Package 'Rseb'

January 11, 2021

Type Package

Title A package for NGS data managing and visualization

Version 0.1.3

Author Sebastian Gregoricchio

Maintainer Sebastian Gregoricchio <sebastian.gregoricchio@gmail.com>

Description
An R-package for daily tasks required to handle biological data as well as avoid re-coding of small functions for quick but necessary data managing.

License GNU GENERAL PUBLIC LICENSE version 3

Depends R (¿= 3.2.0), BiocManager, Biostrings, biomaRt, GO.db, rtracklayer, cowplot, data.table, dplyr, ggplot2, ggrepel, matrixStats, plyr, purrr, robustbase, stringr, tidyr, tools

biocViews

Imports Biostrings, biomaRt, GO.db, rtracklayer, cowplot, data.table, dplyr, ggplot2, ggrepel, matrixStats, plyr, purrr, robustbase, stringr, tidyr, tools

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

git_url https://github.com/sebastian-gregoricchio/Rseb/

R topics documented:

`build.bed` *Bed generator*

Description

Function that helps the building of a bed file providing the columns. It enables also the specification of the track line for software such as IGV in order to pre-define colors, track name, etc.

Usage

```
build.bed(
  chr,
  start,
  end,
  name = NULL,
  score = 0,
  strand = ".",
  thickStart = NULL,
  thickEnd = NULL,
  itemRgb = NULL,
```

```
  blockCount = NULL,
  blockSizes = NULL,
  blockStarts = NULL,
  track.name = NULL,
  display.mode = NULL,
  itemRgb.ON = T,
  useScore = F,
  colorByStrand = NULL,
  track.base.color = NULL,
  sort = T,
  bed.file.name = NULL,
  export.track.line = TRUE,
  return.data.frame = F,
  force.generation = F
)
```

Arguments

chr	String vector containing the name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
start	Numeric vector indicating the starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
end	Numeric vector indicating the ending position of the feature in the chromosome or scaffold.
name	String vector defining the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode. If set as NULL (default) and the column is required, the names will correspond to the mid-point of the region.
score	A single value or a numeric vector with a score between 0 and 1000. If the track line useScore attribute is set as TRUE for this annotation data set, the score value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). By default 0.
strand	A single character or a string vector defining the strand: either "." (=no strand) or "+" or "-". By default ".".
thickStart	A numeric vector indicating the starting position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part (default value, thickStart = NULL) it will be used the start value.
thickEnd	A numeric vector indicating the ending position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part (default value, thickStart = NULL) it will be used the end value.
itemRgb	A single value or a string vector containing the colors for each feature. It can be expressed as an RGB value of the form R,G,B (e.g. "255,0,0") or as any other R-supported color name (it will be converted automatically to RGB version). By default NULL. If the track line itemRgb.ON

attribute is set as TRUE, this color value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.

blockCount — A single number or a numeric vector indicating the number of blocks (exons) in the BED line. By default NULL.

blockSizes — A vector containing a comma-separated list of the block sizes. The number of items in this list should correspond to blockCount. By default NULL.

blockStarts — A vector containing a comma-separated list of block starts. All of the blockStart positions should be calculated relative to start. The number of items in this list should correspond to blockCount. By default NULL.

track.name — A string defining the track label that will be displayed to the left of the track in the Genome Browser window, and also the label of the track control at the bottom of the screen. The name can consist of up to 15 characters. It is recommended that the track_label be restricted to alphanumeric characters and spaces to avoid potential parsing problems. By default NULL.

display.mode — A string that defines the initial display mode of the annotation track. Values for display.mode include: "hide", "dense", "full", "pack", "squish". By default NULL.

itemRgb.ON — Logic value to define whether this attribute should be set to "On", the Genome Browser will use the RGB value shown in the itemRgb field in each data line of the associated BED track to determine the display color of the data on that line. If the itemRgb values are not provided, this parameter will be ignored. By default TRUE.

useScore — Logic value to define if the score field in each of the track's data lines should be used to determine the level of shading in which the data is displayed. By default FALSE.

colorByStrand — A vector composed by two strings for two colors, either in RGB comma separated format (eg. "0,250,30") or any R-supported color string (they will be converted automatically to RGB format). The order of color sets is c("strand +", "strand -"). Parameter ignored when itemRgb is active/provided. By default NULL.

track.base.color — A single string defining the main color for the annotation track. The track color consists of three comma-separated RGB values from 0-255 (eg. "0,250,30") or any R-supported color string (it will be converted automatically to RGB format). Parameter ignored when itemRgb or colorByStrand are active/provided. By default NULL.

sort — Logic value to define whether to sort the bed using the function sort.bed. By default TRUE.

bed.file.name — If a string with a full path to a bed_file is provided, the function will export the bed as a txt file. By default NULL.

export.track.line
: Logic value to define if the track line should be exported. When bed.file.name = NULL this parameter is ignored. By default TRUE.

return.data.frame
: Logic value to define if the to return the data.frame corresponding to the bed (it will show the columns names). By default FALSE.

force.generation
: Force the generation of bed even when certain errors occur (eg. score ¿ 1000, start ¿ end). By default FALSE.

Value

If required the function can export a bed file with or without the track line, return a data.frame (with column names) corresponding to the bed generated, or both. The bed file could be automatically sorted settin the parameter sort = TRUE.

References

- More information about bed format are available at the following link: https://genome.ucsc.edu/FAQ/FAQformat.html#format1.
- More information about track line parameters are available at the following link: https://genome.ucsc.edu/goldenPath/help/hgTracksHelp.html#lines.

---

calculate.mode *Mode calculation*

Description

Calculate the mode value of a vector of numeric values.

Usage

```
calculate.mode(v)
```

Arguments

v	A vector of numeric numbers

Value

A single number corresponding to the mode of the list of numbers give as input

Examples

```
mode = calculate.mode(v = c(6, 8, 4, 845, 8, 5, 55, 84, 8, 84, 45, 5))
```

cmyk *CMYK color converter*

Description

Converts CMYK color values to hexadecimal color values

Usage

```
cmyk(C, M, Y, K)
```

Arguments

C	Value in the 0-100 range for the Cyan component.
M	Value in the 0-100 range for the Magenta component.
Y	Value in the 0-100 range for the Yellow component.
K	Value in the 0-100 range for the Key component.

Value

The result is a string for the color in hexadecimal scale, eg. "#FFFFFF".

Examples

```
color = cmyk(0, 0, 0, 0)
```

combine.lists *List combiner*

Description

Combines two or more lists in a single one keeping the element names.

Usage

```
combine.lists(list.of.lists)
```

Arguments

list.of.lists	A list of lists.

Value

It returns a list that is a combination of the lists in the input list.
If the list is not a nested list of list the original input is returned.

