Package 'peakCombiner'

November 12, 2026

Title The R package to curate and merge enriched genomic regions into consensus peak sets

Version 1.2.0

Description peakCombiner, a fully R based, user-friendly, transparent, and customizable tool that allows even novice R users to create a high-quality consensus peak list. The modularity of its functions allows an easy way to optimize input and output data. A broad range of accepted input data formats can be used to create a consensus peak set that can be exported to a file or used as the starting point for most downstream peak analyses.

Depends R (>= 4.5.0)

License MIT + file LICENSE

LazyData FALSE

biocViews WorkflowStep, Preprocessing, ChipOnChip

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), tidyverse, rmarkdown, styler, cli, lintr, rtracklayer, knitr, devtools, ggplot2, BiocStyle, BiocManager, usethis, utils, AnnotationHub, GenomeInfoDb

Imports tidyr, dplyr (>= 1.1.2), IRanges, GenomicRanges, tidyselect, purrr, readr (>= 2.1.2), tibble (>= 3.2.1), rlang, stringr, here, stats, Seqinfo

URL https://github.com/novartis/peakCombiner/, https://bioconductor.org/packages/peakCombiner

BugReports https://github.com/novartis/peakCombiner/issues

Config/testthat/edition 3

VignetteBuilder knitr

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

git_branch RELEASE_3_23

git_last_commit d72326b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Markus Muckenhuber [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1897-2329>),
Charlotte Soneson [aut] (ORCID:
<https://orcid.org/0000-0003-3833-2169>),
Michael Stadler [aut] (ORCID: <https://orcid.org/0000-0002-2269-4934>),
Kathleen Sprouffske [aut] (ORCID:
<https://orcid.org/0000-0001-7081-2598>),
Novartis Biomedical Research [cph]

Maintainer Markus Muckenhuber <markusmuckenhuber@gmx.at>

Contents

centerExpandRegions 2
checkDataStructure 5
collapseSummits 6
combineRegions 7
crAddSummit 9
crDisjoinFilter 10
crOverlapWithSummits 11
crReduce 12
defineExpansion 13
filterRegions 13
prepareInputRegions 16
syn_blacklist 19
syn_control_rep1_narrowPeak 19
syn_data_bed 20
syn_data_control01 20
syn_data_granges 21
syn_data_tibble 21
syn_data_treatment01 22
syn_sample_sheet 22
syn_treatment_rep1_narrowPeak 23

Index **24**

`centerExpandRegions` *Modifies genomic regions by centering and then expanding them*

Description

centerExpandRegions() is an optional step that re-defines the genomic regions by expanding them from their center. The center information has to be stored in the input data column center, while the information for the expansion can either be user provided or input data derived. The accepted input is a data frame created from prepareInputRegions(). Please see prepareInputRegions() for more details.

Usage

```
centerExpandRegions(
  data,
  centerBy = "center_column",
  expandBy = NULL,
  genome = NA,
  trim_start = TRUE,
  outputFormat = "GenomicRanges",
  showMessages = TRUE
)
```

Arguments

data	PeakCombiner data frame structure with required columns named chrom, start, end, name, score, strand, center, sample_name. Additional columns will be maintained.
centerBy	Allowed values are 'center_column' (default) or 'midpoint'. • 'center_column' uses the value stored in the column center to center. • 'midpoint' replaces the value stored in the column center based on the GenomicRanges::resize() followed by the expansion from based on the user input using GenomicRanges::promoters() to allow symmetric and asymmetic expansion. Note that strand information, if provided is maintained.
expandBy	Allowed values a numeric vector of length 1 or 2, or 'NULL' (default). • The value from the numeric vector of length 1 is expanded in both directions from center to define the genomic region. Thus, the size of the resulting genomic region is 2x the provided value + 1 (for the center coordinate). • The value of the numeric vector of length 2 subtracts the first value from the center and adds the second value to the center to define the genomic region. Thus, the size of the genomic regions is the sum of the first value + the second value + 1 (for the center coordinate). • 'NULL' allows for data-driven definition of the expandBy value. It calculates the median genomic region size of the input data and uses this value like a length 1 numeric vector for expansion.
genome	Character value to define the matching genome reference to the input data. Default value is NA. Allows values are based on GenomicRanges supported genomes like "GRCh38", "GRCh38.p13", "Amel_HAv3.1", "WBcel235", "TAIR10.1", "hg38", "mm10", "rn6", "bosTau9", "canFam3", "musFur1", "galGal6","dm6",

