

# Package ‘EpiCompare’

May 15, 2024

**Type** Package

**Title** Comparison, Benchmarking & QC of Epigenomic Datasets

**Version** 1.9.0

**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 (>= 4.2.0)

**Imports** AnnotationHub, BRGenomics, ChIPseeker, data.table, genomation,

GenomicRanges, IRanges, GenomeInfoDb, ggplot2 (>= 3.5.0),

htmltools, methods, plotly, reshape2, rmarkdown, rtracklayer,

stats, stringr, utils, BiocGenerics, downloadthis, parallel

**Suggests** rworkflows, BiocFileCache, BiocParallel, BiocStyle,

clusterProfiler, GenomicAlignments, grDevices, 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,

ComplexUpset, plyranges, scales, Matrix, consensusSeeker,

heatmaply, viridis

**VignetteBuilder** knitr

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

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** FALSE

**RoxygenNote** 7.3.1

**git\_url** <https://git.bioconductor.org/packages/EpiCompare>

**git\_branch** devel

**git\_last\_commit** e29605b

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-05-15

**Author** Sera Choi [aut] (<<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>>),  
Thomas Roberts [cre]

**Maintainer** Thomas Roberts <tomroberts.work15@gmail.com>

## Contents

as_interactive . . . . .	4
bpplapply . . . . .	4
checkCache . . . . .	6
check_cell_lines . . . . .	6
check_genome_build . . . . .	7
check_grlist_cols . . . . .	7
check_list_names . . . . .	8
check_workers . . . . .	8
clean_granges . . . . .	9
CnR_H3K27ac . . . . .	9
CnR_H3K27ac_picard . . . . .	10
CnT_H3K27ac . . . . .	11
CnT_H3K27ac_picard . . . . .	11
compute_consensus_peaks . . . . .	12
compute_corr . . . . .	14
download_button . . . . .	16
encode_H3K27ac . . . . .	17
EpiCompare . . . . .	18
fig_length . . . . .	22
fragment_info . . . . .	22
gather_files . . . . .	23

gather_files_names . . . . .	24
get_bpparam . . . . .	25
get_chromHMM_annotation . . . . .	26
group_files . . . . .	27
hg19_blacklist . . . . .	27
hg38_blacklist . . . . .	28
is_granges . . . . .	29
liftover_grlist . . . . .	29
messenger . . . . .	30
message_parallel . . . . .	31
mm10_blacklist . . . . .	31
mm9_blacklist . . . . .	32
overlap_heatmap . . . . .	32
overlap_percent . . . . .	33
overlap_stat_plot . . . . .	34
overlap_upset_plot . . . . .	35
peak_info . . . . .	36
plot_ChIPseeker_annotation . . . . .	37
plot_chromHMM . . . . .	38
plot_corr . . . . .	39
plot_enrichment . . . . .	41
plot_precision_recall . . . . .	42
precision_recall . . . . .	44
precision_recall_matrix . . . . .	45
predict_precision_recall . . . . .	46
predict_values . . . . .	47
prepare_blacklist . . . . .	47
prepare_genome_builds . . . . .	48
prepare_peaklist . . . . .	49
prepare_reference . . . . .	50
read_bowtie . . . . .	50
read_peaks . . . . .	51
rebin_peaks . . . . .	51
remove_nonstandard_chrom . . . . .	53
report_command . . . . .	54
report_header . . . . .	54
save_output . . . . .	55
set_min_max . . . . .	55
stopper . . . . .	56
tidy_peakfile . . . . .	56
translate_genome . . . . .	57
tss_plot . . . . .	58
width_boxplot . . . . .	59
write_example_peaks . . . . .	60

as_interactive	<i>As interactive</i>
----------------	-----------------------

---

**Description**

Convert a [ggplot](#) object to [plotly](#), and enable it to be plotted within an Rmarkdown HTML file.

**Usage**

```
as_interactive(
  plt,
  to_widget = isTRUE(getOption("knitr.in.progress")),
  add_boxmode = FALSE
)
```

**Arguments**

plt	ggplot object.
to_widget	Convert to a widget so it works within Rmarkdown HTML files. By default, this will be only be set to TRUE when being run within the context of <b>knitr</b> rendering.
add_boxmode	Add extra <a href="#">layout</a> to enable dodged boxplots.

**Value**

A [plotly](#) object or a [tagList](#) wrapping the [plotly](#) object.

**Source**

[GitHub Issue to check whether knitting](#)

bpplapply	<i>Wrapper for <a href="#">bplapply</a></i>
-----------	---------------------------------------------

---

**Description**

Wrapper function for [bplapply](#) that automatically handles issues with **BiocParallel** related to different OS platforms.

**Usage**

```

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

```

**Arguments**

X	Any object for which methods <code>length</code> , <code>[</code> , and <code>[[</code> 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	<code>logical(1)</code> Enable progress bar (based on <code>plyr::progress_text</code> ).
verbose	Print messages.
use_snowparam	Whether to use <a href="#">SnowParam</a> (default: <code>TRUE</code> ) or <a href="#">MulticoreParam</a> ( <code>FALSE</code> ) when parallelising across multiple workers.
register_now	Register the cores now with <a href="#">register</a> ( <code>TRUE</code> ), or simply return the BPPARAM object (default: <code>FALSE</code> ).
...	Arguments passed on to <a href="#">BiocParallel::bplapply</a>
	BPPARAM An optional <a href="#">BiocParallelParam</a> instance determining the parallel back-end to be used during evaluation, or a list of <a href="#">BiocParallelParam</a> instances, to be applied in sequence for nested calls to <b>BiocParallel</b> functions.
	BPREDO A list of output from <code>bplapply</code> with one or more failed elements. When a list is given in BPREDO, <code>bpok</code> 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 <a href="#">bpoptions</a> .

**Value**

(Named) list.

**Examples**

```

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

```

---

checkCache	<i>Check cache</i>
------------	--------------------

---

**Description**

Quick function to check if object is already saved.

**Usage**

```
checkCache(cache = BiocFileCache::BiocFileCache(ask = FALSE), url)
```

**Arguments**

cache	BiocFileCache.
url	Path to cached file.

**Value**

path

---

check_cell_lines	<i>Check cell lines</i>
------------------	-------------------------

---

**Description**

Check whether a list of cell lines matches any of those that are made available through EpiCompare.

