Package 'genomecompR'

July 10, 2024

Title Division of the genome into functional compartments

Version 1.0.4

Description
genomecompR creates genomic compartments using ChIP-seq and ATAC-seq data as follows:
1) Active promoter: K27ac in TSS-/+1Kb.
2) Transcription initiation: TSS-1/+1kb overlapping with Ser5P peaks.
3) Transcription elongation: TSS+1kb to TES overlapping Ser2P.
4) Transcription termination: TES+50bp
5) Bivalent promoters: H3K4me3/H3K27me3 overlap TSS-/+1Kb.
6) Active enhancer: H3K27ac/H3K4me1/ATAC-seq. They should not overlap with the combination of UCSC refGene, NCBI RefSeq, and GENCODE VM25.
7) Poised enhancer: H3K27me3/H3K4me1/PRC2. They should not overlap with the combination of UCSC refGene, NCBI RefSeq, and GENCODE VM25.
8) Polycomb domain: Suz12 and RING1B overlap.
9) Heterochromatin: H3K9me3
10) SINE: Overlap with SINE annotations of repeat maskers.
11) LINE: Overlap with LINE annotations of repeat maskers.
12) LTR: Overlap with LTR annotations of repeat maskers.
genomecompR also provides several plotting functions.
WARNING: This R package is provided to describe the method section of an accompanying article. It does not aim to be extended in terms of features. Please respect the packages version described below for full functionality.

Depends R (== 4.2.0), tools (== 4.2.0), GenomicRanges (== 1.50.2), tibble (== 3.1.6), ggplot2 (== 3.4.1), ggupset (== 0.3.0), IRanges (== 2.32.0), parallel (== 4.2.0), S4Vectors (== 0.36.2), GenomeInfoDb (== 1.34.9), rtracklayer (== 1.58.0), ComplexUpset (== 1.3.3), chipenrich (== 2.22.0), biomaRt (== 2.54.1)

License GPL-3

Encoding UTF-8

LazyData true

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Imports tools,
GenomicRanges,
tibble,
ggplot2,
ggupset,

IRanges,
parallel,
S4Vectors,
GenomeInfoDb,
rtracklayer,
ComplexUpset,
chipenrich,
biomaRt

Collate 'AllClasses.R'
'AllGenerics.R'
'constructor.R'
'getters.R'
'methods-filteringAnno.R'
'methods-output.R'
'methods-overlap.R'
'methods-plot.R'
'setters.R'
'utils.R'
'biomart.R'

RoxygenNote 7.1.2

R topics documented:

`annoCuration,genomicCompartments-method`
Curate Gene Annotations

Description

This method curates gene annotations by performing several filtering steps on the provided annotation data. It filters out non-canonical chromosomes, retains known coding and non-coding genes, keeps genes with a length greater than 2kb, removes duplicate annotations with the same coordinates, extends the transcription start site (TSS) by 1kb upstream, and removes overlapping annotations. The curated annotations are then set as the curated annotation data of the 'genomicCompartments' object.

Usage

```
annoCuration(theobject, geneannovec, nbcpu = 8)
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`geneannovec`	A character vector containing the paths to gene annotation files.
`nbcpu`	Number of CPU cores to use for parallel processing (default is 8).

Value

Returns the updated 'genomicCompartments' object with curated gene annotations.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Perform annotation curation
gc_obj <- annoCuration(gc_obj, geneannovec)

## End(Not run)
```

`bivalentPromoters,genomicCompartments-method`
Identify Bivalent Promoters

Description

This method identifies bivalent promoters by creating a merged GRanges object from peaks of H3K4me3 and H3K27me3 histone marks at TSS-/+1Kb.

Usage

```
bivalentPromoters(theobject, peakspathvec)
```

Arguments

theobject	An object of class 'genomicCompartments'.
peakspathvec	A named vector specifying paths to the peak files of H3K4me3 and H3K27me3.

Value

Returns the modified 'genomicCompartments' object with bivalent promoters assigned.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Identify bivalent promoters
bivalentPromoters(gc_obj, peakspathvec = c(H3K4me3 = "path/to/H3K4me3.gff",
H3K27me3 = "path/to/H3K27me3.gff"))

## End(Not run)
```

---

boxplotGlcnacLevels,genomicCompartments-method

Generate Boxplot of Glcnac Levels

Description

This method generates boxplots of Glcnac levels across the different genomic compartments stored in the given genomicCompartments object.

Usage