	"ce11", and "sacCer3". Please see also help for `Seqinfo::Seqinfo()` for more details.
`trim_start`	Logical value of TRUE or FALSE (default). If TRUE, and no valid reference genome are provided in `genome`, resulting genomic results with negative starting coordinates will be set to 1.
`outputFormat`	Character value to define format of output object. Accepted values are "GenomicRanges" (default), "tibble" or "data.frame".
`showMessages`	Logical value of TRUE (default) or FALSE. Defines if info messages are displayed or not.

Details

This is an optional function that resizes the genomic regions based on the input peakCombiner standard data frame and the options you select. An expected input data foam contains the following columns with the names: `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Such a data frame is created by the script prepareInputRegions. This step is useful if you want all of your peaks to be the same size for your downstream analyses. In addition, if you want to use the "summit" information, normally obtained by some peak callers (e.g., Macs2), this function allows you to automatically center your regions of interest on these summits. This enables you to capture information about the most important region within a genomic region (e.g., TF-binding site or highest peak) and put that region in the center of your downstream analyses (e.g., applicable to motif-finding or "heatmaps" summarizing multiple genomic regions).

There are two concepts that are relevant for centerExpandRegions: how to define the center, and how much to expand from the center.

How to define the center?:

When you prepared your input regions, it is recommended to use the function prepareInputRegions provided by this package. This pre- populated the `center` column with the absolute genomic coordinate of the center of the peak region. You can either choose to define the center by using pre-defined summit information (e.g., obtained from a peak caller like MACS2) or recompute the arithmetic mean and save that value in the column `center`. (For details see the help for `prepareInputRegions()`).

How much to expand from the center:

You can choose to expand the genomic region from the center either symmetrically or asymmetrically (different lengths before and after the center position).

In the symmetrical case, if you want to choose the size of your genomic region based on the input data, this function can also calculate the median peak size across all of your genomic regions and use that value (`expandBy` = NULL). Alternatively, the user is free to provide a numeric vector to define the expansion. A numeric vector with one value is used to symmetrically expand, while a vector with two values allows to expand asymmetrically.

Value

A tibble with the columns `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. The definitions of these columns are described in full in the prepareInputRegions Details. Use as input for functions `filterRegions()` and `combineRegions()`.

Examples

```
# Load in and prepare a an accepted tibble
utils::data(syn_data_bed)

# Prepare input data
data_prepared <- prepareInputRegions(
  data = syn_data_bed,
  outputFormat = "tibble",
  showMessages = TRUE
)
# Run center and expand
data_center_expand <- centerExpandRegions(
  data = data_prepared,
  centerBy = "center_column",
  expandBy = NULL,
  outputFormat = "tibble",
  showMessages = TRUE
)

data_center_expand

# You can choose to use the midpoint and predefined values to expand

data_center_expand <- centerExpandRegions(
  data = data_prepared,
  centerBy = "midpoint",
  expandBy = c(100, 600),
  outputFormat = "tibble",
  showMessages = FALSE
)

data_center_expand
```

---

checkDataStructure *Control structure of peakCombiner data structure*

Description

This is a general helper function for the package **peakCombiner**. Aim of this function is to check a data frame for the correct column names and classes of each column to ensure to be an accepte inpuut for functions: `centerExpandRegions()`, `filterRegions()` and `combineRegions()`.

Usage

```
checkDataStructure(data, showMessages = TRUE)
```

Arguments

data	A tibble with the columns chrom, start, end, name, score, strand, center, sample_name. Additional columns are tolerated.
showMessages	Logical value of TRUE (default) or FALSE. Defines if info messages are displayed or not.

Value

A tibble with the columns chrom, start, end, name, score, strand, center, sample_name. The definitions of these columns are described in full in the Details below. Use as input for functions centerExpandRegions(), filterRegions() and combineRegions().

---

collapseSummits *collapse_summits*

Description

Helper function for main function prepareInputRegions. Input data is checked for multiple entries of the same genomic region. This can occur when using called peak files as multiple summits can be annotated within the same genomic regions (defined by chrom, start and end). To avoid multiple entries, this script is checking the input for multiple summits within the same regions and maintains only the strongest enriched (based on the values in the column score). This step is mandatory to quantity an optimal result. For details see the details for prepareInputRegions.

