Package 'compEpiTools'

November 11, 2026

Type Package

Title Tools for computational epigenomics

Version 1.46.0

Date 2025-07-22

Description Tools for computational epigenomics developed for the analysis, integration and simultaneous visualization of various (epi)genomics data types across multiple genomic regions in multiple samples.

License GPL

Imports AnnotationDbi, BiocGenerics, Biostrings, Rsamtools, parallel, grDevices, gplots, IRanges, GenomicFeatures, XVector, methylPipe, GO.db, S4Vectors, Seqinfo

Depends R (>= 3.1.1), methods, topGO, GenomicRanges

Suggests BSgenome.Mmusculus.UCSC.mm9, TxDb.Mmusculus.UCSC.mm9.knownGene, org.Mm.eg.db, knitr, rtracklayer

VignetteBuilder knitr

biocViews GeneExpression, Sequencing, Visualization, GenomeAnnotation, Coverage

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

git_branch RELEASE_3_23

git_last_commit 8502cd6

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Mattia Pelizzola [aut], Kamal Kishore [aut], Mattia Furlan [ctb, cre]

Maintainer Mattia Furlan <mattia.furlan@iit.it>

Contents

compEpiTools-package . . . 2
countOverlapsInBins-methods . . . 4
distanceFromTSS-methods . . . 5
enhancers-methods . . . 6
findLncRNA . . . 7
getPromoterClass . . . 9
GR2fasta-methods . . . 10
GRanges2ucsc-methods . . . 10
GRangesInPromoters-methods . . . 11
GRannotate-methods . . . 11
GRannotateSimple-methods . . . 13
GRbaseCoverage-methods . . . 14
GRcoverage-methods . . . 15
GRcoverageSummit-methods . . . 16
GRenrichment-methods . . . 16
GRmidpoint-methods . . . 17
GRsetwidth-methods . . . 18
heatmapData . . . 18
heatmapPlot . . . 20
makeGtfFromDb-methods . . . 22
matchEnhancers-methods . . . 23
overlapOfGRanges-methods . . . 24
palette2d . . . 25
plotStallingIndex . . . 26
simplifyGOterms . . . 27
stallingIndex . . . 28
topGOres . . . 30
TSS . . . 31
ucsc2GRanges . . . 32
unionMaxScore-methods . . . 33

Index **34**

`compEpiTools-package` *Tools for computational epigenomics*

Description

Tools for computational epigenomics

Details

Package:	compEpiTools
Type:	Package
Version:	0.1
Date:	2014-04-07
License:	GPL
Depends:	methods

The package offers the following functionalities, divided by topic

Counting reads in GRanges:

- GRbaseCoverage: based on a GRanges and a BAM file, returns a list of base coverage vectors for each range
- GRcoverage: based on a GRanges and a BAM file, returns the total coverage for each range
- GRcoverageInbins: same as GRcoverage but dividing each range in equally-sized bins
- GRcoverageSummit: based on a GRanges and a BAM file, returns a GRanges with the positions of maximum coverage within each range
- GRenrichment: determines the enrichment over a set of genomic regions given two BAM files
- countOverlapsInBins: given a query and a subject GRanges returns a matrix of counts of subject in bins of query
- stallingIndex: computes the PolII stalling index based on number of ChIP-seq reads in promoter and genebody

Annotation of genomic regions:

- TSS: based on a tXDb returns a GRanges with the TSS positions for all transcripts
- distanceFromTSS: based on a GRanges, returnes the GRanges annotated with info about the closer TSS
- GRangesInPromoters: based on a GRanges and a TxDb, subsets the GRanges to those regions overlapping with promoters
- GRmidpoint: returns a GRanges containing the mid point of a GRanges
- GRannotate: based on a GRanges and a TxDb, returns the GRanges with a series of annotations
- GRannotateSimple: a GRanges method to split a GRanges in three GRanges: promoter, intragenic and intergenic
- makeGtfFromDb: utilities to transform a TranscriptDb into a GTF file

Functional annotation:

- enhancers: a GRanges method to define enhancers based on H3K4me1 peaks
- matchEnhancers: a GRanges method to match enhancers with putative targets sites (either TSS or TF-bound TSS)
- topGOres: determines GeneOntology enriched terms for a set of gene ids

- simplifyGOterms: simplify a list of GeneOntology terms based on the list of genes assigned to each GO term
- findLncRNA: identify putative long non coding RNAs (lncRNA) based on ChIP-seq chromatin features and RNAseq data
- getPromoterClass: determining the CpG promoter class and the average CpG content

Visualization:

- heatmapData: Based on a list of GRanges, determine various kind of counts before displaying a heatmap
- palette2d: build a two dimensional color palette
- heatmapPlot: displays the heatmap based on the data from GRheatmapData
- plotStallingIndex: plot the PolII stalling index

Other:

- GR2fasta: A GRanges method to extract and write to the disk a fasta file containing genomic sequences for the GRanges regions in a genome
- overlapOfGRanges: given a list of GRanges, all pair-wise overlap are evaluated and the percentage of overlapping ranges is visualized in a heatmap
- GRsetwidth: set the width of a GRanges based on the mid point of each region
- unionMaxScore: GRanges method to perform union of peaks keeping the pvalue of the most significant peak
- GRanges2ucsc: a GRanges method to convert ranges information into UCSC format
- ucsc2GRanges: convert UCSC-formatted genomic positions into a GRanges

Author(s)

Computational Epigenomics Unit at the Center for Genomic Sciences of IIT@SEMM, Milan, Italy
http://genomics.iit.it/groups/computational-epigenomics.html <mattia.pelizzola@gmail.com>

`countOverlapsInBins-methods`

given a query and a subject GRanges returns a matrix of counts of subject in bins of query

---

Description

Given a query and a subject GRanges, this function returns a matrix with number of rows equal to the number of regions in query and number of columns equal to the number of bins. For each bin of each region, the occurrence (1) or not (0) of subject is returned.

Methods

The countOverlaps method can be used to determine the overlap between a query and a subject GRanges.

The countOverlapsInBins method add the functionality to partition query in bins.

To be used in this form:

countOverlapsInBins(query, subject, nbins)

where:

query: GRanges

- subject: GRanges
- nbins: numeric, the number of bins

It returns a matrix with number of rows equal to the number of regions in query and number of columns equal to the number of bins.

For each bin of each region 1 is assigned of subject GRanges overlap, 0 if it does not.

Examples

```
gr1 <- GRanges(seqnames=Rle('chr1',2), ranges=IRanges(start=c(10,100), end=c(50, 150)))
gr2 <- GRanges(seqnames=Rle('chr1',2), ranges=IRanges(start=c(2,40), end=c(15, 70)))
countOverlapsInBins(gr1, gr2, nbins=4)
```

`distanceFromTSS-methods`

Returns the GRanges annotated with info about the closer TSS

Description

For each GRanges region it decorates the GRanges with extra columns containing info about the closer TSS.

Methods

To be used in this form:

distanceFromTSS(Object, txdb, EG2GS=NULL)

where:

Object: GRanges

- txdb: TxDb
- EG2GS: an object of class OrgDb; like org.Mm.eg.db, org.Hs.eg.db (use the exact name of object)

The method returns a GRanges with additional columns. If EG2GS is NULL three columns are appended containing info for the gene with the closer TSS:

nearest_tx_name: the transcript id

- distance_fromTSS: the distance in bp
- nearest_gene_id: gene id

If EG2GS is provided, gene symbols are also included as additional column.