```
boxplotGlcnacLevels(theobject, bwpath, outputfolder, includerepeats,
           plotviolin = FALSE)
```

Arguments

theobject	An object of class 'genomicCompartments'.
bwpath	Character string specifying the path to the BigWig file containing Glcnac levels.
outputfolder	Character string specifying the output folder path for saving results.
includerepeats	Logical indicating whether to include repeats in the analysis.
plotviolin	Logical indicating whether to plot to current device (default is FALSE).

Value

Save the boxplot to outputfolder.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Generate boxplots of Glcnac levels
gc_obj <- boxplotGlcnacLevels(gc_obj, bwpath = "path/to/glcnac_levels.bw",
                              outputfolder = "path/to/output_folder",
                              includerepeats = FALSE, plotviolin = FALSE)

## End(Not run)
```

`buildGR` *Build GRanges Object from File*

Description

This function reads a file containing genomic coordinates and constructs a 'GRanges' object.

Usage

```
buildGR(currentpath)
```

Arguments

`currentpath`	A character string representing the GFF file path to read. The file should contain tabular data with the following columns: • `V1` - Chromosome names • `V4` - Start positions • `V5` - End positions • `V9` - Feature names • `V7` - Strand information

Value

A 'GRanges' object with the specified genomic coordinates and feature names.

Examples

```
## Not run:
# Example file path
path <- "path/to/genomic/coordinates/file.gff"
# Build GRanges object
gr <- buildGR(path)

## End(Not run)
```

buildGRNoReading *Build GRanges Object from Data Frame*

Description

This function constructs a 'GRanges' object from a data frame.

Usage

```
buildGRNoReading(fi)
```

Arguments

fi	A data frame with columns representing genomic coordinates and other relevant information in GFF format. The expected columns are: • V1 - Chromosome names • V4 - Start positions • V5 - End positions • V9 - Feature names • V7 - Strand information

Value

A 'GRanges' object with the specified genomic coordinates and feature names.

Examples

```
## Not run:
# Example data frame
df <- data.frame(V1 = "chr1", V4 = 1000, V5 = 2000, V9 = "gene1", V7 = "+")
# Build GRanges object
gr <- buildGRNoReading(df)

## End(Not run)
```

buildIntervalsObject *Build Genomic Intervals Object*

Description

This function constructs a 'genomeCompart' object, defines various genomic features, and optionally includes repeat annotations.

Usage

```
buildIntervalsObject(peakspathvec, geneannos, includerepeats)
```

Arguments

peakspathvec	A named vector of file paths to peak files in GFF format for various ChIP-seq and ATAC-seq data. Expected names include: "H3K27ac", "Ser5P", "Ser2P", "H3K4me3", "H3K27me3", "ATACSeq", "Suz12", "RING1B", "H3K9me3".
geneannos	A list of gene annotation files to be used for defining non-redundant gene sets and other features.
includerepeats	A logical value indicating whether to include repeat annotations (LINE, LTR, SINE).

Details

This function performs the following steps:

- Builds the initial ‘genomeCompart‘ object.
- Defines a non-redundant set of gencode genes by removing overlapping genes, genes closer than 1kb to each other, and genes shorter than 2kb.
- Identifies active promoters using H3K27ac at TSS±1Kb.
- Identifies promoters with Ser5P (initiation) at TSS±1Kb.
- Identifies elongating genes by overlapping Ser2P from TSS+1kb to TES.
- Defines termination sites at TES+1kb.
- Identifies bivalent promoters using H3K4me3 and H3K27me3 at TSS±1kb.
- Defines active enhancers using H3K27ac, H3K4me1, and ATAC-seq, ensuring no overlap with gene annotations.
- Defines poised enhancers using H3K27me3, H3K4me1, and Suz12, ensuring no overlap with gene annotations.
- Defines polycomb domains using Suz12 and RING1B overlaps.
- Defines heterochromatin using H3K9me3.
- Optionally includes repeat annotations (SINE, LINE, LTR) if ‘includerepeats‘ is TRUE.

Value

An updated ‘genomeCompart‘ object with defined genomic features and, optionally, repeat annotations.

Examples

```
## Not run:
peakspathvec <- c(H3K27ac = "path/to/H3K27ac.gff",
     Ser5P = "path/to/Ser5P.gff", Ser2P = "path/to/Ser2P.gff",
     H3K4me3 = "path/to/H3K4me3.gff", H3K27me3 = "path/to/H3K27me3.gff",
     ATACSeq = "path/to/ATACSeq.gff", Suz12 = "path/to/Suz12.gff",
     RING1B = "path/to/RING1B.gff", H3K9me3 = "path/to/H3K9me3.gff")
geneannos <- list("path/to/gencode.gff", "path/to/refGene.gff",
"path/to/refseq.gff")
includerepeats <- FALSE
gc <- buildIntervalsObject(peakspathvec, geneannos, includerepeats)

## End(Not run)
```

checkParams *Validate Parameters given as input to different functions of the package*

Description

This function validates various parameters required by several functions to ensure they meet the expected criteria.