Usage

```
collapseSummits(data)
```

Arguments

data	A tibble with the columns chrom, start, end, name, score, strand, center, sample_name.

Value

A tibble with the columns chrom, start, end, name, score, strand, center, sample_name. The definitions of these columns are described in full in the Details below. Use as input for functions centerExpandRegions(), filterRegions() and combineRegions().

`combineRegions` *Combine overlapping genomic regions from different samples to create a single set of consensus genomic regions*

Description

combineRegions is the main function of this package and combines overlapping genomic regions from different samples to create a single set of consensus genomic regions.

The accepted input is the PeakCombiner data frame is created from the function prepareInputRegions and has optionally already been centered and expanded and / or filtered using centerExpandRegions and filterRegions, respectively. Please see prepareInputRegions for more details.

Usage

```
combineRegions(
  data,
  foundInSamples = 2,
  combinedCenter = "nearest",
  removeFlankOverlaps = TRUE,
  annotateWithInputNames = FALSE,
  combinedSampleName = NULL,
  outputFormat = "GenomicRanges",
  showMessages = TRUE
)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be dropped
`foundInSamples`	Only include genomic regions that are found in at least `foundInSamples` **number** of samples. If `foundInSamples` is a fraction between 0 and 1, then only include genomic regions that ar found in at least `foundInSamples` **fraction** of samples. Default value is 2.
`combinedCenter`	Defines how the column 'center' will be populated for each genomic region in the output data. Allowed options are * `middle` - the mathematical center of the new region * `strongest` - the 'center' of the input region that has the the highest 'score' of all overlapping input regions * `nearest` - the 'center' of the input region that is closest to mean of the 'center's of all overlapping input regions (default)
`removeFlankOverlaps`	TRUE (default) / FALSE. If TRUE, the combined regions are checked for an overlap with an input summit. Regions without such an overlap are considered as false positive regions caused by an artificial overlap of neighboring regions due to the expansion step. If FLASE, this step will be skipped.

`annotateWithInputNames`
: TRUE / FALSE (default). If TRUE, a new column named 'input_names' is created in the output data that is populated for each combined genomic region with the 'name's of all contributing input regions. If the column 'input_names' already exists, it will be overwritten.

`combinedSampleName`
: Optionally defines how the column 'sample_name' is populated for the output data. If not used, then the default is to simply concatenate all input sample_names into a single comma-separated string

`outputFormat`
: Character value to define format of output object. Accepted values are "GenomicRanges" (default), "tibble" or "data.frame".

`showMessages`
: Logical value of TRUE (default) or FALSE. Defines if info messages are displayed or not.

Details

combineRegions creates a set of consensus genomic regions by combining overlapping genomic regions from different samples. The general steps within this function are:

- Identify overlapping genomic regions from the input samples
- Retain overlapping genomic regions that are found in at least `foundInSamples` samples. In this way, you can remove rare or sample-specific regions
- Note that overlapping genomic regions must contain at least one 'center' from its input sample regions to be considered a valid genomic region.
- As you can use the output data from this step again (e.g., to center and expand the new set of consensus regions), we must define the 'center', 'score', 'sample_name', and 'name' values for the new genomic regions. We do this as follows:
  - 'center' is defined by the `combinedCenter` parameter, which has three options. * `middle` - the mathematical center of the new region * `strongest` - the 'center' of the input region that has the the highest 'score' of all overlapping input regions * `nearest` - the 'center' of the input region that is closest to mean of the 'center's of all overlapping input regions (default)
  - 'score' is the score of the genomic region from the sample whose 'center's was used, or the mean of the 'score's if `middle` was selected for the `combinedCenter` parameter
  - 'sample_name' can be user defined (`combinedSampleName`) or is a concatenated string of all input 'sample_names' (default).
  - 'name' is created by combining 'sample_name' and row number to create a unique identifier for each newly created genomic region.

Note, the output data.frame columns `sample_name`, `name` and `score` will be updated.

Value

A tibble with the columns `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`, and optionally `input_names`. The definitions of these columns are described in full in the Details below. Use as input for functions centerExpandRegions and filterRegions.

Examples