**Usage**

```
check_cell_lines(cell_lines = NULL, verbose = TRUE)
```

**Arguments**

cell_lines	A character vector of cell line names. If NULL (default), will return names of all cell lines.
verbose	Print messages.

**Value**

Character vector, or NULL.

---

check_genome_build	<i>Check genome build</i>
--------------------	---------------------------

---

**Description**

Check that the genome build is valid and require specific reference datasets to be installed.

**Usage**

```
check_genome_build(genome_build, type = "txdb")
```

**Arguments**

genome_build	Genome build name.
type	whether to fetch the txdb or bsgen reference data

**Value**

txdb or bsgen

---

check_grlist_cols	<i>Check <a href="#">GRanges</a> list columns</i>
-------------------	---------------------------------------------------

---

**Description**

Check that at least one of the required columns is in a list of [GRanges](#) objects. Elements that do not meet this criterion will be dropped from the list.

**Usage**

```
check_grlist_cols(grlist, target_cols)
```

**Arguments**

grlist	Named list of <a href="#">GRanges</a> objects.
target_cols	A character vector of column names to search for.

**Value**

Named list of [GRanges](#) objects.

---

check_list_names	<i>Check peaklist is named</i>
------------------	--------------------------------

---

**Description**

This function checks whether the peaklist is named. If not, default file names are assigned.

**Usage**

```
check_list_names(peaklist, default_prefix = "sample")
```

**Arguments**

peaklist            A list of peak files as GRanges object.  
default\_prefix    Default prefix to use when creating names for peaklist.

**Value**

named peaklist

---

check_workers	<i>Check workers</i>
---------------	----------------------

---

**Description**

Assign parallel worker cores.

**Usage**

```
check_workers(workers = NULL)
```

**Arguments**

workers            Number of cores to parallelise across (in applicable functions). If NULL, will set to the total number of available cores minus 1.

**Value**

Integer

**Examples**

```
workers <- check_workers()
```



---

clean_granges	<i>Clean GRanges</i>
---------------	----------------------

---

### Description

Remove columns from the metadata (`GenomicRanges::mcols`) that conflicts with [GRanges](#) conventions.

### Usage

```
clean_granges(
  gr,
  nono_cols = c("seqnames", "ranges", "strand", "seqlevels", "seqlengths", "isCircular",
    "start", "end", "width", "element")
)
```

### Arguments

<code>gr</code>	A <a href="#">GRanges</a> object.
<code>nono_cols</code>	Problematic columns to search for and remove (if present).

### Value

Cleaned [GRanges](#) object.

---

CnR_H3K27ac	<i>Example CUT&amp;Run peak file</i>
-------------	--------------------------------------

---

### 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	<i>Example Picard duplication metrics file 2</i>
--------------------	--------------------------------------------------

---

**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**

<code>grlist</code>	Named list of <a href="#">GRanges</a> objects.
<code>groups</code>	A character vector of the same length as <code>grlist</code> defining how to group <a href="#">GRanges</a> objects when computing consensus peaks.
<code>genome_build</code>	Genome build name.
<code>lower, upper</code>	The lower and upper bounds for the slice.
<code>min.gapwidth</code>	Ranges separated by a gap of at least <code>min.gapwidth</code> positions are not merged.
<code>method</code>	Method to call peaks with: <ul style="list-style-type: none"> <li>"granges" : Simple overlap procedure using <a href="#">GRanges</a> functions. Faster but less accurate.</li> <li>"consensusseeker" : Uses <a href="#">findConsensusPeakRegions</a> to compute consensus peaks. Slower but more accurate.</li> </ul>

... 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 chromosomes 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 shrunk 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,
  fill_diag = NA,
  workers = check_workers(),
  save_path = tempfile(fileext = ".corr.csv.gz")
)
```

**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. <code>list("reference_name" = reference_peak)</code> . 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 <b>**all**</b> peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use <a href="#">liftover_gelist</a> 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 peak files (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: <ul style="list-style-type: none"> <li>• "total_signal" : Used by the peak calling software <b>SEACR</b>. <i>NOTE</i>: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal".</li> <li>• "qValue" Used by the peak calling software <b>MACS2/3</b>. Should contain the negative log of the p-values after multiple testing correction.</li> <li>• "Peak Score" : Used by the peak calling software <b>HOMER</b>.</li> </ul>
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).
fill_diag	Fill the diagonal of the overlap matrix.
workers	Number of threads to parallelize across.
save_path	Path to save a table of correlation results to.

### 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 = 200000,
                          workers = 1)
```

---

download_button	<i>Download local file</i>
-----------------	----------------------------

---

## Description

Save an object as RDS and create a download button that can be rendered to Rmarkdown HTML pages. Uses the package **downloadthis**.

## Usage

```
download_button(
  object,
  save_output = FALSE,
  outfile_dir = NULL,
  filename = NULL,
  button_label = paste0("Download: ", "<code>", filename, "</code>"),
  output_extension = ".rds",
  icon = "fa fa-save",
  button_type = "success",
  self_contained = TRUE,
  add_download_button = TRUE,
  verbose = TRUE
)
```

## Arguments

object	R object to serialize.
save_output	Default FALSE. If TRUE, all outputs (tables and plots) of the analysis will be saved in a folder (EpiCompare_file).
outfile_dir	Directory to save the file to.
filename	Name of the file to save.
button_label	Character (HTML), button label
output_extension	Extension of the output file. Currently, .csv, .xlsx, and .rds are supported. If a (named) list is passed to the function, only .xlsx and .rds are supported.
icon	Fontawesome tag e.g.: "fa fa-save"
button_type	Character, one of the standard Bootstrap types
self_contained	A boolean to specify whether your HTML output is self-contained. Default to FALSE.
add_download_button	Add download buttons for each plot or dataset.
verbose	Print messages.



**Value**

Download button as HTML text.

**Source**

[csv2 Issue](#).

