Package 'EpiCompare'

April 10, 2023

Type Package

Title Comparison, Benchmarking & QC of Epigenomic Datasets

Version 1.2.0

Description EpiCompare is used to compare and analyse epigenetic datasets for quality control and benchmarking purposes.
The package outputs an HTML report consisting of three sections:
(1. General metrics) Metrics on peaks (percentage of blacklisted and non-standard peaks, and peak widths) and fragments (duplication rate) of samples,
(2. Peak overlap) Percentage and statistical significance of overlapping and non-overlapping peaks. Also includes upset plot and
(3. Functional annotation) functional annotation (ChromHMM, ChIPseeker and enrichment analysis) of peaks.
Also includes peak enrichment around TSS.

License GPL-3

URL https://github.com/neurogenomics/EpiCompare

BugReports https://github.com/neurogenomics/EpiCompare/issues

Depends R (>= 4.1.0)

Imports AnnotationHub, BRGenomics, ChIPseeker, data.table, genomation, GenomicRanges, IRanges, GenomeInfoDb, ggplot2, htmltools, methods, plotly, reshape2, rmarkdown, rtracklayer, stats, stringr, utils, BiocGenerics

Suggests badger, BiocFileCache, BiocParallel, parallel, BiocStyle, clusterProfiler, GenomicAlignments, grDevices, htmlwidgets, knitr, org.Hs.eg.db, testthat (>= 3.0.0), tidyr, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Mmusculus.UCSC.mm9.knownGene, TxDb.Mmusculus.UCSC.mm10.knownGene, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm9, BSgenome.Mmusculus.UCSC.mm10, UpSetR, plyranges, scales, Matrix, consensusSeekeR

VignetteBuilder knitr

biocViews Epigenetics, Genetics, QualityControl, ChIPSeq, MultipleComparison, FunctionalGenomics, ATACSeq, DNaseSeq

Config/testthat/edition 3

Encoding UTF-8

LazyData FALSE

RoxygenNote 7.2.1.9000

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

git_branch RELEASE_3_16

git_last_commit c1e27ec

git_last_commit_date 2022-11-01

Date/Publication 2023-04-10

Author Sera Choi [aut, cre] (<https://orcid.org/0000-0002-5077-1984>), Brian Schilder [aut] (<https://orcid.org/0000-0001-5949-2191>), Leyla Abbasova [aut], Alan Murphy [aut] (<https://orcid.org/0000-0002-2487-8753>), Nathan Skene [aut] (<https://orcid.org/0000-0002-6807-3180>)

Maintainer Sera Choi <serachoi1230@gmail.com>

R topics documented:

`bpplapply` *Wrapper for bplapply*

Description

Wrapper function for bplapply that automatically handles issues with **BiocParallel** related to different OS platforms.

Usage

```
bpplapply(
  X,
  FUN,
  apply_fun = parallel::mclapply,
  workers = 1,
  progressbar = workers > 1,
  verbose = workers == 1,
  use_snowparam = TRUE,
  register_now = FALSE,
  ...
)
```

Arguments

`X`	Any object for which methods `length`, `[`, and `[[` are implemented.
`FUN`	The `function` to be applied to each element of `X`.
`apply_fun`	Iterator function to use.
`workers`	Number of threads to parallelize across.
`progressbar`	`logical(1)` Enable progress bar (based on plyr:::progress_text).
`verbose`	Print messages.
`use_snowparam`	Whether to use SnowParam (default: `TRUE`) or MulticoreParam (`FALSE`) when parallelising across multiple `workers`.
`register_now`	Register the cores now with register (`TRUE`), or simply return the `BPPARAM object` (default: `FALSE`).
`...`	Arguments passed on to `BiocParallel::bplapply`

BPPARAM An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to **BiocParallel** functions.

BPREDO A list of output from bplapply with one or more failed elements. When a list is given in BPREDO, bpok is used to identify errors, tasks are rerun and inserted into the original results.

BPOPTIONS Additional options to control the behavior of the parallel evaluation, see bpoptions.

Value

(Named) list.

Examples

```
X <- stats::setNames(seq_len(length(letters)), letters)
out <- bpplapply(X, print)
```

CnR_H3K27ac *Example CUT&Run peak file*

Description

Human H3K27ac peak file generated with CUT&Run using K562 cell-line from Meers et al., (2019). Human genome build hg19 was used. Raw peak file (.BED) was obtained from GEO (https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR8581604). Peak calling was performed by Leyla Abbasova using MACS2. The peak file was then processed into GRanges object. Peaks located on chromosome 1 were subsetted to reduce the dataset size.

Usage

```
data("CnR_H3K27ac")
```

Format

An object of class GRanges of length 2707.

Source

The code to prepare the .Rda file from the raw peak file is:

```
# sequences were directly downloaded from https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR8581604
# and peaks (BED file) were generated by Leyla Abbasova (Neurogenomics Lab, Imperial College
London)
CnR_H3K27ac <- ChIPseeker::readPeakFile("path", as = "GRanges")
CnR_H3K27ac <- CnR_H3K27ac[seqnames(CnR_H3K27ac)== "chr1"]
my_label <- c("name","score","strand","signalValue","pValue","qValue","peak")
```

```
colnames(GenomicRanges::mcols(CnR_H3K27ac)) <- my_label
usethis::use_data(CnR_H3K27ac, overwrite = TRUE)
```

`CnR_H3K27ac_picard` *Example Picard duplication metrics file 2*

Description

Duplication metrics output on CUT&Run H3K27ac file (sample accession: SRR8581604). Raw sequences were aligned to hg19 genome and after, Picard was performed by Leyla Abbasova. The duplication summary output generated by Picard was processed to reduce the size of data.

Usage

```
data("CnR_H3K27ac_picard")
```

Format

An object of class `data.frame` with 1 rows and 10 columns.

Source

The code to prepare the .Rda file is:

```
picard <- read.table("path/to/picard/duplication/output", header = TRUE, fill = TRUE)
CnR_H3K27ac_picard <- picard[1,]
usethis::use_data(CnR_H3K27ac_picard, overwrite = TRUE)
```

`CnT_H3K27ac` *Example CUT&Tag peak file*

Description

Human H3K27ac peak file generated with CUT&Tag using K562 cell-line from Kaya-Okur et al., (2019). Human genome build hg19 was used. Raw peak file (.BED) was obtained from GEO (https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR8383507). Peak calling was performed by Leyla Abbasova using MACS2. The peak file was then imported as an GRanges object. Peaks located on chromosome 1 were subsetted to reduce the dataset size.