```
# Load in and prepare a an accepted tibble
utils::data(syn_data_bed)

data_prepared <- prepareInputRegions(
  data = syn_data_bed,
  outputFormat = "tibble",
  showMessages = FALSE
)

# Lets combine the input data by defining all potential option
combineRegions(
  data = data_prepared,
  foundInSamples = 2,
  combinedCenter = "nearest",
  annotateWithInputNames = TRUE,
  combinedSampleName = "consensus",
  outputFormat = "tibble",
  showMessages = TRUE
)
```

`crAddSummit` *Update the center and score information*

Description

Helper function for main function combineRegions. Requires in memory data frame in the standard accepted format for the peakCombiner package. For details see the details for combineRegions.

Usage

```
crAddSummit(
  data,
  input,
  combinedCenter = "nearest",
  annotateWithInputNames = FALSE,
  combinedSampleName = NULL
)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be dropped
`input`	The original input file from `combineRegions` to extract center information

`combinedCenter` Defines how the column 'center' will be populated for each genomic region in the output data. Allowed options are * `middle` - the mathematical center of the new region * `strongest` - the 'center' of the input region that has the the highest 'score' of all overlapping input regions * `nearest` - the 'center' of the input region that is closest to mean of the 'center's of all overlapping input regions (default)

`annotateWithInputNames`
TRUE / FALSE (default). If TRUE, a new column named 'input_names' is created in the output data that is populated for each combined genomic region with the 'name's of all contributing input regions. If the column 'input_names' already exists, it will be overwritten.

`combinedSampleName`
Optionally defines how the column 'sample_name' is populated for the output data. If not used, then the default is to simply concatenate all input sample_names into a single comma-separated string

Details

As you can use the output data from this step again (e.g., to center and expand the new set of consensus regions), we must define the 'center', 'score', 'sample_name', and 'name' values for the new genomic regions. We do this as follows:

- 'center' is defined by the `combinedCenter` parameter, which has three options. * `middle` - the mathematical center of the new region * `strongest` - the 'center' of the input region that has the the highest 'score' of all overlapping input regions * `nearest` - the 'center' of the input region that is closest to mean of the 'center's of all overlapping input regions (default)
- 'score' is the score of the genomic region from the sample whose 'center's was used, or the mean of the 'score's if `middle` was selected for the `combinedCenter` parameter
- 'sample_name' is a concatenated string of all input sample_names

In addition, the output data.frame columns `sample_name`, `name` and `score` will be updated.

Value

A tibble with the following columns: `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`.

`crDisjoinFilter` *Separate genomic regions by coverage (disjoin) and filter input regions*

Description

Helper function for main function combineRegions. Requires in memory data frame in the standard accepted format for the peakCombiner package. For details see the details for combineRegions.

Usage

```
crDisjoinFilter(data, foundInSamples)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be dropped
`foundInSamples`	Only include genomic regions that are found in at least `foundInSamples` **number** of samples. If `foundInSamples` is a fraction between 0 and 1, then only include genomic regions that ar found in at least `foundInSamples` **fraction** of samples. Default value is 2.

Details

Retain overlapping genomic regions that are found in at least `foundInSamples` samples. In this way, you can remove rare or sample-specific regions.

Value

A tibble with the following columns: `chrom`, `start`, `end`, `width`, `strand`, `revmap`, `ranking_comb_ref`, `name`, `rowname_disjoin`.

`crOverlapWithSummits` *Overlap genomic regions with original summits to remove false positive*

Description

Helper function for main function combineRegions. Requires in memory data frame in the standard accepted format for the peakCombiner package. For details see the details for combineRegions.

Usage

```
crOverlapWithSummits(data, input, removeFlankOverlaps = TRUE)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be dropped
`input`	The original input file from `combineRegions` to extract center information
`removeFlankOverlaps`	TRUE (default) / FALSE. If TRUE, the combined regions are checked for an overlap with an input summit. Regions without such an overlap are considered as false positive regions caused by an artificial overlap of neighboring regions due to the expansion step. If FLASE, this step will be skipped.

Details

Overlapping genomic regions must contain at least one 'center' from its input sample regions to be considered a valid genomic region. Regions without overlap might be a consequence of the expansion parameter and are likely to be false positive.

Value

A tibble with the following columns: `chrom`, `start`, `end`, `width`, `strand`, `name`.

