Package 'genomation'

October 9, 2015

Type Package

Title Summary, annotation and visualization of genomic data

Version 1.0.0

Author Altuna Akalin, Vedran Franke

Maintainer Altuna Akalin <aakalin@gmail.com>, Vedran Franke <vedran.franke@gmail.com>

Description A package for summary and annotation of genomic intervals. Users can visualize and quantify genomic intervals over pre-defined functional regions, such as promoters, exons, introns, etc. The genomic intervals represent regions with a defined chromosome position, which may be associated with a score, such as aligned reads from HT-seq experiments, TF binding sites, methylation scores, etc. The package can use any tabular genomic feature data as long as it has minimal information on the locations of genomic intervals. In addition, It can use BAM or BigWig files as input.

License Artistic-2.0

LazyLoad yes

VignetteBuilder knitr

biocViews Annotation, Sequencing, Visualization, CpGIsland

Encoding latin1

Depends R (>= 3.0.0),grid

Imports GenomicRanges, GenomicAlignments, IRanges, Rsamtools, data.table, plyr, reshape2, ggplot2, methods, rtracklayer, gridBase, impute

Suggests RUnit, knitr, RColorBrewer, genomationData, BiocGenerics, rmarkdown, knitrBootstrap

Collate 'documentData.R' 'genomation-classes.R' 'getRandomEnrichment.R' 'findFeatureComb.R' 'plotMatrix.R' 'randomizeFeature.R' 'readAnnotate.R' 'readData.R' 'scoreMatrix.R' 'scoreMatrixBin.R' 'scoreMatrixList.R' 'test_genomation_package.R'

NeedsCompilation no

R topics documented:

`annotateWithFeature` *Function to annotate given GRanges object with a given genomic feature*

Description

Function to annotate given GRanges object with a given genomic feature

Usage

```
annotateWithFeature(target, feature, strand = FALSE, extend = 0,
  feature.name = NULL, intersect.chr = FALSE)

## S4 method for signature 'GRanges,GRanges'
annotateWithFeature(target, feature,
  strand = FALSE, extend = 0, feature.name = NULL,
  intersect.chr = FALSE)
```

Arguments

`target`	a GRanges object storing chromosome locations to be annotated
`feature`	a GRanges object storing chromosome locations of a feature (can be CpG islands, ChIP-seq peaks, etc)
`strand`	If set to TRUE, annotation features and target features will be overlapped based on strand (def:FAULT)
`extend`	specifiying a positive value will extend the feature on both sides as much as `extend`
`feature.name`	name of the annotation feature. For example: H3K4me1,CpGisland etc. by default the name is taken from the given variable
`intersect.chr`	boolean, whether to select only chromosomes that are common to feature and target. FALSE by default

Value

returns an `AnnotationByFeature` object

Examples

```
data(cpgi)
data(promoters)
annot = annotateWithFeature(cpgi, promoters)
```

annotateWithFeatureFlank *Function to annotate a given GRanges object with promoter,exon,intron & intergenic values*

Description

Function to annotate a given GRanges object with promoter,exon,intron & intergenic values

Usage

```
annotateWithFeatureFlank(target, feature, flank, feature.name = NULL,
  flank.name = "flank", strand = FALSE, intersect.chr = FALSE)

## S4 method for signature 'GRanges,GRanges,GRanges'
annotateWithFeatureFlank(target, feature,
  flank, feature.name = NULL, flank.name = "flank", strand = FALSE,
  intersect.chr = FALSE)
```

Arguments

target	a granges object storing chromosome locations to be annotated
feature	a granges object storing chromosome locations of a feature (can be CpG islands, ChIP-seq peaks, etc)
flank	a granges object storing chromosome locations of the flanks of the feature
feature.name	string for the name of the feature
flank.name	string for the name of the flanks
strand	If set to TRUE, annotation features and target features will be overlapped based on strand (def:FAULT)
intersect.chr	boolean, whether to select only chromosomes that are common to feature and target. FALSE by default

Value

returns an AnnotationByFeature object

Examples

```
data(cpgi)
data(cage)
cpgi.flanks = getFlanks(cpgi)
flank.annot = annotateWithFeatureFlank(cage, cpgi, cpgi.flanks)
```

annotateWithGeneParts *Annotate given object with promoter, exon, intron and intergenic regions*

Description

The function annotates GRangesList or GRanges object as overlapping with promoter,exon,intron or intergenic regions.

Usage

```
annotateWithGeneParts(target, feature, strand = FALSE,
  intersect.chr = FALSE)

## S4 method for signature 'GRanges,GRangesList'
annotateWithGeneParts(target, feature,
  strand = FALSE, intersect.chr = FALSE)

## S4 method for signature 'GRangesList,GRangesList'
annotateWithGeneParts(target, feature,
  strand = FALSE, intersect.chr = FALSE)
```

Arguments

target	GRanges or GRangesList object storing chromosome locations to be annotated (e.g. chipseq peaks)
feature	GRangesList object containing GRanges object for promoter, exons, introns and transcription start sites, or simply output of readTranscriptFeatures function
strand	If set to TRUE, annotation features and target features will be overlapped based on strand (def:FALSE)
intersect.chr	boolean, whether to select only chromosomes that are common to feature and target. FALSE by default

Value

AnnotationByGeneParts object or a list of AnnotationByGeneParts objects if target is a GRangesList object.

Examples

```
# data(cage)
# bed.file = system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
# gene.parts = readTranscriptFeatures(bed.file)
# cage.annot = annotateWithGeneParts(cage, gene.parts, intersect.chr=TRUE)
```

`AnnotationByFeature-class`

An S4 class that information on overlap of target features with annotation features

---

Description

This object is desgined to hold statistics and information about genomic feature overlaps

Slots

`members` a matrix showing overlap of target features with annotation genomic features

`annotation` a named vector of percentages

`precedence` a named vector of percentages

`num.annotation` vector

`num.precedence` vector

`no.of.OlapFeat` vector

`perc.of.OlapFeat` vector

`AnnotationByGeneParts-class`

An S4 class that information on overlap of target features with annotation features

---

Description

This object is desgined to hold statistics and information about genomic feature overlaps

Slots

`members` a matrix showing overlap of target features with annotation genomic features

`annotation` a named vector of percentages

`precedence` a named vector of percentages

`num.annotation` vector

`num.precedence` vector

`no.of.OlapFeat` vector

`perc.of.OlapFeat` vector

dist.to.TSS a data frame showing distances to TSS and gene/TSS names and strand

binMatrix *Bins the columns of a matrix using a user provided function*

Description

Bins the columns of a matrix using a user provided function

Usage

```
binMatrix(x, bin.num = NULL, fun = "mean")

## S4 method for signature 'ScoreMatrix'
binMatrix(x, bin.num = NULL, fun = "mean")

## S4 method for signature 'ScoreMatrixList'
binMatrix(x, bin.num = NULL, fun = "mean")
```

Arguments

x	ScoreMatrix or a ScoreMatrixList object
bin.num	integer number of bins in the final matrix
fun	character vector or an anonymous function that will be used for binning

Value

ScoreMatrix or ScoreMatrixList object

Examples

```
# binning the columns in a ScoreMatrix object
#library(GenomicRanges)
#target = GRanges(rep(c(1,2),each=7), IRanges(rep(c(1,1,2,3,7,8,9), times=2), width=5),
#weight = rep(c(1,2),each=7),
#strand=c('-', '-', '-', '-', '+', '-', '+', '-', '-', '-', '-', '-', '-', '+'))
#windows = GRanges(rep(c(1,2),each=2), IRanges(rep(c(1,2), times=2), width=5),
#strand=c('-','+','-','+'))
#sm = ScoreMatrix(target, windows)
#bin = binMatrix(sm, bin.num=2)
```

---

cage *Example CAGE data set.*

Description

Location and tag per million values for CAGE TSS clusters on chr21 and chr22 of human genome (hg19 assembly). The clusters are dowloaded from ENCODE project downloads for NHEK cells.