Usage

```
data("CnT_H3K27ac")
```

Format

An object of class GRanges of length 1670.

Source

The code to prepare the .Rda file from the raw peak file is:

```
# sequences were directly downloaded from https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR8383507
# and peaks (BED file) were generated by Leyla Abbasova (Neurogenomics Lab, Imperial College
London)
CnT_H3K27ac <- ChIPseeker::readPeakFile("path", as = "GRanges")
CnT_H3K27ac <- CnT_H3K27ac[seqnames(CnT_H3K27ac)== "chr1"]
my_label <- c("name","score","strand","signalValue","pValue","qValue","peak")
colnames(GenomicRanges::mcols(CnT_H3K27ac)) <- my_label
usethis::use_data(CnT_H3K27ac)
```

CnT_H3K27ac_picard *Example Picard duplication metrics file 1*

Description

Duplication metrics output of CUT&Tag H3K27ac file (sample accession: SRR8581604). Raw sequences were aligned to hg19 genome and Picard was performed by Leyla Abbasova. The duplication summary output generated by Picard was processed to reduce the size of data.

Usage

```
data("CnT_H3K27ac_picard")
```

Format

An object of class data.frame with 1 rows and 10 columns.

Source

The code to prepare the .Rda file is:

```
picard <- read.table("path/to/picard/duplication/output", header = TRUE, fill = TRUE)]
CnT_H3K27ac_picard <- picard[1,]
usethis::use_data(CnT_H3K27ac_picard, overwrite = TRUE)
```

compute_consensus_peaks *Compute consensus peaks*

Description

Compute consensus peaks from a list of GRanges.

Usage

```
compute_consensus_peaks(
  grlist,
  groups = NULL,
  genome_build,
  lower = 2,
  upper = Inf,
  min.gapwidth = 1L,
  method = c("granges", "consensusseeker"),
  ...
)
```

Arguments

grlist	Named list of GRanges objects.
groups	A character vector of the same length as grlist defining how to group GRanges objects when computing consensus peaks.
genome_build	Genome build name.
lower, upper	The lower and upper bounds for the slice.
min.gapwidth	Ranges separated by a gap of at least min.gapwidth positions are not merged.
method	Method to call peaks with: • "granges" : Simple overlap procedure using GRanges functions. Faster but less accurate. • "consensusseeker" : Uses findConsensusPeakRegions to compute consensus peaks. Slower but more accurate.
...	Arguments passed on to consensusSeekeR::findConsensusPeakRegions narrowPeaks a GRanges containing called peak regions of signal enrichment based on pooled, normalized data for all analyzed experiments. All GRanges entries must have a metadata field called "name" which identifies the region to the called peak. All GRanges entries must also have a row name which identifies the experiment of origin. Each peaks entry must have an associated narrowPeaks entry. A GRanges entry is associated to a narrowPeaks entry by having a identical metadata "name" field and a identical row name.

`peaks` a `GRanges` containing called peaks of signal enrichment based on pooled, normalized data for all analyzed experiments. All `GRanges` entries must have a metadata field called "name" which identifies the called peak. All `GRanges` entries must have a row name which identifies the experiment of origin. Each `peaks` entry must have an associated `narrowPeaks` entry. A `GRanges` entry is associated to a `narrowPeaks` entry by having a identical metadata "name" field and a identical row name.

`chrInfo` a `Seqinfo` containing the name and the length of the chromosomes to analyze. Only the chomosomes contained in this `Seqinfo` will be analyzed.

`extendingSize` a `numeric` value indicating the size of padding on both sides of the position of the peaks median to create the consensus region. The minimum size of the consensus region is equal to twice the value of the `extendingSize` parameter. The size of the `extendingSize` must be a positive integer. Default = 250.

`expandToFitPeakRegion` a `logical` indicating if the region size, which is set by the `extendingSize` parameter is extended to include the entire narrow peak regions of all peaks included in the unextended consensus region. The narrow peak regions of the peaks added because of the extension are not considered for the extension. Default: `FALSE`.

`shrinkToFitPeakRegion` a `logical` indicating if the region size, which is set by the `extendingSize` parameter is shrinked to fit the narrow peak regions of the peaks when all those regions are smaller than the consensus region. Default: `FALSE`.

`minNbrExp` a positive `numeric` or a positive `integer` indicating the minimum number of experiments in which at least one peak must be present for a potential consensus region. The numeric must be a positive integer inferior or equal to the number of experiments present in the `narrowPeaks` and `peaks` parameters. Default = 1.

`nbrThreads` a `numeric` or a `integer` indicating the number of threads to use in parallel. The `nbrThreads` must be a positive integer. Default = 1.

Details

NOTE: If you get the error "`Error in serialize(data, node$con) : error writing to connection`", try running closeAllConnections and rerun compute_consensus_peaks. This error can sometimes occur when compute_consensus_peaks has been disrupted partway through.

Value

Named list of consensus peak GRanges.

Source

GenomicRanges tutorial

consensusSeekeR

Examples

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object
grlist <- list(CnR=CnR_H3K27ac, CnT=CnT_H3K27ac, ENCODE=encode_H3K27ac)

consensus_peaks <- compute_consensus_peaks(grlist = grlist,
                                           groups = c("Imperial",
                                                      "Imperial",
                                                      "ENCODE"))
```

`compute_corr` *Compute correlation matrix*

Description

Compute correlation matrix on all peak files.

Usage