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(TRUE, rep(FALSE, length(isActiveSeq(txdb))-1))
TSSpos <- TSS(txdb)
gr <- TSSpos[1:5]
start(gr) <- start(gr)-1000
end(gr) <- end(gr)-600
mcols(gr) <- NULL
distanceFromTSS(Object=gr, txdb=txdb, EG2GS=NULL)
restoreSeqlevels(txdb)
```

---

enhancers-methods *A GRanges method to define enhancers based on H3K4me1 peaks*

Description

A GRanges method to define enhancers based on H3K4me1 peaks and genome annotation

Methods

To be used in this form:

enhancers(gr, txdb, upstream= 2000, downstream= 1000, CGIgr=NULL)

where:

gr: a GRanges, typically of H3K4me1 peaks

- txdb: an object of class TxDB
- upstream: numeric; the number of bp upstream the TSS
- downstream: numeric; the number of bp downstream the TSS
- CGIgr: GRanges; optional GRanges of CpG Islands (CGI)
  Enhancers are defined as distal H3K4me1 peaks not overlapping with CGI, to avoid unannotated transcriptional units. Distal peaks are those peaks not overlapping with promoters. Alternative marks or proteins, such as H3K27ac or mediator, could be used here in place of H3K4me1. For example, H3K27ac would specifically allow to identify active enhancers.

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
# loading H3K4me1 peaks as a GRanges object
  # built based on the BED file from the GEO GSM1234488 sample
  # limited to chr19:3200000-4000000
  H3K4me1GR <- system.file("extdata", "H3K4me1GR.Rda", package="compEpiTools")
  load(H3K4me1GR)
enhancers(H3K4me1GR, txdb)
```

`findLncRNA` *Identify putative long non coding RNAs (lncRNA)*

Description

Identify putative long non coding RNAs (lncRNA) based on ChIP-seq chromatin features and RNAseq data

Usage

```
findLncRNA(k4me3gr, k4me3bam, k4me1bam, k79bam, k36bam, RNAseqbam,
sizeLNC=10000, extDB= NULL, txdb, org=NULL, Qthr=0.95)
```

Arguments

`k4me3gr`	GRanges; the set of H3K4me3 peaks from a ChIP-seq experiment
`k4me3bam`	character; a path to BAM file containing H3K4me3 ChIP-seq aligned reads
`k4me1bam`	character; a path to BAM file containing H3K4me1 ChIP-seq aligned reads
`k79bam`	either NA or a path to BAM file containing H3K79me2 ChIP-seq aligned reads
`k36bam`	either NA or a path to BAM file containing H3K36me3 ChIP-seq aligned reads
`RNAseqbam`	either NA or a path to BAM file containing RNA-seq aligned reads
`sizeLNC`	numeric; the size of the putative lncRNA
`extDB`	GRanges; a set lncRNAs to be included in the analysis
`txdb`	an object of class TxDb
`org`	either NULL or an object of class BSgenome
`Qthr`	numeric in [0,1]; the percentile of the signal in random genomic regions to be considered as minumum cutoff

Details

Putative long non coding RNAs (lncRNAs) are identified based on the associated chromatin features and, possibly, RNAseq signal. Only putative lncRNAs distal from genebodies are identified. Briefly, H3K4me3 peaks are used as the main mark indicating transcriptional activity. Only H3K4me3 peaks outside genebodies +/- 10Kb are considered. Only peaks where the signal of H3K4me1 is lower than H3K4me3 are kept, to discard possible enhancer sites. Regions of interest of length sizeLNC (default 10Kb) are considered from the mid point of remaining H3K4me3 peaks, either in the forward or reverse direction (ROIs). The rational is that distal H3K4me3 marks could indicate the TSS of distal lncRNAs.

Optionally, to increase the likelihood of having identified a bona-fide transcriptional unit, downstream regions (ROIs on either the forward or reverse strand) are evaluated for the existence of significant transcriptional signal. This is achieved based on the optional data (optional tracks) to be provided as BAM files: ChIP-seq for H3K79me2 or H3K36me3, or RNA-seq. Density of H3K79me2, H3K36me3 and RNAseq reads in the remaining H3K4me3 regions and in 100K random regions of 10Kb each is determined (non overlapping with ROIs), normalized by the respective

library size. ROIs where the signal of any of these is higher than the Qthr percentile of the random regions (profiled for the same mark) are considered as putative lncRNAs.

An optional GRanges containing regions to be considered in any case (for example based on lists of a priori known lncRNAs) can be provided as extDB. These regions will be evaluated as they are, and subject to the same filtering procedure based on H3K79me2, H3K36me3 and RNAseq data, if provided.

Passing at least one of H3K79me2, H3K36me3 and RNAseq while setting Qthr to 0 correspond to profile the ROIs (and the extDB regions if provided) for those optional tracks and avoid filtering based on the signal of random regions. All bam files have to be associated to the corresponding index .bai files. Please refer to the documentation of samtools on how to create them.

Value

Either NULL of a data.frame where putative lncRNAs (UCSC-format coordinates are reported as row names) are reported on the rows and the columns indicate the library-size normalized reads density of the following marks: H3K4me3, H3K4me1, H3K79me2, H3K36me3 and RNAseq reads. Reads density is reported for both up- and down-stream regions of width sizeLNC, see details, while for extDB regions the reads density is reported only for the regions as they are defined.

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
# loading H3K4me3 peaks as a GRanges object
# built based on the BED file from the GEO GSM1234483 sample
# limited to chr19:3200000-4000000
H3K4me3GR <- system.file("extdata", "H3K4me3GR.Rda", package="compEpiTools")
load(H3K4me3GR)
# pointing to Pol2 BAM file (it could be used as a replacement of the K79bam or K36bam ..)
# BAM file from the GEO GSM1234478 sample, limited to chr19:3200000-4000000
Pol2bam <- system.file("extdata", "Pol2.bam", package="compEpiTools")
# pointing to H3K4me3 BAM file
# BAM file from the GEO GSM1234483 sample, limited to chr19:3200000-4000000
H3K4me3bam <- system.file("extdata", "H3K4me3.bam", package="compEpiTools")
# pointing to H3K4me1 BAM file
# BAM file from the GEO GSM1234488 sample, limited to chr19:3200000-4000000
H3K4me1bam <- system.file("extdata", "H3K4me1.bam", package="compEpiTools")
res <- findLncRNA(k4me3gr=H3K4me3GR, k4me3bam=H3K4me3bam, k4me1bam=H3K4me1bam,
  k79bam=Pol2bam, k36bam=NA, RNAseqbam=NA,
  sizeLNC=10000, txdb=txdb, org=NULL, Qthr=0)
```

`getPromoterClass` *Determining the CpG promoter class and the average CpG content*

Description

Determining the CpG promoter class (Low, Intermediate or High CpG Content: lowCG, intCG or highCG, respectively) and the average CpG content for each entry of a transcriptDB.

Methods

To be used in this form:

getPromoterClass(txdb, Nproc=1, org, upstream=1000, downstream=0)

where:

txdb:An object of class TxDb

- Nproc:numeric; the number of processors to be used; one chr is run for each processor
- org:an object of class BSgenome
- upstream:numeric; number of bp upstream transcription start sites defining upstream limit of promoters
- downstream:numeric; number of bp downstream transcription start sites defining downstream limit of promoters

According to Weber M et al, Nature Genet 2007: the CpG content is determined as (W*CpG)/(C*G), where W is the window size and CpG, C and G are the number of CpG, C and G occurrences, respectively. Here W is set to 500bp. The CplusG content is (C*G)/W. The promoter CpG class is determined sliding a 500bp window in 1Kb upstream regions, with step of 5 bp. If the maximum CpG content for a given promoter is < 0.48, the promoter is assigned a lowCG. If the maximum CpG content is > 0.75 and CplusG>0.55, the promoter is assigned a highCG. The remaining promoters are assigned a intCG. The average CpG ratio is the average of the CpG ratios for all the windows.

A GRanges object decorated with the promoterClass and promoterCpG data is returned.

Examples