[Plotly Issue](#)

**Examples**

```
button <- download_button(object=mtcars)
```

---

```
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)
```

**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 = NULL,
  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,
  tss_distance = c(-3000, 3000),
  precision_recall_plot = FALSE,
  n_threshold = 20,
  corr_plot = FALSE,
  bin_size = 5000,
  interact = TRUE,
  add_download_button = FALSE,
  save_output = FALSE,
  output_filename = "EpiCompare",
  output_timestamp = FALSE,
  output_dir,
  display = NULL,
  run_all = FALSE,
  workers = 1,
  quiet = FALSE,
  error = FALSE,
  debug = FALSE
)
```

**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 genomic regions. Blacklists included in **EpiCompare** are:

- NULL (default): Automatically selects the appropriate blacklist based on the `genome_build_output` argument.
- "hg19\_blacklist": Regions of hg19 genome that have anomalous and/or unstructured signals. [hg19\\_blacklist](#)
- "hg38\_blacklist": Regions of hg38 genome that have anomalous and/or unstructured signals. [hg38\\_blacklist](#)
- "mm10\_blacklist": Regions of mm10 genome that have anomalous and/or unstructured signals. [mm10\\_blacklist](#)
- "mm9\_blacklist": Blacklisted regions of mm10 genome that have been lifted over from [mm10\\_blacklist](#). [mm9\\_blacklist](#)
- <user\_input>: A custom user-provided blacklist in [GRanges](#) format.

**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: <ul style="list-style-type: none"> <li>• "K562" = K-562 cells</li> <li>• "Gm12878" = Cellosaurus cell-line GM12878</li> <li>• "H1hesc" = H1 Human Embryonic Stem Cell</li> <li>• "Hepg2" = Hep G2 cell</li> <li>• "Hmec" = Human Mammary Epithelial Cell</li> <li>• "Hsmm" = Human Skeletal Muscle Myoblasts</li> <li>• "Huvec" = Human Umbilical Vein Endothelial Cells</li> <li>• "Nhek" = Normal Human Epidermal Keratinocytes</li> <li>• "Nhlf" = Normal Human Lung Fibroblasts</li> </ul>
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.
tss_distance	A vector specifying the distance upstream and downstream around transcription start sites (TSS). The default value is <code>c(-3000, 3000)</code> ; meaning peak frequency 3000bp upstream and downstream of TSS will be displayed.
precision_recall_plot	Default is FALSE. If TRUE, creates a precision-recall curve plot and an F1 plot using <a href="#">plot_precision_recall</a> .
n_threshold	Number of thresholds to test.
corr_plot	Default is FALSE. If TRUE, creates a correlation plot across all peak files using <a href="#">plot_corr</a> .
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, plots are interactive. If set FALSE, all plots in the report will be static.
add_download_button	Add download buttons for each plot or dataset.
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: <ul style="list-style-type: none"> <li>• "browser" : Display the report in your default web browser.</li> <li>• "rsstudio" : Display the report in Rstudio.</li> <li>• NULL (default) : Do not display the report.</li> </ul>
run_all	Convenience argument that enables all plots/features (without specifying each argument manually) by overriding the default values. Default: FALSE.
workers	Number of threads to parallelize across.
quiet	An option to suppress printing during rendering from knitr, pandoc command line and others. To only suppress printing of the last "Output created: " message, you can set <code>rmarkdown.render.message</code> to FALSE
error	If TRUE, the Rmarkdown report will continue to render even when some chunks encounter errors (default: FALSE). Passed to <a href="#">opts_chunk</a> .
debug	Run in debug mode, where are messages and warnings are printed within the HTML report (default: FALSE).

### 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("CnT_H3K27ac_picard") # example Picard summary output
data("CnR_H3K27ac_picard") # example Picard summary output

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

### Run EpiCompare ###
output_html <- EpiCompare(peakfiles = peakfiles,
  genome_build = list(peakfiles="hg19",
    reference="hg19"),
  picard_files = picard_files,
```

```

        reference = reference,
        output_filename = "EpiCompare_test",
        output_dir = tempdir())
# utils::browseURL(output_html)

```

---

fig_length	<i>Dynamic Figure Length Generator</i>
------------	----------------------------------------

---

### Description

This function calculates the appropriate figure height depending on the number of items.

### Usage

```
fig_length(default_size, number_of_items, max_items)
```

### Arguments

default_size	The default figure length. Must be numeric.
number_of_items	Number of peak files, or terms.
max_items	Maximum number of peak files, or terms.

### Value

Figure height/width. A number.

---

fragment_info	<i>Summary on fragments</i>
---------------	-----------------------------

---

### 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 <code>list()</code> . e.g. <code>list("name1"=file1, "name2"=file2)</code> . To import Picard duplication metrics (.txt file) into R as data frame, use <code>picard &lt;- read.table("/path/to/picard/output", header = TRUE, fill = TRUE)</code> .
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**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 <- list("CnT_H3K27ac"=CnT_H3K27ac_picard,
                   "CnR_H3K27ac"=CnR_H3K27ac_picard)
df <- fragment_info(picard_list = picard_list)
```

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 = check_workers(),
  verbose = TRUE
)
```

**Arguments**

dir	Directory to search within.
type	File type to search for. Options include: <ul style="list-style-type: none"> <li>• "&lt;pattern&gt;" Finds files matching an arbitrary regex pattern specified by user.</li> <li>• "peaks.stringent" Finds files ending in "*.stringent.bed\$"</li> <li>• "peaks.consensus" Finds files ending in "*.consensus.peaks.bed\$"</li> <li>• "peaks.consensus.filtered" Finds files ending in "*.consensus.peaks.filtered.awk.bed\$"</li> <li>• "picard" Finds files ending in "*.target.markdup.MarkDuplicates.metrics.txt\$"</li> </ul>

nfcore_cutandrun	Whether the files were generated by the <b>nf-core/cutandrun</b> 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 <b>GRanges</b> .
rbind_list	Bind all objects into one.
workers	Number of cores to parallelise across (in applicable functions). If NULL, will set to the total number of available cores minus 1.
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\_signal** Maximum 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",
                                  workers = 1)
```

---

gather_files_names	<i>Make file names</i>
--------------------	------------------------

---

### Description

Support function for **gather\_files**.

### Usage