```
compute_corr(
  peakfiles,
  reference = NULL,
  genome_build,
  keep_chr = NULL,
  drop_empty_chr = FALSE,
  bin_size = 5000,
  method = "spearman",
  intensity_cols = c("total_signal", "qValue", "Peak Score", "score"),
  return_bins = FALSE,
  workers = 1
)
```

Arguments

`peakfiles`	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
`reference`	A named list containing reference peak file(s) as GRanges object. Please ensure that the reference file is listed and named i.e. `list("reference_name" = reference_peak)`. If more than one reference is specified, individual reports for each reference will be generated. However, please note that specifying more than one reference can take awhile. If a reference is specified, it enables two analyses: (1) plot showing statistical significance of overlapping/non-overlapping peaks; and (2) ChromHMM of overlapping/non-overlapping peaks.

genome_build	The build of **all** peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use liftover_grlist to convert them all before running. Genome build should be one of hg19, hg38, mm9, mm10.
keep_chr	Which chromosomes to keep.
drop_empty_chr	Drop chromosomes that are not present in any of the peakfiles (default: FALSE).
bin_size	Default of 100. Base-pair size of the bins created to measure correlation. Use smaller value for higher resolution but longer run time and larger memory usage.
method	Default spearman (i.e. non-parametric). A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson", "kendall", or "spearman": can be abbreviated.
intensity_cols	Depending on which columns are present, this value will be used to get quantiles and ultimately calculate the correlations: • "total_signal" : Used by the peak calling software SEACR. *NOTE*: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal". • "qValue"Used by the peak calling software MACS2/3. Should contain the negative log of the p-values after multiple testing correction. • "Peak Score" : Used by the peak calling software HOMER.
return_bins	If TRUE, returns a named list with both the rebinned (standardised) peaks ("bin") and the correlation matrix ("cor"). If FALSE (default), returns only the correlation matrix (unlisted).
workers	Number of cores to parallelise across (in applicable functions).

Value

correlation matrix

Examples

```
data("CnR_H3K27ac")
data("CnT_H3K27ac")
data("encode_H3K27ac")
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac, CnT_H3K27ac=CnT_H3K27ac)
reference <- list("encode_H3K27ac"=encode_H3K27ac)

#increasing bin_size for speed but lower values will give more granular corr
corr_mat <- compute_corr(peakfiles = peakfiles,
                     reference = reference,
                     genome_build = "hg19",
                     bin_size = 5000)
```

encode_H3K27ac *Example ChIP-seq peak file*

Description

Human H3K27ac peak file generated with ChIP-seq using K562 cell-line. Human genome build hg19 was used. The peak file (.BED) was obtained from ENCODE project (https://www.encodeproject.org/files/ENCFF044JNJ/). The BED file was then imported as an GRanges object. Peaks located on chromosome 1 were subsetted to reduce the dataset size.

Usage

```
data("encode_H3K27ac")
```

Format

An object of class GRanges of length 5142.

Source

The code to prepare the .Rda file from the raw peak file is:

```
# dataset was directly downloaded from
# https://www.encodeproject.org/files/ENCFF044JNJ/ encode_H3K27ac <- ChIPseeker::readPeakFile("path",
as = "GRanges")
encode_H3K27ac <- encode_H3K27ac[seqnames(encode_H3K27ac) == "chr1"]
my_label <- c("name","score","strand","signalValue","pValue","qValue","peak")
colnames(GenomicRanges::mcols(encode_H3K27ac)) <- my_label
usethis::use_data(encode_H3K27ac, overwrite = TRUE)
```

EpiCompare *Compare epigenomic datasets*

Description

This function compares and analyses multiple epigenomic datasets and outputs an HTML report containing all results of the analysis. The report is mainly divided into three sections: (1) General Metrics on Peakfiles, (2) Peak Overlaps and (3) Functional Annotation of Peaks.

Usage

```
EpiCompare(
  peakfiles,
  genome_build,
  genome_build_output = "hg19",
  blacklist,
  picard_files = NULL,
  reference = NULL,
  upset_plot = FALSE,
  stat_plot = FALSE,
  chromHMM_plot = FALSE,
  chromHMM_annotation = "K562",
  chipseeker_plot = FALSE,
  enrichment_plot = FALSE,
  tss_plot = FALSE,
  precision_recall_plot = FALSE,
  n_threshold = 15,
  corr_plot = FALSE,
  bin_size = 5000,
  interact = TRUE,
  save_output = FALSE,
  output_filename = "EpiCompare",
  output_timestamp = FALSE,
  output_dir,
  display = NULL,
  workers = 1
)
```

Arguments

peakfiles	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
genome_build	A named list indicating the human genome build used to generate each of the following inputs: • "peakfiles" : Genome build for the `peakfiles` input. Assumes genome build is the same for each element in the `peakfiles` list. • "reference" : Genome build for the `reference` input. • "blacklist" : Genome build for the `blacklist` input. Example input list: `genome_build = list(peakfiles="hg38", reference="hg19", blacklist="hg19")` Alternatively, you can supply a single character string instead of a list. This should *only* be done in situations where all three inputs (`peakfiles`, `reference`,

blacklist) are of the same genome build. For example:
genome_build = "hg19"

genome_build_output
: Genome build to standardise all inputs to. Liftovers will be performed automatically as needed. Default: "hg19".

blacklist
: A GRanges object containing blacklisted regions.

picard_files
: A list of summary metrics output from Picard. Files must be in data.frame format and listed using list() and named using names(). To import Picard duplication metrics (.txt file) into R as data frame, use:
picard <- read.table("/path/to/picard/output", header = TRUE, fill = TRUE).

reference
: A named list containing reference peak file(s) as GRanges object. Please ensure that the reference file is listed and named i.e. list("reference_name" = reference_peak). If more than one reference is specified, individual reports for each reference will be generated. However, please note that specifying more than one reference can take awhile. If a reference is specified, it enables two analyses: (1) plot showing statistical significance of overlapping/non-overlapping peaks; and (2) ChromHMM of overlapping/non-overlapping peaks.

upset_plot
: Default FALSE. If TRUE, the report includes upset plot of overlapping peaks.

stat_plot
: Default FALSE. If TRUE, the function creates a plot showing the statistical significance of overlapping/non-overlapping peaks. Reference peak file must be provided.

chromHMM_plot
: Default FALSE. If TRUE, the function outputs ChromHMM heatmap of individual peak files. If a reference peak file is provided, ChromHMM annotation of overlapping and non-overlapping peaks is also provided.

chromHMM_annotation
: ChromHMM annotation for ChromHMM plots. Default K562 cell-line. Cell-line options are:

- "K562" = K-562 cells
- "Gm12878" = Cellosaurus cell-line GM12878
- "H1hesc" = H1 Human Embryonic Stem Cell
- "Hepg2" = Hep G2 cell
- "Hmec" = Human Mammary Epithelial Cell
- "Hsmm" = Human Skeletal Muscle Myoblasts
- "Huvec" = Human Umbilical Vein Endothelial Cells
- "Nhek" = Normal Human Epidermal Keratinocytes
- "Nhlf" = Normal Human Lung Fibroblasts

chipseeker_plot
: Default FALSE. If TRUE, the report includes a barplot of ChIPseeker annotation of peak files.

enrichment_plot
: Default FALSE. If TRUE, the report includes dotplots of KEGG and GO enrichment analysis of peak files.

tss_plot
: Default FALSE. If TRUE, the report includes peak count frequency around transcriptional start site. Note that this can take awhile.

precision_recall_plot
: Default is FALSE. If TRUE, creates a precision-recall curve plot and an F1 plot using plot_precision_recall.

n_threshold
: Number of thresholds to test.

corr_plot
: Default is FALSE. If TRUE, creates a correlation plot across all peak files using plot_corr.

bin_size
: Default of 100. Base-pair size of the bins created to measure correlation. Use smaller value for higher resolution but longer run time and larger memory usage.

interact
: Default TRUE. By default, all heatmaps are interactive. If set FALSE, all heatmaps in the report will be static.

save_output
: Default FALSE. If TRUE, all outputs (tables and plots) of the analysis will be saved in a folder (EpiCompare_file).

output_filename
: Default EpiCompare.html. If otherwise, the html report will be saved in the specified name.

output_timestamp
: Default FALSE. If TRUE, date will be included in the file name.

output_dir
: Path to where output HTML file should be saved.

display
: After completion, automatically display the HTML report file in one of the following ways:
  - "browser" : Display the report in your default web browser.
  - "rsstudio" : Display the report in Rstudio.
  - NULL (default) : Do not display the report.

workers
: Number of cores to parallelise across (in applicable functions).

Value

Path to one or more HTML report files.

Examples