```
require(BSgenome.Mmusculus.UCSC.mm9)
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(rep(FALSE,20), TRUE, rep(FALSE, 14))
allpromoter <- getPromoterClass(txdb, Nproc=1, org=Mmusculus)
restoreSeqlevels(txdb)
```

GR2fasta-methods *A GRanges method to extract and write to the disk a fasta file containing genomic sequences for the GRanges regions in a genome*

Description

Given a GRanges and a reference genome returns the sequences for all the ranges as in fasta format.

Methods

To be used in this form:

GR2fasta(GR, org, fastaFile=NULL)

where:

GR: GRanges

- org: an object of class BSgenome
- fastaFile: character or NULL; an optional file name for the file to be written on disk

For each range in the gr GRanges , the unmasked reference sequenced is retrieved. All the sequences are returned in fasta format, named by the genomic ranges in UCSC format, and optionally written on disk.

Examples

```
require(BSgenome.Mmusculus.UCSC.mm9)
gr <- GRanges(Rle(c('chr1','chr2')),
 ranges=IRanges(start=c(1e7, 2e7), end=c(1e7+19, 2e7+19)))
show(GR2fasta(GR=gr, org=Mmusculus, fastaFile=NULL))
```

GRanges2ucsc-methods *A GRanges method to convert ranges information into UCSC format*

Description

Chr assignments, start and end positions are converted into UCSC format, in the form chr1:100-500

Methods

To be used in this form:

GRanges2ucsc(Object)

where Object is a GRanges

Examples

```
gr <- GRanges(Rle(c('chr1','chr2')),
 ranges=IRanges(start=c(1e7, 2e7), end=c(1e7+19, 2e7+19)))
GRanges2ucsc(gr)
```

GRangesInPromoters-methods *Based on a GRanges and a TxDb, subsets the GRanges to those regions overalpping with promoters*

Description

Subset a GRanges returning only those ranges which are overlapping (at least 1bp) with promoters

Methods

This method subsets a GRanges returning only those ranges which are overlapping, or not, with promoters defined based on a TxDb, upstream and downstream distance from TSS.

To be used in this form:

GRangesInPromoters(Object, txdb, upstream=2000, downstream=1000, invert=FALSE)

where:

Object: GRanges

- txdb: TxDb
- upstream: numeric
- downstream: numeric
- invert: logical; see below

Promoters are defined based on upstream and downstream distances from Transcription Start Sites (TSS). If invert if FALSE, the subset of Object overlapping with promoter regions is returned, if any, otherwise NULL is returned. If invert if TRUE, the subset of Object which is not overlapping with promoter regions is returned, if any, otherwise NULL is returned.

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb<- TxDb.Mmusculus.UCSC.mm9.knownGene
TSSpos <- TSS(txdb)
gr <- TSSpos[1:5]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
GRangesInPromoters(Object=gr, txdb=txdb, upstream=2000, downstream=1000)
```

GRannotate-methods *Based on a GRanges and a TxDb, returns the GRanges with a series of annotations*

Description

Based on a GRanges and a TxDb, returns the GRanges with a series of annotations

Methods

To be used in this form:

GRannotate(Object, txdb, EG2GS, upstream=2000, downstream=1000, userAnn=NULL)

where:

Object: a GRanges object with ranges of width 1

- txdb: TxDb
- EG2GS: an object of class OrgDb; like org.Mm.eg.db, org.Hs.eg.db (use the exact name of object)
- upstream: numeric, bp upstream the TSS to define promoters
- downstream: numeric, bp downstream the TSS to define promoters
- userAnn: either NULL or a named GRangesList containing user defined annotations as GRanges objects

The method returns a GRanges with extra columns containing the following annotations:

nearest_tx_name: transcript id for the gene with the closer TSS

- distance_fromTSS: distance in bp from the closer TSS
- nearest_gene_id: gene id for the gene with the closer TSS
- nearest_gene_symbol: gene symbol for the gene with the closer TSS
- location: 'intergenic' or a combination of 'promoter' and 'genebody'
- location_tx_id: transcript id(s) corresponding to the matched location, location is not 'intergenic'
- location_gene_id: gene id(s) corresponding to the matched location, location is not 'intergenic'
- location_gene_symbol: gene symbol(s) corresponding to the matched location, location is not 'intergenic'

This method only works with GRanges containing ranges of width 1. In case of annotation of GRanges of different width, such as ChIP-seq peaks, they should be summarized to GRanges of width 1. This could be easily done using the GRmidpoint method, pointing to the peaks mid points, or using GRcoverageSummit, pointing to the positions of maximum coverage provided that the BAM file is available. The obtained annotation only refers to these GRanges of width 1, and should not be referred as the annotation of a wider region. This is necessary to decrease the complexity of the output. Indeed, even given a simple GRanges of width 1, one could have multiple associated genomic features associated to each range. For example, a range could be associated at the same time to a promoter of a transcript and to the intron of another isoform for the same gene.

Additional columns will be reported in the output GRanges if a userAnn GRangesList is provided (if userAnn is not NULL). For each of these extra-columns, 1 (or 0) are used to indicate overlap (or lack of) for each given Object range with at least one range of the correspondent userAnn GRanges. To this purpose, annotation tracks (tables) that are available in UCSC can be used as userAnn GRangesList. Please refer to the documentation of ucscTableQuery in the rtracklayer package to download these tables in R and convert them into a GRangesList object. See the example for a case in which mm9 CpG Islands are retrieved and provided to GRannotate to match each genomic region of interest (the mid points of gr) to this annotation source (CGIgr). Promoter regions are defined based on upstream and downstream bp from TSS.

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(TRUE, rep(FALSE, length(isActiveSeq(txdb))-1))
require(org.Mm.eg.db)
require(rtracklayer)
TSSpos <- TSS(txdb)
gr <- TSSpos[1:5]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
mcols(gr) <- NULL
res <- GRannotate(Object=GRmidpoint(gr), txdb=txdb, EG2GS=org.Mm.eg.db,
 upstream=2000, downstream=1000)
isActiveSeq(txdb) <- rep(TRUE, length(isActiveSeq(txdb)))
## alternatively, CGI can be incoirporated as follow:
## retrieving CGI mm9 islands from UCSC annotation tables
# session <- browserSession()
# genome(session) <- 'mm9'
# query <- ucscTableQuery(session, 'cpgIslandExt')
# CGIgr <- as(track(query), 'GRanges')
# res <- GRannotate(Object=GRmidpoint(gr), txdb=txdb, EG2GS=org.Mm.eg.db,
# upstream=2000, downstream=1000, userAnn=GRangesList(CGI=CGIgr))
```

GRannotateSimple-methods

a GRanges method to split a GRanges in three GRanges: promoter, intragenic and intergenic

---

Description

Given a GRanges and a TxDB object, the GRanges are divided according to their overlap to promoters (defined based on upstream and downstream bp from transcription start sites, TSS) and genebody (intragenic), while the remaining GRanges are assigned to intergenic

Methods

To be used in this form:

GRannotateSimple(gr, txdb, upstream=2000, downstream=1000)

where:

gr: a GRanges

- txdb: an object of class TxDB
- upstream: numeric; the number of bp upstream the TSS
- downstream: numeric; the number of bp downstream the TSS

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
gr <- GRanges(Rle(c('chr1','chr1')),
 ranges=IRanges(start=c(100,200), end=c(150,250)))
GRannotateSimple(gr, txdb)
```

GRbaseCoverage-methods

Based on a GRanges and a BAM file, returns a list of base coverage vectors for each range

---

Description

Based on a GRanges and a BAM file, returns a list of base coverage vectors for each range

Methods

To be used in this form:

GRbaseCoverage(Object, bam, Nnorm=FALSE)

where:

Object: GRanges

- bam: path to BAM file
- Nnorm: logical; whether to apply library size normalization

The method determines for each base in each region of the GRanges the number of reads in the BAM file. If Nnorm is TRUE the coverage is divided by million of mapped reads contained in the BAM file. The method returns a list of length equal to the length of the Object GRanges. Each list item is a vector of lenth equal to the length of the corresponding range width. The vector reports the (normalized) base coverage. The bam file has to be associated to the corresponding index .bai file. Please refer to the documentation of samtools on how to create it.