```
gather_files_names(paths, type, nfcore_cutandrun, verbose = TRUE)
```



**Arguments**

paths	Character vector of file paths.
type	File type to search for. Options include: <ul style="list-style-type: none"> <li>• "&lt;pattern&gt;" Finds files matching an arbitrary regex pattern specified by user.</li> <li>• "peaks.stringent" Finds files ending in "/*.stringent.bed"</li> <li>• "peaks.consensus" Finds files ending in "/*.consensus.peaks.bed"</li> <li>• "peaks.consensus.filtered" Finds files ending in "/*.consensus.peaks.filtered.awk.bed"</li> <li>• "picard" Finds files ending in "/*.target.markdup.MarkDuplicates.metrics.txt"</li> </ul>
nfcore_cutandrun	Whether the files were generated by the <b>nf-core/cutandrun</b> Nextflow pipeline. If TRUE, can use the standardised folder structure to automatically generate more descriptive file names with sample IDs.
verbose	Print messages.

**Value**

Named character vector.

---

get_bpparam	<i>Get <a href="#">BiocParallel</a> parameters</i>
-------------	----------------------------------------------------

---

**Description**

Get (and optionally register) [BiocParallel](#) parameter (BPPARAM). [SnowParam](#) is the default function as it tends to be more robust. However, because it doesn't work on Windows, this function automatically detected the Operating System and switches to [SerialParam](#) as needed.

**Usage**

```
get_bpparam(
  workers,
  progressbar = workers > 1,
  use_snowparam = TRUE,
  register_now = FALSE
)
```

**Arguments**

workers	Number of threads to parallelize across.
progressbar	logical(1) Enable progress bar (based on <code>plyr:::progress_text</code> ).
use_snowparam	Whether to use <a href="#">SnowParam</a> (default: TRUE) or <a href="#">MulticoreParam</a> (FALSE) when parallelising across multiple workers.
register_now	Register the cores now with <a href="#">register</a> (TRUE), or simply return the BPPARAM object (default: FALSE).

**Value**

BPPARAM

---

get\_chromHMM\_annotation*Download ChromHMM annotation file(s)*

---

**Description**

Download ChromHMM annotation file(s) for a given cell-line (returned as a [GRanges](#) object) or a list of cell-lines (returned as a named list of [GRanges](#) objects). All annotations are aligned to the hg19 genome build. All data can be found on the UCSC Genome Browser [here](#).

**Usage**

```
get_chromHMM_annotation(
  cell_line,
  cache = BiocFileCache::BiocFileCache(ask = FALSE)
)
```

**Arguments**

cell\_line      ChromHMM annotation for user-specified 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

**Value**

Cell-line specific ChromHMM annotation file. Default K562 cell-line.

---

group_files	<i>Group files</i>
-------------	--------------------

---

**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.

**Value**

Named peak files

**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	<i>Human genome hg19 blacklisted regions</i>
----------------	----------------------------------------------

---

**Description**

Obtained from <https://www.encodeproject.org/files/ENCFF001TD0/>. 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/ENCFF001TD0/
hg19_blacklist <-ChIPseeker::readPeakFile(file.path(path), as = "GRanges")
usethis::use_data(hg19_blacklist, overwrite = TRUE)
```

---

hg38_blacklist	<i>Human genome hg38 blacklisted regions</i>
----------------	----------------------------------------------

---

**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)
```

---

is_granges	<i>Is an object of class GRanges</i>
------------	--------------------------------------

---

### Description

Check whether an object is of the class [GRanges](#).

### Usage

```
is_granges(obj)
```

### Arguments

obj                   Any R object.

### Value

Boolean.

---

liftover_grlist	<i>Liftover peak list</i>
-----------------	---------------------------

---

### 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 <a href="#">GRanges</a> objects, or simply a single unlisted <a href="#">GRanges</a> 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 <a href="#">seqlevelsStyle</a> . <ul style="list-style-type: none"><li>• "UCSC" : Uses the chromosome style "chr1".</li><li>• "NCBI" : Uses the chromosome style "1"</li></ul>
keep_chr	Which chromosomes to keep.
as_grangeslist	Return as a <a href="#">GRangesList</a> .
merge_all	Merge all <a href="#">GRanges</a> into a single <a href="#">GRanges</a> 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")
```

---

messenger	<i>Print messages</i>
-----------	-----------------------

---

Description

Conditionally print messages. Allows developers to easily control verbosity of functions, and meet Bioconductor requirements that dictate the message must first be stored to a variable before passing to [message](#).

Usage

```
messenger(..., v = TRUE, parallel = FALSE)
```

Arguments

v	Whether to print messages or not.
parallel	Whether to enable message print when wrapped in parallelised functions.

**Value**

Null

---

message_parallel	<i>Message parallel</i>
------------------	-------------------------

---

**Description**

Send messages to console even from within parallel processes

**Usage**

```
message_parallel(...)
```

**Value**

A message

---

mm10_blacklist	<i>Mouse genome mm10 blacklisted regions</i>
----------------	----------------------------------------------

---

**Description**

Obtained from <https://www.encodeproject.org/files/ENCFF547MET/>. The ENCODE blacklist includes regions of the mm10 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("mm10_blacklist")
```

**Format**

An object of class GRanges of length 164.

**Source**

The code to prepare the .Rda file is:

```
## blacklisted regions were directly downloaded
## from https://www.encodeproject.org/files/ENCFF547MET/
mm10_blacklist <-ChIPseeker::readPeakFile(file.path(path), as = "GRanges")
usethis::use_data(mm10_blacklist, overwrite = TRUE)
```

---

mm9_blacklist	<i>Mouse genome mm9 blacklisted regions</i>
---------------	---------------------------------------------

---

**Description**

Blacklisted regions of the mm9 genome build obtained by lifting over the mm10\_blacklist.

**Usage**

```
data("mm9_blacklist")
```

**Format**

An object of class GRanges of length 292.

**Source**

```
tmp <- base::get("mm10_blacklist", asNamespace("EpiCompare")) mm9_blacklist <- liftover_grlist(grlist
= tmp, input_build = "mm10", output_build = "mm9", keep_chr = NULL) usethis::use_data(mm9_blacklist,
overwrite = TRUE)
```

---

overlap_heatmap	<i>Generate heatmap of percentage overlap</i>
-----------------	-----------------------------------------------

---

**Description**

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

**Usage**

```
overlap_heatmap(
  peaklist,
  interact = TRUE,
  draw_cellnote = TRUE,
  fill_diag = NA,
  verbose = 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.
draw_cellnote	Draw the numeric values within each heatmap cell.
fill_diag	Fill the diagonal of the overlap matrix.
verbose	Print messages.



