Package 'EpiCompare'

October 18, 2022

Type Package

Title Comparison, Benchmarking & QC of Epigenetic Datasets

Version 1.0.0

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.

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

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

License GPL-3

Depends R (>= 4.1.0)

LazyData FALSE

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

Imports GenomicRanges, genomation, IRanges, reshape2, ggplot2, ChIPseeker, BRGenomics, clusterProfiler, plotly, stringr, dplyr, tidyr, UpSetR, rmarkdown, rtracklayer, AnnotationHub, utils, stats, methods, org.Hs.eg.db, S4Vectors, magrittr, plyranges

Suggests testthat (>= 3.0.0), badger, knitr, htmlwidgets, BiocStyle, data.table, BiocParallel, BiocFileCache, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene

VignetteBuilder knitr

RoxygenNote 7.1.2.9000

Encoding UTF-8

Config/testthat/edition 3

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

git_branch RELEASE_3_15

git_last_commit fba3043

git_last_commit_date 2022-04-27

Date/Publication 2022-10-18

Author Sera Choi [aut, cre] (<https://orcid.org/0000-0002-5077-1984>),
Brian Schilder [aut] (<https://orcid.org/0000-0001-5949-2191>),
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:

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).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). 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)
```

encode_H3K27ac *Example ChIP-seq peak file*

Description

Human H3K27ac peak file generated with ChIP-seq using K562 cell-line. Raw 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 epigenetic datasets*

Description

This function compares epigenetic datasets and performs various functional analyses.The function outputs an HTML report containing results from the analysis. The report is mainly divided into three areas: (1) Peakfile information, (2) Overlapping peaks and (3) Functional annotations.

Usage

```
EpiCompare(
  peakfiles,
  genome_build,
  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,
  interact = TRUE,
  save_output = FALSE,
  output_filename = "EpiCompare",
  output_timestamp = FALSE,
  output_dir
)
```

Arguments

`peakfiles`	A list of peak files as GRanges object and/or as paths to BED files. If paths are provided, EpiCompare creates GRanges object. EpiCompare also accepts a list

containing a mix of GRanges object and paths. Files must be listed using 'list()' and named using 'names()'. If not named, default file names will be assigned.

`genome_build` The human genome reference build used to generate peakfiles. "hg19" or "hg38".

`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 reference peak file as GRanges object. 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. Please make sure that the reference file is listed and named i.e. 'list("reference_name" = reference_peak)'.

`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.

`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.

Value

An HTML report

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
data("hg19_blacklist") # hg19 blacklist dataset
data("hg38_blacklist") # hg38 blacklist dataset
data("CnT_H3K27ac_picard") # example Picard summary output
data("CnR_H3K27ac_picard") # example Picard summary output

# prepare input data
peaks <- list(CnR_H3K27ac, CnT_H3K27ac) # create list of peakfiles
names(peaks) <- c("CnR", "CnT") # set names
picard <- list(CnR_H3K27ac_picard, CnT_H3K27ac_picard) # create list of picard outputs
names(picard) <- c("CnR", "CnT") # set names
reference_peak <- list("ENCODE" = encode_H3K27ac) # reference peak file

EpiCompare(peakfiles = peaks,
           genome_build = "hg19",
           blacklist = hg19_blacklist,
           picard_files = picard,
           reference = reference_peak,
           upset_plot = FALSE,
           stat_plot = FALSE,
           chromHMM_plot = FALSE,
           chromHMM_annotation = "K562",
           chipseeker_plot = FALSE,
           enrichment_plot = FALSE,
           tss_plot = FALSE,
           interact = FALSE,
           save_output = FALSE,
           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 listed using 'list()'. To import Picard duplication metrics (.txt file) into R as data frame, use 'picard <- read.table("/path/to/picard/output", header = TRUE, fill = TRUE)'. The list must be named e.g. 'names(picard_list) <- c("name1","name2")'

Value

A table summarizing metrics on fragments.

Examples

```
data(CnT_H3K27ac_picard) # load example picard output
data(CnR_H3K27ac_picard) # load example picard output

# 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)

picard <- list(CnT_H3K27ac_picard, CnR_H3K27ac_picard) # convert into list
names(picard) <- c("CnT_H3K27ac", "CnR_H3K27ac") # set names
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.consensus.filtered",
  nfcore_cutandrun = FALSE,
  mc.cores = 1
)
```

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 "*.peaks.bed.stringent.bed$" • "peaks.consensus"Finds files ending in "*.peaks.bed.stringent.bed$" • "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.
mc.cores	Number of cores to parallelise importing

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="*.narrowPeaks.bed$")
```