Format

GRanges object

---

calculateOverlapSignificance *function that calculates the significance of overlaps of two sets of features using randomization*

Description

This function calculates the significance of overlaps of two sets of features using randomization. #' It returns a distributon of overlaps of a target set with a given randomized feature set. The randomization can be constrained by supplied arguments. The function is still in Beta mode - the regions can overlap excluded regions, and the randomized regions are not disjoint. Please take care that the excluded and included regions are not too strict when compared to the total width of the ranges.

Usage

```
calculateOverlapSignificance(target, feature, chrom.sizes = NULL,
  stranded = TRUE, keep.strand.prop = TRUE, keep.chrom = TRUE,
  exclude = NULL, include = NULL, seed = NULL, nrand = 1)

## S4 method for signature 'GRanges,GRanges'
calculateOverlapSignificance(target, feature,
  chrom.sizes = NULL, stranded = TRUE, keep.strand.prop = TRUE,
  keep.chrom = TRUE, exclude = NULL, include = NULL, seed = NULL,
  nrand = 1)
```

Arguments

target	a GRanges object for which the overlap needs to be calculates
feature	a GRanges object to be randomized
chrom.sizes	sizes of chromosomes as a named vector (names are chromsomes names and elements of the vectors are lengths). , if not given sizes in GRanges object will be used if no sizes there the end of each chr will be the end last feature on each chr

stranded	if FALSE, all of the returned features will be strandless (will have "*" in the strand slot)
keep.strand.prop	If TRUE strands will have the same proportion as the features
keep.chrom	If TRUE, number of features and randomized features for a chromosome will match. Currently seeting this to FALSE is not supported.
exclude	A GRanges object where no randomized feature should overlap, can be gaps or unmappable regions in the genome as an example.
include	A GRanges object which defines the boundaries of randomized features
seed	random number generator seed
nrand	number of randomizations (default:1)

Value

returns a GRanges object which is randomized version of the feature

---

`convertBed2Exons` *convert a data frame read-in from a bed file to a GRanges object for exons*

Description

convert a data frame read-in from a bed file to a GRanges object for exons

Usage

```
convertBed2Exons(bed.df)

## S4 method for signature 'data.frame'
convertBed2Exons(bed.df)
```

Arguments

bed.df	a data.frame where column order and content resembles a bed file with 12 columns

Value

GRanges object

Note

one bed track per file is only accepted, the bed files with multiple tracks will cause en error

Examples

```
file = system.file('extdata/chr21.refseq.hg19.bed', package='genomation')
bed12 = read.table(file)
exons = convertBed2Exons(bed12)
head(exons)
```

---

convertBed2Introns *convert a data frame read-in from a bed file to a GRanges object for introns*

Description

convert a data frame read-in from a bed file to a GRanges object for introns

Usage

```
convertBed2Introns(bed.df)

## S4 method for signature 'data.frame'
convertBed2Introns(bed.df)
```

Arguments

bed.df	a data.frame where column order and content resembles a bed file with 12 columns

Value

GRanges object

Note

one bed track per file is only accepted, the bed files with multiple tracks will cause en error

Examples

```
file = system.file('extdata/chr21.refseq.hg19.bed', package='genomation')
bed12 = read.table(file)
introns = convertBed2Introns(bed12)
head(introns)
```

convertBedDf *convert a data frame read-in from a bed file to a GRanges object*

Description

convert a data frame read-in from a bed file to a GRanges object

Usage

```
convertBedDf(bed)

## S4 method for signature 'data.frame'
convertBedDf(bed)
```

Arguments

bed	a data.frame where column order and content resembles a bed file with 12 columns

Value

GRanges object

Note

one bed track per file is only accepted, the bed files with multiple tracks will cause en error

bed files are expected not to have header lines

cpgi *Example CpG island data set.*

Description

CpG islands of hg19 assembly of human genome on chr21 and chr22. Downloaded from UCSC genome browser.

Format

GRanges object

`findFeatureComb` *Find combitations of genomic features*

Description

Provided a GRangesList, finds the combinations of sets of ranges. It is mostly used to look at the combinatorics of transcription factor binding. The function works by, firstly, constructing a union of all ranges in the list, which are then designated by the combinatorics of overlap with the original sets. A caveat of this approach is that the number of possible combinations increases exponentially, so we would advise you to use it with up to 6 data sets. If you wish to take a look at a greater number of factors, methods like self organizing maps or ChromHMM might be more appropriate.

Usage

```
findFeatureComb(gl, width=0, use.names=FALSE, collapse.char=':')

## S4 method for signature 'GRangesList'
findFeatureComb(gl, width = 0, use.names = FALSE,
  collapse.char = ":")
```

Arguments

`gl`	a `GRangesList` object, containing ranges for which represent regions enriched for transcription factor binding
`width`	`integer` is the requested width of each enriched region. If 0 the ranges are not resized, if a positive integer, the width of all ranges is set to that number. Ranges are resized relative to the center of original ranges.
`use.names`	a boolean which tells the function whether to return the resulting ranges with a numeric vector which designates each class (the default), or to construct the names of each class using the names from the GRangesList
`collapse.char`	a character which will be used to separate the class names if use.names=TRUE. The default is ':'

Value

a GRanges object

Examples

```
library(GenomicRanges)
g = GRanges(paste('chr',rep(1:2, each=3), sep=''), IRanges(rep(c(1,5,9), times=2), width=3))
gl = GRangesList(g1=g, g2=g[2:5], g3=g[3:4])
findFeatureComb(gl)
findFeatureComb(gl, use.names=TRUE)
```

genes *Example RefSeq genes data set.*

Description

RefSeq genes of hg19 assembly of human genome on chr21 and chr22. Downloaded from UCSC genome browser.

Format

GRanges object

getAssociationWithTSS *Get distance to nearest TSS and gene id from AnnotationByGeneParts*

Description

This accessor function gets the nearest TSS, its distance to target feature, strand and name of TSS/gene from AnnotationByGeneParts object

Usage

```
getAssociationWithTSS(x)

## S4 method for signature 'AnnotationByGeneParts'
getAssociationWithTSS(x)
```

Arguments

x	a AnnotationByGeneParts object

Value

RETURNS a data.frame containing row number of the target features, distance of target to nearest TSS, TSS/Gene name, TSS strand

Examples

```
data(cage)
bed.file = system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
gene.parts = readTranscriptFeatures(bed.file)
cage.annot = annotateWithGeneParts(cage, gene.parts, intersect.chr=TRUE)
head(getAssociationWithTSS(cage.annot))
```

getFeatsWithTargetsStats *Get the percentage/count of annotation features overlapping with target features from AnnotationByFeature*

Description

This function retrieves percentage/number of annotation features overlapping with targets. For example, if AnnotationByFeature object is containing statistics of differentially methylated regions overlapping with gene annotation. This function will return number/percentage of introns,exons and promoters overlapping with differentially methylated regions.

