stranded=TRUE, transform=NA,
outPrefix=NULL, rmOutlier=FALSE, heatRange=NULL, Ylab="Coverage/base/gene",
statsMethod="wilcox.test", nc=2)
```

`plot_start_end` *Plot signals around the start and the end of genomic features*

Description

Plot reads or peak Coverage/base/gene of samples in the query files around start and end of custom features. The upstream and downstream windows can be given separately, within the window, a smaller window can be defined to highlight region of interest. A line plot will be displayed for both start and end of feature. If Input files are provided, ratio over Input is computed and displayed as well.

Usage

```
plot_start_end(
  queryFiles,
  inputFiles = NULL,
  centerFiles,
  txdb = NULL,
  handleInputParams = NULL,
  binSize = 10,
  insert = 0,
  verbose = FALSE,
  ext = c(-500, 100, -100, 500),
  hl = c(-50, 50, -50, 50),
  stranded = TRUE,
  scale = FALSE,
  smooth = FALSE,
  rmOutlier = FALSE,
  outPrefix = NULL,
  transform = NA,
  shade = TRUE,
  Ylab = "Coverage/base/gene",
  nc = 2
)
```

Arguments

queryFiles	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
inputFiles	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
centerFiles	bed files that define the custom features, or features in c("utr3", "utr5", "cds", "intron", "exon", "transcript", "gene"), multiple features are allowed.
txdb	a TxDb object defined in the GenomicFeatures package. Default NULL, needed only when genomic features are used in the place of centerFiles.
handleInputParams	a list of parameters for handle_input
binSize	an integer defines bin size for intensity calculation
insert	an integer specifies the length of the center regions to be included, in addition to the start and end of the feature
verbose	logical, whether to output additional information (including data used for plotting or statistical test results)
ext	a vector of four integers defining upstream and downstream boundaries of the plot window, flanking the start and end of features
hl	a vector of four integers defining upstream and downstream boundaries of the highlight window, flanking the start and end of features
stranded	logical, indicating whether the strand of the feature should be considered
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)
transform	a string in c("log", "log2", "log10"), default = NA, indicating no transformation of data matrix
shade	logical indicating whether to place a shaded rectangle around the point of interest
Ylab	a string for y-axis label
nc	integer, number of cores for parallel processing

Value

a list of two objects, the first is a GRanges object, the second is a GRangesList object

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))

queryfiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryfiles) <- "query"

inputfiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputfiles) <- "input"

centerfiles <- system.file("extdata", "test_chip_peak_chr19.narrowPeak", package="GenomicPlot")
names(centerfiles) <- "narrowPeak"

handleInputParams <- list(CLIP_reads=FALSE, fix_width=150, fix_point="start", norm=FALSE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

df <- plot_start_end(queryFiles=queryfiles, inputFiles=inputfiles,
centerFiles=c("gene", centerfiles), txdb=txdb, handleInputParams=handleInputParams, binSize=10,
insert=100, verbose=TRUE, ext=c(-500, 100, -100, 500), hl=c(-50, 50, -50, 50), stranded=TRUE,
scale=FALSE, smooth=TRUE, rmOutlier=FALSE, outPrefix=NULL, transform=NA, shade=TRUE,
Ylab="Coverage/base/gene", nc=2)
```

`plot_start_end_with_random` *Plot signals around the start and the end of genomic features and random regions*

Description

Plot reads or peak Coverage/base/gene of samples in the query files around stat, end and center of genomic features or custom feature given in a .bed file. The upstream and downstream windows can be given separately. If Input files are provided, ratio over Input is computed and displayed as well. A random feature can be generated to serve as a background for contrasting.

Usage

```
plot_start_end_with_random(
  queryFiles,
  inputFiles = NULL,
  txdb = NULL,
  centerFile,
  handleInputParams = NULL,
  binSize = 10,
  insert = 0,
  verbose = FALSE,
  ext = c(-500, 200, -200, 500),
  hl = c(-50, 50, -50, 50),
  randomize = FALSE,
  stranded = TRUE,
  scale = FALSE,
  smooth = FALSE,
  rmOutlier = FALSE,
```

```
  outPrefix = "plots",
  transform = NA,
  shade = TRUE,
  nc = 2,
  Ylab = "Coverage/base/gene"
)
```

