Package 'metagene'

April 10, 2015

Version 1.0.0

Date 2014-05-05

Title A package to produce metagene plots

Author Charles Joly Beauparlant <charles.joly-beauparlant@crchul.ulaval.ca>,
Fabien Claude Lamaze <fabien.lamaze.1@ulaval.ca>,
Rawane Samb <rawane.samb.1@ulaval.ca>,
Astrid Louise Deschenes <Astrid-Louise.Deschenes@crchudequebec.ulaval.ca> and
Arnaud Droit <arnaud.droit@crchuq.ulaval.ca>.

Author@R c(person(``Charles", ``Joly Beauparlant",
email=``charles.joly-beauparlant@crchul.ulaval.ca"),
person(``Fabien Claude", ``Lamaze",
email=``fabien.lamaze.1@ulaval.ca"), person(``Rawane", ``Samb",
email=``rawane.samb.1@ulaval.ca"), person(``Astrid
Louise",``Deschenes",
email=``Astrid-Louise.Deschenes@crchudequebec.ulaval.ca")
person(``Arnaud", ``Droit",
email=``arnaud.droit@crchuq.ulaval.ca"))

Maintainer Charles Joly Beauparlant <charles.joly-beauparlant@crchul.ulaval.ca>

Description

This package produces metagene plots to compare the behavior of DNA-interacting proteins at selected groups of genes/features. Pre-calculated features (such as transcription start sites of protein coding gene or enhancer) are available. Bam files are used to increase the resolution. Multiple combination of group of features and or group of bam files can be compared in a single analysis. Bootstraping analysis is used to compare the groups and locate regions with statistically different enrichment profiles.

biocViews ChIPSeq, Genetics, MultipleComparison

License Artistic-2.0

BugReports https://github.com/CharlesJB/metagene/issues

VignetteBuilder knitr

Depends rtracklayer, GenomicRanges, GenomicAlignments, ggplot2

Enhances parallel, biomaRt

Suggests RUnit, BiocGenerics, knitr

R topics documented:

`applyOnGroups` *Apply a function on every groups of the main data structure*

Description

This function takes a `list` of elements and apply a user-specified function on them. Works like lapply, but can use multiple cores if asked.

Usage

```
applyOnGroups(
  groups,
  cores=1,
  FUN,
 ...)
```

Arguments

groups	The main data structure
cores	Number of cores for parallel processing. Require parallel package.
FUN	The function to apply on every groups.
...	Extract arguments for FUN.

Value

The applyOnGroups return value will vary depending on the function specified by the user.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: a <- list("a", "b", "c")
## Not run: metagene:::applyOnGroups(a, cores=1, FUN=print)
```

---

binBootstrap *Estimate mean and confidence interval of a column using bootstrap.*

Description

Since the sample size can be small, the use of a bootstrap function can help to calculate a better approximation of the mean and reduce the confidence intervals.

Usage

```
binBootstrap(
  data,
  alpha,
  sampleSize,
  cores=1)
```

Arguments

data	A vector representing a column from the binned matrix.
alpha	Confidence interval.
sampleSize	Number of time each bin will be resampled (hould be at least 1000).
cores	Number of cores for parallel processing (require parallel package).

Value

binBootstrap returns a list with the mean of the bootstrapped vector and the quartile of order alpha/2 and (1-alpha/2)

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: metagene:::binBootstrap(1:100, alpha=0.05, sampleSize=100)
```

`binMatrix` *Bin matrix columns*

Description

This function will split a matrix into multiple segments.

Usage

```
binMatrix(
  data,
  binSize)
```

Arguments

`data`	The matrix to bin.
`binSize`	The number of nucleotides in each bin.