`crReduce` *Recombine (reduce) input regions*

Description

Helper function for main function combineRegions. Requires in memory data frame in the standard accepted format for the peakCombiner package. For details see the details for combineRegions.

Usage

```
crReduce(data)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be dropped

Details

Recombine filtered genomic regions from disjoin function to create the consensus regions.

Value

A tibble with the following columns: `chrom`, `start`, `end`, `width`, `strand`, `name`.

defineExpansion *Calculate expansion value from median input region size*

Description

Calculates the parameter expandBy when it was set to 'NULL' in the main function. 'NULL' allows for data-driven definition of the expandBy value. It calculates the median genomic region size of the input data and uses this value like a length 1 numeric vector for expansion.

Usage

```
defineExpansion(data = data, expandBy = expandBy)
```

Arguments

data	PeakCombiner data frame structure with required columns named chrom, start, end, name, score, strand, center, sample_name. Additional columns will be maintained.
expandBy	Allowed values a numeric vector of length 1 or 2, or 'NULL' (default). • The value from the numeric vector of length 1 is expanded in both directions from center to define the genomic region. Thus, the size of the resulting genomic region is 2x the provided value + 1 (for the center coordinate). • The value of the numeric vector of length 2 subtracts the first value from the center and adds the second value to the center to define the genomic region. Thus, the size of the genomic regions is the sum of the first value + the second value + 1 (for the center coordinate). • 'NULL' allows for data-driven definition of the expandBy value. It calculates the median genomic region size of the input data and uses this value like a length 1 numeric vector for expansion.

Value

A vector of length 1 to define region expansion.

filterRegions *Apply user-defined filtering options to genomic regions.*

Description

filterRegions is an optional step that allows inclusion or exclusion of genomic regions based on 4 different criteria:

- Include regions by their chromosome names (optional).
- Exclude blacklisted regions (optional).

- Include regions above a given score (optional).
- Include top n regions per sample, ranked from highest to lowest score (optional).

The accepted input is the PeakCombiner data frame is created from the function prepareInputRegions. Please see prepareInputRegions for more details.

The filterRegions can be used multiple times on the same data set, which allows a user to step-wise optimize selection criteria of regions of interest.

Usage

```
filterRegions(
  data,
  includeByChromosomeName = NULL,
  excludeByBlacklist = NULL,
  includeAboveScoreCutoff = NULL,
  includeTopNScoring = NULL,
  outputFormat = "GenomicRanges",
  showMessages = TRUE
)
```

Arguments

`data`	PeakCombiner data frame structure with required columns named `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. Additional columns will be maintained.
`includeByChromosomeName`	• 'NULL' (default) - No chromosome name filtering will be done. * Character vector that contains chromosomes names to be retained.
`excludeByBlacklist`	• 'NULL' (default) - No blacklist filtering will be done. * GenomicRanges object (default setup) or data frame/tibble with columns `chrom`, `start`, and `end`.
`includeAboveScoreCutoff`	• 'NULL' (default) - No score filtering will be done. * Single numeric value that defines the `score` threshold above which all genomic regions will be retained. This results in variable number of sites per sample.
`includeTopNScoring`	• 'NULL' (default) - No score filtering will be done. * Single numeric value representing the number of genomic regions per sample to be retained. The genomic regions are selected from highest to lowest score, and if includeTopNScoring > number of regions, then no filtering is done.
`outputFormat`	Character value to define format of output object. Accepted values are "GenomicRanges" (default), "tibble" or "data.frame".
`showMessages`	Logical value of TRUE (default) or FALSE. Defines if info messages are displayed or not.

Details

This is an optional step which enables commonly-needed filtering steps to focus in on the key genomic regions of interest. This can be useful when there are many genomic regions identified in your peak-caller or input BED files.

filterRegions can be used multiple times on the same data set, allowing a user to select regions of interest using a step-wise optimization approach.

- `includeByChromosomeName` - Retains only chromosomes that are in the provided vector. By not including mitochondrial, sex, or non-classical chromosomes, genomic regions found on these chromosomes can be removed. If set to 'NULL' (default), this step will be skipped (optional).
- `excludeByBlacklist` - A GenomicRanges file, dataframe or tibble can be provided listing the genomic regions to remove (having `chrom` (`seqnames` for GenomicRanges) , `start`, and `end` column names). If set to 'NULL' (default), this step will be skipped (optional). Please note that if there are not matching entries in the 'chrom' columns of input and blacklist, an information message is displayed. This can happend and does not cause any problems with the script.
- `includeAboveScoreCutoff` - Single numeric value that defines the `score` threshold above which all genomic regions will be retained. The `score` column in the peakCombiner input data should be non-zero for this parameter to be used. It is populated by prepareInputRegions, and by default takes the value of -log10(FDR) if possible (e.g., using a .narrowPeak file from MACS2 as input). Importantly, applying this filter retains a variable number of genomic regions per sample, all having a score greater than the `includeAboveScoreCutoff` parameter. If set to 'NULL' (default), this step will be skipped (optional).
- `includeTopNScoring` - Single numeric value that defines how many of the top scoring genomic regions (using the column `score`) are retained. All other genomic regions are discarded. Importantly, applying this filter retains `includeTopNScoring` regions per sample, which means that the minimum enrichment levels may vary between samples. Note that if multiple genomic regions have the same `score` cutoff value, then all of those genomic regions are included. In this case, the number of resulting regions retained may be a bit higher than the input parameter. If set to 'NULL' (default), this step will be skipped (optional).

Value

A tibble with the columns `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. The definitions of these columns are described in full in the prepareInputRegions Details. Use as input for functions centerExpandRegions and combineRegions.

Examples