Examples

```
combined_list = combine.lists(list.of.lists = list(list(c(1:2), c(1:3)), list("X" = c("A", "B"), "Y" = 2)))

combined_list = combine.lists(list.of.lists = list(c(1:2), c(1:3)))
```

`computeMatrix.deeptools` *Score matrix NGS data builder at specific regions (by* `deeptools/computeMatrix` *function).*

Description

This function runs a command line that uses `deeptools` to calculate scores per genome regions and to prepare an intermediate file that can be used with plot.density.profile. Typically, the genome regions are genes, but any other regions defined in a BED file can be used. computeMatrix accepts multiple score files (bigWig format) and multiple regions files (BED format). This tool can also be used to filter and sort regions according to their score.

Usage

```
computeMatrix.deeptools(
  mode,
  scoreFileName,
  regionsFileName,
  outFileName,
  outFileNameMatrix = NULL,
  outFileSortedRegions = NULL,
  referencePoint = "TSS",
  nanAfterEnd = FALSE,
  regionBodyLength = 1000,
  startLabel = "TSS",
  endLabel = "TES",
  unscaled5prime = 0,
  unscaled3prime = 0,
  upstream = 500,
  downstream = 500,
  binSize = 10,
  sortRegions = "keep",
  sortUsing = "mean",
  sortUsingSamples = NULL,
  averageTypeBins = "mean",
  missingDataAsZero = FALSE,
  skipZeros = FALSE,
  minThreshold = NULL,
  maxThreshold = NULL,
  blackListFileName = NULL,
```

```
  samplesLabel = NULL,
  smartLabels = TRUE,
  scale = 1,
  numberOfProcessors = "max",
  metagene = FALSE,
  transcriptID = "transcript",
  exonID = "exon",
  transcript_id_designator = "transcript_id",
  srun = FALSE,
  computeMatrix.deeptools.command = "computeMatrix",
  return.command = FALSE,
  run.command = TRUE,
  quiet = FALSE,
  verbose = FALSE
)
```

Arguments

`mode` The type of matrix computation. Allowed values are "reference-point" or "scale-region". No default.

- `reference-point`:
  Reference-point refers to a position within a BED region (e.g., the starting point). In this mode, only those genomicpositions before (upstream) and/or after (downstream) of the reference point will be plotted;
- `scale-region`:
  In the scale-regions mode, all regions in the BED file are stretched or shrunken to the length (in bases) indicated by the user.

`scoreFileName` String vector with the full paths to bigWig file(s) containing the scores to be plotted.

`regionsFileName`
String vector with the full paths to .BED or .GTF files containing the regions to plot. If multiple bed files are given, each one is considered a group that can be plotted separately. Also, adding a "#" symbol in the bed file causes all the regions until the previous "#" to be considered one group.

`outFileName` String containing the full file name to save the gzipped matrix file (.gz) needed by plot.density.profile.

`outFileNameMatrix`
If this option is given, then the matrix of values underlying the heatmap will be saved using the indicated name, e.g. IndividualValues.tab. This matrix can easily be loaded into R or other programs. By default `NULL`.

`outFileSortedRegions`
File name in which the regions are saved after skiping zeros or min/max threshold values. The order of the regions in the file follows the sorting order selected. This is useful, for example, to generate other heatmaps keeping the sorting of the first heatmap. Example: Heatmap1sortedRegions.bed. By default `NULL`.

`referencePoint` Possible choices: TSS, TES, center. The reference point for the plotting could be either the region start (TSS), the region end (TES) or the center of the region. Note that regardless of what you specify, plotHeatmap/plotProfile will default to using "TSS" as the label. By default `TSS`.

`nanAfterEnd` Logic value. If set (`TRUE`), any values after the region end are discarded. This is useful to visualize the region end when not using the scale-regions mode and when the reference-point is set to the TSS. By default `FALSE`.

`regionBodyLength`
Distance in bases to which all regions will be fit. (Default: 1000).

`startLabel` Label shown in the plot for the start of the region. Default is TSS (transcription start site), but could be changed to anything, e.g. "peak start". Note that this is only useful if you plan to plot the results yourself and not, for example, with plotHeatmap, which will override this. (Default: "TSS").

`endLabel` Label shown in the plot for the region end. Default is TES (transcription end site). See the –startLabel option for more information. (Default: "TES").

`unscaled5prime` Number of bases at the 5-prime end of the region to exclude from scaling. By default, each region is scaled to a given length (see the –regionBodyLength option). In some cases it is useful to look at unscaled signals around region boundaries, so this setting specifies the number of unscaled bases on the 5-prime end of each boundary. (Default: 0).

`unscaled3prime` Number of bases at the 3-prime end of the region to exclude from scaling. By default, each region is scaled to a given length (see the –regionBodyLength option). In some cases it is useful to look at unscaled signals around region boundaries, so this setting specifies the number of unscaled bases on the 3-prime end of each boundary. (Default: 0).

`upstream` Distance upstream of the reference-point selected. (Default: 500).

`downstream` Distance downstream of the reference-point selected. (Default: 500).

`binSize` Length, in bases, of the non-overlapping bins for averaging the score over the regions length. (Default: 10).

`sortRegions` Possible choices: "descend", "ascend", "no", "keep". Whether the output file should present the regions sorted. The default is to not sort the regions. Note that this is only useful if you plan to plot the results yourself and not, for example, with plotHeatmap, which will override this. Note also that unsorted output will be in whatever order the regions happen to be processed in and not match the order in the input files. If you require the output order to match that of the input regions, then either specify "keep" or use computeMatrixOperations to resort the results file. (Default: "keep").

`sortUsing` Possible choices: "mean", "median", "max", "min", "sum", "region_length". Indicate which method should be used for sorting. The value is computed for each row.Note that the region_length option will lead to a dotted line within the heatmap that indicates the end of the regions. (Default: "mean").

`sortUsingSamples`
: List of sample numbers (order as in matrix), that are used for sorting by –sortUsing, no value uses all samples, example: –sortUsingSamples 1 3. By default `NULL`.

`averageTypeBins`
: Possible choices: "mean", "median", "min", "max", "std", "sum". Define the type of statistic that should be used over the bin size range. (Default: "mean").

`missingDataAsZero`
: Logic value to define if set, missing data (NAs) will be treated as zeros. The default is to ignore such cases (`NULL`). If not included, this parameter can be changed later in the function plot.density.profile.

`skipZeros`
: Logic value to understand whether regions with only scores of zero should be included or not. Default is to include them (`FALSE`).

`minThreshold`
: Numeric value. Any region containing a value that is less than or equal to this will be skipped. This is useful to skip, for example, genes where the read count is zero for any of the bins. This could be the result of unmappable areas and can bias the overall results. (Default: `NULL`).