```
### Load Data ###
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object
data("hg19_blacklist") # hg38 blacklist dataset
data("CnT_H3K27ac_picard") # example Picard summary output
data("CnR_H3K27ac_picard") # example Picard summary output

#### Prepare Input ####
# create named list of peakfiles
peaks <- list(CnR=CnR_H3K27ac, CnT=CnT_H3K27ac)
# create named list of picard outputs
picard <- list(CnR=CnR_H3K27ac_picard, CnT=CnT_H3K27ac_picard)
# reference peak file
reference_peak <- list("ENCODE" = encode_H3K27ac)
```

```
### Run EpiCompare ###
EpiCompare(peakfiles = peaks,
           genome_build = list(peakfiles="hg19",
                               reference="hg19",
                               blacklist="hg19"),
           genome_build_output = "hg19",
           blacklist = hg19_blacklist,
           picard_files = picard,
           reference = reference_peak,
           output_filename = "EpiCompare_test",
           output_dir = tempdir())
```

`fragment_info` *Summary on fragments*

Description

This function outputs a summary on fragments using metrics generated by Picard. Provides the number of mapped fragments, duplication rate and number of unique fragments.

Usage

```
fragment_info(picard_list)
```

Arguments

`picard_list`	Named list of duplication metrics generated by Picard as data frame. Data frames must be named and listed using `list()`. e.g. `list("name1"=file1, "name2"=file2)`. To import Picard duplication metrics (.txt file) into R as data frame, use `picard <- read.table("/path/to/picard/output", header = TRUE,fill = TRUE)`.

Value

A table summarizing metrics on fragments.

Examples

```
### Load Data ###
data(CnT_H3K27ac_picard) # example picard output
data(CnR_H3K27ac_picard) # example picard output

### Import Picard Metrics ###
# To import Picard duplication metrics (.txt file) into R as data frame
# CnT_H3K27ac_picard <- read.table("/path/to/picard/output.txt", header = TRUE,fill = TRUE)

### Create Named List ###
picard <- list("CnT_H3K27ac"=CnT_H3K27ac_picard,
               "CnR_H3K27ac"=CnR_H3K27ac_picard)
```

```
### Run ###
df <- fragment_info(picard_list = picard)
```

`gather_files` *Gather files*

Description

Recursively find peak/picard files stored within subdirectories and import them as a list of GRanges objects.

Usage

```
gather_files(
  dir,
  type = "peaks.stringent",
  nfcore_cutandrun = FALSE,
  return_paths = FALSE,
  rbind_list = FALSE,
  workers = 1,
  verbose = TRUE
)
```

Arguments

`dir`	Directory to search within.
`type`	File type to search for. Options include: • "<pattern>"Finds files matching an arbitrary regex pattern specified by user. • "peaks.stringent"Finds files ending in "*.stringent.bed$" • "peaks.consensus"Finds files ending in "*.consensus.peaks.bed$" • "peaks.consensus.filtered" Finds files ending in"*.consensus.peaks.filtered.awk.bed$" • "picard"Finds files ending in "*.target.markdup.MarkDuplicates.metrics.txt$"
`nfcore_cutandrun`	Whether the files were generated by the nf-core/cutandrun Nextflow pipeline. If `TRUE`, can use the standardised folder structure to automatically generate more descriptive file names with sample IDs.
`return_paths`	Return only the file paths without actually reading them in as GRanges.
`rbind_list`	Bind all objects into one.
`workers`	`integer(1)` Number of workers. Defaults to the maximum of 1 or the number of cores determined by `detectCores` minus 2 unless environment variables `R_PARALLELLY_AVAILABLECORES_FALLBACK` or `BIOCPARALLEL_WORKER_NUMBER` are set otherwise.
`verbose`	Print messages.

Details

For "peaks.stringent" files called with SEACR, column names will be automatically added:

- total_signal : Total signal contained within denoted coordinates.
- max_signalMaximum bedgraph signal attained at any base pair within denoted coordinates.
- max_signal_region Region representing the farthest upstream and farthest downstream bases within the denoted coordinates that are represented by the maximum bedgraph signal.

Value

A named list of GRanges objects.

Examples

```
#### Make example files ####
save_paths <- EpiCompare::write_example_peaks()
dir <- unique(dirname(save_paths))
#### Gather/import files ####
peaks <- EpiCompare::gather_files(dir=dir, type="peaks.narrow")
```

---

group_files *Group files*

Description

Assign group names to each file in a named list based on a series of string searches based on combinations of relevant metadata factors.

Usage

```
group_files(peakfiles, searches)
```

Arguments

peakfiles	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
searches	A named list of substrings to group `peakfiles` by.

Examples

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac,
                  CnT_H3K27ac=CnT_H3K27ac,
                  encode_H3K27ac=encode_H3K27ac)

peaks_grouped <- group_files(peakfiles = peakfiles,
                             searches=list(assay=c("H3K27ac"),
                                           source=c("Cn","ENCODE")))
```

---

hg19_blacklist *Human genome hg19 blacklisted regions*

Description

Obtained from https://www.encodeproject.org/files/ENCFF001TDO/. The ENCODE blacklist includes regions of the hg19 genome that have anomalous and/or unstructured signals independent of the cell-line or experiment. Removal of ENCODE blacklist is recommended for quality measure.