Examples

```
bampath <- system.file("extdata", "ex1.bam", package="Rsamtools")
gr <- GRanges(seqnames=Rle(c('seq1','seq2')),
 ranges=IRanges(start=c(1000, 100), end=c(2000, 1000)))
res <- GRbaseCoverage(Object=gr, bam=bampath)
str(res)
```

GRcoverage-methods *based on a GRanges and a BAM file, returns the total coverage for each range, or for each bin of the range.*

Description

GRcoverage returns the total coverage for each range in GRanges, while GRcoverageInbins divides each range in equally-sized bins and returns the total coverage for each bin.

Methods

To be used in this form:

GRcoverage(Object, bam, Nnorm=TRUE, Snorm=TRUE)

GRcoverageInbins(Object, bam, Nnorm=TRUE, Snorm=TRUE, Nbins)

where:

Object: GRanges

- bam: character; a path to a BAM file
- Nnorm: logical; whether to perform library size normalization
- Snorm: logical; whether to perform normalization based on the region width in bp
- Nbins: numeric; the number of equally sized bins each range in Object is divide into

For each range in the GRanges (or for each bin within the range), the sum of the base coverage in the range is determined.

If Nnorm is TRUE, the coverage is divided by the number of aligned reads in the BAM file, and multiplied by 1e6.

If Snorm is TRUE, the coverage is divided by the region width in bp.

GRcoverage returns an array of region coverages with length equal to the length of the Object GRanges.

GRcoverageInbins returns a matrix with nrow equal to the length of the Object GRanges and ncol equal to Nbins; if Nbins is equal to 1 a matrix of 1 column is returned.

An increasing number of bins, does not correspond to a significant increase in the computation time (65sec for 43k regions and 10 bins).

GRcoverageInbins with Nbins equal to 1 returns counts identical to GRcoverage but it is slightly slower (64sec vs 46sec for 43k regions).

GRcoverageInbins will return NAs and a warning for the rows corresponding to those ranges whose width is lower than the number of bins.

The coverage for sequences not available in the BAM file is set to 0.

The BAM file has to be associated to the corresponding index .bai file. Please refer to the documentation of samtools on how to create it.

Examples

```
bampath <- system.file("extdata", "ex1.bam", package="Rsamtools")
gr <- GRanges(seqnames= Rle(c('seq1','seq2')),
 ranges=IRanges(start=c(1000, 100), end=c(2000, 1000)))
GRcoverage(Object=gr, bam=bampath, Nnorm=TRUE, Snorm=TRUE)
GRcoverageInbins(Object=gr, bam=bampath, Nnorm=TRUE, Snorm=TRUE, Nbins=4)
```

GRcoverageSummit-methods *Based on a GRanges and a BAM file, returns a GRanges with the positions of maximum coverage within each range*

Description

Based on a GRanges and a BAM file (and optionally a control BAM file), returns a GRanges with the positions of maximum coverage within each range; it can be used to identify peak summits in ChIPseq enriched regions.

Methods

To be used in this form:

GRcoverageSummit(Object, bam, bamControl=NULL)

where:

Object: GRanges

- bam: a BAM file path
- bamControl: a BAM file path

The method returns a GRanges with regions of width 1 pointing to the position of higher coverage.

If the optional bamControl is provided, typically the ChIP-seq input sample, the bamControl coverage is subtracted from the bam coverage before identifying the maximum.

If multiple maxima exist in a range, one is returned at random.

The bam file has to be associated to the corresponding index .bai file. Please refer to the documentation of samtools on how to create it.

Examples

```
bampath <- system.file("extdata", "ex1.bam", package="Rsamtools")
gr <- GRanges(seqnames=Rle(c('seq1','seq2')),
  ranges=IRanges(start=c(1000, 100), end=c(2000, 1000)))
GRcoverageSummit(Object=gr, bam=bampath)
```

GRenrichment-methods *Determines the enrichment over a set of genomic regions given two BAM files*

Description

Given a GRanges, and two BAM files, this method detetermines the coverage of the 1st BAM file (bam) and of the 2nd BAM file (bamRef), both normalized by millions of reads in the library. Subsequently, the enrichment is computed as log2(bam - bamRef).

Methods

To be used in this form:

GRenrichment(Object, bam, bamRef)

where:

Object: a GRanges

- bam: path to a BAM file
- bamRef: path to a BAM file, to be used as reference
  This is typically useful when applied to BAM files derived from ChIP-seq experiments, where bam and bamRef are the result of the alignment of ChIP and input reads, respectively.

An array of length equal to the length of Object is returned, containing the enrichment values. The enrichment of a given genomic region in Object is set to NA if in either bam or bamRef there are no reads, while it is -Inf of the same normalized coverage is found for both bam and bamRef (as in the example).

The bam file has to be associated to the corresponding index .bai file. Please refer to the documentation of samtools on how to create it.

Examples

```
bampath <- system.file("extdata", "ex1.bam", package="Rsamtools")
gr <- GRanges(seqnames=Rle(c('seq1','seq2')),
 ranges=IRanges(start=c(1000, 100), end=c(2000, 1000)))
 # bam and bamRef should not be pointing to the same file in real life ..
GRenrichment(Object=gr, bam=bampath, bamRef=bampath)
```

GRmidpoint-methods *Returns a GRanges containing the mid point of a GRanges*

Description

Returns a GRanges containing the mid point of a GRanges

Methods

To be used in this form:

GRmidpoint(Object)

where Object is a GRanges.

A GRanges with width 1 containing the mid points of each range in Object is returned.

Examples

```
gr <- GRanges(seqnames=Rle('chr1',2),
 ranges=IRanges(start=c(10,100), end=c(50, 150)))
GRmidpoint(gr)
```

GRsetwidth-methods *Set the width of a GRanges based on the mid point of each region*

Description

Given a GRanges, this method sets the width of a GRanges based on the mid point of each region

Methods

To be used in this form:
GRsetwidth(gr, newWidth)
where:
gr: a GRanges
- newWidth: a positive numeric

Examples

```
gr <- GRanges(Rle(c('chr1','chr1')),
 ranges=IRanges(start=c(100,200), end=c(150,250)))
GRsetwidth(gr, 1000)
```

heatmapData *Based on a list of GRanges, determine various kind of counts before displaying a heatmap*

Description

Based on a list of GRanges, determine various kind of counts before displaying a heatmap

Usage

```
heatmapData(grl, refgr=grl[[1]], useScore=rep(FALSE,
length(grl)), type, Nnorm=TRUE, Snorm=TRUE, txdb=NULL, nbins=5)
```

Arguments

grl	list; a list of GRanges or paths to BAM files
refgr	GRanges; the reference set of genomic regions
useScore	logical; an optional boolean array of length equal to the length of grl
type	character; an array of length equal to the length of grl, with a combination of 'mcols', 'gr' or 'cov'
Nnorm	logical; whether to perform library size normalization, only applied if some of the element in type are equal to 'cov'
Snorm	logical; whether to perform normalization based on the refgr widths, only applied if some of the element in type are equal to 'cov'
nbins	numeric; the number of bins the ranges in refgr have to be divided into
txdb	an object of class TxDb

Details

The functions is used to determine various kind of counts for each object in grl in each range of refgr and is typically used to prepare the input for the heatmapPlot method.