Usage

```
getFeatsWithTargetsStats(x,percentage=TRUE)

## S4 method for signature 'AnnotationByFeature'
getFeatsWithTargetsStats(x, percentage = TRUE)
```

Arguments

x	a AnnotationByFeature object
percentage	TRUE|FALSE. If TRUE percentage of annotation features will be returned. If FALSE, number of annotation features will be returned

Value

RETURNS a vector of percentages or counts showing quantity of annotation features overlapping with target features

Examples

```
data(cage)
bed.file=system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
gene.parts = readTranscriptFeatures(bed.file)
cage.annot = annotateWithGeneParts(cage, gene.parts, intersect.chr=TRUE)
getFeatsWithTargetsStats(cage.annot)
```

getFlanks *Function to get upstream and downstream adjecent regions to a genomic feature such as CpG islands*

Description

Function to get upstream and downstream adjecent regions to a genomic feature such as CpG islands

Usage

```
getFlanks(grange,flank=2000,clean=TRUE)

## S4 method for signature 'GRanges'
getFlanks(grange, flank = 2000, clean = TRUE)
```

Arguments

grange	GRanges object for the feature
flank	number of basepairs for the flanking regions
clean	If set to TRUE, flanks overlapping with other main features will be trimmed, and overlapping flanks will be removed. This will remove multiple counts when other features overlap with flanks

Value

GRanges object for flanking regions

Examples

```
data(cpgi)
cpgi.flanks = getFlanks(cpgi)
head(cpgi.flanks)
```

getMembers *Get the membership slot of AnnotationByFeature*

Description

Membership slot defines the overlap of target features with annotation features For example, if a target feature overlaps with an exon

Usage

```
getMembers(x)

## S4 method for signature 'AnnotationByFeature'
getMembers(x)
```

Arguments

x	a AnnotationByFeature object

Value

matrix showing overlap of target features with annotation features. 1 for overlap, 0 for non-overlap

---

getRandomEnrichment *get enrichment based on randomized feature overlap*

Description

This function measures the association between two genomic features by randomizing one feature and counting the overlaps in randomized sets. That is to say, query feature will be randomly distributed over the genome (constrained by provided options), and the overlap of target with these randomized features will be measured.

Usage

```
getRandomEnrichment(target, query, randomizations = 1000, rand.set = NULL,
  ...)

## S4 method for signature 'GRanges,GRanges'
getRandomEnrichment(target, query,
  randomizations = 1000, rand.set = NULL, ...)
```

Arguments

target	a GRanges object to be overlapped with query
query	a GRanges object that will be randomly placed across the genome and overlap of these random regions with target will be the background distribution of association between target and query.
randomizations	number of times the features to be shuffled
rand.set	instead of randomly placing features in query one can supply an already shuffled set of query genomic features.
...	other parameters to be passed to randomizeFeature function. These parameters ccontrol how randomization is done.

Value

returns a RandomEnrichment object

See Also

randomizeFeature

Examples

```
# data(cage)
# data(cpgi)

# enr = getRandomEnrichment(cage, cpgi, randomizations=50)
```

`getTargetAnnotationStats` *Get the percentage of target features overlapping with annotation from AnnotationByFeature*

Description

This function retrieves percentage/number of target features overlapping with annotation

Usage

```
getTargetAnnotationStats(x,percentage=TRUE,precedence=TRUE)

## S4 method for signature 'AnnotationByFeature'
getTargetAnnotationStats(x, percentage = TRUE,
  precedence = TRUE)
```

Arguments

`x`	a `AnnotationByFeature` object
`percentage`	TRUE|FALSE. If TRUE percentage of target features will be returned. If FALSE, number of target features will be returned
`precedence`	TRUE|FALSE. If TRUE there will be a hierachy of annotation features when calculating numbers (with promoter>exon>intron precedence) That means if a feature overlaps with a promoter it will be counted as promoter overlapping only, or if it is overlapping with a an exon but not a promoter, #' it will be counted as exon overlapping only whether or not it overlaps with an intron.

Value

a vector of percentages or counts showing quantity of target features overlapping with annotation

Examples

```
data(cage)
bed.file=system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
gene.parts = readTranscriptFeatures(bed.file)
cage.annot=annotateWithGeneParts(cage, gene.parts, intersect.chr=TRUE)
getTargetAnnotationStats(cage.annot)
```

gffToGRanges *Converts a gff formated data.frame into a GenomicRanges object. The GenomicRanges object needs to be properly formated for the function to work.*

Description

Converts a gff formated data.frame into a GenomicRanges object. The GenomicRanges object needs to be properly formated for the function to work.

Usage

```
gffToGRanges(gff.file, split.group = FALSE, split.char = ";",
  filter = NULL, zero.based = FALSE)
```

Arguments

gff.file	path to a gff formatted file
split.group	boolean, whether to split the 9th column of the file
split.char	character that is used as a separator of the 9th column. ';' by default
filter	a character designating which elements to retain from the gff file (e.g. exon, CDS, ...)
zero.based	boolean whether the coordinates are 0 or 1 based. 0 is the default

Value

returns a GenomicRanges object

Examples

```
# gff.file = system.file('extdata/chr21.refseq.hg19.gtf', package='genomation')
# gff = gffToGRanges(gff.file, split.group=TRUE)
```

heatMatrix *Draw a heatmap of a given ScoreMatrix object*

Description

The function makes a heatmap out of given ScoreMatrix object. If desired it can use clustering using k-means and plot cluster color codes as a sidebar. In addition, user can define groups of rows using 'group' argument.

Usage

```
heatMatrix(mat, grid = FALSE, col = NULL, xcoords = NULL, group = NULL,
  group.col = NULL, order = FALSE, winsorize = c(0, 100),
  kmeans = FALSE, k = 3, main = "", legend.name = NULL,
  cex.legend = 1, xlab = NULL, cex.main = 1, cex.lab = 1,
  cex.axis = 1, newpage = TRUE)
```

Arguments

mat	a ScoreMatrix object
grid	if TRUE, grid graphics will be used. if FALSE, base graphics will be used on the top level, so users can use par(mfrow) or par(mfcol) prior to calling the function. Default:FALSE
col	a vector of colors, such as the ones created by heat.colors(10). If NULL (which is default), jet color scheme (common in matlab plots) will be used.
xcoords	a vector of numbers showing relative positions of the bases or windows. It must match the number of columns in the ScoreMatrix. Alternatively, it could be a numeric vector of two elements. Such as c(0,100) showing the relative start and end coordinates of the first and last column of the ScoreMatrix object.
group	a list of vectors of row numbers or a factor. This grouping is used for rowside colors of the heatmap. If it is a list, each element of the list must be a vector of row numbers. Names of the elements of the list will be used as names of groups. If group is a factor , it's length must match the number of rows of the matrix, and factor levels will be used as the names of the groups in the plot.
group.col	a vector of color names to be used at the rowside colors if group argument is given or kmeans=TRUE
order	Logical indicating if the rows should be ordered or not (Default:FALSE). If order=TRUE the matrix will be ordered with rowSums(mat) values in descending order. If kmeans=TRUE or group argument is provided, first the groups/clusters will be ordered in descending order of sums of rows then, everything within the clusters will be ordered by sums of rows.
winsorize	Numeric vector of two, defaults to c(0,100). This vector determines the upper and lower percentile values to limit the extreme values. For example, c(0,99) will limit the values to only 99th percentile, everything above the 99 percentile will be equalized to the value of 99th percentile.This is useful for visualization of matrices that have outliers.
kmeans	Logical indicating if kmeans clustering should be done on the rows or not (Default:FALSE).
k	Defaults to 3. It designates the number of clusters to be returned by kmeans clustering.
main	a character string for the plot title
legend.name	a character label plotted next to the legend
cex.legend	A numerical value giving the amount by which legend axis marks should be magnified relative to the default

xlab	label a character string for x-axis of the heatmap
cex.main	A numerical value giving the amount by which plot title should be magnified
cex.lab	A numerical value giving the amount by which axis labels (including 'legend.name') should be magnified relative to the default.
cex.axis	A numerical value giving the amount by which axis marks should be magnified relative to the default
newpage	logical indicating if grid.newpage() function should be invoked if grid=TRUE.