Usage

```
data("hg19_blacklist")
```

Format

An object of class GRanges of length 411.

Source

The code to prepare the .Rda file is:

```
# blacklisted regions were directly downloaded
# from https://www.encodeproject.org/files/ENCFF001TDO/
hg19_blacklist <- ChIPseeker::readPeakFile(file.path(path), as = "GRanges")
usethis::use_data(hg19_blacklist, overwrite = TRUE)
```

hg38_blacklist *Human genome hg38 blacklisted regions*

Description

Obtained from https://www.encodeproject.org/files/ENCFF356LFX/. The ENCODE blacklist includes regions of the hg38 genome that have anomalous and/ or unstructured signals independent of the cell-line or experiment. Removal of ENCODE blacklist is recommended for quality measure.

Usage

```
data("hg38_blacklist")
```

Format

An object of class GRanges of length 910.

Source

The code to prepare the .Rda file is:

```
## blacklisted regions were directly downloaded
## from https://www.encodeproject.org/files/ENCFF356LFX/
hg38_blacklist <- ChIPseeker::readPeakFile(file.path(path), as = "GRanges")
usethis::use_data(hg38_blacklist, overwrite = TRUE)
```

liftover_grlist *Liftover peak list*

Description

Perform genome build liftover to one or more GRanges objects at once.

Usage

```
liftover_grlist(
  grlist,
  input_build,
  output_build = "hg19",
  style = "UCSC",
  keep_chr = paste0("chr", c(seq_len(22), "X", "Y")),
  as_grangeslist = FALSE,
  merge_all = FALSE,
  verbose = TRUE
)
```

Arguments

grlist	A named list of GRanges objects, or simply a single unlisted GRanges object. Can perform liftover within species or across species.
input_build	The genome build of grlist.
output_build	Desired genome build for grlist to be lifted over to.
style	Chromosome style, set by seqlevelsStyle. • "UCSC" : Uses the chromosome style "chr1". • "NCBI" : Uses the chromosome style "1"
keep_chr	Which chromosomes to keep.
as_grangeslist	Return as a GRangesList.
merge_all	Merge all GRanges into a single GRanges object.
verbose	Print messages.

Value

Named list of lifted GRanges objects.

Examples

```
grlist <- list("gr1"=GenomicRanges::GRanges("4:1-100000"),
               "gr2"=GenomicRanges::GRanges("6:1-100000"),
               "gr3"=GenomicRanges::GRanges("8:1-100000"))

grlist_lifted <- liftover_grlist(grlist = grlist,
                                 input_build = "hg19",
                                 output_build="hg38")
```

---

overlap_heatmap *Generate heatmap of percentage overlap*

Description

This function generates a heatmap showing percentage of overlapping peaks between peak files.

Usage

```
overlap_heatmap(peaklist, interact = TRUE)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed and named using list(). e.g. list("name1"=file1, "name2"=file2). If not named, default file names will be assigned.
interact	Default TRUE. By default heatmap is interactive. If FALSE, heatmap is static.

Value

An interactive heatmap

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object

### Create Named List ###
peaks <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)

### Run ###
my_heatmap <- overlap_heatmap(peaklist = peaks)
```

---

overlap_percent *Calculate percentage of overlapping peaks*

Description

This function calculates the percentage of overlapping peaks and outputs a table or matrix of results.

Usage

```
overlap_percent(
  peaklist1,
  peaklist2,
  invert = FALSE,
  precision_recall = TRUE,
  suppress_messages = TRUE
)
```

Arguments

peaklist1	A list of peak files as GRanges object. Files must be listed and named using list(). e.g. list("name1"=file1, "name2"=file2). If not named, default file names will be assigned.
peaklist2	peaklist1 A list of peak files as GRanges object. Files must be listed and named using list(). e.g. list("name1"=file1, "name2"=file2).
invert	If TRUE, keep only the ranges in x that do *not* overlap ranges.
precision_recall	Return percision-recall results for all combinations of peaklist1 (the "query") and peaklist2 (the "subject"). See subsetByOverlaps for more details on this terminology.
suppress_messages	Suppress messages.

Value

data frame

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object
data("CnR_H3K27ac") # example peakfile GRanges object

### Create Named Peaklist ###
peaks <- list("CnT"=CnT_H3K27ac, "CnR"=CnR_H3K27ac)
reference_peak <- list("ENCODE"=encode_H3K27ac)

### Run ###
overlap <- overlap_percent(peaklist1=peaks,
                           peaklist2=reference_peak)
```

---

overlap_stat_plot *Statistical significance of overlapping peaks*

Description

This function calculates the statistical significance of overlapping/ non-overlapping peaks against a reference peak file.If the reference peak file has the BED6+4 format (peak called by MACS2), the function generates a series of boxplots showing the distribution of q-values for sample peaks that are overlapping and non-overlapping with the reference. If the reference peak file does not have the BED6+4 format, the function uses 'enrichPeakOverlap()' from 'ChIPseeker' package to calculate the statistical significance of overlapping peaks only. In this case, please provide an annotation file as TxDb object.

Usage

```
overlap_stat_plot(reference, peaklist, annotation = NULL)
```

Arguments

reference	A reference peak file as GRanges object.
peaklist	A list of peak files as GRanges object. Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor. This is required only if the reference file does not have BED6+4 format.

Value

A boxplot or barplot showing the statistical significance of overlapping/non-overlapping peaks.

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object
data("CnR_H3K27ac") # example peakfile GRanges object

### Create Named Peaklist & Reference ###
peaklist <- list('CnT'=CnT_H3K27ac, "CnR"=CnR_H3K27ac)
reference <- list("ENCODE"=encode_H3K27ac)

### Run ###
out <- overlap_stat_plot(reference = reference,
                         peaklist = peaklist)
stat_plot <- out[[1]] # plot
stat_df <- out[[2]] # df
```

`overlap_upset_plot` *Generate Upset plot for overlapping peaks*

Description

This function generates upset plot (UpSetR package) of overlapping peaks.