The type of counts is determined through the corresponding type setting. If type is mcols, the counts are expected to be pre-calculated and available in the mcols of the correponding grl GRanges. If type is gr, the corresponding grl GRanges (gr) is considered and the counts are the number of occurrencies of gr for each ranges of refgr; if nbins is greater than 1 and type is gr, the counts are determined for each bin of each range of refgr. A score (the lower, the more significant) can be provided in the first column of the mcols of gr; the minimum score over the gr ranges associated to every given refgr range is determined and stored in the corresponding column of the scoreMat output matrix.

If type is cov, the corresponding grl has to be a path to a BAM file, and the counts are the coverage within each range of refgr; if nbins is greater than 1 and type is cov, the counts are determined for each bin of each range of refgr. If Nnorm is TRUE and type is cov, the counts are divided by the million mapped reads in the BAM file. If Snorm is TRUE and type is cov, the counts are divided by the range width in bp.

If a TxDb is provided, the presence of an intron or exon is registered for each range of refgr; intron is assigned 0.6, exon 0.4, and they will be rendered using the heatmapPlot function as red and pink, respectively. If nbins is greater than 1 and a TxDb is provided, the presence of an intron or exon is registered for each bin of each range of refgr.

The bam files have to be associated to the corresponding index .bai files. Please refer to the documentation of samtools on how to create them.

Value

A list of two items, matList and scoreMat is returned. matList: if a TxDb is not provided, matList is a list of length equal to the length of grl; each item of the list is a matrix with number of rows equal to the number of ranges in refgr, and number of columns equal to nbins; if a TxDb is provided, matList is a list of length equal to the length of grl + 2 is returned; the two extra items contain the count for introns and exons. scoreMat: if useScore is all FALSE then scoreMat is set to NULL, otherwise it is a matrix whose number of rows is equal to the length of refgr and the number of columns is equal to the length of grl; row Nr and column Nc contain the minimum score of mcols(grl[[Nc]])[,1] for the ranges overlapping with refgr[Nr], if any (0 otherwise).

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(TRUE, rep(FALSE, length(isActiveSeq(txdb)) - 1))
TSSpos <- TSS(txdb)
gr <- TSSpos[1:5]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
extgr <- GRanges(seqnames(gr), ranges=IRanges(start(gr) - 1000, end(gr) + 1000))
```

```
data <- heatmapData(grl=list(ChIPseq= gr), refgr=extgr, type='gr')
restoreSeqlevels(txdb)
```

`heatmapPlot` *displays the heatmap based on the data from heatmapData*

Description

displays heatmap of counts for a list of GRanges, typically computed based on the heatmapData function

Usage

```
heatmapPlot(matList, sigMat=NULL, qnorm=NULL, tnorm=NULL,
rowLab=FALSE, colLab=TRUE, margins=NULL, colors=NULL,
clusterInds=1:length(matList), dendrogram=TRUE)
```

Arguments

`matList`	list; a list of matrices, all with the same number of rows and columns, typically the output of heatmapData
`sigMat`	matrix; a matrix of p-values with nrow equal to the nrow of the matList matrices and ncol equal to the length of matList
`qnorm`	array; an array with length equal to the length of matList containing either NULL or thresholds for quantile normalization, see details
`tnorm`	array; an aray with length equal to the length of matList containing either NULL or threshold for data normalization, see details
`rowLab`	logical; whether to add row labels to the heatmap, taken from the rownames of matList[[1]]
`colLab`	logical; whether to add column names to the heatmap, taken from the names of matList
`colors`	either NULL or character with an array of valid colors; it only works if sigMat is NULL
`margins`	either NULL or a numeric array of length 2
`clusterInds`	either NULL or numerics, defining with matList items have to be used to drive the clustering of heatmap rows
`dendrogram`	logical; whether to display the dendrogram folling the clustering of rows

Details

Each matrix in matList is either ranging from 0 to 1 or will be forced to by dividing to its maxumum. Alternatively, matrix normalization can be obtained using qnorm or tnorm. Setting qrnorm to X [0,1] for a given matList matrix will force the maximum of the matrix to be quantile(matrix, X). Setting trnorm to X for a given matList matrix will force the maximum of the matrix to be X. Using either qnorm or tnorm matrix will be finally normalized to 1 by dividing it by its maximum.

If sigMat is not NULL it is expected to contain a list of pvalues or scores [0,1] for each range for each matList dataset. The colorscale of the heatmap will be adjusted to display the significance, by lightening the observation colors as a function of its significance, see the example. A minimum pvalue of 1e-10 is forced. 50 levels of intensity (white to orange to red palette, as displayed in the colorscale) and 10 levels of significance (white for the less significance, full color according to the intensity palette, as displayed in the colorsclae) are considered. Plese ignore the values reported in the colorscale. If sigMat is NULL, the normalized intensity in each matList item is reported as it is on a white to beige to red default palette, or based on the colors defined in the colors argument.

If margins is set to NULL the row and columns margins used to display labels are computed automatically, otherwise a numeric array of length two can be set to define them (in lines).

clusterInds can be used to define which matList items drive the clustering of heatmap rows. If clusterInds is NULL no clustering is performed and no dendrogram is displayed. If clusterInds is an array of index in 1:length(matList), only those matList items will be used to determine the clustering and the dendrogram, while all matList data will be displayed.

If a TxDB was provided to heatmapData beofre calling heatmapPlot, the last two tracks are about the overlap with exons and introns in the forward and reverse strand, repectively. If the default white to red color palette is used, and sigMat is NULL exons will be plotted in red and introns in pink. Rather, if sigMat is defined, introns will be in orange.

Value

A list with two items

`data`	the normalized matrix used for the final heatmap visualization
`heatRes`	the list invisibly returned by the heatmap.2 function, see its documentation

Be careful that the rowInds contained in heatRes poiting to the new order of clustered data rows, is intended to list the reordered rows starting from the bottom of the heatmap.

References

http://genomics.iit.it/groups/computational-epigenomics.html

See Also

See Also as `heatmap.2`

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(TRUE, rep(FALSE, length(isActiveSeq(txdb)) - 1))
TSSpos <- TSS(txdb)
names(TSSpos) <- TSSpos$tx_name
gr <- TSSpos[1:50]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
pvalues <- c(runif(20,1e-20,1e-8), runif(15,1e-4,1e-2), runif(15,0.5,1))
mcols(gr) <- pvalues
extgr <- GRanges(seqnames(gr), ranges= IRanges(start(gr) - 1000, end(gr) + 1000))
```

```
data <- heatmapData(grl=list(ChIPseq=gr), refgr=extgr, type='gr', useScore=TRUE,
  Nnorm=TRUE, Snorm=TRUE, nbins=6, txdb=txdb)
rownames(data[[1]][[1]]) <- paste(1:50, signif(pvalues,1), sep=' # ')
heatmapPlot(matList=data$matList, qnorm=NULL, tnorm=NULL,
  rowLab=TRUE, colLab=TRUE, clusterInds=1:3)
dev.new()
heatmapPlot(matList=data$matList, sigMat=data$scoreMat, qnorm=NULL, tnorm=NULL,
  rowLab=TRUE, colLab=TRUE, clusterInds=1:3)
restoreSeqlevels(txdb)
```

`makeGtfFromDb-methods` *Utilities to transform a TxDb into a GTF file*

Description

Given a TxDB, the method 'makeGtfFromDb' creates and writes to file a GTF file of exons either at the gene level or at the transcript level. This methods can be used to generate a GTF file that is compliant with Bioconductor TxDB libraries, to be used for example with external tools for counting NGS reads over genes or transcripts. The method 'featuresLength' determines the length of all the features associated to every gene or transctipt and returns the sum. All the exons associated to either a gene or a transcript are non-overlapping. This can be useful to determine read counts into summaries such as expression RPKMs.