Value

An interactive heatmap

Examples

```
### Load Data ###
data("encode_H3K27ac") # example peakfile GRanges object
data("CnT_H3K27ac") # example peakfile GRanges object
### Create Named List ###
peaklist <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)
### Run ###
my_heatmap <- overlap_heatmap(peaklist = peaklist)
```

---

overlap_percent	<i>Calculate percentage of overlapping peaks</i>
-----------------	--------------------------------------------------

---

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	<i>Statistical significance of overlapping peaks</i>
-------------------	------------------------------------------------------

---

## 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 box plots 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 a TxDb object.

## Usage

```
overlap_stat_plot(
  reference,
  peaklist,
  txdb = NULL,
  interact = FALSE,
  nShuffle = 50,
  digits = 4,
  workers = check_workers()
)
```

## 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.
txdb	A TxDb annotation object from Bioconductor. This is required only if the reference file does not have BED6+4 format.

interact	Default TRUE. By default, plots are interactive. If set FALSE, all plots in the report will be static.
nShuffle	shuffle numbers
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used. For round, negative values are allowed (see ‘Details’).
workers	Number of threads to parallelize across.

### Value

A named list.

- "plot"boxplot/barplot showing the statistical significance of overlapping/non-overlapping peaks.
- "data"Plot data.

### 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)
out <- overlap_stat_plot(reference = reference,
                        peaklist = peaklist,
                        workers = 1)
```

---

overlap_upset_plot	<i>Generate Upset plot for overlapping peaks</i>
--------------------	--------------------------------------------------

---

### Description

This function generates upset plot of overlapping peaks files using the **ComplexUpset** package.

### Usage

```
overlap_upset_plot(peaklist, verbose = TRUE)
```

### 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.
verbose	Print messages

### Value

Upset plot of overlapping peaks.

Examples

```
### Load Data ###
data("encode_H3K27ac") # load example data
data("CnT_H3K27ac") # load example data
peaklist <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)
my_plot <- overlap_upset_plot(peaklist = peaklist)
```

---

peak_info	<i>Summary of Peak Information</i>
-----------	------------------------------------

---

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,
  txdb = NULL,
  tss_distance = c(-3000, 3000),
  interact = FALSE
)
```

## 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.
txdb	A TxDb annotation object from Bioconductor.
tss_distance	A vector specifying the distance upstream and downstream around transcription start sites (TSS). The default value is c(-3000, 3000); meaning peak frequency 3000bp upstream and downstream of TSS will be displayed.
interact	Default TRUE. By default, plots are interactive. If set FALSE, all plots in the report will be static.

## Value

## ggplot barplot

## Examples

[illegible]

---

plot_chromHMM	<i>Plot ChromHMM heatmap</i>
---------------	------------------------------

---

## Description

Creates a heatmap using outputs from ChromHMM using ggplot2. The function takes a list of peak-files, 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,
  genome_build,
  cell_line = NULL,
  interact = FALSE,
  return_data = FALSE
)
```

## Arguments

peaklist	A named <a href="#">list</a> of peak files as GRanges object. If list is not named, default names will be assigned.
chromHMM_annotation	ChromHMM annotation list.
genome_build	The human genome reference build used to generate peakfiles. "hg19" or "hg38".
cell_line	If not cell_line, will replace chromHMM_annotation by importing chromHMM data for a given cell line using <a href="#">get_chromHMM_annotation</a> .
interact	Default TRUE. By default, the heatmaps are interactive. If FALSE, 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 = NULL,
  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,
  draw_cellnote = TRUE,
  fill_diag = NA,
  workers = check_workers(),
  show_plot = TRUE,
  save_path = tempfile(fileext = ".corr.csv.gz")
)
```

## 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 <b>**all**</b> peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use <a href="#">liftover_grlist</a> 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: <ul style="list-style-type: none"> <li>• "total_signal" : Used by the peak calling software <b>SEACR</b>. <i>NOTE</i>: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal".</li> <li>• "qValue" Used by the peak calling software <b>MACS2/3</b>. Should contain the negative log of the p-values after multiple testing correction.</li> <li>• "Peak Score" : Used by the peak calling software <b>HOMER</b>.</li> </ul>
interact	Default TRUE. By default heatmap is interactive. If FALSE, heatmap is static.
draw_cellnote	Draw the numeric values within each heatmap cell.
fill_diag	Fill the diagonal of the overlap matrix.
workers	Number of threads to parallelize across.
show_plot	Show the plot.
save_path	Path to save a table of correlation results to.

## 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 here,
## but lower values will give more precise results (and lower correlations)
cp <- plot_corr(peakfiles = peakfiles,
               reference = reference,
               genome_build = "hg19",
               bin_size = 5000,
               workers = 1)
```





---

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 = 20,
  max_threshold = 1,
  workers = check_workers(),
  plot_f1 = TRUE,
  subtitle = NULL,
  color = "peaklist1",
  shape = color,
  facets = "peaklist2 ~ .",
  interact = FALSE,
  show_plot = TRUE,
  save_path = tempfile(fileext = "precision_recall.csv"),
  verbose = TRUE
)
```

### Arguments

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

**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** **[Deprecated]** Please use rows and cols instead.

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

**show\_plot** Show the plot.

**save\_path** File path to save precision-recall results to.

**verbose** Print messages.

**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,
                               workers = 1)
```

---

```
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 = 20,
  max_threshold = 1,
  cast = TRUE,
  workers = 1,
  verbose = TRUE,
  save_path = tempfile(fileext = "precision_recall.csv"),
  ...
)
```

## Arguments

- |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| peakfiles         | A list of peak files as <a href="#">GRanges</a> object and/or as paths to BED files. If paths are provided, EpiCompare imports the file as <a href="#">GRanges</a> object. EpiCompare also accepts a list containing a mix of <a href="#">GRanges</a> objects and paths. Files must be listed and named using <code>list()</code> . E.g. <code>list("name1"=file1, "name2"=file2)</code> . If no names are specified, default file names will be assigned.                                                                                                                           |
| reference         | A named list containing reference peak file(s) as <a href="#">GRanges</a> object. Please ensure that the reference file is listed and named i.e. <code>list("reference_name" = reference_peak)</code> . 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 | <p>Depending on which columns are present, <a href="#">GRanges</a> will be filtered at each threshold according to one or more of the following:</p> <ul style="list-style-type: none"> <li>• "total_signal" : Used by the peak calling software <a href="#">SEACR</a>. <i>NOTE</i>: Another <a href="#">SEACR</a> column (e.g. "max_signal") can be used together or instead of "total_signal".</li> <li>• "qValue" Used by the peak calling software <a href="#">MACS2/3</a>. Should contain the negative log of the p-values after multiple testing correction.</li> </ul>        |

- "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.

**cast** Cast the data into a format that's more compatible with **ggplot2**.

**workers** Number of threads to parallelize across.

**verbose** Print messages.

**save\_path** File path to save precision-recall results to.

**...** 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

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,
                        workers = 1)
```

---

```
precision_recall_matrix
```

*Create a precision-recall matrix*

---

## Description

Converts a list of peak files to a symmetric matrix where the y-axis indicates precision and the x-axis indicates recall.