Value

returns kmeans clustering result invisibly, if kmeans=TRUE

Examples

```
# data(cage)
# data(promoters)
# scores1=ScoreMatrix(target=cage,windows=promoters,strand.aware=TRUE,
#                   weight.col="tpm")



# heatMatrix(mat=scores1,legend.name="tpm",winsorize=c(0,99),xlab="region around TSS",
#           xcoords=-1000:1000,
#           cex.legend=0.8,main="CAGE clusters on promoters",cex.lab=1,
#           cex.axis=0.9,grid=FALSE)

# set.seed(1000)
# heatMatrix(mat=scores1,legend.name="tpm",winsorize=c(0,99),xlab="region around TSS",
#         xcoords=-1000:1000,kmeans=TRUE,k=3,
#         cex.legend=0.8,main="CAGE clusters on promoters",cex.lab=1,
#         cex.axis=0.9,grid=FALSE)
```

---

heatMeta *Heatmap for meta-region profiles*

Description

Function calculates meta-profile(s) from a ScoreMatrix or a ScoreMatrixList, then produces a heatmap or a set of stacked heatmaps for meta-region profiles

Usage

```
heatMeta(mat, profile.names = NULL, xcoords = NULL, col = NULL,
  meta.rescale = FALSE, legend.name = NULL, cex.legend = 1, xlab = NULL,
  main = "", cex.lab = 1, cex.axis = 1)
```

Arguments

mat	ScoreMatrix or ScoreMatrixList to be plotted
profile.names	a character vector for names of profiles. If NULL, the names will be taken from names(mat) if mat is a ScoreMatrixList object.
xcoords	a vector of numbers showing relative positions of the bases or windows. It must match the number of columns in the ScoreMatrix For example: if there are 2001 elements in the matrices which are base-pair resolution and they are centered around an anchor point like TSS, the xcoords argument should be -1000:1000. This argument is used to plot accurate x-axis labels for the plots.If NULL it will be equal to 1:ncol(mat).
col	a vector of color pallete. color scheme to be used. If NULL, a version of jet colors will be used.
meta.rescale	if TRUE meta-region profiles are scaled to 0 to 1 range by subracting the min from profiles and dividing them by max-min.
legend.name	a character label plotted next to the legend
cex.legend	A numerical value giving the amount by which legend axis marks should be magnified relative to the default
xlab	label a character string for x-axis
main	a character string for the plot title
cex.lab	A numerical value giving the amount by which axis labels (including 'legend.name') should be magnified relative to the default.
cex.axis	A numerical value giving the amount by which axis marks should be magnified relative to the default

Value

returns meta-profile matrix invisibly.

Examples

```
# data(cage)
# data(promoters)
# scores1=ScoreMatrix(target=cage,windows=promoters,strand.aware=TRUE)
# data(cpgi)
# scores2=ScoreMatrix(target=cpgi,windows=promoters,strand.aware=TRUE)

# x=new("ScoreMatrixList",list(scores1,scores2))
# heatMeta(mat=x,legend.name="fg",cex.legend=0.8,main="fdf",cex.lab=6,
#          cex.axis=0.9)
```

`intersectScoreMatrixList` *Get common rows from all matrices in a ScoreMatrixList object*

Description

Returns a intersection of rows for each matrix in a ScoreMatrixList object. This is done using the rownames of each element in the list.

Usage

```
intersectScoreMatrixList(sml, reorder = FALSE)

## S4 method for signature 'ScoreMatrixList'
intersectScoreMatrixList(sml, reorder = FALSE)
```

Arguments

`sml`	a `ScoreMatrixList` object
`reorder`	if TRUE `ScoreMatrix` objects in the list are sorted based on their common row ids.

Value

`ScoreMatrixList` object

Examples

```
# library(GenomicRanges)
# target = GRanges(rep(c(1,2),each=7),
#                  IRanges(rep(c(1,1,2,3,7,8,9), times=2), width=5),
#                  weight = rep(c(1,2),each=7))

# windows1 = GRanges(rep(c(1,2),each=2),
#                    IRanges(rep(c(1,2), times=2), width=5),
#                    strand=c('-','+','-','+'))
# windows2 = windows1[c(1,3)]
# sml = as(list(ScoreMatrix(target, windows1),
#               ScoreMatrix(target, windows2)), 'ScoreMatrixList')
# sml
# intersectScoreMatrixList(sml)
```

multiHeatMatrix *Draw multiple heatmaps from a ScoreMatrixList object*

Description

The function plots multiple heatmaps for a ScoreMatrixList object side by side. Each matrix can have different color schemes but it is essential that each matrix is obtained from same regions or neighbouring regions.

Usage

```
multiHeatMatrix(sml, grid = TRUE, col = NULL, xcoords = NULL,
  group = NULL, group.col = NULL, order = FALSE, winsorize = c(0, 100),
  kmeans = FALSE, k = 3, column.scale = TRUE, matrix.main = NULL,
  common.scale = FALSE, legend = TRUE, legend.name = NULL,
  cex.legend = 0.8, xlab = NULL, cex.lab = 1, cex.main = 1,
  cex.axis = 0.8, newpage = TRUE)
```

Arguments

sml	a ScoreMatrixList object
grid	if TRUE, grid graphics will be used. if FALSE, base graphics will be used on the top level, so users can use par(mfrow) or par(mfcol) prior to calling the function. Default:FALSE
col	a color palette or list of color palettes, such as list(heat.colors(10),topo.colors(10)). If it is a list, it is length must match the number of matrices to be plotted. If it is a single palette every heatmap will have the same colors.
xcoords	a vector of numbers showing relative positions of the bases or windows or a list of vectors. The elements of the list must match the number of columns in the corresponding ScoreMatrix. Alternatively, the elements could be a numeric vector of two elements. Such as c(0,100) showing the relative start and end coordinates of the first and last column of the ScoreMatrix object. The remaining coordinates will be automatically matched in this case. If the argument is not a list but a single vector, then all heatmaps will have the same coordinate on their x-axis.
group	a list of vectors of row numbers or a factor. The rows will be reordered to match their grouping. The grouping is used for rowside colors of the heatmap. If it is a list, each element of the list must be a vector of row numbers. Names of the elements of the list will be used as names of groups. If group is a factor , it's length must match the number of rows of the matrix, and factor levels will be used as the names of the groups in the plot.
group.col	a vector of color names to be used at the rowside colors if group argument is given or kmeans=TRUE