Methods

The 'makeGtfFromDb' has to be used in this form:

makeGtfFromDb(object, type, filename)

where:

object: TxDb

- type: character, either 'gene' or 'tx'
- filename: either NULL (a data.frame is returned) or a character containing the path to the file that has to be written

The 'featuresLength' has to be used in this form:

featuresLength(object, type)

where:

object: TxDb

- type: character, either 'gene' or 'tx'

It returns a vector of integers with number of items equal to the number genes or transcripts annotated in the TxDb given containing the widths of the features.

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(TRUE, rep(FALSE, length(isActiveSeq(txdb)) - 1))
chr1geneLengths <- featuresLength(txdb, 'gene')
res <- makeGtfFromDb(txdb, 'gene')
isActiveSeq(txdb) <- rep(TRUE, length(isActiveSeq(txdb)))
```

`matchEnhancers-methods` *GRanges method to match enhancers with putative targets sites*

Description

GRanges method to match enhancers with putative targets sites (either TSS or TF-bound TSS)

Methods

To be used in this form:

matchEnhancers(enhGR, minD=2e4, maxD=2e5, txdb, EG2GS, TFGR=NULL)

where:

enhGR: a GRanges of enhancer sites, such as those provided by the enhancers method

- minD: a positive numeric; the minimum distance between an enhancer and a target genomic region
- maxD: a positive numeric; the maximum distance between an enhancer and a target genomic region
- txdb: an object of class TxDb
- EG2GS: an object of class OrgDb; like org.Mm.eg.db, org.Hs.eg.db (use the exact name of object)
- TFGR: an optional GRanges collecting genomic regions bound from a transcription factor which also binds promoters

This methods relies on a previous identification of enhancer sites, as the one performed by the enhancers compEpiTools method based on H3K4me1 peaks (or alternatively H3K27ac, mediator, ..). Genomic regions which are evaluated as putative target sites of these enhancers are either transcription start sites (TSS) or TSS which are bound at least by a transcription factor (TF) whose binding sites are provided wuth TFGR.

Putative target sites of the provided enhancers are here defined based on the maximum and minimum distance from an enhancer. In addition, no additional TSS (belonging to differe genes, isoforms of the same gene do not count) have to be in between the enhancer and the reference target region.

If a set of TF bound regions is also provided, which is supposed to contain binding sites at the level of promoters, this method returns a list with 5 items:

XmP: a GRanges with location of TSS with no enhancers, based on the distance contraints

- EmP.E: a GRanges of TF-unbound enhancers putatively associated to the TF-bound promoters EmP.mP
- EmP.mP: a GRanges of TF-bound promoters putatively associated to the TF-unbound enhancer sites EmP.E
- mEmP.mE: a GRanges of TF-bound enhancers putatively associated to the TF-bound promoter EmP.mP
- mEmP.mP: a GRanges of TF-bound promoters putatively associated to the TF-bound enhancer sites EmP.E

otherwise this method returns a list with 3 items

XP: a GRanges with location of TSS with no enhancers, based on the distance contraints

- EP.E: a GRanges of enhancers putatively associated to the promoters EP.P
- EP.P: a GRanges of promoters putatively associated to the enhancer sites EP.E

Examples

```
require(org.Mm.eg.db)
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
# loading H3K4me1 peaks as a GRanges object
  # built based on the BED file from the GEO GSM1234488 sample
  # limited to chr19:3200000-4000000
  H3K4me1GR <- system.file("extdata", "H3K4me1GR.Rda", package="compEpiTools")
  load(H3K4me1GR)
enh <- enhancers(H3K4me1GR, txdb)
m.enh <- matchEnhancers(enhGR=enh, minD=2e4, maxD=2e5,
 txdb=txdb, EG2GS=org.Mm.eg.db)
```

---

overlapOfGRanges-methods

visualization of GRanges overlap

Description

given a list of GRanges, all pair-wise overlap are evaluated and the percentage of overlapping ranges is visualized in a heatmap

Methods

To be used in this form:

overlapOfGRanges(GRlist, plot=TRUE)

where:

GRlist: a list of GRanges

- plot: logical

This function uses countOverlaps to count the number of shared ranges for all pairs of GRanges in the input list. The result is returned as an heatmap, white to beige to red, corresponding to increased overlap. The percentage overlap is added on each heatmap cell. For each GRanges on the column of the heatmap (GRC) it represents the percentage of each GRanges in the heatmap rows which is overlapping with GRC.

Examples

```
starts <- seq(100, 500, length.out=5)
gr1 <- GRanges(Rle('chr1'),
 ranges=IRanges(start=starts, end=starts+100))
starts <- seq(300, 700, length.out=5)
```

```
gr2 <- GRanges(Rle('chr1'),
 ranges=IRanges(start=starts+50, end=starts+120))
overlapOfGRanges(GRlist=list(gr1, gr2), plot=FALSE)
```

---

palette2d *build a two dimensional color palette*

Description

build a two dimensional color palette

Usage

```
palette2d(n, k, col1='white', col2='orange', col3='red')
```

Arguments

n	numeric
k	numeric
col1	character; a color, the lower limit of the color palette
col2	character; a color, the mid point of the color palette
col3	character; a color, the upper limit of the color palette

Details

A bidimensional color palette is built. First a set1 of round(n/2) colors ranging from col1 and col2 is defined. Then a set2 of round(n/2) colors ranging from col2 and col3 is defined. Set1 and set2 are concatenated, as the first row of a k X n matrix M. Finally, for each column i of M, k colors ranging from white to M[1,i] are defined.

It is used in the GRheatmapPlot function to have a color palette simultaneously representing the intensity of an event (n) and its significance (k). See the example.

Value

A matrix of color codes, as described in details

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
res <- palette2d(50, 10)
plot(rep(1:ncol(res),each=nrow(res)), rep(1:nrow(res),times=ncol(res)), col=res,
 pch=20, xlab='intensity', ylab='significance')
```

plotStallingIndex *Stalling Index plots*

Description

generates the plot from the output of the stallingIndex function. The plot has 3 panels: Stalling Index, TSS and gene body

Usage

```
plotStallingIndex(matlist, xlimlist=NULL,
colors=rainbow(length(matlist)))
```

Arguments

matlist	List of matrices; each matrix must have a TSS, GB and SI column, as in the output of the function stallingIndex
xlimlist	List of numeric vectors (optional); ranges for the x axis of the three plots. The list must have 3 elements named 'SI', 'TSS' and 'GB'. Default:NULL
colors	array; names of the colors used for the lines in the plot. Default:rainbow palette

Details

Generates a 3-panel plot for the Stalling Index data (Stalling Index, TSS, and gene body).