```
# Load in and prepare a an accepted tibble
utils::data(syn_data_bed)

data_prepared <- prepareInputRegions(
  data = syn_data_bed,
  outputFormat = "tibble",
  showMessages = TRUE
)
```

```
# Here use options for all four filtering methods.

filterRegions(
  data = data_prepared,
  includeByChromosomeName = c("chr1", "chr2", "chr4"),
  excludeByBlacklist = NULL,
  includeAboveScoreCutoff = 10,
  includeTopNScoring = 100,
  outputFormat = "tibble",
  showMessages = TRUE
)
```

`prepareInputRegions` *Prepare input data for peakCombiner package*

Description

prepareInputRegions prepares the input data in the format needed for all of the following steps within peakCombiner. It accepts the following formats:

- in memory data frame listing each sample's peak file location,
- in memory data frame listing the peaks themselves that are found in each sample, or
- in memory GRanges object listing the peaks themselves that are found in each sample.

Usage

```
prepareInputRegions(
  data,
  outputFormat = "GenomicRanges",
  genome = NA,
  startsAreBased = 1,
  showMessages = TRUE
)
```

Arguments

`data`	Data frame or GRanges object with the input data. Several formats are accepted, which are described in full in the Details below. • in memory data frame listing each sample's peak file location, • in memory data frame listing the peaks themselves that are found in each sample, or • in memory GRanges object listing the peaks themselves that are found in each sample.
`outputFormat`	Character value to define format of output object. Accepted values are "GenomicRanges" (default), "tibble" or "data.frame".

genome	Character value to define the matching genome reference to the input data. Default value is NA. Allows values are based on GenomicRanges supported genomes like "GRCh38", "GRCh38.p13", "Amel_HAv3.1", "WBcel235", "TAIR10.1", "hg38", "mm10", "rn6", "bosTau9", "canFam3", "musFur1", "galGal6","dm6", "ce11", and "sacCer3". Please see also help for `Seqinfo::seqinfo()` for more details.
startsAreBased	Either 0, 1 (Default), or NA. Define if the provided input data is 0 or 1-based. Only, if paramter is NA then GenomicRanges object, tibbles and dataframes are considered 1-based, while data loaded from a sample_sheet is considered 0-based (expected to load a BED file).
showMessages	Logical value of TRUE (default) or FALSE. Defines if info messages are displayed or not.

Details

Accepted inputs are one of the three following options:

1. In memory data frame listing each sample's peak file location
   - `sample_name` - Unique name for each sample (required).
   - `file_path` - Path to the file in which the genomic regions are stored. For example, the path to a bed file or `.narrowPeak` file (required).
   - `file_format` - The expected file format. Needed to correctly label the columns of the input. Acceptable values are: `bed`, `narrowPeak`, and `broadPeak` (required).
   - `score_colname` - Either column name or number of the column having the the metric used to rank peak importance, where bigger values are more important. Entries have to be identical, mutliple entries are not supported. If not provided, column 9 will be used for `.narrowPeak` or `.broadPeak` file formats. Column 9 corresponds to the `qValue` as described in the UCSC documentation here. Other alternatives for `narrowPeak` or `broadPeak` could be columns 7 or 8, which correspond to `signalValue` or `pValue` (optional).
2. In memory data frame listing the peaks themselves that are found in each sample. The columns can be provided in any order and have the following names. Note that additional columns will be dropped.
   - `chrom` - chromosome name (required).
   - `start` - start coordinate of range (1-based coordinate system, NOT like bed files which are 0-based) (required).
   - `end` - end coordinate of range (required).
   - `sample_name` - unique identifier for a sample. No restrictions on characters (required).