order	Logical indicating if the rows should be ordered or not (Default:FALSE). If order=TRUE the matrix will be ordered with rowSums of all matrices in descending order. If kmeans=TRUE or group argument is provided, first the groups/clusters will be ordered in descending order by the sums of rows, then everything within the clusters will be ordered by the sums of rows.
winsorize	Numeric vector of two, defaults to c(0,100). This vector determines the upper and lower percentile values to limit the extreme values. For example, c(0,99) will limit the values to only 99th percentile for a matrix, everything above the 99 percentile will be equalized to the value of 99th percentile.This is useful for visualization of matrices that have outliers.
kmeans	Logical indicating if kmeans clustering should be done on the rows or not (Default:FALSE).
k	Defaults to 3. It designates the number of clusters to be returned by kmeans clustering.
column.scale	Logical indicating if matrices should be scaled or not, prior to k-means clustering or ordering. Setting this to TRUE scales the columns of the matrices using, scale() function. scaled columns are only used for clustering or ordering. Original scores are displayed for heatmaps.
matrix.main	a vector of strings for the titles of the heatmaps. If NULL titles will be obtained from names of the ScoreMatrix objects in the ScoreMatrixList objects.
common.scale	if TRUE (Default:FALSE) all the heatmap colors will be coming from the same score scale, although each heatmap color scale can be different. The color intensities will be coming from the same scale. The scale will be determined by minimum of all matrices and maximum of all matrices. This is useful when all matrices are on the same score scale. If FALSE, the color scale will be determined by minimum and maximum of each matrix individually.
legend	if TRUE and color legend for the heatmap is drawn.
legend.name	a vector of legend labels to be plotted with legends of each heatmap. If it is a length 1 vector, all heatmaps will have the same legend label.
cex.legend	A numerical value giving the amount by which legend axis marks should be magnified relative to the default
xlab	a vector of character strings for x-axis labels of the heatmaps. if it is length 1, all heatmaps will have the same label.
cex.lab	A numerical value giving the amount by which axis labels (including 'legend.name') should be magnified relative to the default.
cex.main	A numerical value giving the amount by which plot title should be magnified
cex.axis	A numerical value giving the amount by which axis marks should be magnified relative to the default
newpage	logical indicating if grid.newpage() function should be invoked if grid=TRUE.

Value

invisibly returns the order of rows, if kmeans=TRUE and/or order=TRUE

Examples

```
# data(cage)
# data(promoters)
# scores1=ScoreMatrix(target=cage,windows=promoters,strand.aware=TRUE)

# data(cpgi)
# scores2=ScoreMatrix(target=cpgi,windows=promoters,strand.aware=TRUE)

# sml=new("ScoreMatrixList",list(a=scores1,b=scores2))
# multiHeatMatrix(sml,kmeans=TRUE,k=2,matrix.main=c("cage","CpGi"),cex.axis=0.8)

# use with K-means
# multiHeatMatrix(sml,kmeans=TRUE,k=2,cex.axis=0.8,xcoords=c(-1000,1000),
#                 winsorize=c(0,99),
#                 legend.name=c("tpm","coverage"),xlab="region around TSS")

# use different colors
# require(RColorBrewer)
# col.cage= brewer.pal(9,"Blues")
# col.cpgi= brewer.pal(9,"YlGn")
# multiHeatMatrix(sml,kmeans=TRUE,k=2,cex.axis=0.8,xcoords=c(-1000,1000),
#                 winsorize=c(0,99),col=list(col.cage,col.cpgi),
#                 legend.name=c("tpm","coverage"),xlab="region around TSS")
```

---

orderBy *Reorder all elements of a ScoreMatrixList to a given ordering vector*

Description

Reorder all elements of a ScoreMatrixList to a given ordering vector

Usage

```
orderBy(sml, ord.vec)

## S4 method for signature 'ScoreMatrixList'
orderBy(sml, ord.vec)
```

Arguments

sml	ScoreMatrixList object
ord.vec	an integer vector

Value

ScoreMatrixList object

Examples

```
#  library(GenomicRanges)
#  data(cage)
#  data(cpgi)
#  data(promoters)

#  cage$tpm = NULL
#  targets = GRangesList(cage=cage, cpgi=cpgi)
#  sml = ScoreMatrixList(targets, promoters, bin.num=10)
#  kmeans.clust = kmeans(sml$cage,3)

#  sml.ordered = orderBy(sml, kmeans.clust$cluster)
#  multiHeatMatrix(sml.ordered)
```

`plotGeneAnnotation` *Plots the enrichment of each feature in the set in the gene annotation*

Description

This function plots a heatmap of enrichment of each range in given gene feature

Usage

```
plotGeneAnnotation(l, cluster = FALSE, col = c("white", "cornflowerblue"))

## S4 method for signature 'list'
plotGeneAnnotation(l, cluster = FALSE, col = c("white",
  "cornflowerblue"))
```

Arguments

`l`	a `list` of `AnnotationByGeneParts` objects
`cluster`	TRUE/FALSE. If TRUE the heatmap is going to be clustered using hierarchical clustering
`col`	a vector of two colors that will be used for interpolation. The first color is the lowest one, the second is the highest one

Value

plots a heatmap of enrichment of target in each gene functional category

Examples

```
# library(GenomicRanges)
# data(cage)
# data(cpgi)

# cage$tpm = NULL
```

```
# gl = GRangesList(cage=cage, cpgi=cpgi)

# bed.file = system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
# gene.parts = readTranscriptFeatures(bed.file)
# annot = annotateWithGeneParts(gl, gene.parts, intersect.chr=TRUE)
# plotGeneAnnotation(annot)
```

`plotMeta` *Line plot(s) for meta-region profiles*

Description

Function calculates meta-profile(s) from a ScoreMatrix or a ScoreMatrixList, then produces a line plot or a set of line plots for meta-region profiles

Usage

```
plotMeta(mat, overlay = TRUE, profile.names = NULL, xcoords = NULL,
  meta.rescale = FALSE, line.col = NULL, ylim = NULL,
  ylab = "average score", xlab = "bases", ...)
```

Arguments

`mat`	`ScoreMatrix` or `ScoreMatrixList` object. If it is a `ScoreMatrixList` object, all matrices in the ScoreMatrixList should have the same number of columns.
`overlay`	If TRUE multiple profiles will be overlayed in the same plot (Default:TRUE). If FALSE, and mat is a ScoreMatrixList, consider using par(mfrow=c(1,length(mat))) to see the plots from all matrices at once.
`profile.names`	a character vector for names of the profiles. The order should be same as the as the order of ScoreMatrixList.
`xcoords`	a numeric vector which designates relative base positions of the meta-region profiles. For example, for a 2001 column ScoreMatrix, xcoord=-1000:1000 indicates relative positions of each column in the score matrix. If NULL (Default), xcoords equals to 1:ncol(mat)
`meta.rescale`	if TRUE meta-region profiles are scaled to 0 to 1 range by subracting the min from profiles and dividing them by max-min.
`line.col`	color of lines for the meta-region profiles. Defaults to colors from `rainbow()` function.
`ylim`	same as `ylim` at `plot` function. if NULL ylim is estimated from all meta-region profiles.
`ylab`	same as `ylab` at `plot` function. Default: "average score"
`xlab`	same as `xlab` at `plot` function. Default: "bases"
`...`	other options to `plot`

Value

returns the meta-region profiles invisibly as a matrix.