Value

`binMatrix` returns a `matrix` with each columns representing the mean of `binSize` nucleotides.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: metagene:::binMatrix(matrix(1:40, nrow=4, ncol=10), 4)
```

`bootstrapAnalysis` *Perform the bootstrap analysis*

Description

This function wraps the diffentent steps needed to do a bootstrap analysis on a matrix.

Usage

```
bootstrapAnalysis(
  currentMatrix,
  binSize,
  alpha,
  sampleSize,
  cores=1)
```

Arguments

`currentMatrix`	The matrix to use for the bootstrap analysis
`binSize`	The number of nucleotides in each bin for the bootstrap step.
`alpha`	Confidence interval.
`sampleSize`	Number of time each bin will be resampled (hould be at least 1000).
`cores`	Number of cores for parallel processing (require parallel package).

Value

`bootstrapAnalysis` returns a `list` with the mean of the bootstraped vector and the quartile of order alpha/2 and (1-alpha/2).

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: metagene:::bootstrapAnalysis(matrix(1:40, nrow=4, ncol=10), 4, alpha=0.05, sampleSize=100)
```

`getDataFrame` *Convert the bootstrapped data into a data.frame*

Description

This function will convert the results of the bootstrap analysis into a `data.frame` that can be used directly with ggplot2 to produce the final plot.

Usage

```
getDataFrame(bootstrapData, range, binSize)
```

Arguments

`bootstrapData`	Data produced during the bootstrap analysis
`range`	The start and the end for the x-label
`binSize`	The number of nucleotides in each bin for the bootstrap step.

Value

`getDataFrame` a `data.frame` with the condensed results from the main data structure.

Columns: * Groups: name of current group * distances: the number of bin for each entry * means: the means to plot * qinf: the lower end of the confidence interval * qsup: the higher end of the confidence interval

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Get bootstrap results
## Not run: bootstrapResults <- list()
## Not run: bootstrapResults$a <- metagene:::bootstrapAnalysis(matrix(1:40, nrow=4, ncol=10), 4, alpha=0.05, sam

# Get the data.frame
## Not run: DF <- metagene:::getDataFrame(bootstrapResults, c(-1,1))
```

getGenes *Fetch the annotation of all genes.*

Description

This function will fetch the positions of all known coding genes for a given specie. Currently supported species are: “mouse”, “human” (default).

Usage

```
getGenes(
  specie="human")
```

Arguments

specie	human: Homo sapiens (default) / mouse: Mus musculus

Details

This function will fetch all the ensembl_gene_id for a given specie ("human" or "mouse").

Value

getGenes return a GRanges object with a feature metadata that correspond to the ensembl_gene_id.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: knownGenes <- getGenes("human")
```

getGenesBiomart *Fetch the annotation of all genes from biomart.*

Description

This function will fetch the positions of all known coding genes for a given specie. Currently supported species are: “mouse”, “human” (default).

This function was used to create external datasets for the getGenes function.

Usage

```
getGenesBiomart(
  specie="human")
```

Arguments

specie	human: Homo sapiens (default) / mouse: Mus musculus

Details

Using biomaRt package, this function will fetch all the ensembl_gene_id for a given specie ("human" or "mouse").

Value

getGenesBiomart return a GRanges object with a feature metadata that corresponds to emsembl_gene_id.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: knownGenes <- getGenesBiomart("human")
```

---

mergeMatrix *Convert list of vectors in matrix for a single group*

Description

This function combine a list of list of vectors. The first step is to merge every list of vector in a group into a matrix. Then every matrices in the same group are combined.

Usage

```
mergeMatrix(
  group,
  level)
```

Arguments

group	Correspond to an element in the data subsection of the main data structure.
level	The names of the element to merge.

Details

Pre-calculated features (such as transcription start sites of protein coding gene or enhancer) are available. Bam files are used to increase the resolution. Multiple combination of group of features and or group of bam files can be compared in a single analysis. Bootstraping analysis is used to compare the groups and locate regions with statistically different enrichment profiles.

Value

mergeMatrix returns the group element from the arguments with an extra element named matrix.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Create the lists of vector
## Not run: vector1 <- list(c(1,2,3,4), c(5,6,7,8))
## Not run: vector2 <- list(c(9,10,11,12), c(13,14,15,16))

# Add them to a list
## Not run: group <- list()
## Not run: group$vectors <- list(vector1, vector2)

# We will modify the group objet to add an element names
# matrix that contains the matrix that correspond to the
# vectors binded by row
## Not run: metagene:::mergeMatrix(group, "vectors")
```

---

parseBamFile *Parse a single bam file*

Description

This function will fetch all the aligned reads from a bam file that overlap with a list of regions.