**Usage**

```
precision_recall_matrix(peaklist, fill_diag = NA, verbose = TRUE)
```

**Arguments**

fill_diag	Fill the diagonal of the overlap matrix.
verbose	Print messages.

**Value**

matrix

---

predict\_precision\_recall

*Predict precision-recall*

---

**Description**

Predict specific values of precision or recall by fitting a model to a precision-recall curve. Predictions that are <0 will automatically be set to 0. Predictions that are >100 will automatically be set to 100.

**Usage**

```
predict_precision_recall(
  pr_df,
  fun = stats::loess,
  precision = seq(10, 100, 10),
  recall = seq(10, 100, 10)
)
```

**Arguments**

pr_df	Precision-recall data.frame generated by <a href="#">precision_recall</a> .
fun	Function to fit the data with.
precision	Precision values to predict recall from.
recall	Recall values to predict precision from.

**Value**

A named list of fitted models and predictions.

**Source**

Fix for producing NAs from loess fun.

**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)
predictions <- predict_precision_recall(pr_df = pr_df)
```

---

predict_values	<i>Predict values</i>
----------------	-----------------------

---

**Description**

Fit a model and make predictions from it.

**Usage**

```
predict_values(df, fun, values, input_var, predicted_var)
```

**Arguments**

df	data.frame
fun	Function to fit the data with.
values	Values to make predictions from.
input_var	Input variable column name.
predicted_var	Predicted variable name.

**Value**

data.frame

---

prepare_blacklist	<i>Prepare blacklist as GRanges</i>
-------------------	-------------------------------------

---

**Description**

Selects the appropriate blacklist in a variety of conditions.

**Usage**

```
prepare_blacklist(
  blacklist,
  output_build,
  blacklist_build = NULL,
  verbose = TRUE
)
```

**Arguments**

output_build	Desired genome build for grlist to be lifted over to.
blacklist_build	Genome build of the blacklist. Only used when blacklist is a user-supplied <a href="#">GRanges</a> object.
verbose	Print messages.

**Value**

A [GRanges](#) objects of blacklisted genomic regions from the relevant genome build.

---

prepare\_genome\_builds *Prepare genome builds*

---

**Description**

Parse the genome\_build argument into peaklist\_build and reference\_build.

**Usage**

```
prepare_genome_builds(genome_build, blacklist = NULL)
```

**Arguments**

genome_build	<p>A named list indicating the human genome build used to generate each of the following inputs:</p> <ul style="list-style-type: none"> <li>"peakfiles" : Genome build for the peakfiles input. Assumes genome build is the same for each element in the peakfiles list.</li> <li>"reference" : Genome build for the reference input.</li> <li>"blacklist" : Genome build for the blacklist input.</li> </ul>
--------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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"
```



- blacklist** A [GRanges](#) object containing blacklisted genomic regions. Blacklists included in **EpiCompare** are:
- NULL (default): Automatically selects the appropriate blacklist based on the `genome_build_output` argument.
  - "hg19\_blacklist": Regions of hg19 genome that have anomalous and/or unstructured signals. [hg19\\_blacklist](#)
  - "hg38\_blacklist": Regions of hg38 genome that have anomalous and/or unstructured signals. [hg38\\_blacklist](#)
  - "mm10\_blacklist": Regions of mm10 genome that have anomalous and/or unstructured signals. [mm10\\_blacklist](#)
  - "mm9\_blacklist": Blacklisted regions of mm10 genome that have been lifted over from [mm10\\_blacklist](#). [mm9\\_blacklist](#)
  - <user\_input>: A custom user-provided blacklist in [GRanges](#) format.

**Value**

Named list.

---

prepare_peaklist	<i>Prepare peaklist as GRanges</i>
------------------	------------------------------------

---

**Description**

Prepare peaklist as GRanges

**Usage**

```
prepare_peaklist(peaklist, remove_empty = TRUE, as_grangeslist = FALSE)
```

**Arguments**

- peaklist** A named list of peaks as [GRanges](#) or paths to BED files.
- remove\_empty** Remove any empty elements in the list.
- as\_grangeslist** Convert output to class [GRangesList](#) before returning.

**Value**

A list of [GRanges](#) objects

---

prepare_reference	<i>Prepare reference as GRanges</i>
-------------------	-------------------------------------

---

**Description**

Prepare reference as GRanges

**Usage**

```
prepare_reference(  
  reference,  
  max_elements = NULL,  
  remove_empty = TRUE,  
  as_list = TRUE,  
  as_grangeslist = FALSE  
)
```

**Arguments**

reference	A named list of <a href="#">GRanges</a> objects, or a single <a href="#">GRanges</a> object to be converted into a named list.
max_elements	Max number of elements to use within the list. Set to NULL (default) to use all elements.
remove_empty	Remove any empty elements in the list.
as_list	Return as a list.
as_grangeslist	Return as a <a href="#">GRangesList</a> (overrides as_list).

**Value**

A list of [GRanges](#) objects

---

read_bowtie	<i>Read bowtie</i>
-------------	--------------------

---

**Description**

Read a bowtie file.

**Usage**

```
read_bowtie(path, verbose = TRUE)
```

**Arguments**

path	Path to bowtie file.
verbose	Print messages.

**Value**

[data.table](#)

---

read_peaks	<i>Read peaks</i>
------------	-------------------

---

**Description**

Read peak files.