   - `score` - the metric used to rank peak importance, where bigger values are more important. For example, qValue from Macs2, -log10FDR from another method, or fold enrichment over background computed from your favorite method. If not provided, defaults to 0 (optional).
   - `strand` - values are '+', '-', or '.'. If not provided, defaults to '.' (optional).
   - `summit` - distance of the strongest signal ("summit") of the peak region from the start coordinate (optional).

3. In memory GRanges object listing the peaks themselves that are found in each sample. This object is very similar to the data frame above, except that `chrom`, `start`, and `end` are instead described using the `GRanges` nomenclature. Note that additional columns will be dropped.

This function parses the inputs provided and returns a data frame having the columns listed below.

- `chrom` - chromosome name
- `start` - start coordinate of range (1-based coordinate system, NOT like bed files which are 0-based)
- `end` - end coordinate of range
- `name` - unique identifier for a region. auto-generated by this function
- `score` - the metric used to rank peak importance, where bigger values are more important. For example, qValue from MACS2, -log10FDR from another method, or fold enrichment over background computed from your favorite method
- `strand` - values are '+', '-', or '.'. Chromatin data are typically non- stranded so will have a '.'.
- `center` - absolute genomic coordinate of the nucleotide at the center of the peak region, or alternatively the strongest signal ("summit") of the peak region. If no value is provided by the user, `center` defaults to the arithmetic center of the peak region.
- `sample_name` - unique identifier for a sample. No restrictions on characters

In addition, input data is checked for mutliple entries of the same genomic region. This can occure when using called peak files as multiple summits can be annotated within the sqme genomic regions (defined by `chrom`, `start` and `end`). To avoid mutliple entries, this script is checking the input for multiple summits within the same regions and maintains only the strongest enriched (based on the values in the column `score`). This step is mandatory to guaranty an optimal result.

An additional option is to provide already here a genome (details see below) and maintain this information for the function `centerExpandRegions()`.

Value

A tibble with the columns `chrom`, `start`, `end`, `name`, `score`, `strand`, `center`, `sample_name`. The definitions of these columns are described in full in the Details below. Use as input for functions `centerExpandRegions()`, `filterRegions()` and `combineRegions()`.

Examples

```
# Load in and prepare a an accepted tibble
utils::data(syn_data_tibble)

data_prepared <- prepareInputRegions(
  data = syn_data_tibble,
  outputFormat = "tibble",
  showMessages = TRUE
)
data_prepared

# Or a pre-loaded tibble with genomic regions and named columns.
```

```
utils::data(syn_data_control01)
utils::data(syn_data_treatment01)

combined_input <- syn_data_control01 |>
  dplyr::mutate(sample_name = "control-rep1") |>
  rbind(syn_data_treatment01 |>
    dplyr::mutate(sample_name = "treatment-rep1"))

prepareInputRegions(
  data = combined_input,
  outputFormat = "tibble",
  showMessages = FALSE
)
```

`syn_blacklist` *Synthetic file with blacklisted regions for peakCombiner*

Description

Synthetic example blacklisted regions file as tibble with columns "chrom", "start", and "end".

Usage

```
data(syn_blacklist)
```

Format

`syn_blacklist` A tibble with 2 rows and 3 columns:

Source

Created for R package peakCombiner.

`syn_control_rep1_narrowPeak`
Synthetic data set for control rep 1 sample in narrowPeak file format

---

Description

Synthetic example data set as minimal required input file with columns "chrom", "start", "end", "name", "score", "strand", "signalValue", "pValue", "qValue" and "peak".

Usage