Usage

```
parseBamFile(
  bamFile,
  features,
  cores=1)
```

Arguments

bamFile	The name of the bam file to parse. Must be sorted and indexed.
features	A GRanges corresponding to the regions to parse.
cores	Number of cores for parallel processing (require parallel package).

Value

parseBamFile returns a list with an element for every feature to parse. Each element contains a vector of reads coverage.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Get bamFile
## Not run: bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")

# Get features
## Not run: bedFileName <- system.file("extdata/list1.bed", package="metagene")
## Not run: features <- read.table(bedFileName, header=FALSE, stringsAsFactors=FALSE, nrow=10)
## Not run: colnames(features) <- c("space", "start_position", "end_position")
## Not run: features$end_position <- features$start_position + 200
## Not run: features$start_position <- features$start_position - 200

# Parse the bam file
## Not run: bamDensity <- metagene:::parseBamFile(bamFileName, features)
```

---

parseBamFiles *Parse multiple bamFiles*

Description

This function will fetch the aligned reads from multiple bam files that overlap with a list of regions.

Usage

```
parseBamFiles(
  bamFiles,
  featuresGroups,
  cores=1)
```

Arguments

bamFiles	A vector of bam files
featuresGroups	A list of data.frame. One data.frame by group of features. The names of each element of the list correspond to the name of the group.
cores	Number of cores for parallel processing (require parallel package).

Value

parseBamFiles returns a list of list of list that contains the raw counts for every features groups.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Get a bam file name
## Not run: bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")

# Load a feature file
## Not run: featuresFileName <- system.file("extdata/list1.txt", package="metagene")
## Not run: features <- metagene:::prepareFeatures(featuresFileName, specie="mouse")

# Parse multiple bam files
## Not run: bam <- metagene:::parseBamFiles(bamFileName, features)
```

`parseFeatures` *Parse an experiment using a list of features*

Description

This function produces the list object that contains all the information necessary to produce a metagene-like plot with the `plotGraphic` function. Currently supported species are: "mouse", "human" (default).

Usage

```
parseFeatures(
  bamFiles,
  features=NULL,
  specie="human",
  maxDistance=5000,
  design=NULL,
  cores=1,
  debug=FALSE)
```

Arguments

`bamFiles`	A `vector` of BAM files to plot. All BAM files must exist.
`features`	Either a filename of a `vector` of filenames. Supported features: ensembl_gene_id. If value is NULL, all known RefSeq genes will be used.
`specie`	human: Homo sapiens (default). mouse: Mus musculus.
`maxDistance`	The distance around feature to include in the plot. The maximum distance has to be a positive integer.
`design`	A `data.frame` explaining the relationship between multiple samples. One line per samples. One column per group of samples. For example, biological replicates and corresponding controls are in the same group. 1: treatment file(s). 2: control file(s).
`cores`	Number of cores for parallel processing. Require parallel package. The number of cores has to be a positive integer.
`debug`	Keep the intermediate files (can use a lot of memory). TRUE or FALSE.

Details

This function will extract the read density from alignments files (bam) in the viscinity of transcription start sites of one or multiple list of genes.

The values are normalized as read per millions aligned (RPM).

It is possible to parse multiple groups of gene by saving each list in a separate file and by listing the file names in a vector as the `features` parameter.

By using the `design` parameter, the `parseFeatures` function will deal with more complex experimental design such as the use of replicates and/or controls. The values of controls are substracted from every replicates.

Value

`parseFeatures` returns a list that contains the data necessary to produce a plot.

The data structure is a list of lists.