`maxThreshold`
: Numeric value. Any region containing a value greater than or equal to this will be skipped. The maxThreshold is useful to skip those few regions with very high read counts (e.g. micro satellites) that may bias the average values. (Default: `NULL`).

`blackListFileName`
: A BED file containing regions that should be excluded from all analyses. Currently this works by rejecting genomic chunks that happen to overlap an entry. Consequently, for BAM files, if a read partially overlaps a blacklisted region or a fragment spans over it, then the read/fragment might still be considered. (Default: `NULL`).

`samplesLabel`
: Labels for the samples. This will then be passed to plot.density.profile function. The default is to use the file name of the sample. The sample labels should be separated by spaces and quoted if a label itself contains a space E.g. –samplesLabel label-1 "label 2".

`smartLabels`
: Instead of manually specifying labels for the input bigWig and BED/GTF files, this causes deepTools to use the file name after removing the path and extension. (Default: `TRUE`).

`scale`
: If set, all values are multiplied by this number. (Default: 1).

`numberOfProcessors`
: Number of processors to use. Type "max/2" to use half the maximum number of processors or "max" to use all available processors. (Default: "max").

`metagene`
: When either a BED12 or GTF file are used to provide regions, perform the computation on the merged exons, rather than using the genomic interval defined by the 5-prime and 3-prime most transcript bound (i.e., columns 2 and 3 of a BED file). If a BED3 or BED6 file is used as input, then columns 2 and 3 are used as an exon. (Default: `FALSE`).

`transcriptID` When a GTF file is used to provide regions, only entries with this value as their feature (column 3) will be processed as transcripts. (Default: "transcript").

`exonID` When a GTF file is used to provide regions, only entries with this value as their feature (column 3) will be processed as exons. CDS would be another common value for this. (Default: "exon").

`transcript_id_designator`
Each region has an ID (e.g., ACTB) assigned to it, which for BED files is either column 4 (if it exists) or the interval bounds. For GTF files this is instead stored in the last column as a key:value pair (e.g., as 'transcript_id "ACTB"', for a key of transcript_id and a value of ACTB). In some cases it can be convenient to use a different identifier. To do so, set this to the desired key. (Default: "transcript_id").

`srun` Logic value to define whether the command should be run in `srun` mode. By default `FALSE`.

`computeMatrix.deeptools.command`
String to define the command to use to recall the computeMatrix function of deeptools. An example: "/home/user/anaconda3/bin/computeMatrix". By default `"computeMatrix"`.

`return.command` Logic value to define whether to return the string corresponding to the command for deeptools. By default `FALSE`.

`run.command` Logic value to define whether to run the the command line on system terminal and generate the score matrix by deeptools. By default `TRUE`.

`quiet` Logic value to define if to remove any warning or processing messages. By default `FALSE`.

`verbose` Logic value to define if to be VERY verbose in the status messages. –quiet will disable this. By default `FALSE`.

Details

To know more about the deeptools's `computeMatrix` function see the package manual at the following link:
https://deeptools.readthedocs.io/en/develop/content/tools/computeMatrix.html.

Value

The function generates the files indicated by the output parameters. The matrix.gz output file can be read by the function read.computeMatrix.file.

Examples

```
computeMatrix.deeptools(
   mode = "reference-point",
   scoreFileName = c("path_to/signal_file1.bw", "path_to/signal_file2.bw"),
   regionsFileName = c("path.to/regions1.bed", "path.to/regions2.bed"),
   upstream = 1000,
   downstream = 1000,
   outFileName = "path_to/output_matrix.gz",
```

```
   computeMatrix.deeptools.command = "/home/user/anaconda3/bin/computeMatrix",
   referencePoint = "peakMax")

computeMatrix.deeptools(
   mode = "scale-regions",
   scoreFileName = c("path_to/signal_file1.bw", "path_to/signal_file2.bw"),
   regionsFileName = c("path.to/regions1.bed", "path.to/regions2.bed"),
   upstream = 1000,
   downstream = 1000,
   regionBodyLength = 300,
   startLabel = "geneStart",
   endLabel = "geneEnd",
   outFileName = "path_to/output_matrix.gz",
   computeMatrix.deeptools.command = "/home/user/anaconda3/bin/computeMatrix",
   referencePoint = "peakMax")
```

`convert_sequence` *Nucleic acid sequences converter.*

Description

Obtains de complementary, reverse complementary or the reverse of a DNA/RNA sequence.

Usage

```
convert_sequence(sequence = NULL, mode = "not specified", nucleic.acid = "DNA")
```

Arguments

`sequence`	A string containing the sequence to be converted. By default `NULL`, it returns an help for the mode.
`mode`	A string value to define the modality of convertion. Possible options: - Reverse complement = revComp — RC — rc — reverseComplement - Reverse = rev — R — r — reverse - Complement = comp — C — c — complement. By default `"not specified"`, it returns an help for the mode.
`nucleic.acid`	A string to define the type of nucleic acid to which the input sequence belongs. Available options "DNA", default value, or "RNA".

Value

It returns a string with the converted sequence.

Examples

```
convert_sequence(sequence = "AATTTCCCGTCGAT",
                 mode = "reverse",
                 nucleic.acid = "DNA")
```

---

data.frame.to.list *Data frame conversion to a list of columns.*

Description

Converts each column of a data.frame in a element of a list with the corresponding name of the original column. Useful for further use in functions such as purrr::pmap().

Usage

```
data.frame.to.list(x)
```

Arguments

x	A data.frame to be converted

Value

A list of vectors in which each element is a column of input the data.frame.

Examples

```
data.frame.to.list(mtcars)
```

---

data.summary *Statistical data summary generator*

Description

Produces a table with a summary of the statistics for a specific column of an input data.frame by a group of values defined by a group defined by another column.

Usage

```
data.summary(data, variable, group.names)
```

Arguments

data	Input data.frame to be analyzed.
variable	A string with the name of the column to be analyzed.
group.names	A string with the name of the column indicating the groups.

Value

It returns a list that is a combination of the lists in the input list.
If the list is not a nested list of list the original input is returned.

Examples

```
data.summary(data = mtcars, variable = "mpg", group.names = "disp")
```

DE.status *Differential Expression status calculator for RNA-seq data*

Description

Defines the differential expression status of genes from RNA-seq data depending on fold change expression and adjusted p-value.