Usage

```
checkParams(peakspathqueryvec, glcnacbwvec, querynamevec,
       geneannovec, peakspathcategoriesvec, repeatsannovec, countstable,
       includerepeats, countsannotype)
```

Arguments

peakspathqueryvec	A vector of file paths to peak files in GFF format.
glcnacbwvec	A vector of file paths to glc bigwig files.
querynamevec	A vector of names corresponding to the peak files.
geneannovec	A vector of file paths to gene annotation files, expected to be in the order of gencode, refGene, and refseq.
peakspathcategoriesvec	A vector of names for each peak file, expected to contain strings identifying categories such as H3K27ac, H3K4me1, etc.
repeatsannovec	A vector of file paths to repeat annotation files in GFF format. The file names are expected to contain the substrings LINE, LTR, and SINE.
countstable	A vector containing the path to a count table file.
includerepeats	A logical value indicating whether to include repeat annotations.
countsannotype	A string specifying the annotation type for the counts table, either "ensembl" or "entrez".

Details

This function performs the following validation checks:

- Ensures there are no more than two files in peakspathqueryvec.
- Checks that the number of peak files matches the number of bigwig files.
- Verifies that the number of query names matches the number of peak files.
- Validates that gene annotation files are ordered as gencode, refGene, and refseq.
- Confirms that category peak files contain the expected strings for categories like H3K27ac, H3K4me1, etc.
- If repeat annotations are included, checks that the repeat files contain the expected substrings LINE, LTR, and SINE.
- Ensures only one counts table is provided.
- Validates that the counts table annotation type is either "ensembl" or "entrez".

Value

None. The function stops with an error message if any validation check fails.

Examples

```
## Not run:
peakspathqueryvec <- c("path/to/peak1.gff", "path/to/peak2.gff")
glcnacbwvec <- c("path/to/bw1", "path/to/bw2")
querynamevec <- c("query1", "query2")
geneannovec <- c("gencode", "refGene", "refseq")
peakspathcategoriesvec <- c("H3K27ac_path", "H3K4me1_path", "H3K27me3_path",
"H3K4me3_path", "Suz12_path", "RING1B_path", "H3K9me3_path", "Ser5P_path",
"Ser2P_path", "ATAC_path")
repeatsannovec <- c("LINE_path", "LTR_path", "SINE_path")
countstable <- c("path/to/counts_table")
includerepeats <- TRUE
countsannotype <- "ensembl"
checkParams(peakspathqueryvec, glcnacbwvec, querynamevec, geneannovec,
peakspathcategoriesvec, repeatsannovec, countstable, includerepeats,
countsannotype)

## End(Not run)
```

---

complexUpsetDiagram,list-method

Generate Complex Upset Diagram

Description

This method generates a plot showing the upset diagram of the overlaps of GlcNAc peaks with different genomic compartments and the corresponding average binding values for each peak.

Usage

```
complexUpsetDiagram(theobject, includerepeats, outfold)
```

Arguments

`theobject`	A list of objects of class ‘genomicCompartments‘.
`includerepeats`	Logical indicating whether to include repeats in the analysis.
`outfold`	Character string specifying the output folder path for saving the upset diagram and boxplot.

Value

No return value. The function generates and saves a complex upset plot in the specified output folder.

Examples

```
## Not run:
# Create a list of genomicCompartments objects
gc_list <- list(gc_obj1, gc_obj2, gc_obj3)
# Generate complex upset diagram
complexUpsetDiagram(gc_list, includerepeats = FALSE,
 outfold = "path/to/output_folder")

## End(Not run)
```

---

enhancer,genomicCompartments-method

Identify Enhancers

Description

This method identifies active or poisedd enhancers based on the overlap of histone marks and ATAC-seq peaks.

Usage

```
enhancer(theobject, peakspath1, peakspath2, peakspath3, label1, label2,
label3, statelabel)
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`peakspath1`	Path to the peaks file of the first histone mark.
`peakspath2`	Path to the peaks file of the second histone mark.
`peakspath3`	Path to the peaks file of the third histone mark or ATAC-seq.
`label1`	Label for the first histone mark.
`label2`	Label for the second histone mark.
`label3`	Label for the third histone mark or ATAC-seq.
`statelabel`	Label indicating the state of enhancers ('active' or 'poised').

Details

Active enhancers are defined as the overlap of H3K27ac/H3K4me1/ATAC-seq. Poised enhancers are defined as the overlap of H3K27me3/H3K4me1/PRC2. They should not overlap with the combination of UCSC refGene, NCBI RefSeq, and GENCODE VM25.

Value

Returns the modified 'genomicCompartments' object with enhancers assigned.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Identify active enhancers
enhancer(gc_obj, peakspath1 = "path/to/H3K27ac.gff",
 peakspath2 = "path/to/H3K4me1.gff",
 peakspath3 = "path/to/ATAC-seq.gff",
label1 = "H3K27ac", label2 = "H3K4me1", label3 = "ATAC-seq",
statelabel = "active")

## End(Not run)
```