The first level contain the following fields:

- design: The information from the design file.
- param: The values of the argument used with parseFeatures.
- bamFilesDescription: A `data.frame` with the following columns:
  - bam: The names of the sorted bam files
  - oldBam: The names of the original bam files
  - alignedCount: The number of aligned reads
- matrix: A list of matrix that will be used to produce the plot. One element by combination of features/design groups.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
featuresFileName <- system.file("extdata/list1.txt", package="metagene")
groups <- parseFeatures(bamFileName, featuresFileName, specie="mouse")
```

`parseRegions` *Parse an experiment using regions that can be of different length.*

Description

This function produces the list object that contains all the information necessary to produce a metagene-like plot with the `plotGraphic` function.

Usage

```
parseRegions(
  regions,
  bamFiles,
  specie="human",
  design=NULL,
  paddingSize,
  cores=1,
  debug=FALSE)
```

Arguments

regions	A vector of bed file names corresponding to the regions to include in the analysis, a GRanges object or a GRangesList object.
bamFiles	A vector of BAM files to plot. All BAM files must exist.
specie	human: Homo sapiens (default). mouse: Mus musculus.
design	A data.frame explaining the relationship between multiple samples. One line per samples. One column per group of samples. For example, biological replicates and corresponding controls are in the same group. 1: treatment file(s). 2: control file(s).
paddingSize	The length of padding we want to add on each side of each regions. The padding size has to be a positive integer
cores	Number of cores for parallel processing. Require parallel package. The number of cores has to be a positive integer.
debug	Keep the intermediate files (can use a lot of memory). TRUE or FALSE.

Details

This function will extract the read density from alignments files (BAM) in one or multiple list of regions.

The values are normalized as read per millions aligned (RPM).

It is possible to parse multiple groups of regions by saving each regions in a separate bed file and by listing the file names in a vector as the regions parameter.

By using the design parameter, the parseRegions function will deal with more complex experimental design such as the use of replicates and/or controls. The values of controls are substracted from every replicates.

Value

plotRegions returns a list that contains the data necessary to produce a plot.

The data structure is a list of lists.

The first level contain the following fields:

- design: The information from the design file.
- param: The values of the argument used with parseFeatures.

- bamFilesDescription: A `data.frame` with the following columns;
  - The names of the original bam files.
  - The names of the sorted bam files
  - The number of aligned reads.
- matrix: A list of matrix that will be used to produce the plot. One element by combination of features/design groups.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Minimally, we need a bam file name and a list of regions
bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
listFileName <- system.file("extdata/list1.bed", package="metagene")
groups <- parseRegions(listFileName, bamFileName, specie="mouse")
```

`plotGraphic` *Produce a plot with based on a data.frame*

Description

This function will produce the final plot for an analysis.

Usage

```
plotGraphic(
  DF,
  title,
  binSize)
```

Arguments

`DF`	The data frame produced by the getDataFrame function
`title`	The title of the graph
`binSize`	The number of nucleotides in each bin for the bootstrap step.

Value

`plotGraphic` plots a graph on the current device.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: DF <- data.frame(Groups=rep("group1",11), distances=seq(-5,5), means=rep(30, 11), qinf=rep(20, 11), c
 ## Not run: metagene:::plotGraphic(DF, "Example")
```

---

plotMatrices *Create a graph*

Description

This function wraps all the steps necessary to produce the final plot using the results of either `parseFeatures` or `parseRegions` functions.

Usage

```
plotMatrices(
  matricesGroups,
  data,
  binSize=100,
  alpha=0.05,
  sampleSize=1000,
  cores=1)
```

Arguments

`matricesGroups`	A list with every groups to include in the graph. A group is one or more combination of featureGroup/designGroup. The names must correspond to the names of the matrix object returned from the parse functions (parseFeatures or parseRegions).
`data`	The object returned by the parse functions. (parseFeatures or parseRegions).
`binSize`	The number of nucleotides in each bin for the bootstrap step.
`alpha`	Confidence interval.
`sampleSize`	Number of time each bin will be resampled (hould be at least 1000).
`cores`	Number of cores for parallel processing (require parallel package).

Value

`plotMatrices` return the `data.frame` used to produce the graph.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Prepare the groups
bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
listFileName <- system.file("extdata/list1.txt", package="metagene")
groups <- parseFeatures(bamFileName, listFileName, specie="mouse")

# Do the graph
DF <- plotMatrices(list(g1="list1"), groups)
```

`prepareBamFiles` *Prepare bam files before parsing.*

Description

Sort and index bam files, if necessary. Return the number of aligned reads for each bam file.