Examples

```
# data(cage)
# data(promoters)
# scores1=ScoreMatrix(target=cage,windows=promoters,strand.aware=TRUE)

# data(cpgi)
# scores2=ScoreMatrix(target=cpgi,windows=promoters,strand.aware=TRUE)

# create a new ScoreMatrixList
# x=new("ScoreMatrixList",list(scores1,scores2))
# plotMeta(mat=x,overlay=TRUE,main="my plotowski")
```

---

plotTargetAnnotation *Plot annotation categories from AnnotationByGeneParts or AnnotationByFeature*

Description

This function plots a pie or bar chart for showing percentages of targets annotated by genic parts or other query features

Usage

```
plotTargetAnnotation(x,precedence=TRUE,col,...)

## S4 method for signature 'AnnotationByFeature'
plotTargetAnnotation(x, precedence = TRUE,
  col = getColors(length(x@annotation)), ...)
```

Arguments

x	a AnnotationByFeature or AnnotationByGeneParts object
precedence	TRUE|FALSE. If TRUE there will be a hierachy of annotation features when calculating numbers (with promoter>exon>intron precedence). This option is only valid when x is a AnnotationByGeneParts object
col	a vector of colors for piechart or the bar plot
...	graphical parameters to be passed to pie or barplot functions

Value

plots a piechart or a barplot for percentage of the target features overlapping with annotation

Examples

```
data(cage)

bed.file = system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
gene.parts = readTranscriptFeatures(bed.file)
annot = annotateWithGeneParts(cage, gene.parts, intersect.chr=TRUE)
# plotTargetAnnotation(annot)
```

---

promoters *Example promoter data set.*

Description

promoters of hg19 assembly of human genome on chr21 and chr22. Promoter set is derived from refseq TSS.

Format

GRanges object

RandomEnrichment-class *An S4 class for storing* getRandomEnrichment *function results*

Description

The resulting object stores the results of getRandomEnrichment function

Slots

orig.cnt: number of features overlapping with query at getRandomEnrichment

rand.olap.dist: set of number of features overlapping with randomized queries at getRandomEnrichment

log2fc: log2 fold change calculated by dividing orig.cnt by mean(rand.olap.dist) and taking log2 of that result

p.value: P-value assuming rand.olap.dist has a normal distribution and comparing orig.cnt with that distribution

rand.p.value: p-value from randomization by calculation the proportion of how many times a random number of overlap exceeds the original number of overlap

See Also

getRandomEnrichment

`randomizeFeature` *function that randomizes the genomic coordinates*

Description

This function randomly distributes the coordinates of genomic features which is stored in a `GRanges` object. The randomization can be constrained by supplied arguments. The function is still in Beta mode - the regions can overlap excluded regions, and the randomized regions are not disjoint. Please take care that the excluded and included regions are not too strict when compared to the total width of the ranges.

Usage

```
randomizeFeature(feature, chrom.sizes = NULL, stranded = TRUE,
  keep.strand.prop = TRUE, keep.chrom = TRUE, exclude = NULL,
  include = NULL, seed = NULL, nrand = 1)

## S4 method for signature 'GRanges'
randomizeFeature(feature, chrom.sizes = NULL,
  stranded = TRUE, keep.strand.prop = TRUE, keep.chrom = TRUE,
  exclude = NULL, include = NULL, seed = NULL, nrand = 1)
```

Arguments

`feature`	a GRanges object to be randomized
`chrom.sizes`	sizes of chromosomes as a named vector (names are chromsomes names and elements of the vectors are lengths). , if not given sizes in GRanges object will be used if no sizes there the end of each chr will be the end last feature on each chr
`stranded`	if FALSE, all of the returned features will be strandless (will have "*" in the strand slot)
`keep.strand.prop`	If TRUE strands will have the same proportion as the features
`keep.chrom`	If TRUE, number of features and randomized features for a chromosome will match. Currently seeting this to FALSE is not supported.
`exclude`	A GRanges object where no randomized feature should overlap, can be gaps or unmappable regions in the genome as an example.
`include`	A GRanges object which defines the boundaries of randomized features. If not provided the whole genome is used, as defined using the chrom.sizes parameter.
`seed`	random number generator seed
`nrand`	number of randomizations (default:1)

Value

returns a GRanges object which is randomized version of the feature, along with a "set" column in the metadata which designates to which iteration of the randomization the range belong.

readBed *Read a BED file and convert it to GRanges.*

Description

The function reads a BED file that contains location and other information on genomic features and returns a GRanges object. The minimal information that the BED file has to have is chromosome, start and end columns. it can handle all BED formats up to 12 columns.

Usage

```
readBed(file, track.line = FALSE, remove.unusual = FALSE)
```

Arguments

file	location of the file, a character string such as: "/home/user/my.bed"
track.line	logical, indicates if the bed file has a track line or not. default:FALSE.
remove.unusual	if TRUE(default) remove the chromosomes with unsual names, such as chrX_random (Default:FALSE)

Value

GRanges object

Examples

```
# my.file=system.file("extdata","chr21.refseq.hg19.bed",package="genomation")
# refseq = readBed(my.file,track.line=FALSE,remove.unusual=FALSE)
# head(refseq)
```

readBroadPeak *A function to read the Encode formatted broad peak file into a GRanges object*

Description

A function to read the Encode formatted broad peak file into a GRanges object

Usage

```
readBroadPeak(file)
```

Arguments

file	a abosulte or relative path to a bed file formatted by the Encode broadPeak standard

Value

a GRanges object

Examples

```
# broad.peak.file = system.file('extdata',"ex.broadPeak", package='genomation')

# broad.peak = readBroadPeak(broad.peak.file)
# head(broad.peak)
```

`readFeatureFlank` *A function to read-in genomic features and their upstream and downstream adjecent regions such as CpG islands and their shores*

Description

A function to read-in genomic features and their upstream and downstream adjecent regions such as CpG islands and their shores

Usage

```
readFeatureFlank(location,remove.unusual=TRUE,flank=2000,clean=TRUE,feature.flank.name=NULL)

## S4 method for signature 'character'
readFeatureFlank(location, remove.unusual = TRUE,
  flank = 2000, clean = TRUE, feature.flank.name = NULL)
```

Arguments

`location`	for the bed file of the feature
`remove.unusual`	remove chromsomes with unsual names random, Un and antyhing with "_" character
`flank`	number of basepairs for the flanking regions
`clean`	If set to TRUE, flanks overlapping with other main features will be trimmed
`feature.flank.name`	the names for feature and flank ranges, it should be a character vector of length 2. example: c("CpGi","shores")

Value

a GenomicRangesList contatining one GRanges object for flanks and one for GRanges object for the main feature. NOTE:This can not return a GRangesList at the moment because flanking regions do not have to have the same column name as the feature. GRangesList elements should resemble eachother in the column content. We can not satisfy that criteria for the flanks

Examples

```
# cgi.path = system.file('extdata/chr21.CpGi.hg19.bed', package='genomation')
# cgi.shores = readFeatureFlank(cgi.path)
```

`readGeneric` *Read a tabular file and convert it to GRanges.*

Description

The function reads a tabular text file that contains location and other information on genomic features and returns a `GRanges` object. The minimal information that the file has to have is chromosome, start and end columns. Strand information is not compulsory.

Usage

```
readGeneric(file, chr = 1, start = 2, end = 3, strand = NULL,
  meta.cols = NULL, keep.all.metadata = FALSE, zero.based = FALSE,
  remove.unusual = FALSE, header = FALSE, skip = 0, sep = "\t")
```

Arguments

`file`	location of the file, a character string such as: "/home/user/my.bed"
`chr`	number of the column that has chromsomes information in the table (Def:1)
`start`	number of the column that has start coordinates in the table (Def:2)
`end`	number of the column that has end coordinates in the table (Def:3)
`strand`	number of the column that has strand information, only -/+ is accepted (Default:NULL)
`meta.cols`	named `list` that maps column numbers to meta data columns. e.g. list(name=5, score=10), which means 5th column will be named "name", and 10th column will be named "score" and their contents will be a part of the returned GRanges object. If header = TRUE, meta.cols parameter will over-write the column names given by the header line of the data frame.
`keep.all.metadata`	`logical` determining if the extra columns (the ones that are not designated by chr,start,end,strand and meta.cols arguments) should be kept or not. (Default:FALSE)
`zero.based`	a boolean which tells whether the ranges in the bed file are 0 or 1 base encoded. (Default: FALSE)
`remove.unusual`	if TRUE(default) remove the chromosomes with unsual names, such as chrX_random (Default:FALSE)
`header`	whether the original file contains a header line which designates the column names. If `TRUE` header will be used to construct column names. These names can be over written by meta.cols argument.

skip — number of lines to skip. If there is a header line(s) you do not wish to include you can use skip argument to skip that line.

sep — a single character which designates the separator in the file. The default value is tab.