---

extractCompCoordWithPeak,genomicCompartments-method

Extract Compartment Coordinates with Peaks

Description

This method performs overlap analysis between reference peaks and each compartment defined in the ‘genomicCompartments‘ object. It extracts compartments containing peaks and outputs their coordinates in GFF format.

Usage

```
extractCompCoordWithPeak(theobject, outfold, includerep)
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
outfold	Path to the folder where output files will be written.
includerep	Logical indicating whether to include repeat regions in the analysis.

Value

Returns the modified ‘genomicCompartments‘ object with compartments containing peaks.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Extract compartment coordinates with peaks
extractCompCoordWithPeak(gc_obj, outfold = "output_folder",
 includerep = TRUE)

## End(Not run)
```

`filterChromAndStrand` *Filter Chromosomes and Strand Information in GRanges*

Description

This function filters out unwanted chromosomes and standardizes strand information in a ‘GRanges‘ object.

Usage

```
filterChromAndStrand(currentgr)
```

Arguments

`currentgr`	A ‘GRanges‘ object containing genomic ranges to be filtered.

Details

This function performs the following operations:

- If the strand information contains neither "+" and "-" strands, all strand information is set to "+".
- Chromosomes with names containing "Un", "random", or "chrM" are removed.
- The sequence levels of the resulting ‘GRanges‘ object are updated to include only the remaining chromosomes.

Value

A filtered ‘GRanges‘ object with unwanted chromosomes removed and strand information standardized.

Examples

```
## Not run:
# Example GRanges object
gr <- GRanges(seqnames = c("chr1", "chr2", "chrUn", "chrM"),
              ranges = IRanges(start = c(1, 2, 3, 4), end = c(5, 6, 7, 8)),
              strand = c("+", "-", "+", "*"))
# Filter the GRanges object
filtered_gr <- filterChromAndStrand(gr)

## End(Not run)
```

genomeCompart *Create Genomic Compartments Object*

Description

This function is the constructor of the genomicCompartments object. It is called by the function buildIntervalsObject.

Usage

```
genomeCompart(peakspathvec, geneannovec)
```

Arguments

peakspathvec	Character vector containing the path to the peaks file.
geneannovec	Character vector containing the paths to gene annotation files.

Details

peakspathvec is made of the GFF file containing glc peaks and of the GFF files containing the ChIP-seq and ATAC-seq peak coordinates. The geneannovec contains path to the gencode, refgene, and refseq gff files.

Value

Returns an object of class 'genomicCompartments' containing the processed peaks and gene annotations.

See Also

buildIntervalsObject

Examples

```
## Not run:
peakspathquery <- "/path/to/glc.gff"
peakspathcategoriesvec <- c(H3K27ac = "/path/to/H3K27ac.gff",
       H3K4me1 = "/path/to/H3K4me1.gff", H3K27me3 = "/path/to/H3K27me3.gff",
       H3K4me3 = "/path/to/H3K4me3.gff", Suz12 = NA, RING1B = NA,
       H3K9me3 = "/path/to/H3K9me3.gff", Ser5P = "/path/to/Ser5P.gff",
       Ser2P = "/path/to/Ser2P.gff", ATACSeq = "/path/to/ATACseq.gff")
peakspathvec <- c(peakspathquery, peakspathcategoriesvec)
geneannovec <- c(gencode = "/path/to/gencode.gff",
       refgene = "/path/to/refGeneUCSC.gff",
       refseq = "/path/to/refseq.gff")

genomic_compartments <- genomeCompart(peakspathvec, geneannovec)

## End(Not run)
```

genomicCompartments-class
Genomic Compartments Class

Description

The 'genomicCompartments' class is designed to store various genomic data, including reference peaks, gene annotations, and different types of genomic regions such as promoters, enhancers, and chromatin domains.

Slots

`referencePeaks` A 'GRanges' object containing reference peaks (required).

`geneAnnotations` A 'GRangesList' object containing gene annotations (required).

`curatedAnno` A 'GRanges' object for curated annotations.

`activeProm` A 'GRanges' object for active promoters defined as K27ac in TSS-/+1Kb.