Usage

```
DE.status(
  log2FC,
  p.value.adjusted,
  FC_threshold = 1.5,
  FC_NoResp_left = 0.9,
  FC_NoResp_rigth = NULL,
  p.value_threshold = 0.05,
  low.FC.status.label = "DOWN",
  high.FC.status.label = "UP",
  unresponsive.label = "NoResp",
  null.label = "NULL"
)
```

Arguments

`log2FC`	Numeric vector of log2(fold change expression) values.
`p.value.adjusted`	Numeric vector of p-values. Use of adjusted p-values is recommended.
`FC_threshold`	Value of the threshold to use for the fold change expression to define differentially expressed genes, expressed as linear value. By default 1.5 and by consequence 1/1.5.
`FC_NoResp_left`	Value of the threshold to use for the fold change expression to define unresponsive genes when `FC < 1`, expressed as linear value. By default 0.9. If `NULL` it will be calculated symmetrically from `FC_NoResp_rigth` as 1/`FC_NoResp_rigth`.

FC_NoResp_rigth Value of the threshold to use for the fold change expression to define unresponsive genes when FC > 1, expressed as linear value. By default 1.1. If NULL it will be calculated symmetrically from FC_NoResp_left as 1/FC_NoResp_left.

p.value_threshold
Value of the threshold to use for the p-values to define differentially expressed genes, expressed as linear value. By default 0.05.

low.FC.status.label
String to define the label indicating the differentially expressed genes with a FoldChange < FC_threshold.

high.FC.status.label
String to define the label indicating the differentially expressed genes with a FoldChange > FC_threshold.

unresponsive.label
String to define the label indicating the unresponsive genes identified as FC_NoResp_left < FoldChange < FC_NoResp_rigth and p.value > p.value.threshold.

null.label String to define the label indicating the null genes.

Value

It returns a vector containing the differential expression status for each original value in the same order used in the input.

`density_plot` *Plot density signal of NGS data.*

Description

Plots the density profile of NGS data (e.g. ChIP-seq, ATAC-seq, MeDIP-seq, etc.). Used by the function `plot.density.profile`.

Usage

```
density_plot(
  samples,
  scores,
  positions,
  variance_scores,
  xlab = "Distance from regions center [bp]",
  ylab = "Average density signal",
  line_type = "solid",
  y_lim = NULL,
  x_lim = NULL,
  x_intercept = 0,
  colors = c("blue", "red", "purple", "orange", "green"),
  title = "Density profile",
```

```
  text_size = 12,
  variance = T,
  print_plot = F,
  line_width = 1,
  variance_opacity = 0.25
)
```

Arguments

samples	A character vector containing the samples list.
scores	A numeric vector containing the scores for the Y-axis.
positions	A numeric vector containing the position for the X-axis.
variance_scores	A numeric vector containing the variance/error value at each position.
xlab	A string containing the label for the X-axis. By default "Distance from regions center [bp]".
ylab	A string containing the label for the Y-axis. By default "Average density signal".
line_type	Vector to define each line type. Both numeric and string codes are accepted. if only one element is given this will be applied to all the lines. By default "solid". Example 1: `c("solid","dashed")`. Example 2: `c(1,2)`
y_lim	List of numeric vectors with two elements each to define the range of the Y-axis. To set only one side use NA for the side to leave automatic. If only one range is given this one will be applied to all the plots. By default `NULL`, the range will be defined automatically. Example `list(c(0,20),c(NA,30),c(0,NA),c(NA,NA))`.,
x_lim	List of numeric vectors with two elements each to define the range of the X-axis. To set only one side use NA for the side to leave automatic. If only one range is given this one will be applied to all the plots. By default `NULL`, the range will be defined automatically. Example `list(c(0,20),c(NA,30),c(0,NA),c(NA,NA))`.,
x_intercept	A vector indicating the X intercepts for the vertical lines. By default 0.
colors	Vector to define the line and error area colors. If only one value is provided or the number of values is lower than the required ones only the first value will be used. All standard R.colors values are accepted. By default `c("blue","red","purple","orange","green")`.
title	A string containing the label for the X-axis. By default "Density profile".
text_size	Numeric value to define the size of the text for the labels of all the plots. By default 12.
variance	Logic value to define whether to plot the error/variance around the signal. By default `TRUE`.
print_plot	Logic value to define whether to print the plot once generated or not. By default `FALSE`.

`line_width` Numeric value to define the line width for all the plots. By default 1.,

`variance_opacity`
Numeric value to define the alpha/transparency of the error/variance. By default 0.25. Parameter considered only when `variance = TRUE)`.

Value

Returns a plot in ggplot2 format.

`doughnut` *Donut/Doughnut plot*

Description

Generation of a donut/doughnut plot (equivalent of a pie chart)

Usage

```
doughnut(
  x,
  labels = as.character(x),
  edges = 200,
  outer.radius = 0.8,
  inner.radius = 0.4,
  clockwise = FALSE,
  init.angle = if (clockwise) 90 else 0,
  density = NULL,
  angle = 45,
  col = NULL,
  border = FALSE,
  lty = NULL,
  main = NULL,
  ...
)
```

Arguments

`x`	A vector containing the values to be plotted.
`labels`	A string vector for the labels of the different sectors. By default as.character(x).
`edges`	Number of edges of the shape. By default 200.
`outer.radius`	Fraction of the area to dedicate to the outer circle. By default 0.8.
`inner.radius`	Fraction of the area to dedicate to the inner circle. By default 0.4.
`clockwise`	Logic value to define whether the values should be plotted in clockwise sense. By default `FALSE`.
`init.angle`	Numeric value to define the starting angle for the data. By default if `clockwise = TRUE` 90, otherwise 0.

density	A vector or single number to define de density of the lines in the filling color of each value plotted. By default NULL.
angle	A vector or single number to define de angle of the lines in the filling color of each value plotted. By default 45.
col	A vector of R standard colors for each value to be plotted. By default NULL.
border	Logic value to define whether plot the border of the sectors. By default FALSE.
lty	Numeric value to define the type of line for the borders. By default NULL.
main	String to set the title of the plot. By default NULL.

References

https://magesblog.com/

Examples

```
doughnut(x = c(3,5,9,12), inner.radius=0.5, col=c("red", "blue", "green", "yellow"))
```

---

get.gene.name *Conversion of ENSEMBL gene IDs.*

Description

Conversion of ENSEMBL gene IDs to gene symbols.

Usage

```
get.gene.name(ensembl.id, type = "gene", organism = "mmusculus")
```

Arguments

ensembl.id	String vector of ENSEMBL genes IDs
type	String to define the type of ENSEMBL inputs. By default gene to indicate "ensembl_gene_id". If different from "gene" it will be set to "ensembl_transcript_id_version".
organism	String to define de organism, e.g. mmusculus, hsapiens, etc. By default mmusculus.

Value

A string vector with the corresponding gene_symbols.