Value

GRanges object

Examples

```
# my.file=system.file("extdata","chr21.refseq.hg19.bed",package="genomation")
# refseq = readGeneric(my.file,chr=1,start=2,end=3,strand=NULL,
#                      meta.cols=list(score=5,name=4),
#                      keep.all.metadata=FALSE, zero.based=TRUE)
# head(refseq)
```

---

readNarrowPeak *A function to read the Encode formatted narrowPeak file into a GRanges object*

Description

A function to read the Encode formatted narrowPeak file into a GRanges object

Usage

```
readNarrowPeak(file)
```

Arguments

file — a abosulte or relative path to a bed file formatted by the Encode narrowPeak standard

Value

a GRanges object

Examples

```
# narrow.peak.file = system.file('extdata',"ex.narrowPeak", package='genomation')

# narrow.peak = readBroadPeak(narrow.peak.file)
# head(narrow.peak)
```

readTranscriptFeatures *Function for reading exon intron and promoter structure from a given bed file*

Description

Function for reading exon intron and promoter structure from a given bed file

Usage

```
readTranscriptFeatures(location,remove.unusual=TRUE,up.flank=1000,down.flank=1000,unique.prom=TRUE)

## S4 method for signature 'character'
readTranscriptFeatures(location, remove.unusual = TRUE,
  up.flank = 1000, down.flank = 1000, unique.prom = TRUE)
```

Arguments

location	location of the bed file with 12 or more columns
remove.unusual	remove the chromomesomes with unsual names, mainly random chromsomes etc
up.flank	up-stream from TSS to detect promoter boundaries
down.flank	down-stream from TSS to detect promoter boundaries
unique.prom	get only the unique promoters, promoter boundaries will not have a gene name if you set this option to be TRUE

Value

a GRangesList containing locations of exon/intron/promoter/TSS

Note

one bed track per file is only accepted, the bed files with multiple tracks will cause en error

Examples

```
# my.bed12.file = system.file("extdata/chr21.refseq.hg19.bed", package = "genomation")
# my.bed12.file
# feats = readTranscriptFeatures(my.bed12.file)
# names(feats)
# sapply(feats, head)
```

scaleScoreMatrix *Scales the values in the matrix by rows and/or columns*

Description

Scales the values in the matrix by rows and/or columns

Usage

```
scaleScoreMatrix(mat, columns = FALSE, rows = TRUE, scalefun = NULL)

## S4 method for signature 'ScoreMatrix'
scaleScoreMatrix(mat, columns = FALSE, rows = TRUE,
  scalefun = NULL)
```

Arguments

mat	ScoreMatrix object
columns	columns whether to scale the matrix by columns. Set by default to FALSE.
rows	rows Whether to scale the matrix by rows. Set by default to TRUE
scalefun	function object that takes as input a matrix and returns a matrix. By default the argument is set to (x - mean(x))/(max(x)-min(x)+1)

Value

ScoreMatrix object

Examples

```
# scale the rows of a scoreMatrix object
#library(GenomicRanges)
#target = GRanges(rep(c(1,2),each=7), IRanges(rep(c(1,1,2,3,7,8,9), times=2), width=5),
#         weight = rep(c(1,2),each=7),
#        strand=c('-', '-', '-', '-', '+', '-', '+', '-', '-', '-', '-', '-', '-', '+'))
#windows = GRanges(rep(c(1,2),each=2), IRanges(rep(c(1,2), times=2), width=5),
#           strand=c('-','+','-','+'))
#sm = ScoreMatrix(target, windows)
#ssm = scaleScoreMatrix(sm, rows=TRUE)
```

scaleScoreMatrixList *Scale the ScoreMatrixList*

Description

Scales each ScoreMatrix in the ScoreMatrixList object, by rows and/or columns

Usage

```
scaleScoreMatrixList(sml, columns, rows, scalefun)

## S4 method for signature 'ScoreMatrixList'
scaleScoreMatrixList(sml, columns = FALSE,
  rows = TRUE, scalefun = NULL)
```

Arguments

sml	a ScoreMatrixList object
columns	a columns whether to scale the matrix by columns. Set by default to FALSE
rows	a rows Whether to scale the matrix by rows. Set by default to TRUE
scalefun	a function object that takes as input a matrix and returns a matrix. By default the argument is set to the R scale function with center=TRUE and scale=TRUE

Value

ScoreMatrixList object

Examples

```
# library(GenomicRanges)
# data(cage)
# data(cpgi)
# data(promoters)

# cage$tpm = NULL
# targets = GRangesList(cage=cage, cpgi=cpgi)
# sml = ScoreMatrixList(targets, promoters, bin.num=10, strand.aware=TRUE)
# sml.scaled = scaleScoreMatrixList(sml, rows=TRUE)
# multiHeatMatrix(sml)
```

ScoreMatrix *Get base-pair score for bases in each window*

Description

The funcion produces a base-pair resolution matrix of scores for given equal width windows of interest. The returned matrix can be used to draw meta profiles or heatmap of read coverage or wig track-like data. The windows argument can be a predefined region around transcription start sites or other regions of interest that have equal lengths The function removes all window that fall off the Rle object - have the start coordinate < 1 or end coordinate > length(Rle) The function takes the intersection of names in the Rle and GRanges objects. On Windows OS the function will give an error if the target is a file in .bigWig format.

Usage

```
ScoreMatrix(target, windows, strand.aware = FALSE, weight.col = NULL,
  is.noCovNA = FALSE, type = "", rpm = FALSE, unique = FALSE,
  extend = 0, param = NULL)

\S4method{ScoreMatrix}{RleList,GRanges}(target,windows,strand.aware)

\S4method{ScoreMatrix}{GRanges,GRanges}(target, windows, strand.aware, weight.col, is.noCovNA)

\S4method{ScoreMatrix}{character,GRanges}(target,windows, strand.aware, type='', rpm=FALSE, unique=F
```

Arguments

target	RleList , GRanges, a BAM file or a BigWig to be overlapped with ranges in windows
windows	GRanges object that contains the windows of interest. It could be promoters, CpG islands, exons, introns. However the sizes of windows have to be equal.
strand.aware	If TRUE (default: FALSE), the strands of the windows will be taken into account in the resulting ScoreMatrix. If the strand of a window is -, the values of the bins for that window will be reversed
weight.col	if the object is GRanges object a numeric column in meta data part can be used as weights. This is particularly useful when genomic regions have scores other than their coverage values, such as percent methylation, conservation scores, GC content, etc.
is.noCovNA	(Default:FALSE) if TRUE,and if 'target' is a GRanges object with 'weight.col' provided, the bases that are uncovered will be preserved as NA in the returned object. This useful for situations where you can not have coverage all over the genome, such as CpG methylation values.
type	if target is a character vector of file paths, then type designates the type of the corresponding files (bam or bigWig)
rpm	boolean telling whether to normalize the coverage to per milion reads. FALSE by default.