Usage

```
overlap_upset_plot(peaklist)
```

Arguments

`peaklist`	A named list of peak files as GRanges object. Objects must be listed and named using `list()`. e.g. `list("name1"=file1, "name2"=file2)`. If not named, default file names are assigned.

Value

Upset plot of overlapping peaks

Examples

```
### Load Data ###
data("encode_H3K27ac") # load example data
data("CnT_H3K27ac") # load example data

### Create Named List ###
peakfile <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)

### Run ###
my_plot <- overlap_upset_plot(peaklist = peakfile)
```

peak_info *Summary of Peak Information*

Description

This function outputs a table summarizing information on the peak files. Provides the total number of peaks and the percentage of peaks in blacklisted regions.

Usage

```
peak_info(peaklist, blacklist)
```

Arguments

peaklist	A named list of peak files as GRanges object. Objects listed using `list("name1" = peak, "name2" = peak2)`.
blacklist	A GRanges object containing blacklisted regions.

Value

A summary table of peak information

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object
data("hg19_blacklist") # example blacklist GRanges object

### Named Peaklist ###
peaklist <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)

### Run ###
df <- peak_info(peaklist = peaklist,
                blacklist = hg19_blacklist)
```

plot_ChIPseeker_annotation *Create ChIPseeker annotation plot*

Description

This function annotates peaks using `ChIPseeker::annotatePeak`. It outputs functional annotation of each peak file in a barplot.

Usage

```
plot_ChIPseeker_annotation(peaklist, annotation)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed and named using `list()`. e.g. `list("name1"=file1, "name2"=file2)`. If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.

Value

barplot

Examples

```
### Load Data ###
data("CnT_H3K27ac") # example peakfile GRanges object
data("CnR_H3K27ac") # example peakfile GRanges object

### Create Named Peaklist ###
peaks <- list("CnT"=CnT_H3K27ac, "CnR"=CnR_H3K27ac)

## not run
# txdb<-TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
# my_plot <- plot_ChIPseeker_annotation(peaklist = peaks
#                                       annotation = txdb)
```

---

plot_chromHMM *Plot ChromHMM heatmap*

Description

Creates a heatmap using outputs from ChromHMM using ggplot2.The function takes a list of peakfiles, performs ChromHMM and outputs a heatmap. ChromHMM annotation file must be loaded prior to using this function. ChromHMM annotations are aligned to hg19, and will be automatically lifted over to the `genome_build` to match the build of the `peaklist`.

Usage

```
plot_chromHMM(
  peaklist,
  chromHMM_annotation,
  cell_line = NULL,
  genome_build,
  interact = TRUE,
  return_data = FALSE
)
```

Arguments

peaklist	A named list of peak files as GRanges object. If list is not named, default names will be assigned.
chromHMM_annotation	ChromHMM annotation list.
cell_line	If not cell_line, will replace chromHMM_annotation by importing chromHMM data for a given cell line using get_chromHMM_annotation.
genome_build	The human genome reference build used to generate peakfiles. "hg19" or "hg38".
interact	Default TRUE. By default, the heatmaps are interactive. IfFALSE, the function generates a static ChromHMM heatmap.
return_data	Return the plot data as in addition to the plot itself.

Value

ChromHMM heatmap, or a named list.

Examples

```
### Load Data ###
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object

### Create Named Peaklist ###
peaklist <- list(CnT=CnT_H3K27ac, CnR=CnR_H3K27ac)

### Run ###
my_plot <- plot_chromHMM(peaklist = peaklist,
                         cell_line = "K562",
                         genome_build = "hg19")
```

---

plot_corr *Plot correlation of peak files*

Description

Plot correlation by binning genome and measuring correlation of peak quantile ranking. This ranking is based on p-value or other peak intensity measure dependent on the peak calling approach.

Usage

```
plot_corr(
  peakfiles,
  reference,
  genome_build,
  bin_size = 5000,
```

```
  keep_chr = NULL,
  drop_empty_chr = FALSE,
  method = "spearman",
  intensity_cols = c("total_signal", "qValue", "Peak Score", "score"),
  interact = FALSE,
  workers = 1,
  show_plot = TRUE
)
```

Arguments

peakfiles	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
reference	A named list containing reference peak file(s) as GRanges object. Please ensure that the reference file is listed and named i.e. `list("reference_name" = reference_peak)`. If more than one reference is specified, individual reports for each reference will be generated. However, please note that specifying more than one reference can take awhile. If a reference is specified, it enables two analyses: (1) plot showing statistical significance of overlapping/non-overlapping peaks; and (2) ChromHMM of overlapping/non-overlapping peaks.
genome_build	The build of **all** peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use liftover_grlist to convert them all before running. Genome build should be one of hg19, hg38, mm9, mm10.
bin_size	Default of 100. Base-pair size of the bins created to measure correlation. Use smaller value for higher resolution but longer run time and larger memory usage.
keep_chr	Which chromosomes to keep.
drop_empty_chr	Drop chromosomes that are not present in any of the `peakfiles` (default: `FALSE`).
method	Default spearman (i.e. non-parametric). A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson", "kendall", or "spearman": can be abbreviated.
intensity_cols	Depending on which columns are present, this value will be used to get quantiles and ultimately calculate the correlations: • "total_signal" : Used by the peak calling software SEACR. *NOTE*: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal". • "qValue"Used by the peak calling software MACS2/3. Should contain the negative log of the p-values after multiple testing correction. • "Peak Score" : Used by the peak calling software HOMER.
interact	Default TRUE. By default, all heatmaps are interactive. If set FALSE, all heatmaps in the report will be static.
workers	Number of cores to parallelise across (in applicable functions).
show_plot	Show the plot.

Value

list with correlation plot (corr_plot) and correlation matrix (data)

Examples

```
data("CnR_H3K27ac")
data("CnT_H3K27ac")
data("encode_H3K27ac")
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac, CnT_H3K27ac=CnT_H3K27ac)
reference <- list("encode_H3K27ac"=encode_H3K27ac)

#increasing bin_size for speed but lower values will give more granular corr
cp <- plot_corr(peakfiles = peakfiles,
                reference = reference,
                genome_build = "hg19",
                bin_size = 5000)
```

---

plot_enrichment *Generate enrichment analysis plots*

Description

This function runs KEGG and GO enrichment analysis of peak files and generates dot plots.