```
data(syn_control_rep1_narrowPeak)
```

Format

syn_control_rep1_narrowPeak A tibble with 11 rows and 6 columns:

Source

Created for R package peakCombiner.

---

syn_data_bed *Synthetic data set of genomic coordinates and meta data columns*

Description

Synthetic example data set as minimal required input file with columns "chrom", "start", "end", and "sample_name".

Usage

```
data(syn_data_bed)
```

Format

syn_data_bed A tibble with 55 rows and 4 columns:

Source

Created for R package peakCombiner.

---

syn_data_control01 *Synthetic data set of genomic coordinates and meta data columns filtered for control rep 1 sample*

Description

Synthetic example data set as minimal required input file with columns "chrom", "start", "end", "score", "strand", and "center".

Usage

```
data(syn_data_control01)
```

Format

syn_data_control01 A tibble with 11 rows and 6 columns:

Source

Created for R package peakCombiner.

syn_data_granges *Synthetic data set of genomic coordinates and meta data columns*

Description

Synthetic example data set from GenomicRanges object with columns "seqnames", "start", "end", "width", "strand", "score", "center", and "sample_name".

Usage

```
data(syn_data_granges)
```

Format

`syn_data_granges` A data frame with 55 rows and 8 columns:

Source

Created for R package peakCombiner.

syn_data_tibble *Synthetic data set of genomic coordinates and meta data columns as tibble*

Description

Synthetic example data set as tibble with columns "chrom", "start", "end", "name", "score", "strand", "center", and "sample_name".

Usage

```
data(syn_data_tibble)
```

Format

`syn_data_tibble` A tibble with 55 rows and 8 columns:

Source

Created for R package peakCombiner.

syn_data_treatment01 *Synthetic data set of genomic coordinates and meta data columns filtered for treatment rep 1 sample*

Description

Synthetic example data set as minimal required input file with columns "chrom", "start", "end", "score", "strand", and "center".

Usage

```
data(syn_data_treatment01)
```

Format

`syn_data_treatment01` A tibble with 10 rows and 6 columns:

Source

Created for R package peakCombiner.

syn_sample_sheet *Synthetic sample sheet to load example data with peakCombiner*

Description

Synthetic example sample sheet as tibble with columns "sample_name", "file_path", "file_format", and "score_colname".

Usage

```
data(syn_sample_sheet)
```

Format

`syn_sample_sheet` A tibble with 6 rows and 4 columns.

Source

Created for R package peakCombiner.

`syn_treatment_rep1_narrowPeak` *Synthetic data set for treatment rep 1 sample in narrowPeak file format*

Description

Synthetic example data set as minimal required input file with columns "chrom", "start", "end", "name", "score", "strand", "signalValue", "pValue", "qValue" and "peak".

Usage

```
data(syn_treatment_rep1_narrowPeak)
```

Format

`syn_treatment_rep1_narrowPeak` A tibble with 11 rows and 6 columns:

Source

Created for R package peakCombiner.

Index

∗ **datasets**
  syn_blacklist, 19
  syn_control_rep1_narrowPeak, 19
  syn_data_bed, 20
  syn_data_control01, 20
  syn_data_granges, 21
  syn_data_tibble, 21
  syn_data_treatment01, 22
  syn_sample_sheet, 22
  syn_treatment_rep1_narrowPeak, 23

centerExpandRegions, 2, *4*, *7, 8*, *15*
centerExpandRegions(), *3*, *5, 6*, *18*
checkDataStructure, 5
collapseSummits, 6
combineRegions, *7*, 7, *8–12*, *15*
combineRegions(), *4–6*, *18*
crAddSummit, 9
crDisjoinFilter, 10
crOverlapWithSummits, 11
crReduce, 12

defineExpansion, 13

filterRegions, *7, 8*, *13*, 13, *14, 15*
filterRegions(), *4–6*, *18*

GenomicRanges::promoters(), *3*
GenomicRanges::resize(), *3*

prepareInputRegions, *4*, *6, 7*, *14–16*, 16
prepareInputRegions(), *3, 4*

Seqinfo::Seqinfo(), *4*
Seqinfo::seqinfo(), *17*
syn_blacklist, 19
syn_control_rep1_narrowPeak, 19
syn_data_bed, 20
syn_data_control01, 20
syn_data_granges, 21
syn_data_tibble, 21
syn_data_treatment01, 22
syn_sample_sheet, 22
syn_treatment_rep1_narrowPeak, 23