Usage

```
prepareBamFiles(
  bamFiles,
  cores=1)
```

Arguments

`bamFiles`	Vector containing the list of every bam filename to be included in the analysis.
`cores`	Number of cores used by the function.

Details

This function will take a list of bam files and first check if they are indexed. If this is not the case, it will call the `Rsamtools` package to sort and index the files.

The names of the orginal bam files will be stored in the `oldBam` column of the results and the sorted bam file name will be stored in the `bam` column. If the files already indexed, the names in both columns will be the same.

The aligned read counts are then calculated for each files and added to the `alignedCount` column of the result.

Value

`prepareBamFiles` returns a `data.frame` with three columns and as many rows as there are bam files in the `bamFiles` argument. The `data.frame` has the following columns: * `oldBam` * `bam` * `alignedCount`

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
## Not run: bamFileDescription <- metagene:::prepareBamFiles(bamFileName)
```

---

prepareFeatures *Convert a list of IDs into a GRangesList*

Description

This function will convert a list of Ensembl Gene IDs into a list of regions.

Usage

```
prepareFeatures(
  features,
  specie="human",
  maxDistance=5000,
  cores=1)
```

Arguments

features	Either a filename of a vector of filenames. Supported features: ensembl_gene_id If value is NULL: every ensembl_gene_id will be returned.
specie	human: Homo sapiens (default) / mouse: Mus musculus
maxDistance	The distance around features to include in the plot.
cores	Number of cores for parallel processing (require parallel package).

Value

prepareFeatures returns a list of GRangesList. One GRanges by group of features. The names of each GRanges of the list correspond to the name of the group.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Load a feature file
## Not run: featuresFileName <- system.file("extdata/list1.txt", package="metagene")
## Not run: features <- metagene:::prepareFeatures(featuresFileName, specie="mouse")
```

prepareGroups *Distribute the bam filenames in their respective groups.*

Description

This function will initialize every groups in the main data structure.

Usage

```
prepareGroups(
  featuresGroupsNames,
  bamFiles,
  design=NULL)
```

Arguments

featuresGroupsNames	The name of the group of features in the current analysis.
bamFiles	A vector of bam filename(s).
design	A matrix explaining the relationship between multiple samples. * One line per samples. * One column per group of samples. For example, biological replicates and corresponding controls are in the same group. 1: treatment file(s) 2: control file(s)

Value

prepareGroups returns a list of list of list.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

prepareRegions *Parse bed files and convert them in a list of GRangesList.*

Description

This function will convert a bed files into a GRangesList.

Usage

```
prepareRegions(
  regions,
  cores=1)
```

Arguments

regions	A vector of bed file names corresponding to the regions to include in the analysis. The file name (minus the extension) will be used as the name of the region.
cores	Number of cores for parallel processing (require parallel package).

Value

prepareRegions returns a GRangesList. One GRanges by group of features. The names of each element of the list correspond to the name of the group.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: regionsFileName <- system.file("extdata/list1.bed", package="metagene")
## Not run: regions <- metagene:::prepareRegions(regionsFileName)
```

---

rawCountsToRPM *Convert the raw counts into reads per million aligned (rpm)*

Description

This function will convert raw counts into reads per million aligned (rpm) using the aligned reads counts obtained with the prepareBamFiles function.

Usage