Arguments

queryFiles	a vector of sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
inputFiles	a vector of input sample file names. The file should be in .bam, .bed, .wig or .bw format, mixture of formats is allowed
txdb	a TxDb object defined in the GenomicFeatures package. Default NULL, needed only when genomic features are used in the place of centerFile.
centerFile	a bed file that defines the custom feature, or a feature in c("utr3", "utr5", "cds", "intron", "exon", "transcript", "gene"), multiple features are not allowed.
handleInputParams	a list of parameters for handle_input
binSize	an integer defines bin size for intensity calculation
insert	an integer specifies the length of the center regions to be included, in addition to the start and end of the feature
verbose	logical, whether to output additional information (data used for plotting or statistical test results)
ext	a vector of four integers defining upstream and downstream boundaries of the plot window, flanking the start and end of features
hl	a vector of four integers defining upstream and downstream boundaries of the highlight window, flanking the start and end of features
randomize	logical, indicating if randomized feature should generated and used as a contrast to the real feature. The ransomized feature is generated by shifting the given feature with a random offset within the range of ext[1] and ext[4]
stranded	logical, indicating whether the strand of the feature should be considered
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
smooth	logical, indicating whether the line should smoothed with a spline smoothing algorithm
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
outPrefix	a string specifying output file prefix for plots (outPrefix.pdf)
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
shade	logical indicating whether to place a shaded rectangle around the point of interest
nc	integer, number of cores for parallel processing
Ylab	a string for y-axis label

Value

a list of two objects, the first is a GRanges object, the second is a GRangesList object

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql", package="GenomicPlot"))

queryFiles <- system.file("extdata", "treat_chr19.bam", package="GenomicPlot")
names(queryFiles) <- "query"

inputFiles <- system.file("extdata", "input_chr19.bam", package="GenomicPlot")
names(inputFiles) <- "input"

ext <- c(-500, 200, -200, 500)
hl <- c(-50, 50, -50, 50)

handleInputParams <- list(CLIP_reads=TRUE, fix_width=150, fix_point="start", norm=TRUE,
useScore=FALSE, outRle=TRUE, useSizeFactor=TRUE, genome="hg19")

plot_start_end_with_random(queryFiles=c(queryFiles), inputFiles=c(inputFiles), txdb=txdb,
centerFile="intron", binSize=10, handleInputParams=handleInputParams, ext=ext, hl=hl,
randomize=TRUE, verbose=TRUE, insert=100, stranded=TRUE, scale=FALSE, smooth=TRUE,
outPrefix=NULL, nc=2)
```

---

prepare_3parts_genomic_features

Demarcate genes into promoter, gene body and TTS features

Description

This is a helper function for 'plot_3parts_metagene', used to speed up plotting of multiple data sets with the same configuration. Use featureName='transcript' and meta=FALSE and longest=TRUE for genes.

Usage

```
prepare_3parts_genomic_features(
  txdb,
  featureName = "transcript",
  meta = TRUE,
  nbins = 100,
  fiveP = -1000,
  threeP = 1000,
  longest = TRUE,
  protein_coding = TRUE,
  verbose = FALSE
)
```

Arguments

txdb	a TxDb object defined in the GenomicFeatures package
featureName	one of the gene feature in c("utr3", "utr5", "cds", "intron", "exon", "transcript")
meta	logical, indicating whether a metagene (intron excluded) or gene (intron included) plot should be produced
nbins	an integer defines the total number of bins
fiveP	extension out of the 5' boundary of gene
threeP	extension out of the 3' boundary of gene
longest	logical, indicating whether the output should be limited to the longest transcript of each gene
protein_coding	logical, indicating whether to limit to protein_coding genes
verbose	logical, whether to output additional information

Value

a named list with the elements c("windowRs", "nbins", "scaled_bins", "fiveP", "threeP", "meta", "longest")

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("data", "txdb_chr19.sql", package="GenomicPlot"))

gf <- prepare_3parts_genomic_features(txdb, meta=FALSE, nbins=100, fiveP=-1000, threeP=1000,
longest=FALSE)
```

---

prepare_5parts_genomic_features

Demarcate genes into promoter, 5'UTR, CDS, 3'UTR and TTS features

---

Description

This is a helper function for 'plot_5parts_metagene', used to speed up plotting of multiple data sets with the same configuration.