Examples

```
gene_symbols =
get.gene.name(
   ensembl.id = c("ENSMUSG00000002111", "ENSMUSG00000027381"),
   type = "gene",
   organism = "mmusculus")
```

`grepl.data.frame` *Grep a pattern in a full data.frame.*

Description

The function helps to define which rows of an input data.frame contain a specific patter.

Usage

```
grepl.data.frame(
  data.frame,
  pattern,
  ignore.case = FALSE,
  perl = FALSE,
  fixed = FALSE,
  useBytes = FALSE
)
```

Arguments

`data.frame`	Input data.frame.
`pattern`	Character string containing a regular expression (or character string for `fixed = TRUE`) to be matched in the given character vector. Coerced by `as.character` to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are allowed except for `regexpr` and `gregexpr`.
`ignore.case`	If `FALSE`, the pattern matching is case sensitive and if `TRUE`, case is ignored during matching. By default `FALSE`.
`perl`	Logical value to define if Perl-compatible regexps should be used. By default `FALSE`.
`fixed`	Logical value to define if the pattern is a string to be matched as is. Overrides all conflicting arguments. By default `FALSE`.
`useBytes`	Logical value to define if the matching is done byte-by-byte rather than character-by-character. By default `FALSE`.

Value

It will be return a logic vector with an element per each row of the data.frame. The value is `TRUE` when the patter is found at least once in the corresponding data.frame row.

Examples

```
iris = iris %>% filter(grepl.data.frame(iris, pattern = "setosa"))
```

GSEA.to.GOnumber *Conversion of GSEA terms into Gene Ontology numbers*

Description

Helps to convert the terms of GSEA analyses into Gene Ontology (GO) ID numbers.

Usage

```
GSEA.to.GOnumber(
  input_terms,
  input_pvalue,
  return_table = T,
  export_table = F,
  output_file_name = paste(getwd(), "GO_numbers_table.tsv", sep = "/")
)
```

Arguments

input_terms	A character vector containing the GSEA terms to be converted.
input_pvalue	A numeric vector containing the p-values of the GSEA terms.
return_table	Logic value to define whether to return the resulting data.frame. By default TRUE.
export_table	Logic value to define whether to export the resulting data.frame. By default FALSE.
output_file_name	Path and file name of the output table if export is required. By default <working.directory>/GO_numbers_table.tsv.

Details

This functions requires the package GO.db.
If problems are encountered during the installation see https://www.biostars.org/p/50564/.

Value

If required, returns a data.frame with 3 columns: GO_number, GO_annotation, p.value. This table could be directly exported.

IGVsnap *Script generator for Integrative Genomics Viewer (IGV) batch tasks.*

Description

The function builds a script file that can be run on IGV to generate multiple screenshots at specific genomic regions.

Usage

```
IGVsnap(
  loci_vector,
  input_type,
  biomart = "ensembl",
  dataset = "mmusculus_gene_ensembl",
  reference_genome = NULL,
  fivePrime = 1000,
  threePrime = 1000,
  snap_names = NULL,
  IGV_batch_file = paste(getwd(), "/IGV_batch.txt", sep = ""),
  snap_image_format = "png",
  snap_directory = getwd(),
  maxPanelHeight = 1000,
  session = NULL,
  exit = FALSE
)
```

Arguments

loci_vector	Either a gene name vector (e.g. `c("Gapdh","Spi1",...)`) or a regions vector (eg. `c('chr1:253000-256503',...)`. All IGV formats are allowed.
input_type	Define the input type. Allowed values are `genes` and `regions`.
biomart	Defines the `biomart` parameter for `biomaRt` package, by default `ensembl`.
dataset	Defines the `dataset` parameter for `biomaRt` package, by default `mmusculus_gene_ensembl`.
reference_genome	[optional] Defines the genome to use, e.g. "mm10", "hg19", ... By default `NULL`.
fivePrime	Numeric value to define how many bases [bp] exapand from full gene position at it's 5'-end, default 1000bp.
threePrime	Numeric value to define how many bases [bp] exapand from full gene position at it's 3'-end, default 1000bp.
snap_names	[optional] String vector to define the names of images (without extention), by default uses `loci_vector`.
IGV_batch_file	String for the batch_script_file_name/path, by default `<working_directory>/IGV_batch.txt`.

`snap_image_format`
: String to define the format of the images, e.g. "png", "jpeg", "svg", ... By default `png`.

`snap_directory`
: String for the output directory for the snapshoots. By default ¡working_directory¿.

`maxPanelHeight`
: Numeric value to define the height in pixel of the IGV pannel that will be captured on IGV.

`session`
: [optional] FULL path to an IGV session file (session.xml) to use for the images. By default `NULL`.

`exit`
: Logical value to indicate whether exit IGV after image capture ended. By default `FALSE`.

Details

To run the script on IGV: Tools ¿ Run Batch Script... ¿ choose the .txt output file from this function.
For more info on how batch tasks work on IGV see:
https://software.broadinstitute.org/software/igv/PortCommands.

Value

Exports a .txt file ready-to-use on IGV.

`install.pkg.source` *Package installer from source archive.*

Description

Allows the installation of R packages using the source archive file.

Usage

```
install.pkg.source(pkg.path)
```

Arguments

`pkg.path`
: String to define the path for the archive file to be installed.

Value

No returned value. The package required will be installed.

`intersect.bedtools` *Intersect two or more bed files (by* `bedtools intersect` *function).*

Description

This function runs a command line that uses `bedtools intersect` to intersect one or more .bed files.

Usage

```
intersect.bedtools(
  a,
  b,
  outputFileName = paste(getwd(), "intersected.bed", sep = "/"),
  abam = FALSE,
  ubam = FALSE,
  bed = FALSE,
  wa = FALSE,
  wb = FALSE,
  loj = FALSE,
  wo = FALSE,
  wao = FALSE,
  u = FALSE,
  c = FALSE,
  C = FALSE,
  v = FALSE,
  f = NULL,
  F. = NULL,
  r = NULL,
  e = NULL,
  s = FALSE,
  S = FALSE,
  split = FALSE,
  sorted = FALSE,
  g = NULL,
  srun = FALSE,
  intersect.bedtools.command = "intersectBed",
  return.command = FALSE,
  return.bed = FALSE,
  delete.output = FALSE,
  run.command = TRUE
)
```