`activeEnh` A 'GRanges' object for active enhancers defined as H3K27ac/H3K4me1/ATAC-seq. They should not overlap with the combination of UCSC refGene, NCBI RefSeq, and GENCODE VM25.

`poisedEnh` A 'GRanges' object for poised enhancers defined as H3K27me3/H3K4me1/PRC2. They should not overlap with the combination of UCSC refGene, NCBI RefSeq, and GENCODE VM25. Note: use a non redundant set of genes i.e. no overlap, remove genes closer to 1kb from one another, remove genes that are too short i.e. length < 2kb.

`PcGDomain` A 'GRanges' object for PcG domains defined as Suz12 and RING1B overlap.

`heteroChrom` A 'GRanges' object for heterochromatin regions (H3K9me3)

`bivalentProm` A 'GRanges' object for bivalent promoters defined as the overlap of H3K4me3/H3K27me3.

`initiation` A 'GRanges' object for initiation regions defined as TSS-1/+1kb overlapping with Ser5P peaks.

`elongation` A 'GRanges' object for elongation regions defined as TSS+1kb to TES overlapping Ser2P.

`termination` A 'GRanges' object for termination regions defined as TES+1kb

`SINE` A 'GRanges' object for SINE elements.

`LINE` A 'GRanges' object for LINE elements.

`LTR` A 'GRanges' object for LTR elements.

`regionsMatrix` A matrix representing various regions.

`glcPeakVals` A list of glc peaks.

`activePromWithPeaks` A 'GRanges' object for active promoters with glc peaks.

`activeEnhWithPeaks` A 'GRanges' object for active enhancers with glc peaks.

`poisedEnhWithPeaks` A 'GRanges' object for poised enhancers with glc peaks.

`PcGDomainWithPeaks` A 'GRanges' object for PcG domains with glc peaks.

`heteroChromWithPeaks` A 'GRanges' object for heterochromatin regions with glc peaks.

`bivalentPromWithPeaks` A 'GRanges' object for bivalent promoters with glc peaks.

`initiationWithPeaks` A 'GRanges' object for initiation regions with glc peaks.

`elongationWithPeaks` A 'GRanges' object for elongation regions with glc peaks.

`terminationWithPeaks` A 'GRanges' object for termination regions with glc peaks.

Constructor

new("genomicCompartments", referencePeaks = grglc, geneAnnotations = grlist, ...)

Accessors

In the following snippets, x is a genomicCompartments object.

getRefPeaks(x): Get the reference glc peaks
getGenesFullAnno(x): Get the full gene annotations
getCuratedAnno(x): Get the curated annotations
getActiveProm(x): Get the active promoters
getActiveEnh(x): Get the active enhancers
getPoisedEnh(x): Get the poised enhancers
getPcGDomain(x): Get the PcG domains
getHeteroChrom(x): Get the heterochromatin regions
getBivalentProm(x): Get the bivalent promoters
getInitiation(x): Get the initiation regions
getElongation(x): Get the elongation regions
getTermination(x): Get the termination regions
getSINE(x): Get the SINE regions
getLINE(x): Get the LINE regions
getLTR(x): Get the LTR regions
getRegionsMatrix(x): Get the regions matrix
getGlcPeakVal(x): Get the glc peak values
getActivePromWithPeaks(x): Get the active promoters with glc peaks
getActiveEnhWithPeaks(x): Get the active enhancers with glc peaks
getPoisedEnhWithPeaks(x): Get the poised enhancers with glc peaks
getPcGDomainWithPeaks(x): Get the PcG domains with glc peaks
getHeteroChromWithPeaks(x): Get the heterochromatin regions with glc peaks
getBivalentPromWithPeaks(x): Get the bivalent promoters with glc peaks
getInitiationWithPeaks(x): Get the initiation regions with glc peaks
getElongationWithPeaks(x): Get the elongation regions with glc peaks
getTerminationWithPeaks(x): Get the termination regions with glc peaks

Subsetting

In the following snippets, x is a genomicCompartments object.