Value

A plot

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
require(org.Mm.eg.db)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(rep(FALSE,18), TRUE, rep(FALSE, length(isActiveSeq(txdb))-19))
# pointing to Pol2 BAM file
# BAM file from the GEO GSM1234478 sample, limited to chr19:3200000-4000000
Pol2bam <- system.file("extdata", "Pol2.bam", package="compEpiTools")
# loading Pol2 peaks as a GRanges object
# built based on the BED file from the GEO GSM1234478 sample
# limited to chr19:3200000-4000000
Pol2GR <- system.file("extdata", "Pol2GR.Rda", package="compEpiTools")
load(Pol2GR)
egs <- distanceFromTSS(Pol2GR, txdb=txdb)
```

```
egs <- unique(egs$nearest_gene_id)
SI_matrix <- stallingIndex(BAMlist=list(Pol2bam), peakGRlist=list(Pol2GR),
  genesList=list(egs), transcriptDB=txdb, countMode='gene')
plotStallingIndex(SI_matrix)
restoreSeqlevels(txdb)
```

`simplifyGOterms` *simplify a list of GO terms*

Description

simplify a list of GeneOntology terms based on the list of genes assigned to each GO term

Usage

```
simplifyGOterms(goterms, maxOverlap= 0.8, ontology, go2allEGs)
```

Arguments

`goterms`	character; a vector of GO ids
`maxOverlap`	numeric in (0,1) see details
`ontology`	character; one of BP, MF or CC
`go2allEGs`	the species specific assignment of each GO term to EntrezGene ids

Details

Given a pair of parent and child GO terms, and the Entrez Gene ids (genes sets) which can be assigned to them, the parent GO term is defined as redundant if the two gene sets overlap more than 100*maxOverlap percent of the parent genes. Given a set goterms, this rule is applied to discard redundant parent terms while keeping the corresponding children terms.

Value

A subset of the (possibly unaltered) input goterms

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(org.Mm.eg.db)
simplifyGOterms(goterms=c('GO:0002320','GO:0002244'), maxOverlap= 0.1, ontology='BP', go2allEGs= org.Mm.egGO2ALL
```

`stallingIndex` *returns a list with average read count on TSS, gene body, and stalling index for a number of samples*

Description

based on a TxDb, and a list of Pol2 ChIP-seq samples with a GRange corresponding to the peak call, returns a list whose elements contain the stalling index and average read count on TSS and gene body for a user-defined list of genes

Usage

```
stallingIndex(BAMlist,inputList=NULL, peakGRlist, peakGB=FALSE,
  genesList, transcriptDB, countMode="transcript", upstream=300,
  downstream=300, cutoff=600, elongationOffset=0)
```

Arguments

`BAMlist`	List of characters; paths to BAM files containing ChIP-seq aligned reads
`inputList`	List of characters (optional); paths to BAM files containing ChIP-seq aligned reads of inputs relative to the ChIP-seq samples in BAMlist
`peakGRlist`	List of GRanges; Pol2 peaks relative to the ChIP-seq samples in BAMlist. Only genes with a peak on the TSS will be used for counting
`peakGB`	logical; whether or not to consider only genes which have also a Pol2 peak on the gene body. Deafult=FALSE
`genesList`	List of characters; Entrez gene IDs defining the genes where the stalling index will be computed
`transcriptDB`	An object of class transcriptDb
`countMode`	either 'transcript' or 'average' or 'gene'. Genomic units used to count reads: 'transcript' considers all transcripts separately, 'average' averages over different isoforms of a gene, 'gene' considers the longest isoform only. Default='transcript'
`upstream`	numeric; upstream end of the TSS interval where reads will be counted. Default=300
`downstream`	numeric; downstream end of the TSS interval where reads will be counted. Default=300
`cutoff`	numeric; minimum length required for a gene to be considered for the counts. Default=600
`elongationOffset`	numeric; downstream end of the gene body interval where reads will be counted. Default=0

Details

Given a set of Pol2 ChIP-seq samples, computes the average base coverage on TSS and gene body and their ratio (Pol2 stalling index) computed over specific sets of genes for a list of samples.

For each sample, reads will be counted only for genes contained in the corresponding element of the genesList, and only for those genes having a Pol2 peak (given in peakGRlist as list of GRanges) on the TSS region (defined through upstream and downstream). If peakGB is set to TRUE, a Pol2 peak on the body of a gene will be required. The lists of genes peakGRlist must be provided in terms of entrezIDs.

Reads can be counted on individual gene isoforms (countMode='transcript'), averaging over isoforms (countMode='average') or taking the longest isoform for each gene (countMode='gene').

The read counts will be performed on the intervals [TSS-upstream, TSS+downstream] (TSS) and [TSS+downstream, TES+elongationOffset] (genebody), where upstream, downstream, elongationOffset are user-defined parameters. Moreover, only genes having a length bigger than the user-defined parameter cutoff will be considered.

BAM files have to be associated to the corresponding index .bai files. Please refer to the documentation of samtools on how to create them.

Value

A list of matrices, each containing TSS counts, gene body counts and stalling index for each sample

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
require(org.Mm.eg.db)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
isActiveSeq(txdb) <- c(rep(FALSE,18), TRUE, rep(FALSE, length(isActiveSeq(txdb))-19))
# pointing to Pol2 BAM file
# BAM file from the GEO GSM1234478 sample, limited to chr19:3200000-4000000
Pol2bam <- system.file("extdata", "Pol2.bam", package="compEpiTools")
# loading Pol2 peaks as a GRanges object
# built based on the BED file from the GEO GSM1234478 sample
# limited to chr19:3200000-4000000
Pol2GR <- system.file("extdata", "Pol2GR.Rda", package="compEpiTools")
load(Pol2GR)
egs <- distanceFromTSS(Pol2GR, txdb=txdb)
egs <- unique(egs$nearest_gene_id)
SI_matrix <- stallingIndex(BAMlist=list(Pol2bam), peakGRlist=list(Pol2GR),
  genesList=list(egs), transcriptDB=txdb, countMode='gene')
restoreSeqlevels(txdb)
```

topGOres *determines GeneOntology (GO) enriched terms for a set of Entrez gene ids*

Description

determines GeneOntology (GO) enriched terms for a set of Entrez gene ids

Usage

```
topGOres(ids, ontology='BP', Pthr=1e-05, maxN=5000, minN=5,
orgdb, allEG=keys(orgdb),showTerms=NULL)
```

Arguments

ids	can be either a vector or a list of human or mouse EntrezGene ids
ontology	one of the three GO ontologies: BP (Biological Processes), CC (Cellular Components) or MF (Molecular Functions)
Pthr	numeric [0,1]; the p-value for an enrichment to be considered significant
maxN	numeric; only GO terms with a total up to maxN genes annotated on the genome are considered
minN	numeric; only GO terms with a minimum of minN genes annotated on the genome are considered
orgdb	An object of class OrgDb; either org.Hs.eg.db, org.Mm.eg.db or org.Dm.eg.db
allEG	character; the reference universe of EntrezGene ids, by default all of them
showTerms	numeric: the number of GO terms to plot; NULL: no plotting

Details

Determines GeneOntology (GO) enriched terms for a set of Entrez gene ids. Based on the topGO Bioconductor library. Both maxN and minN refer to the number of genes annotated in the reference genome for a given GO term (indipendently from the ids that the enrichment is being evaluated for). This can be used for excluding GOterm very generic or very specific, since these would mostly be ignored in the final output. This would also save time in the analysis and decrease the severity of the multiple testing issue.

Value

A matrix containing enriched GO terms ranked by significance is returned, with the following columns:

GO.ID	GO id
Term	text description of the GO id
Annotated	total number of genes annotated with the considered GOterm
Significant	number of genes in ids for the specific GOterms

Expected	expected number of GOterms genes in ids in case of random enrichment
Classic	pvalue for the enrichment as reported from the topGO package
Genes	Gene ids of significantly annotated genes for each specific GOterms

References

http://genomics.iit.it/groups/computational-epigenomics.html

See Also

topGOdata in the topGO Bioconductor package

Examples