Usage

```
plot_enrichment(peaklist, annotation)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed and named using list(). e.g. list("name1"=file1, "name2"=file2). If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.

Value

KEGG and GO dot plots

Examples

```
### Load Data ###
data("CnT_H3K27ac") # example peakfile GRanges object
data("CnR_H3K27ac") # example peakfile GRanges object

### Create Named Peaklist ###
peaks <- list("CnT"=CnT_H3K27ac, "CnR"=CnR_H3K27ac)
```

```
## not run
# txdb<-TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
# my_plot <- plot_enrichment(peaklist = peaks,
#                            annotation = txdb)
```

`plot_precision_recall` *Plot precision-recall curves*

Description

Plot precision-recall curves (and optionally F1 plots) by iteratively testing for peak overlap across a series of thresholds used to filter `peakfiles`. Each GRanges object in `peakfiles` will be used as the "query" against each GRanges object in `reference` as the subject. Will automatically use any columns that are specified with `thresholding_cols` and present within each GRanges object to create percentiles for thresholding. *NOTE* : Assumes that all GRanges in `peakfiles` and `reference` are already aligned to the same genome build.

Usage

```
plot_precision_recall(
  peakfiles,
  reference,
  thresholding_cols = c("total_signal", "qValue", "Peak Score"),
  initial_threshold = 0,
  n_threshold = 10,
  max_threshold = 1,
  workers = 1,
  plot_f1 = TRUE,
  subtitle = NULL,
  color = "peaklist1",
  shape = color,
  facets = "peaklist2 ~ .",
  interact = FALSE,
  show_plot = TRUE
)
```

Arguments

`peakfiles`	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
`reference`	A named list containing reference peak file(s) as GRanges object. Please ensure that the reference file is listed and named i.e. `list("reference_name"`

= reference_peak). If more than one reference is specified, individual reports for each reference will be generated. However, please note that specifying more than one reference can take awhile. If a reference is specified, it enables two analyses: (1) plot showing statistical significance of overlapping/non-overlapping peaks; and (2) ChromHMM of overlapping/non-overlapping peaks.

thresholding_cols
: Depending on which columns are present, GRanges will be filtered at each threshold according to one or more of the following:
  - "total_signal" : Used by the peak calling software SEACR. *NOTE*: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal".
  - "qValue"Used by the peak calling software MACS2/3. Should contain the negative log of the p-values after multiple testing correction.
  - "Peak Score" : Used by the peak calling software HOMER.

initial_threshold
: Numeric threshold that was provided to SEACR (via the parameter --ctrl) when calling peaks without an IgG control.

n_threshold
: Number of thresholds to test.

max_threshold
: Maximum threshold to test.

workers
: Number of cores to parallelise across (in applicable functions).

plot_f1
: Generate a plot with the F1 score vs. threshold as well.

subtitle
: Plot subtitle.

color
: Variable to color data points by.

shape
: Variable to set data point shapes by.

facets
: This argument is soft-deprecated, please use rows and cols instead.

interact
: Default TRUE. By default, all heatmaps are interactive. If set FALSE, all heatmaps in the report will be static.

show_plot
: Show the plot.

Value

list with data and precision recall and F1 plots

Examples

```
data("CnR_H3K27ac")
data("CnT_H3K27ac")
data("encode_H3K27ac")
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac, CnT_H3K27ac=CnT_H3K27ac)
reference <- list("encode_H3K27ac" = encode_H3K27ac)

pr_out <- plot_precision_recall(peakfiles = peakfiles,
                                reference = reference)
```

`precision_recall` *Compute precision-recall*

Description

Compute precision and recall using each GRanges object in `peakfiles` as the "query" against each GRanges object in `reference` as the subject.

Usage

```
precision_recall(
  peakfiles,
  reference,
  thresholding_cols = c("total_signal", "qValue", "Peak Score"),
  initial_threshold = 0,
  n_threshold = 15,
  max_threshold = 1,
  workers = 1,
  ...
)
```

Arguments

`peakfiles`	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
`reference`	A named list containing reference peak file(s) as GRanges object. Please ensure that the reference file is listed and named i.e. `list("reference_name" = reference_peak)`. If more than one reference is specified, individual reports for each reference will be generated. However, please note that specifying more than one reference can take awhile. If a reference is specified, it enables two analyses: (1) plot showing statistical significance of overlapping/non-overlapping peaks; and (2) ChromHMM of overlapping/non-overlapping peaks.
`thresholding_cols`	Depending on which columns are present, GRanges will be filtered at each threshold according to one or more of the following: • "total_signal" : Used by the peak calling software SEACR. *NOTE*: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal". • "qValue"Used by the peak calling software MACS2/3. Should contain the negative log of the p-values after multiple testing correction. • "Peak Score" : Used by the peak calling software HOMER.

initial_threshold
: Numeric threshold that was provided to SEACR (via the parameter `--ctrl`) when calling peaks without an IgG control.

n_threshold
: Number of thresholds to test.

max_threshold
: Maximum threshold to test.

workers
: Number of cores to parallelise across (in applicable functions).

...
: Arguments passed on to `bpplapply`

  `apply_fun` Iterator function to use.

  `verbose` Print messages.

  `register_now` Register the cores now with register (`TRUE`), or simply return the `BPPARAM object` (default: `FALSE`).

  `use_snowparam` Whether to use SnowParam (default: `TRUE`) or MulticoreParam (`FALSE`) when parallelising across multiple `workers`.

  `progressbar logical(1)` Enable progress bar (based on plyr:::progress_text).

  `X` Any object for which methods `length`, `[`, and `[[` are implemented.

  `FUN` The `function` to be applied to each element of `X`.

Value

Overlap

Examples

```
data("CnR_H3K27ac")
data("CnT_H3K27ac")
data("encode_H3K27ac")
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac, CnT_H3K27ac=CnT_H3K27ac)
reference <- list("encode_H3K27ac" = encode_H3K27ac)

pr_df <- precision_recall(peakfiles = peakfiles,
                          reference = reference)
```

---

rebin_peaks *Rebin peaks*

Description

Standardise a list of peak files by rebinning them into fixd-width tiles across the genome.

Usage