setCurratedAnno(x, value): Set the curated annotations
setActivePromoters(x, value): Set the active promoters
setInitiationSer5PProm(x, value): Set the initiation Ser5P promoters
setElongationSer2P(x, value): Set the elongation Ser2P regions
setTermination(x, value): Set the termination regions
setBivalentProm(x, value): Set the bivalent promoters
setActiveEnhancer(x, value): Set the active enhancers
setPoisedEnhancer(x, value): Set the poised enhancers
setPolycombsDomain(x, value): Set the PcG domains
setHeterochromatin(x, value): Set the heterochromatin regions
setSINE(x, value): Set the SINE regions
setLINE(x, value): Set the LINE regions
setLTR(x, value): Set the LTR regions
setRegionsMatrix(x, value): Set the regions matrix
setGlcPeakVal(x, value): Set the glc peak values

setActivePromWithPeaks(x, value): Set the active promoters with glc peaks
setActiveEnhWithPeaks(x, value): Set the active enhancers with glc peaks
setPoisedEnhWithPeaks(x, value): Set the poised enhancers with glc peaks
setPcGDomainWithPeaks(x, value): Set the PcG domains with glc peaks
setHeteroChromWithPeaks(x, value): Set the heterochromatin regions with glc peaks
setBivalentPromWithPeaks(x, value): Set the bivalent promoters with glc peaks
setInitiationWithPeaks(x, value): Set the initiation regions with glc peaks
setElongationWithPeaks(x, value): Set the elongation regions with glc peaks
setTerminationWithPeaks(x, value): Set the termination regions with glc peaks

---

`matrixForUpset,genomicCompartments-method`
Create Matrix for UpSet Plot

Description

This method creates a matrix for generating an UpSet plot, indicating overlaps between glc peaks and genomic compartments.

Usage

```
matrixForUpset(theobject, includerepeats, glcpeakvalues)
```

Arguments

`theobject`	An object of class ‘genomicCompartments‘.
`includerepeats`	Logical indicating whether to include repeats in the analysis.
`glcpeakvalues`	Numeric vector of glc peak values.

Value

Returns the modified ‘genomicCompartments‘ object with the regions matrix assigned.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Create matrix for UpSet plot
matrixForUpset(gc_obj, includerepeats = FALSE, glcpeakvalues = c(1, 2, 3))

## End(Not run)
```

`outputGlcPeaksCoordPerCompartment,genomicCompartments-method`
Output Glc Peaks Coordinates per Compartment

Description

This method calculates the overlap of glc peaks with each compartment defined in the 'genomicCompartments' object.

Usage

```
outputGlcPeaksCoordPerCompartment(theobject, outputfolder, glcpeakspath,
                        includerepeats)
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`outputfolder`	Path to the folder where output files will be written.
`glcpeakspath`	Path to the GFF file containing glc peaks coordinates.
`includerepeats`	Logical indicating whether to include repeat regions in the analysis.

Value

Returns nothing explicitly; Write GFF and BED files of the coordinates of the glc peaks overlapping with a specific compartment.

Examples

```
## Not run:
# Parameters
peakspathquery <- "/path/to/glc.gff"
peakspathcategoriesvec <- c(H3K27ac = "/path/to/H3K27ac.gff",
       H3K4me1 = "/path/to/H3K4me1.gff", H3K27me3 = "/path/to/H3K27me3.gff",
       H3K4me3 = "/path/to/H3K4me3.gff", Suz12 = NA, RING1B = NA,
       H3K9me3 = "/path/to/H3K9me3.gff", Ser5P = "/path/to/Ser5P.gff",
       Ser2P = "/path/to/Ser2P.gff", ATACSeq = "/path/to/ATACseq.gff")
peakspathvec <- c(peakspathquery, peakspathcategoriesvec)
geneannovec <- c(gencode = "/path/to/gencode.gff",
       refgene = "/path/to/refGeneUCSC.gff",
       refseq = "/path/to/refseq.gff")

# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)

# Output glc peaks coordinates per compartment
outputGlcPeaksCoordPerCompartment(gc_obj, outputfolder, peakspathquery,
includerepeats = FALSE)

## End(Not run)
```

overlapOnGenes,genomicCompartments-method
Overlap Peaks on Genes

Description

This method performs overlap analysis between peaks and curated annotations at specified genomic regions (TSS or TES). It assigns overlapping peaks to specific compartments based on the provided labels.

Usage

```
overlapOnGenes(theobject, regionlabel, peaklabel = NA, peakpath = NA)
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
regionlabel	Character specifying the genomic region label ("TSS", "GBTES", or "TES").
peaklabel	Character specifying the peak label ("H3K27ac", "Ser5P", or "Ser2P").
peakpath	Path to the file containing peaks data. NA if regionlabel is "TES".

Value

Returns the modified ‘genomicCompartments‘ object with assigned compartments.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Perform overlap analysis on TSS with H3K27ac peaks
overlapOnGenes(gc_obj, regionlabel = "TSS", peaklabel = "H3K27ac",
peakpath = "path/to/peaks.gff")

## End(Not run)
```

polycombsDomains,genomicCompartments-method
Identify Polycomb Domains

Description

This method identifies Polycomb domains based on the overlap of two marks.