Arguments

a	A single string defining the BAM/BED/GFF/VCF file "A". Each feature in A is compared to B in search of overlaps. Use "stdin" if passing A with

	a UNIX pipe.
b	A character vector with one or more BAM/BED/GFF/VCF file(s) "B". It could be also a single string containing wildcard (*) character(s).
outputFileName	Full path to output file name. By default <working.directory>/intersected.bed.
abam	Logic value to define if file A is a BAM. Each BAM alignment in A is compared to B in search of overlaps. By default FALSE.
ubam	Logic value to define if to write the output as uncompressed BAM. The default is to write compressed BAM output (ubam = FALSE).
bed	Logic value to define whether to write output as BED when using a BAM input abam = TRUE. The default is to write output in BAM (bed = FALSE).
wa	Logic value to define if to write the original entry in A for each overlap. By default FALSE.
wb	Logic value to define if to write the original entry in B for each overlap. Useful for knowing what A overlaps. Restricted by -f and -r. By default FALSE.
loj	Logic value to define if to perform a "left outer join". That is, for each feature in A report each overlap with B. If no overlaps are found, report a NULL feature for B. By default FALSE.
wo	Logic value to define if to write the original A and B entries plus the number of base pairs of overlap between the two features. Only A features with overlap are reported. Restricted by -f and -r. By default FALSE.
wao	Logic value to define if to write the original A and B entries plus the number of base pairs of overlap between the two features. However, A features w/o overlap are also reported with a NULL B feature and overlap = 0. Restricted by -f and -r. By default FALSE.
u	Logic value to define if to write original A entry once if any overlaps found in B. In other words, just report the fact at least one overlap was found in B. Restricted by -f and -r. By default FALSE.
c	Logic value to define if to for each entry in A, report the number of hits in B while restricting to -f. Reports 0 for A entries that have no overlap with B. Restricted -f, -F, -r, and -s. By default FALSE.
C	Logic value to define if to for each entry in A, separately report the number of overlaps with each B file on a distinct line. Reports 0 for A entries that have no overlap with B. Overlaps restricted by -f, -F, -r, and -s. By default FALSE.
v	Logic value to define if to only report those entries in A that have no overlap in B. Restricted by -f and -r.
f	Numeric value defining the minimum overlap required as a fraction of A. Default is 1E-9 (i.e. 1bp). By default NULL.
F.	Numeric value defining the minimum overlap required as a fraction of B. Default is 1E-9 (i.e., 1bp). By default NULL.
r	Numeric value defining the required reciprocal fraction of overlap for A and B. In other words, if -f is 0.90 and -r is used, this requires that B overlap at least 90% of A and that A also overlaps at least 90% of B. By default NULL.

`e` Numeric value defining the minimum fraction to be satisfied for A _OR_ B. In other words, if -e is used with -f 0.90 and -F 0.10 this requires that either 90% of A is covered OR 10% of B is covered. Without -e, both fractions would have to be satisfied. By default `NULL`.

`s` Logic value to define if to force "strandedness". That is, only report hits in B that overlap A on the same strand. By default, overlaps are reported without respect to strand. By default `FALSE`.

`S` Logic value to define if to require different strandedness. That is, only report hits in B that overlap A on the _opposite_ strand. By default, overlaps are reported without respect to strand. By default `FALSE`.

`split` Logic value to define if to treat "split" BAM (i.e., having an "N" CIGAR operation) or BED12 entries as distinct BED intervals. By default `FALSE`.

`sorted` Logic value to define, for very large B files, if to invoke a "sweeping" algorithm that requires position-sorted input. When using -sorted, memory usage remains low even for very large files. By default `FALSE`. It is possible to sort a bed file on terminal by `(sort -k1,1 -k2,2n unsorted.bed > sorted.bed)` or by the function sort.bed.

`g` Specify a genome file the defines the expected chromosome order in the input files for use with the -sorted option. By default `NULL`.

`srun` Logic value to define whether the command should be run in `srun` mode. By default `FALSE`.

`intersect.bedtools.command` String to define the command to use to recall the `bedtools intersect` function. An example: "/home/user/anaconda3/bin/intersectBed". By default `"intersectBed"`.

`return.command` Logic value to define whether to return the string corresponding to the command for bedtools. By default `FALSE`.

`return.bed` Logic value to define whether to return the resulting bed as data.frame. By default `FALSE`. Parameter not active when inputs are bam files.

`delete.output` Logic value to define whether to delete the exported intersected bed file. By default `FALSE`. Parameter active only when `return.bed = TRUE`. Useful when is sufficient to get the result as a data.frame without saving it.

`run.command` Logic value to define whether to run the the command line on system terminal and generate the bed resulting from the intersection. By default `TRUE`.

Details

To know more about the `bedtools intersect` function see the package manual at the following link:
https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html.

Value

The function generates the files indicated by the output parameters. If required the command line used and/or the resulting intersected bed file. If both outputs are required, the output will be a named list with two values: "command" and "intersected.bed".

Examples

```
intersect.bedtools(a = bed_file1.bed,
                   b = c("bed_file2.bed", "bed_file3.bed"),
                   wb = TRUE,
                   intersect.bedtools.command = "/home/user/anaconda3/bin/intersectBed")

intersect.bedtools(a = bed_file1.bed,
                   b = c("bed_file2.bed", "bed_file3.bed"),
                   wa = TRUE,
                   return.bed = TRUE,
                   delete.output = T,
                   intersect.bedtools.command = "/home/user/anaconda3/bin/intersectBed")
```

---

is.nan_df — is.nan() *applied to a data.frame*

Description

Applies the function `is.nan()` to a full data.frame.

Usage

```
is.nan_df(data.frame)
```

Arguments

`data.frame`	Input data.frame.

Value

It returns a matrix/array containing logic values for each element of the input data.frame. When `TRUE` it means that the corresponding element is a `NaN`.

Examples

```
is.nan.df(mtcars)
```

`mass.to.volume` *Get solvent volume to make a solution with a given amount of a compound.*

Description

Given a specific ammount of solute calculates the volume of solvent necessary to obtain a certain final molarity concentration.

Usage

```
mass.to.volume(
  final_concentration,
  final_concentration_unit = "M",
  mass,
  mass_unit = "g",
  MW
)
```

Arguments

`final_concentration`	Numeric value for the final concentration wanted.
`final_concentration_unit`	String to define the unit of the final concentration wanted. Available units are: "M", "mM", "uM", "nM", "pM", "fM". By default "M".
`mass`	Numeric value for the solute mass ammount.
`mass_unit`	String to define the unit of the mass. Available units are: "kg", "g", "mg", "ug", "ng". By default "g".
`MW`	Numeric value for the Molecular Weigth (MW) of the compound expressed in g/mol.

Value

It returns a string with the volume of solvent to use.

Examples

```
mass.to.volume(final_concentration = 5, mass = 10, MW = 215)
```

`molarity.to.mass` *Get solvent volume to make a solution with a given amount of a compound.*

Description

Given a specific volume of solution wanted calculates the mass of solute necessary to obtain a certain final molarity concentration.