unique	boolean which tells the function to remove duplicated reads based on chr, start, end and strand
extend	numeric which tells the function to extend the reads to width=extend
param	ScanBamParam object

Value

returns a ScoreMatrix object

See Also

ScoreMatrixBin

Examples

```
# When target is GRanges
#data(cage)
#data(promoters)
#scores1=ScoreMatrix(target=cage,windows=promoters,strand.aware=TRUE,
#                                weight.col="tpm")

# When target is RleList
#library(GenomicRanges)
#covs = coverage(cage)
#scores2 = ScoreMatrix(target=covs,windows=promoters,strand.aware=TRUE)

# When target is a bam file
# bam.file = system.file('tests/test.bam', package='genomation')
# windows = GRanges(rep(c(1,2),each=2), IRanges(rep(c(1,2), times=2), width=5))
# scores3 = ScoreMatrix(target=bam.file,windows=windows, type='bam')
```

ScoreMatrix-class *An S4 class for storing* ScoreMatrix *function results*

Description

The resulting object is an extension of a matrix object, and stores values (typically genome-wide scores) for a predefined set of regions Each row on the ScoreMatrix is a predefined region (Ex: CpG islands, promoters) and columns are values across those regions.

See Also

ScoreMatrix

ScoreMatrixBin *Get bin score for bins on each window*

Description

The function firsts bins each window to equal number of bins, and calculates the a summary metrix for scores of each bin (currently, mean, max and min supported) A scoreMatrix object can be used to draw average profiles or heatmap of read coverage or wig track-like data. windows can be a predefined region such as CpG islands or gene bodies that are not necessarily equi-width. Each window will be chopped to equal number of bins based on bin.num option.

Usage

```
ScoreMatrixBin(target, windows, bin.num = 10, bin.op = "mean",
  strand.aware = FALSE, weight.col = NULL, is.noCovNA = FALSE,
  type = "", rpm = FALSE, unique = FALSE, extend = 0, param = NULL)

\S4method{ScoreMatrixBin}{RleList,GRanges}(target, windows, bin.num, bin.op, strand.aware)

\S4method{ScoreMatrixBin}{GRanges,GRanges}(target,windows,bin.num,bin.op,strand.aware,weight.col,is

\S4method{ScoreMatrixBin}{character,GRanges}(target, windows, bin.num=10, bin.op='mean', strand.awar
```

Arguments

target	RleList, GRanges, BAM file or a bigWig file object to be overlapped with ranges in windows
windows	GRanges object that contains the windows of interest. It could be promoters, CpG islands, exons, introns. However, the sizes of windows does NOT have to be equal.
bin.num	single integer value denoting how many bins there should be for each window
bin.op	bin operation that is either one of the following strings: "max","min","mean". The operation is applied on the values in the bin. Defaults to "mean"
strand.aware	If TRUE (default: FALSE), the strands of the windows will be taken into account in the resulting scoreMatrix. If the strand of a window is -, the values of the bins for that window will be reversed
weight.col	if the object is GRanges object a numeric column in meta data part can be used as weights. This is particularly useful when genomic regions have scores other than their coverage values, such as percent methylation, conservation scores, GC content, etc.
is.noCovNA	(Default:FALSE) if TRUE,and if 'target' is a GRanges object with 'weight.col' provided, the bases that are uncovered will be preserved as NA in the returned object. This useful for situations where you can not have coverage all over the genome, such as CpG methylation values.

type	if target is a character vector of file paths, then type designates the type of the corresponding files (bam or bigWig)
rpm	boolean telling whether to normalize the coverage to per milion reads. FALSE by default.
unique	boolean which tells the function to remove duplicated reads based on chr, start, end and strand
extend	numeric which tells the function to extend the reads to width=extend
param	ScanBamParam object

Value

returns a scoreMatrix object

See Also

ScoreMatrix

Examples

```
#  data(cage)
#  data(cpgi)
#  data(promoters)
#  myMat=ScoreMatrixBin(target=cage,
#                       windows=cpgi,bin.num=10,bin.op="mean",weight.col="tpm")
# plot(colMeans(myMat,na.rm=TRUE),type="l")

# myMat2=ScoreMatrixBin(target=cage,
#                        windows=promoters,bin.num=10,bin.op="mean",
#                        weight.col="tpm",strand.aware=TRUE)
# plot(colMeans(myMat2,na.rm=TRUE),type="l")
```

ScoreMatrixList *Make ScoreMatrixList from multiple targets*

Description

The function constructs a list of ScoreMatrix objects in the form of ScoreMatrixList object. This object can be visualized using multiHeatMatrix, heatMeta or plotMeta

Usage

```
ScoreMatrixList(targets, windows = NULL, bin.num = NULL, bin.op = "mean",
  strand.aware = FALSE, weight.col = NULL, is.noCovNA = FALSE,
  type = "", rpm = FALSE, unique = FALSE, extend = 0, param = NULL)
```

Arguments

targets	can be a list of scoreMatrix objects, that are coerced to the ScoreMatrixList, a list of RleList objects, or a character vector specifying the locations of mulitple bam files or bigWig files that are used to construct the scoreMatrixList. If it is either a RleList object or a character vector of files, it is obligatory to give a windows argument.
windows	GenomicRanges containing viewpoints for the scoreMatrix or ScoreMatrixList functions
bin.num	an integer telling the number of bins to bin the score matrix
bin.op	an name of the function that will be used for smoothing windows of ranges
strand.aware	a boolean telling the function whether to reverse the coverage of ranges that come from - strand (e.g. when plotting enrichment around transcription start sites)
weight.col	if the object is GRanges object a numeric column in meta data part can be used as weights. This is particularly useful when genomic regions have scores other than their coverage values, such as percent methylation, conservation scores, GC content, etc.
is.noCovNA	(Default:FALSE) if TRUE,and if 'targets' is a GRanges object with 'weight.col' provided, the bases that are uncovered will be preserved as NA in the returned object. This useful for situations where you can not have coverage all over the genome, such as CpG methylation values.
type	if targets is a character vector of file paths, then type designates the type of the corresponding files (bam or bigWig)
rpm	boolean telling whether to normalize the coverage to per milion reads. FALSE by default.
unique	boolean which tells the function to remove duplicated reads based on chr, start, end and strand
extend	numeric which tells the function to extend the features (i.e aligned reads) to total length ofwidth+extend
param	ScanBamParam object

Value

returns a ScoreMatrixList object

Examples

```
# visualize the distribution of cage clusters and cpg islands around promoters
# library(GenomicRanges)
# data(cage)
# data(cpgi)
# data(promoters)

# cage$tpm = NULL
# targets = GRangesList(cage=cage, cpgi=cpgi)
# sml = ScoreMatrixList(targets, promoters, bin.num=10, strand.aware=TRUE)
# multiHeatMatrix(sml)
```

ScoreMatrixList-class *An S4 class for storing a set of* ScoreMatrixList

Description

The resulting object is an extension of a `list` object, where each element corresponds to a score matrix object

See Also

ScoreMatrixList

show,RandomEnrichment-method *show method for some of the genomation classes*

Description

show method for some of the genomation classes

Usage

```
## S4 method for signature 'RandomEnrichment'
show(object)

## S4 method for signature 'AnnotationByGeneParts'
show(object)

## S4 method for signature 'AnnotationByFeature'
show(object)

## S4 method for signature 'ScoreMatrix'
show(object)

## S4 method for signature 'ScoreMatrixList'
show(object)
```

Arguments

`object`	object of class RandomEnrichment

Value

Shows the dimension of the ScoreMatrix

Shows the number of matrices and their sizes

Index