Usage

```
polycombsDomains(theobject, peakspath1, peakspath2, label1, label2,
domainname)
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
peakspath1	Path to the peaks file of the first polycomb.
peakspath2	Path to the peaks file of the second polycomb.
label1	Label for the first polycomb.
label2	Label for the second polycomb.
domainname	Name or label for the identified Polycomb domains.

Value

Returns the modified ‘genomicCompartments‘ object with Polycomb domains assigned.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Identify Polycomb domains
polycombsDomains(gc_obj, peakspath1 = "path/to/SUZ12.gff",
 peakspath2 = "path/to/RING1B.gff",
 label1 = "SUZ12", label2 = "RING1B", domainname = "PolycombDomains")

## End(Not run)
```

---

retrieveconversiontab *Retrieve Gene Conversion Table*

Description

This function retrieves a conversion table of gene identifiers from Ensembl BioMart.

Usage

```
retrieveconversiontab(biomartstr, datasetstr, hoststr, alternativemirroropt)
```

Arguments

biomartstr	A string specifying the BioMart database to use (e.g., "ENSEMBL_MART_ENSEMBL").
datasetstr	A string specifying the dataset to use within the BioMart database (e.g., "hsapiens_gene_ensembl").
hoststr	A string specifying the host URL for BioMart (e.g., "http://www.ensembl.org").
alternativemirroropt	A string specifying an alternative mirror option for BioMart, if needed.

Details

The function filters the retrieved data to include only rows with non-empty and non-NA gene IDs.

Value

A data frame containing the conversion table with columns for RefSeq, Entrez, and Ensembl gene identifiers.

Examples

```
## Not run:
biomartstr <- "ENSEMBL_MART_ENSEMBL"
datasetstr <- "hsapiens_gene_ensembl"
hoststr <- "http://www.ensembl.org"
alternativemirroropt <- "useast"
geneconvert <- retrieveconversiontab(biomartstr, datasetstr, hoststr,
alternativemirroropt)
head(geneconvert)

## End(Not run)
```

---

retrieveGlcPeakVal,genomicCompartments-method

Retrieve Glc Peak Values

Description

This method retrieves glc peak values associated with genomic compartments.

Usage

```
retrieveGlcPeakVal(theobject, includerepeats, bwpath)
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
includerepeats	Logical indicating whether to include repeats in the analysis.
bwpath	Character string specifying the path to the bigWig file containing glc peak values.

Value

Returns the modified ‘genomicCompartments‘ object with glc peak values assigned.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Retrieve glc peak values
gc_obj <- retrieveGlcPeakVal(gc_obj, includerepeats = FALSE,
bwpath = "path/to/bigWig/files")

## End(Not run)
```

setHeterochromatin<-,genomicCompartments-method

Allocates heterochromatic region coordinates

Description

This method stores the heterochromatin region coordinates in an object of class 'genomicCompartments'.

Usage

```
setHeterochromatin(obj) <- value
```

Arguments

theobject	An object of class 'genomicCompartments'.
value	GRanges object of heterochromatic region coordinates.

Value

The updated object of class 'genomicCompartments' with the new heterochromatin regions.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- new("genomicCompartments")
# Define new heterochromatin regions
new_heterochromatin <- GRanges(seqnames = "chr1",
ranges = IRanges(start = 1, end = 1000000))
# Set the heterochromatin regions
setHeterochromatin(gc_obj) <- new_heterochromatin

## End(Not run)
```

setLINE<-,genomicCompartments-method

Allocate LINE coordinates

Description

This method stores the LINE coordinates in an object of class 'genomicCompartments'.

Usage

```
setLINE(obj) <- value
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
value	GRanges object of LINE region coordinates.

Value

The updated object of class ‘genomicCompartments‘ with the new LINE regions.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- new("genomicCompartments")
# Define new LINE regions
new_LINE <- GRanges(seqnames = "chr1",
ranges = IRanges(start = 1, end = 1000000))
# Set the LINE regions
setLINE(gc_obj) <- new_LINE

## End(Not run)
```

---

setLTR<-,genomicCompartments-method
Allocate LTR coordinates

Description

This method stores the heterochromatin region coordinates in an object of class ‘genomicCompartments‘.

Usage

```
setLTR(obj) <- value
```

Arguments

theobject	An object of class ‘genomicCompartments‘.
value	GRanges object of LTR region coordinates.

Value

The updated object of class ‘genomicCompartments‘ with the new LTR regions.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- new("genomicCompartments")
# Define new LTR regions
new_LTR <- GRanges(seqnames = "chr1",
ranges = IRanges(start = 1, end = 1000000))
```

```
# Set the LTR regions
setLTR(gc_obj) <- new_LINE

## End(Not run)
```

`setSINE<-,genomicCompartments-method` *Allocates SINE coordinates*

Description

This method stores the SINE coordinates in an object of class 'genomicCompartments'.