**Usage**

read\_peaks(path, type, verbose = TRUE)

**Arguments**

path	Path to peak file.
type	File type to search for. Options include: <ul style="list-style-type: none"><li>• "&lt;pattern&gt;" Finds files matching an arbitrary regex pattern specified by user.</li><li>• "peaks.stringent" Finds files ending in "*.stringent.bed\$"</li><li>• "peaks.consensus" Finds files ending in "*.consensus.peaks.bed\$"</li><li>• "peaks.consensus.filtered" Finds files ending in "*.consensus.peaks.filtered.awk.bed\$"</li><li>• "picard" Finds files ending in "*.target.markdup.MarkDuplicates.metrics.txt\$"</li></ul>
verbose	Print messages.

**Value**

[GRanges](#)

---

rebin_peaks	<i>Rebin peaks</i>
-------------	--------------------

---

**Description**

Standardise a list of peak files by rebinning them into fixed-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,
  sep = c(":", "-"),
  drop_empty_chr = FALSE,
  as_sparse = TRUE,
  workers = check_workers(),
  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 <b>**all**</b> peak and reference files to calculate the correlation matrix on. If all peak and reference files are not of the same build use <a href="#">liftover_grlist</a> 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: <ul style="list-style-type: none"> <li>• "total_signal" : Used by the peak calling software <b>SEACR</b>. <i>NOTE</i>: Another SEACR column (e.g. "max_signal") can be used together or instead of "total_signal".</li> <li>• "qValue" Used by the peak calling software <b>MACS2/3</b>. Should contain the negative log of the p-values after multiple testing correction.</li> <li>• "Peak Score" : Used by the peak calling software <b>HOMER</b>.</li> </ul>
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.
sep	Separator to be used after chromosome name (first item) and between start/end genomic coordinates (second item).
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 threads to parallelize across.
verbose	Print messages.
...	Arguments passed on to <a href="#">bpplapply</a>

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,
                                  workers = 1)

```

---

remove\_nonstandard\_chrom

*Remove non-standard chromosomes*

---

## Description

Remove non-standard chromosomes from a list of [GRanges](#) objects.

## Usage

```

remove_nonstandard_chrom(
  grlist,
  keep_chr = paste0("chr", c(seq_len(22), "X", "Y")),
  verbose = TRUE
)

```

## Arguments

grlist	Named list of <a href="#">GRanges</a> objects.
keep_chr	Which chromosomes to keep.
verbose	Print messages.

**Value**

Named list of [GRanges](#) objects.

---

report_command	<i>Report command</i>
----------------	-----------------------

---

**Description**

Reconstruct the [EpiCompare](#) command used to generate the current Rmarkdown report.

**Usage**

```
report_command(params, peaklist_tidy, reference_tidy)
```

**Arguments**

params	Parameters supplied to the Rmarkdown template.
peaklist_tidy	Post-processed target peaks.
reference_tidy	Post-processed reference peaks.

**Value**

String reconstructing R function call.

**Examples**

```
# report_command()
```

---

report_header	<i>Report header</i>
---------------	----------------------

---

**Description**

Generate a header for [EpiCompare](#) reports generated using the *EpiCompare.Rmd* template.

**Usage**

```
report_header()
```

**Value**

Header string to be rendering within Rmarkdown file.

**Examples**

```
report_header()
```

---

save_output	<i>Save output</i>
-------------	--------------------

---

**Description**

This function saves data frames and plots generated by EpiCompare.

**Usage**

```
save_output(  
  save_output = FALSE,  
  file,  
  file_type,  
  filename,  
  outpath,  
  interactive = FALSE,  
  verbose = TRUE  
)
```

**Arguments**

save_output	Default FALSE. If TRUE, outputs are saved.
file	Tables and plots to be saved.
file_type	Type of file to be saved. "data.frame", "ggplot", "image"
filename	Name of file.
outpath	Outpath
interactive	Default FALSE. If TRUE, interactive plots are saved as html.
verbose	Print messages.

**Value**

Saved data frames and plots.

---

set_min_max	<i>Set min/max</i>
-------------	--------------------

---

**Description**

Set the min/max values in a data.frame.

**Usage**

```
set_min_max(df, colname, min_val = 0, max_val = 100)
```

**Arguments**

df	data.frame
colname	Column name to check.
min_val	Minimum value.
max_val	Maximum value.

**Value**

data.frame

---

stopper	<i>Stop messages</i>
---------	----------------------

---

**Description**

Conditionally print stop messages. Allows developers to easily control verbosity of functions, and meet Bioconductor requirements that dictate the stop message must first be stored to a variable before passing to [stop](#).

**Usage**

```
stopper(..., v = TRUE)
```

**Arguments**

v	Whether to print messages or not.
---	-----------------------------------

**Value**

Null

---

tidy_peakfile	<i>Tidy peakfiles in GRanges</i>
---------------	----------------------------------

---

**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	<i>Translate genome</i>
------------------	-------------------------

---

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	<i>Read count frequency around TSS</i>
----------	----------------------------------------

---

**Description**

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

**Usage**

```
tss_plot(
  peaklist,
  txdb = NULL,
  tss_distance = c(-3000, 3000),
  conf = 0.95,
  resample = 500,
  interact = FALSE,
  workers = check_workers()
)
```

**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.
txdb	A TxDb annotation object from Bioconductor.
tss_distance	A vector specifying the distance upstream and downstream around transcription start sites (TSS). The default value is c(-3000, 3000); meaning peak frequency 3000bp upstream and downstream of TSS will be displayed.
conf	Confidence interval threshold estimated by bootstrapping (0.95 means 95 Argument passed to <a href="#">plotAvgProf</a> .
resample	Number of bootstrapped iterations to run. Argument passed to <a href="#">plotAvgProf</a> .
interact	Default TRUE. By default, plots are interactive. If set FALSE, all plots in the report will be static.
workers	Number of cores to parallelise bootstrapping across. Argument passed to <a href="#">plotAvgProf</a> .

**Value**

A named list of profile plots.

**Examples**

```
### Load Data ###
data("CnT_H3K27ac") # example peaklist GRanges object
data("CnR_H3K27ac") # example peaklist GRanges object
### Create Named Peaklist ###
peaklist <- list("CnT"=CnT_H3K27ac, "CnR"=CnR_H3K27ac)
my_plot <- tss_plot(peaklist = peaklist,
                    tss_distance=c(-50,50),
                    workers = 1)
```

---

width_boxplot	<i>Peak width boxplot</i>
---------------	---------------------------

---

**Description**

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

**Usage**