```
rebin_peaks(
  peakfiles,
  genome_build,
  intensity_cols = c("total_signal", "qValue", "Peak Score", "score"),
  bin_size = 5000,
```

```
  keep_chr = NULL,
  drop_empty_chr = FALSE,
  as_sparse = TRUE,
  workers = 1,
  verbose = TRUE,
  ...
)
```

Arguments

peakfiles	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as GRanges object. EpiCompare also accepts a list containing a mix of GRanges objects and paths.Files must be listed and named using `list()`. E.g. `list("name1"=file1, "name2"=file2)`. If no names are specified, default file names will be assigned.
genome_build	The build of **all** peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use liftover_grlist to convert them all before running. Genome build should be one of hg19, hg38, mm9, mm10.
intensity_cols	Depending on which columns are present, this value will be used to get quantiles and ultimately calculate the correlations: • "total_signal" : Used by the peak calling software SEACR. *NOTE*: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal". • "qValue"Used by the peak calling software MACS2/3. Should contain the negative log of the p-values after multiple testing correction. • "Peak Score" : Used by the peak calling software HOMER.
bin_size	Default of 100. Base-pair size of the bins created to measure correlation. Use smaller value for higher resolution but longer run time and larger memory usage.
keep_chr	Which chromosomes to keep.
drop_empty_chr	Drop chromosomes that are not present in any of the `peakfiles` (default: `FALSE`).
as_sparse	Return the rebinned peaks as a sparse matrix (default: `TRUE`), which is more efficiently stored than a dense matrix (`FALSE`).
workers	Number of cores to parallelise across (in applicable functions).
verbose	Print messages.
...	Arguments passed on to bpplapply `apply_fun` Iterator function to use. `register_now` Register the cores now with register (`TRUE`), or simply return the `BPPARAM object` (default: `FALSE`). `use_snowparam` Whether to use SnowParam (default: `TRUE`) or MulticoreParam (`FALSE`) when parallelising across multiple `workers`. `progressbar logical(1)` Enable progress bar (based on plyr:::progress_text). `X` Any object for which methods `length`, `[`, and `[[` are implemented. `FUN` The `function` to be applied to each element of `X`.

Value

Binned peaks matrix

Examples

```
data("CnR_H3K27ac")
data("CnT_H3K27ac")
peakfiles <- list(CnR_H3K27ac=CnR_H3K27ac, CnT_H3K27ac=CnT_H3K27ac)

#increasing bin_size for speed
peakfiles_rebinned <- rebin_peaks(peakfiles = peakfiles,
                                  genome_build = "hg19",
                                  bin_size = 5000)
```

---

tidy_peakfile *Tidy peakfiles in GRanges*

Description

This function filters peak files by removing peaks in blacklisted regions and in non-standard chromosomes. It also checks that the input list of peakfiles is named. If no names are provided, default file names will be used.

Usage

```
tidy_peakfile(peaklist, blacklist)
```

Arguments

peaklist	A named list of peak files as GRanges object. Objects must be named and listed using `list()`. e.g. `list("name1"=file1, "name2"=file2)` If not named, default names are assigned.
blacklist	Peakfile specifying blacklisted regions as GRanges object.

Value

list of GRanges object

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object
data("hg19_blacklist") # blacklist region for hg19 genome

### Create Named Peaklist ###
peaklist <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)
```

```
### Run ###
peaklist_tidy <- tidy_peakfile(peaklist = peaklist,
                               blacklist = hg19_blacklist)
```

`translate_genome` *Translate genome*

Description

Translate the name of a genome build from one format to another.

Usage

```
translate_genome(
  genome,
  style = c("UCSC", "Ensembl", "NCBI"),
  default_genome = NULL,
  omit_subversion = TRUE
)
```

Arguments

`genome`	A character vector of genomes equivalent to UCSC version or Ensembl Assemblies
`style`	A single value equivalent to "UCSC" or "Ensembl" specifying the output genome
`default_genome`	Default genome build when `genome` is `NULL`.
`omit_subversion`	Omit any subversion suffixes after the ".".

Value

Standardized genome build name as a character string.

Examples

```
genome <- translate_genome(genome="hg38", style="Ensembl")
genome2 <- translate_genome(genome="mm10", style="UCSC")
```

tss_plot *Read count frequency around TSS*

Description

This function generates a plot of read count frequency around TSS.

Usage

```
tss_plot(
  peaklist,
  annotation,
  upstream = 3000,
  downstream = upstream,
  conf = 0.95,
  resample = 500,
  workers = 1
)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed and named using `list()`. e.g. `list("name1"=file1, "name2"=file2)` If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.
upstream	upstream from TSS site
downstream	downstream from TSS site
conf	Confidence interval threshold estimated by bootstrapping (`0.95` means 95 Argument passed to plotAvgProf.
resample	Number of bootstrapped iterations to run. Argument passed to plotAvgProf.
workers	Number of cores to parallelise bootstrapping across. Argument passed to plotAvgProf.

Value

profile plot in a list.

Examples

```
### Load Data ###
data("CnT_H3K27ac") # example peaklist GRanges object
data("CnR_H3K27ac") # example peaklist GRanges object
### Create Named Peaklist ###
peaks <- list("CnT"=CnT_H3K27ac, "CnR"=CnR_H3K27ac)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
my_plot <- tss_plot(peaklist = peaks,
                    annotation = txdb)
```

width_boxplot *Peak width boxplot*

Description

This function creates boxplots showing the distribution of widths in each peak file.

Usage

```
width_boxplot(peaklist)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be named and listed using `list()`. e.g. `list("name1"=file1, "name2"=file2)`

Value

A boxplot of peak widths.

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peaklist GRanges object
data("CnT_H3K27ac") # example peaklist GRanges object

### Create Named Peaklist ###
peaks <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)

### Run ###
my_plot <- width_boxplot(peaklist = peaks)
```

write_example_peaks *Write example peaks*

Description

Write example peaks datasets to disk.

Usage

```
write_example_peaks(
  dir = file.path(tempdir(), "processed_results"),
  datasets = c("encode_H3K27ac", "CnT_H3K27ac", "CnR_H3K27ac")
)
```

Arguments

dir	Directory to save peak files to.
datasets	Example datasets from **EpiCompare** to write.

Value

Named vector of paths to saved peak files.

Examples

```
save_paths <- EpiCompare::write_example_peaks()
```

Index