```
require(org.Mm.eg.db)
egs <- keys(org.Mm.egACCNUM)[1:50]
topGOres(ids=egs, Pthr=0.006, maxN=5000, minN=5,
 orgdb=org.Mm.eg.db, allEG=keys(org.Mm.eg.db)[1:5000])
```

TSS *based on a TxDb returns a GRanges with the TSS positions for all transcripts*

Description

based on a TxDb returns a GRanges of width 1 with the TSS positions for all transcripts

Usage

```
TSS(txdb)
```

Arguments

txdb	An object of class TxDb

Details

based on a TxDb returns a GRanges of width 1 with the TSS positions for all transcripts

Value

A GRanges

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
res <- TSS(txdb)
res[1:2]
```

---

ucsc2GRanges *Convert UCSC-formatted genomic positions into a GRanges*

Description

Convert UCSC-formatted genomic positions to a GRanges object

Usage

```
ucsc2GRanges(ucscPositions)
```

Arguments

ucscPositions	character; a vector of UCSC formatted genomic positions.

Details

UCSC formatted genomic positions such as chr5:10000-35000 are converted into GRanges format.

Value

a GRanges with chr, start and end derived from the UCSC format

References

http://genomics.iit.it/groups/computational-epigenomics.html

Examples

```
ucsc2GRanges(c('chr1:100-500','chrX:20-1000'))
```

`unionMaxScore-methods` *GRanges method to perform union of peaks keeping the score of the most significant peak*

Description

the union between two GRanges associated to scores is performed, and the most significant score for the overlapping regions is returned

Methods

To be used in this form:

unionMaxScore(gr1, gr2, score1=mcols(gr1)[,1], score2=mcols(gr2)[,1])

where:

gr1: GRanges

- gr2: GRanges
- score1: numeric; the scores (or any kind of score) associated to gr1
- score2: numeric; the scores (or any kind of score) associated to gr2

The method returns the union GRanges as in union(gr1, g2). For each range, if it resulted from overlapping ranges, the maximum score for those ranges is returned. In case pvalues (P) are adopted, they should be transformed in a score with something like -log10(P)

Examples

```
gr1 <- GRanges(Rle(c('chr1','chr1')),
 ranges=IRanges(start=c(100,200), end=c(150,250)), pvalue=c(200, 100))
gr2 <- GRanges(Rle(c('chr1','chr1')),
 ranges=IRanges(start=c(80,400), end=c(300,500)), pvalue=c(50, 150))
unionMaxScore(gr1, gr2)
```

Index

∗ **methods**
countOverlapsInBins-methods, 4
distanceFromTSS-methods, 5
enhancers-methods, 6
getPromoterClass, 9
GR2fasta-methods, 10
GRanges2ucsc-methods, 10
GRangesInPromoters-methods, 11
GRannotate-methods, 11
GRannotateSimple-methods, 13
GRbaseCoverage-methods, 14
GRcoverage-methods, 15
GRcoverageSummit-methods, 16
GRenrichment-methods, 16
GRmidpoint-methods, 17
GRsetwidth-methods, 18
makeGtfFromDb-methods, 22
matchEnhancers-methods, 23
overlapOfGRanges-methods, 24
unionMaxScore-methods, 33
∗ **package**
compEpiTools-package, 2

compEpiTools *(*compEpiTools-package*)*, 2
compEpiTools-package, 2
countOverlapsInBins
*(*countOverlapsInBins-methods*)*, 4
countOverlapsInBins,GRanges-method
*(*countOverlapsInBins-methods*)*, 4
countOverlapsInBins-methods, 4

distanceFromTSS
*(*distanceFromTSS-methods*)*, 5
distanceFromTSS,GRanges-method
*(*distanceFromTSS-methods*)*, 5
distanceFromTSS-methods, 5

enhancers *(*enhancers-methods*)*, 6
enhancers,GRanges-method
*(*enhancers-methods*)*, 6
enhancers-methods, 6

featuresLength *(*makeGtfFromDb-methods*)*, 22
featuresLength,TxDb-method
*(*makeGtfFromDb-methods*)*, 22
featuresLength-methods
*(*makeGtfFromDb-methods*)*, 22
findLncRNA, 7

getPromoterClass, 9
getPromoterClass,TxDb-method
*(*getPromoterClass*)*, 9
getPromoterClass-methods
*(*getPromoterClass*)*, 9
GR2fasta *(*GR2fasta-methods*)*, 10
GR2fasta,GRanges-method
*(*GR2fasta-methods*)*, 10
GR2fasta-methods, 10
GRanges, 9
GRanges2ucsc *(*GRanges2ucsc-methods*)*, 10
GRanges2ucsc,GRanges-method
*(*GRanges2ucsc-methods*)*, 10
GRanges2ucsc-methods, 10
GRangesInPromoters
*(*GRangesInPromoters-methods*)*, 11
GRangesInPromoters,GRanges-method
*(*GRangesInPromoters-methods*)*, 11
GRangesInPromoters-methods, 11
GRannotate *(*GRannotate-methods*)*, 11
GRannotate,GRanges-method
*(*GRannotate-methods*)*, 11
GRannotate-methods, 11
GRannotateSimple
*(*GRannotateSimple-methods*)*, 13

GRannotateSimple,GRanges-method
(GRannotateSimple-methods), 13
GRannotateSimple-methods, 13
GRbaseCoverage
(GRbaseCoverage-methods), 14
GRbaseCoverage,GRanges-method
(GRbaseCoverage-methods), 14
GRbaseCoverage-methods, 14
GRcoverage (GRcoverage-methods), 15
GRcoverage,GRanges-method
(GRcoverage-methods), 15
GRcoverage-methods, 15
GRcoverageInbins (GRcoverage-methods),
15
GRcoverageInbins,GRanges-method
(GRcoverage-methods), 15
GRcoverageInbins-methods
(GRcoverage-methods), 15
GRcoverageSummit
(GRcoverageSummit-methods), 16
GRcoverageSummit,GRanges-method
(GRcoverageSummit-methods), 16
GRcoverageSummit-methods, 16
GRenrichment (GRenrichment-methods), 16
GRenrichment,GRanges-method
(GRenrichment-methods), 16
GRenrichment-methods, 16
GRmidpoint (GRmidpoint-methods), 17
GRmidpoint,GRanges-method
(GRmidpoint-methods), 17
GRmidpoint-methods, 17
GRsetwidth (GRsetwidth-methods), 18
GRsetwidth,GRanges-method
(GRsetwidth-methods), 18
GRsetwidth-methods, 18

heatmap.2, *21*
heatmapData, 18
heatmapPlot, 20

makeGtfFromDb (makeGtfFromDb-methods),
22
makeGtfFromDb,TxDb-method
(makeGtfFromDb-methods), 22
makeGtfFromDb-methods, 22
matchEnhancers
(matchEnhancers-methods), 23
matchEnhancers,GRanges-method
(matchEnhancers-methods), 23
matchEnhancers-methods, 23

overlapOfGRanges
(overlapOfGRanges-methods), 24
overlapOfGRanges,GRanges-method
(overlapOfGRanges-methods), 24
overlapOfGRanges-methods, 24

palette2d, 25
plotStallingIndex, 26

simplifyGOterms, 27
stallingIndex, 28

topGOres, 30
TSS, 31

ucsc2GRanges, 32
unionMaxScore (unionMaxScore-methods),
33
unionMaxScore,GRanges-method
(unionMaxScore-methods), 33
unionMaxScore-methods, 33