```
width_boxplot(peaklist, interact = FALSE)
```

**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)
interact	Default TRUE. By default, plots are interactive. If set FALSE, all plots in the report will be static.

**Value**

A boxplot of peak widths.

**Examples**

```
### Load Data ###
data("encode_H3K27ac") # example peaklist GRanges object
data("CnT_H3K27ac") # example peaklist GRanges object
peaklist <- list("encode"=encode_H3K27ac, "CnT"=CnT_H3K27ac)
my_plot <- width_boxplot(peaklist = peaklist)
```

---

write_example_peaks	<i>Write example peaks</i>
---------------------	----------------------------

---

**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 <b>EpiCompare</b> to write.

**Value**

Named vector of paths to saved peak files.

**Examples**

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

# Index

## \* datasets

- CnR\_H3K27ac, [9](#)
- CnR\_H3K27ac\_picard, [10](#)
- CnT\_H3K27ac, [11](#)
- CnT\_H3K27ac\_picard, [11](#)
- encode\_H3K27ac, [17](#)
- hg19\_blacklist, [27](#)
- hg38\_blacklist, [28](#)
- mm10\_blacklist, [31](#)
- mm9\_blacklist, [32](#)

## \* internal

- as\_interactive, [4](#)
- check\_cell\_lines, [6](#)
- check\_genome\_build, [7](#)
- check\_grlist\_cols, [7](#)
- check\_list\_names, [8](#)
- checkCache, [6](#)
- clean\_granges, [9](#)
- fig\_length, [22](#)
- gather\_files\_names, [24](#)
- get\_bpparam, [25](#)
- get\_chromHMM\_annotation, [26](#)
- is\_granges, [29](#)
- message\_parallel, [31](#)
- messenger, [30](#)
- precision\_recall\_matrix, [45](#)
- predict\_values, [47](#)
- prepare\_blacklist, [47](#)
- prepare\_genome\_builds, [48](#)
- prepare\_peaklist, [49](#)
- prepare\_reference, [50](#)
- read\_bowtie, [50](#)
- read\_peaks, [51](#)
- remove\_nonstandard\_chrom, [53](#)
- save\_output, [55](#)
- set\_min\_max, [55](#)
- stopper, [56](#)

as\_interactive, [4](#)

BiocParallel, [25](#)

BiocParallel::bplapply, [5](#)

BiocParallelParam, [5](#)

bplapply, [4](#)

bpoptions, [5](#)

bpplapply, [4](#), [45](#), [52](#)

check\_cell\_lines, [6](#)

check\_genome\_build, [7](#)

check\_grlist\_cols, [7](#)

check\_list\_names, [8](#)

check\_workers, [8](#)

checkCache, [6](#)

clean\_granges, [9](#)

closeAllConnections, [13](#)

CnR\_H3K27ac, [9](#)

CnR\_H3K27ac\_picard, [10](#)

CnT\_H3K27ac, [11](#)

CnT\_H3K27ac\_picard, [11](#)

compareCluster, [41](#)

compute\_consensus\_peaks, [12](#), [13](#)

compute\_corr, [14](#)

consensusSeeker::findConsensusPeakRegions,  
[13](#)

data.table, [51](#)

download\_button, [16](#)

encode\_H3K27ac, [17](#)

enrichPeakOverlap, [34](#)

EpiCompare, [18](#), [54](#)

fig\_length, [22](#)

findConsensusPeakRegions, [12](#)

fragment\_info, [22](#)

gather\_files, [23](#), [24](#)

gather\_files\_names, [24](#)

get\_bpparam, [25](#)

get\_chromHMM\_annotation, [26](#), [38](#)

ggplot, [4](#)

GRanges, [7](#), [9](#), [12](#), [13](#), [19](#), [23](#), [24](#), [26](#), [29](#), [30](#),  
[42–44](#), [48–51](#), [53](#), [54](#)  
GRangesList, [30](#), [49](#), [50](#)  
group\_files, [27](#)  
  
hg19\_blacklist, [19](#), [27](#), [49](#)  
hg38\_blacklist, [19](#), [28](#), [49](#)  
  
is\_granges, [29](#)  
  
layout, [4](#)  
liftover\_grlist, [15](#), [29](#), [40](#), [52](#)  
list, [38](#)  
  
message, [30](#)  
message\_parallel, [31](#)  
messenger, [30](#)  
mm10\_blacklist, [19](#), [31](#), [49](#)  
mm9\_blacklist, [19](#), [32](#), [49](#)  
MulticoreParam, [5](#), [25](#), [45](#), [53](#)  
  
opts\_chunk, [21](#)  
overlap\_heatmap, [32](#)  
overlap\_percent, [33](#)  
overlap\_stat\_plot, [34](#)  
overlap\_upset\_plot, [35](#)  
  
peak\_info, [36](#)  
plot\_ChIPseeker\_annotation, [37](#)  
plot\_chromHMM, [38](#)  
plot\_corr, [20](#), [39](#)  
plot\_enrichment, [41](#)  
plot\_precision\_recall, [20](#), [42](#)  
plotAvgProf, [58](#)  
plotly, [4](#)  
precision\_recall, [44](#), [46](#)  
precision\_recall\_matrix, [45](#)  
predict\_precision\_recall, [46](#)  
predict\_values, [47](#)  
prepare\_blacklist, [47](#)  
prepare\_genome\_builds, [48](#)  
prepare\_peaklist, [49](#)  
prepare\_reference, [50](#)  
  
read\_bowtie, [50](#)  
read\_peaks, [51](#)  
rebin\_peaks, [51](#)  
register, [5](#), [25](#), [45](#), [53](#)  
remove\_nonstandard\_chrom, [53](#)  
report\_command, [54](#)  
  
report\_header, [54](#)  
  
save\_output, [55](#)  
seqlevelsStyle, [30](#)  
SerialParam, [25](#)  
set\_min\_max, [55](#)  
SnowParam, [5](#), [25](#), [45](#), [53](#)  
stop, [56](#)  
stopper, [56](#)  
subsetByOverlaps, [33](#)  
  
tagList, [4](#)  
tidy\_peakfile, [56](#)  
translate\_genome, [57](#)  
tss\_plot, [58](#)  
  
width\_boxplot, [59](#)  
write\_example\_peaks, [60](#)