```
rawCountsToRPM(
  rawCounts,
  bamFilesDescription,
  cores=1)
```

Arguments

rawCounts	The data structure returned by the parseBamFiles function.
bamFilesDescription	The data.frame obtained with the prepareBamFiles function.
cores	Number of cores for parallel processing (require parallel package).

Value

rawCountsToRPM returns a list of list of list that contains the rpm for every features groups.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
# Get the description of the bam files
## Not run: bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
## Not run: bamFileDescription <- metagene:::prepareBamFiles(bamFileName)

# Get the raw counts
## Not run: bamFileName <- system.file("extdata/align1_rep1.bam", package="metagene")
## Not run: featuresFileName <- system.file("extdata/list1.txt", package="metagene")
## Not run: features <- metagene:::prepareFeatures(featuresFileName, specie="mouse")
## Not run: bamRawCounts <- metagene:::parseBamFiles(bamFileName, features)

# Get the RPM values
## Not run: RPMcounts <- metagene:::rawCountsToRPM(bamRawCounts, bamFileDescription)
```

---

removeControls *Substract controls from a single group*

Description

This function will substract controls for every replicates in the same group. If there are multiple controls, their values will be averaged before substracting them from the input. Values cannot be lower than 0.

Usage

```
removeControls(
  group,
  data.rpm,
  design,
  controlCores=1)
```

Arguments

group	A group extracted from the main data structure
data.rpm	The normalized data for every bam files.
design	The line from matrix explaining the relationship between current samples.
controlCores	Number of cores for parallel processing (require parallel package).

Value

removeControls The group extracted from the main data structure from which the controls were substracted then deleted

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
 # Prepare bam files
 ## Not run: bamFileInput <- system.file("extdata/align1_rep1.bam", package="metagene")
 ## Not run: bamFileControl <- system.file("extdata/ctrl.bam", package="metagene")
 ## Not run: bamFiles <- c(bamFileInput, bamFileControl)
 ## Not run: groups <- list()
 ## Not run: groups$bamFilesDescription <- metagene:::prepareBamFiles(bamFiles)

 # Parse features and convert to RPM
 ## Not run: features <- system.file("extdata/list1.txt", package="metagene")
## Not run: groups$design <- data.frame(Samples=c(bamFileInput, bamFileControl), Exp1=c(1,2))
## Not run: groups$regionsGroups <- metagene:::prepareFeatures(features=features, specie="mouse", maxDistance=1
## Not run: groups$raw <- metagene:::parseBamFiles(groups$bamFilesDescription$bam, groups$regionsGroups)
## Not run: groups$rpm <- metagene:::rawCountsToRPM(groups$raw, groups$bamFilesDescription)
## Not run: groups$data <- metagene:::prepareGroups(names(groups$regionsGroups), bamFiles=groups$bamFilesDescri

 # Remove controls
## Not run: groups$data <- metagene:::applyOnGroups(groups=groups$data, cores=1, FUN=metagene:::removeControls,
```

---

scaleVector *Scale the values of a vector to fit with predetermined size*

Description

This function will resize the values of the vector so that they fit in a pre-defined range. The number of elements will changed, but the general distribution will remain the same.

Usage

```
scaleVector(
  values,
  domain)
```

Arguments

values	the values to scale
domain	the range to fit the value to

Value

scaleVector returns a vector with the scaled data

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: values <- 1:10
## Not run: scaledValues <- metagene:::scaleVector(values, 100)
```

`scaleVectors` *Resize the vectors of every group so they have the same length.*

Description

This function takes a list of list of vectors and scale them so they have the same size

Usage

```
scaleVectors(
  group,
  level,
  domain,
  scaleCores=1)
```

Arguments

`group`	A `list` that contains a list of list of vectors
`level`	The names of the element of the group to scale
`domain`	The target length for the vectors
`scaleCores`	Number of cores for parallel processing (require parallel package)

Details

A group can contain multiple list of list of vectors (i.e.: if `debug == TRUE`). We need to to specify which one to use with the `level` argument.

Each second level list can contain multiple list corresponding to each bam files associated with current group. Each vector of each bam files sublist will be scaled to the size specified with the `domain` argument.

Value

`scaleVectors` returns the same group that was used in input with an extra element named scaled.

Author(s)

Charles Joly Beauparlant <Charles.Joly-Beauparlant@crchul.ulaval.ca>

Examples

```
## Not run: group <- list(l1=list(element1=list(1:10, 1:100)))
## Not run: metagene:::scaleVectors(group, "l1", 100)
```

Index