Usage

```
molarity.to.mass(
  final_concentration,
  final_concentration_unit = "M",
  final_volume,
  final_volume_unit = "mL",
  MW
)
```

Arguments

`final_concentration`	Numeric value for the final concentration wanted.
`final_concentration_unit`	String to define the unit of the final concentration wanted. Available units are: "M", "mM", "uM", "nM", "pM", "fM". By default "M".
`final_volume`	Numeric value for the final volume wanted.
`final_volume_unit`	String to define the unit of the volume. Available units are: "L", "mL", "uL". By default "mL".
`MW`	Numeric value for the Molecular Weigth (MW) of the compound expressed in g/mol.

Value

It returns a string with the mass of compound to use.

Examples

```
molarity.to.mass(final_concentration = 5, final_volume = 10, MW = 215)
```

`move.df.col` *Function to change easily the order of specific columns in a data.frame.*

Description

Allows to change the position of a column in a data.frame using other columns as reference.

Usage

```
move.df.col(data.frame, move.command)
```

Arguments

`data.frame`	An input data.frame.
`move.command`	A string containing the moving command. The command is formed as follows: "columnA movingCommand columnB". The basic options are: "first", "last", "before", "after". Compounded moves must be separated by a semicolon. Example: `"g first; a last; e before c"`.

Value

It returns the original data.frame but with the columns moved as demanded.

References

https://stackoverflow.com/questions/3369959/moving-columns-within-a-data-frame-without-retyping

Examples

```
new.mtcars = move.df.col(mtcars, "mpg last")

new.mtcars = move.df.col(mtcars, "wt before carb")

new.mtcars = move.df.col(mtcars, "am before carb; cyl first")
```

`pkg.check` *Check package installation.*

Description

Function to check if a package is installed. It works with bioconductor or CRAN packages.

Usage

```
pkg.check(package, archive)
```

Arguments

package	A single string indicating the name of the package to check.
archive	A single string indicating the type of archive. Possible values "CRAN" and "bioconductor" (not case sensitive). Parameter without default..

Value

If the pkg is not already installed it will be installed.

Examples

```
pkg.check("ggplot2", "cran")

pkg.check("biomaRt", "bioconductor")
```

---

pkg.version *Get session info and package versions.*

Description

Retrieves the information of the current session and the version of the packages loaded.

Usage

```
pkg.version(return.session = F, print.versions = T, return.versions = F)
```

Arguments

return.session	Logic value to define if to save the session info. By default FALSE.
print.versions	Logic value to define if to print the session and version info. By default TRUE.
return.versions	Logic value to define if to save package versions info. By default FALSE.

Value

If return.session and/or return.versions TRUE a list with these informations is returned. Otherwise nothing is returned.

plot.density.profile *Plot of NGS density signal at specific regions from deepTools matrices.*

Description

Plots the density profile of NGS data signals, using as input a score matrix computed by deeptools's computeMatrix function or by computeMatrix.deeptools from this package.

Usage

```
## S3 method for class 'density.profile'
plot(
  matrix.file,
  plot.by.group = T,
  missing.data.as.zero = NULL,
  sample.names = NULL,
  region.names = NULL,
  signal.type = "mean",
  error.type = "sem",
  plot.error = T,
  error.transparency = 0.125,
  title = NULL,
  x.lab = NULL,
  y.lab = NULL,
  line.type = "solid",
  line.width = 0.5,
  x.lim = NULL,
  y.lim = NULL,
  y.identical.auto = T,
  y.ticks.number = 5,
  text.size = 12,
  plot.vertical.lines = T,
  write.reference.points = T,
  colors = c("#00A5CF", "#F8766D", "#AC88FF", "#E08B00", "#00BA38", "#BB9D00",
    "#FF61C9", "gray30"),
  n.row.multiplot = 1,
  export.multiplot = F,
 multiplot.export.file = paste(getwd(), "/multiplot.", Sys.Date(), "_", gsub(pattern =
    ":", replacement = ".", x = format(Sys.time(), "%X")), ".pdf", sep = ""),
  real.width.single.plot = 2.5,
  real.height.single.plot = 3,
  print.multiplot = F
)
```

Arguments

matrix.file	A single string indicating a full path to a matrix.gz file generated by deepTools/computeMatrix or by computeMatrix.deeptools, or a list generated by the function read.computeMatrix.file.
plot.by.group	Logic value to define whether plot by group of regions or by sample. By default TRUE.
missing.data.as.zero	Logic value to define whether treat missing data as 0. If set as FALSE missing data will be converted to NA and will be excluded from the computations of the signal. By default TRUE.
sample.names	Samples names could be defined by a string vector. If set as NULL sample names will be get automatically by the matrix file. By default NULL. Example: c("sample1","sample2","sample3")
region.names	Region names could be defined by a string vector. If set as NULL sample names will be get automatically by the matrix file. By default NULL. Example: c("regionA","regionB")
signal.type	String indicating the signal to be computed and plotted. Available parameters are "mean", "median" and "sum". By default "mean".
error.type	String indicating the type of error to be computed and plotted. Available parameters are "sem" and "sd", standard error mean and standard deviation respectively. By default "sem". Parameter considered only when plot.error = TRUE).
plot.error	Logic value to define whether to plot the error around the signal. By default TRUE.
error.transparency	Numeric value to define the alpha/transparency of the error. By default 0.125. Parameter considered only when plot.error = TRUE).
title	Title of each plot could be defined by a string vector. If set as NULL titles will be generated automatically. By default NULL. Example: c("Title1","Title2")
x.lab	Single string to define the X-axis label for all the plots. By default NULL, the label will be defined automatically.
y.lab	Single string to define the Y-axis label for all the plots. By default NULL, the label will be defined automatically.
line.type	Vector to define each line type. Both numeric and string codes are accepted. If only one element is given this will be applied to all the lines. By default "solid". Example 1: c("solid","dashed"). Example 2: c(1,2)
line.width	Numeric value to define the line width for all the plots. By default 0.5.
x.lim	List of numeric vectors with two elements each to define the range of the X-axis. To set only one side use NA for the side to leave automatic. If only one range is given this one will be applied to all the plots. By default NULL, the range will be defined automatically. Example list(c(0,20),c(NA,30),c(0,NA),c(NA,NA)).,

`y.lim` List of numeric vectors with two elements each to define the range of the Y-axis. To set only one side use NA for the side to leave automatic. If only one range is given this one will be applied to all the plots. By default `NULL`, the range will be defined automatically.
Example `list(c(0,20),c(NA,30),c(0,NA),c(NA,NA))`.,

`y.identical.auto` Logical value to define whether use the same Y-axis range for all the plots automatically depending on the values. Not used when `y.lim` is not `NULL`. By default `TRUE`.

`y.ticks.number` Define the number of ticks to display in the Y-axis. By default 5. Active only when `y.identical.auto = TRUE`.