Usage

```
setSINE(obj) <- value
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`value`	GRanges object of SINE region coordinates.

Value

The updated object of class 'genomicCompartments' with the new SINE regions.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- new("genomicCompartments")
# Define new SINE regions
new_SINE <- GRanges(seqnames = "chr1",
ranges = IRanges(start = 1, end = 1000000))
# Set the SINE regions
setSINE(gc_obj) <- new_SINE

## End(Not run)
```

`tryGetBM` *Attempt to Retrieve Information from Ensembl Using biomaRt*

Description

This function attempts to retrieve information about genes from Ensembl using the 'biomaRt' package. It tries up to 5 times before giving up and raising an error. The function supports specifying attributes, values, and filters for the query.

Usage

```
tryGetBM(attributes, ensembl, values = NULL, filters = NULL)
```

Arguments

attributes	Character vector specifying the attributes to retrieve.
ensembl	'Mart' object representing the connection to the Ensembl database.
values	Optional. List or vector of values for the filter.
filters	Optional. Character vector specifying the filters to apply.

Value

Returns a data frame with the retrieved information if the query is successful.

Examples

```
## Not run:
ensembl <- tryUseMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl",
host = "https://www.ensembl.org")
attributes <- c("ensembl_gene_id", "external_gene_name", "chromosome_name",
"start_position", "end_position")
gene_info <- tryGetBM(attributes = attributes, ensembl = ensembl)

## End(Not run)
```

---

tryUseMart *Attempt to Connect to Ensembl Using biomaRt*

Description

This function attempts to establish a connection to the Ensembl database using #' the 'biomaRt' package. It tries up to 5 times before giving up and raising an error. The function supports using an alternative mirror if specified.

Usage

```
tryUseMart(biomart = "ensembl", dataset, host, alternativemirror)
```

Arguments

biomart	Character string specifying the BioMart database to use. Default is "ensembl".
dataset	Character string specifying the dataset to use within the BioMart database.
host	Character string specifying the host URL to connect to.
alternativemirror	Logical value indicating whether to use an alternative mirror. Default is 'FALSE'.

Value

Returns a 'Mart' object if the connection is successful.

Examples

```
## Not run:
ensembl <- tryUseMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl",
host = "https://www.ensembl.org")

## End(Not run)
```

`upsetDiagram,genomicCompartments-method`
Generate UpSet Diagram

Description

This method generates an UpSet diagram based on the overlap matrix of glc peaks and genomic compartments.

Usage

```
upsetDiagram(theobject, outputfolder, includerepeats, plotupset = FALSE)
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`outputfolder`	Character string specifying the output folder where the UpSet diagram will be saved.
`includerepeats`	Logical indicating whether to include repeats in the analysis.
`plotupset`	Logical indicating whether to plot on device the upset plot ('TRUE') or not ('FALSE').

Value

Does not return anything but saves the plot to outpufolder.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Generate UpSet diagram
upsetDiagram(gc_obj, outputfolder = "results/upset_plots",
includerepeats = FALSE, plotupset = TRUE)

## End(Not run)
```

`violinplotExpression,genomicCompartments-method`
Generate Violin Plot for Gene Expression

Description

This method generates violin plots for gene expression across genomic compartments.

Usage

```
violinplotExpression(theobject, outfold, includerep, rnacounts, refseqpath,
 geneidtab, countsannotype, plotviolin = FALSE)
```

Arguments

`theobject`	An object of class 'genomicCompartments'.
`outfold`	Character string specifying the output folder path for saving results.
`includerep`	Logical indicating whether to include repeats in the analysis.
`rnacounts`	Character string specifying the path to the RNA-seq count table.
`refseqpath`	Character string specifying the path to the original RefSeq annotation file.
`geneidtab`	Table containing genes ID.
`countsannotype`	Character string specifying the type of counts (e.g., entrez or not) used for expression values.
`plotviolin`	Logical indicating whether to plot on current device (default is FALSE).

Value

Save the violin plot to outfold.

Examples

```
## Not run:
# Create a genomicCompartments object
gc_obj <- genomeCompart(peakspathvec, geneannovec)
# Generate violin plots for gene expression
violinplotExpression(gc_obj, outfold = "path/to/output_folder",
 includerep = TRUE, rnacounts = "path/to/rna_counts",
 refseqpath = "path/to/refseq_annotation",
 geneidtab = geneidtab, countsannotype = "entrez", plotviolin = TRUE)

## End(Not run)
```

Index