Usage

```
prepare_5parts_genomic_features(
  txdb,
  meta = TRUE,
  nbins = 100,
  fiveP = -1000,
  threeP = 1000,
  longest = TRUE,
  verbose = FALSE,
  subsetTx = NULL
)
```

Arguments

txdb	a TxDb object defined in the GenomicFeatures package
meta	logical, indicating whether a metagene (intron excluded) or gene (intron included) plot should be produced
nbins	an integer defines the total number of bins
fiveP	extension out of the 5' boundary of gene
threeP	extension out of the 3' boundary of gene
longest	logical, indicating whether the output should be limited to the longest transcript of each gene
verbose	logical, whether to output additional information
subsetTx	a vector of transcript names (eg. ENST00000587541.1) for subsetting the genome

Value

a named list with the elements c("windowRs", "nbins", "scaled_bins", "fiveP", "threeP", "meta", "longest")

Author(s)

Shuye Pu

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "txdb_chr19.sql",
package="GenomicPlot"))

gf <- prepare_5parts_genomic_features(txdb, meta=TRUE, nbins=100, fiveP=-1000, threeP=1000,
longest=TRUE)
```

---

process_scoreMatrix *Preprocess scoreMatrix before plotting*

Description

This is a helper function for manipulate the score matrix produced by ScoreMatrix or ScoreMatrinBin functions defined in the 'genomation' package.

Usage

```
process_scoreMatrix(
  fullmatrix,
  scale = FALSE,
  rmOutlier = FALSE,
  transform = NA,
  pc = 0,
  verbose = FALSE
)
```

Arguments

fullmatrix	a numeric matrix, with bins in columns and genomic windows in rows
scale	logical, indicating whether the score matrix should be scaled to the range 0:1, so that samples with different baseline can be compared
rmOutlier	logical, indicating whether a row with abnormally high values in the score matrix should be removed
transform	a string in c("log", "log2", "log10"), default = NA indicating no transformation of data matrix
pc	pseudo-count added to the data matrix before log transformation to avoid taking log of zero
verbose	logical, indicating whether to output additional information (data used for plotting or statistical test results)

Value

a numeric matrix with the same dimension as the fullmatrix

Author(s)

Shuye Pu

---

rank_rows *Rank rows of a matrix based on user input*

Description

The rows of a input numeric matrix is ordered based row sum, row maximum, or hierarchical clustering of the rows with euclidean distance and centroid linkage. This a helper function for drawing matrix heatmaps.

Usage

```
rank_rows(fullmatrix, ranking = "Hierarchical")
```

Arguments

fullmatrix	a numeric matrix
ranking	a string in c("Sum", "Max", "Hierarchical", "None")

Value

a numeric matrix

Author(s)

Shuye Pu

rm_outlier *Remove outliers from scoreMatrix*

Description

This is a helper function for dealing with excessively high values using Hampel filter. If outliers are detected, replace the outliers with the up bound = median(rowmax) + multiplier*mad(rowmax). This function is experimental. For data with normal distribution, the multiplier is usually set at 3. As the read counts data distribution is highly skewed, it is difficult to define a boundary for outliers, try the multiplier values between 10 to 1000.

Usage

```
rm_outlier(fullmatrix, verbose = FALSE, multiplier = 1000)
```

Arguments

fullmatrix	a numeric matrix, with bins in columns and genomic windows in rows
verbose	logical, whether to output the outlier information to a log file
multiplier	a numeric value to multiple the 'mad', default 1000, maybe adjusted based on data

Value

a numeric matrix

Author(s)

Shuye Pu

Examples

```
fullmatrix <- matrix(rnorm(100), ncol=10)
maxm <- max(fullmatrix)
fullmatrix[3,9] <- maxm + 1000
fullmatrix[8,1] <- maxm + 500
rm_outlier(fullmatrix, verbose=TRUE, multiplier=100)
rm_outlier(fullmatrix, verbose=TRUE, multiplier=1000)
```

start_parallel *Prepare for parallel processing*

Description

Method for starting a virtual cluster needed for parallel processing

Usage

```
start_parallel(nc = 2, verbose = FALSE)
```

Arguments

nc	a positive integer greater than 1, denoting number of cores requested
verbose	logical, whether to output additional information

Value

an object of class c("SOCKcluster", "cluster"), depending on platform

Author(s)

Shuye Pu

Examples

```
cl <- start_parallel(2L)
```

---

stop_parallel *Stop parallel processing*

Description

Method for stopping a virtual cluster needed for parallel processing

Usage

```
stop_parallel(cl)
```

Arguments

cl	a cluster or SOCKcluster object depending on platform

Author(s)

Shuye Pu

Examples

```
cl <- start_parallel(2L)
stop_parallel(cl)
```

Index