---

hg19_blacklist *Human genome hg19 blacklisted regions*

Description

Obtained from https://www.encodeproject.org/files/ENCFF001TDO/. The ENCODE blacklist includes regions of the 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 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)
```

---

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 using 'list()' and named using 'names()' 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

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object

peaks <- list(encode_H3K27ac, CnT_H3K27ac) # create list
names(peaks) <- c("encode", "CnT") # set names

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

---

overlap_percent *Calculate percentage of overlapping peaks*

Description

Calculate percentage of overlapping peaks

Usage

```
overlap_percent(peaklist, reference, invert = FALSE)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed using 'list()' and named using 'names()' If not named, default file names will be assigned.
reference	reference peaks file
invert	whether to invert

Value

data frame

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

peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names
reference_peak <- list("ENCODE"=encode_H3K27ac)

overlap <- overlap_percent(peaklist=peaks,
                           reference=reference_peak[[1]])
```

`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 using 'list()' and named using 'names()' 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

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object

peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names
reference_peak <- list("ENCODE"=encode_H3K27ac)

out <- overlap_stat_plot(reference = reference_peak,
                         peaklist = peaks)
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 listed using 'list()' and named using 'names()'. If not named, default file names are assigned.

Value

Upset plot of overlapping peaks

Examples

```
data("encode_H3K27ac") # load example data
data("CnT_H3K27ac") # load example data
peakfile <- list(encode_H3K27ac, CnT_H3K27ac) # create list
names(peakfile) <- c("ENCODE","CnT") # name list
my_plot <- overlap_upset_plot(peaklist = peakfile) # run function
```

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(peak_list, blacklist)
```

Arguments

peak_list	A named list of peak files as GRanges object. Objects listed using 'list()' and named using 'names()'.
blacklist	A GRanges object containing blacklisted regions.

Value

A summary table of peak information

Examples

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("hg19_blacklist") # example blacklist dataset as GRanges object

peaklist <- list(encode_H3K27ac, CnT_H3K27ac) # list two peakfiles
names(peaklist) <- c("encode", "CnT") # set names

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

plot_ChIPseeker_annotation
Create ChIPseeker annotation plot

Description

This function annotates peaks using 'annotatePeak' from 'ChIPseeker' package. 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 using 'list()' and named using 'names()' If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.

Value

barplot

Examples

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

peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names

## 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.

Usage

```
plot_chromHMM(peaklist, chromHMM_annotation, genome_build, interact = TRUE)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed using 'list()' and named using 'names()' If not named, default file names will be assigned.
chromHMM_annotation	ChromHMM annotation list
genome_build	The human genome reference build used to generate peakfiles. "hg19" or "hg38".
interact	Default TRUE. By default, the heatmaps are interactive. If set FALSE, the function generates a static ChromHMM heatmap.

Value

ChromHMM heatmap

Examples

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

peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names

## not run
## import ChromHMM annotation
# chromHMM_annotation_K562 <- get_chromHMM_annotation("K562")

# my_plot <- plot_chromHMM(peaklist=peaks,
#                          chromHMM_annotation=chromHMM_annotation_K562,
#                          genome_build = "hg19")
```

---

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 using 'list()' and named using 'names()' If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.

Value

KEGG and GO dot plots

Examples

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

peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names
```

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

`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 listed using 'list()' and named using 'names()'. If not named, default file names are assigned.
`blacklist`	Peakfile specifying blacklisted regions as GRanges object.

Value

list of GRanges object

Examples

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object
data("hg19_blacklist") # blacklist region for hg19 genome

peaklist <- list(encode_H3K27ac, CnT_H3K27ac) # list two peakfiles
names(peaklist) <- c("encode", "CnT") # set names

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

tss_plot *Read count frequency around TSS*

Description

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

Usage

```
tss_plot(peaklist, annotation)
```

Arguments

peaklist	A list of peak files as GRanges object. Files must be listed using 'list()' and named using 'names()' If not named, default file names will be assigned.
annotation	A TxDb annotation object from Bioconductor.

Value

profile plot in a list.

Examples

```
data("CnT_H3K27ac") # example dataset as GRanges object
data("CnR_H3K27ac") # example dataset as GRanges object
peaks <- list(CnT_H3K27ac, CnR_H3K27ac) # create a list
names(peaks) <- c("CnT", "CnR") # set names

## not run
# txdb<-TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
# my_plot <- tss_plot(peaklist = peaks,
#                     annotation = txdb)
## first plot
# my_plot[1]
```

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 listed using 'list()'. Files must be named using 'names(peaklist) <- c("sample1","sample2)'. If not named, default file names will be assigned.

Value

A boxplot of peak widths.

Examples

```
data("encode_H3K27ac") # example dataset as GRanges object
data("CnT_H3K27ac") # example dataset as GRanges object

peaks <- list(encode_H3K27ac, CnT_H3K27ac) # create list
names(peaks) <- c("encode", "CnT") # set names

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