`text.size` Numeric value to define the size of the text for the labels of all the plots. By default 12.

`plot.vertical.lines` Logic value to define whether to plot a dashed gray vertical line in correspondence of the reference points of each plot. By default `TRUE`.

`write.reference.points` Logic value to define whether to indicate the reference points on each plot. Applied only when `x.lim` is `NULL`. By default `TRUE`.

`colors` Vector to define the line and error area colors. If only one value is provided it will applied to all the samples/groups. If the number of values is lower than the the required one, a random set of colors will be generated. All standard R.colors values are accepted. By default `c("#00A5CF","#F8766D","#AC88FF","#E08B00","#00BA38","#BB9D00","#FF61C9","gray30")`.

`n.row.multiplot` Numeric value to define the number of rows in the final multiplot.

`export.multiplot` Logic value to define whether to export the multiplot generated by the function. By default `FALSE`.

`multiplot.export.file` Name of the PDF file of the multiplot to be exported when `export.multiplot = T`.
By default "/working_directory/multiplot_current.date_current.time.pdf".

`real.width.single.plot` Numeric value, in inches, to define the real width of each plot in the multiplot exported, if required. By default 2.5 inches.

`real.height.single.plot` Numeric value, in inches, to define the real height of each plot in the multiplot exported, if required. By default 3 inches.

`print.multiplot` Logic value to define whether to print the multiplot once created. By default `FALSE`.

Details

To know more about the deepTools's function `computeMatrix` see the package manual at the following link:
https://deeptools.readthedocs.io/en/develop/content/tools/computeMatrix.html.

Value

The functions returns a list containing:

- `data.table` with the computed values used for the plot;
- `metadata` table with the information gotten from the matrix_file.gz;
- `plot.list` with a plot for each list element;
- `multiplot` with the image of all the plots together.

Examples

```
plot.density.profile(
  matrix.file = "/path.to/matrix.file.gz", plot.by.group = TRUE,
  missing.data.as.zero = NULL, sample.names = NULL, region.names = NULL,
  signal.type = "mean", error.type = "sem", plot.error = TRUE,
  error.transparency = 0.125, title = NULL, x.lab = NULL, y.lab = NULL,
  line.type = "solid", line.width = 0.5, x.lim = NULL, y.lim = NULL,
  y.identical.auto = TRUE, y.ticks.number = 5, text.size = 12,
  plot.vertical.lines = TRUE, colors = c("red", "blue", "#00BA38"),
  n.row.multiplot = 1, export.multiplot = FALSE,
  multiplot.export.file = "/path.to/multiplot.pdf",
  real.width.single.plot = 2.5, real.height.single.plot = 3,
  print.multiplot = FALSE)
```

---

pStars *P-value significance stars definer.*

Description

Converts a p-value score in equivalent stars of significance.

Usage

```
pStars(p.value, one = 0.05, two = 0.01, three = 0.001, four = 1e-04)
```

Arguments

`p.value`	A single numeric value indicating the p-value to evaluate.
`one`	A numeric value to define the p-value threshold for the first level of significance (*). By default 0.05.
`two`	A numeric value to define the p-value threshold for the second level of significance (**). By default 0.01.
`three`	A numeric value to define the p-value threshold for the third level of significance (***). By default 0.001.
`four`	A numeric value to define the p-value threshold for the fourth level of significance (****). By default 0.0001.

Value

It returns a string with the corresponding level of significance: NS, *, **, ***, ****.

Examples

```
significance = pStars(0.002)

require(dplyr)
data.frame =
   data.frame %>%
   mutate(p.stars = sapply(data.frame$p.value.column, pStars))
```

---

read.computeMatrix.file

computeMatrix **.gz file reader*

Description

The function reads a matrix.file.gz generated by deepTools/computeMatrix function or by computeMatrix.deeptools. The value can be passed to plot.density.profile function.

Usage

```
read.computeMatrix.file(matrix.file)
```

Arguments

matrix.file	A string indicating indicating the full path to the matrix.file.gz generated by deepTools/computeMatrix function or by computeMatrix.deeptools.

Value

The functions returns a named list containing:

- metadata data.frame with the information gotten from the matrix_file.gz;
- matrix.data data.frame with the scores gotten from;
- original.file.path with full path to the original matrix_file.gz.

This list can be passed as it is to the function plot.density.profile.

`restore_packages` *Restores packages installed from a .rda file.*

Description

Installs the packages contained in a .rda file. This file can be generated by the `store_packages` function of this package.

Usage

```
restore_packages(rda_file)
```

Arguments

`rda_file`	Path to the .rda from which get the information for the packages to re-install.

Value

If it was not possible to re-install al packages, the list of not restored packages will be returned.

`restriction.sites.to.bed` *Generator of a bed file for enzymatic restriction sites.*

Description

The function allows to create a bed file that can be added on IGV both as regions and track. It will show the restriction sites of a sequences if starting from the cut positions depending on sequence lenght. Chromosome, start and end of the input sequence are required.

Usage

```
restriction.sites.to.bed(
  cut_positions,
  chromosome,
  genome_start,
  return_bed = TRUE,
  export_bed_file = FALSE,
  output_file_name = paste(getwd(), "restriction_positions.bed", sep = "/"),
  enzyme_cut_length = 4,
  include_region_description = TRUE,
  region_name = "site",
  append = FALSE
)
```

Arguments

`cut_positions`	A numeric vector with the list of the restriction/cut positions.
`chromosome`	Chromosome number of the region analyzed.
`genome_start`	Start position on the genome of the region analyzed.
`return_bed`	Logic value to define if to return the bed as data.frame. By default `TRUE`.
`export_bed_file`	Logic value to define if to export the resulting .bed file. By default `FALSE`.
`output_file_name`	String corresponding to the path to the exported .bed file. By default `"<working.directory>/restriction_positions.bed"`.
`enzyme_cut_length`	Numeric value to define the length of cut of the restriction enzyme. By default 4.
`include_region_description`	Logic value to define whether to include a fourth column containing the region name define by the parameter `region_description`. By default `TRUE`.
`region_name`	Regions base name. Automatically it will be added a number to the base name. By default `"site"`, the resulting regions will be: site_1, site_2, ...
`append`	Logic value to define if to append the result to the file. By default `FALSE`, the file will be overwritten.

Details

To map the positions of restriction enzymes it is possible to use http://restrictionmapper.org/ with the option `Map (version 3)`.

Value

If required, it will be returned a classic bed file (chr start end [name]) with the regions centered on the cut position in the genome.

Examples

```
restriction.sites.to.bed(cut_positions = c(230, 235, 1250, 36),
                         chromosome = 10,
                         genome_start = 1205126,
                         region_name = "EcoRI_cut_site")
```

`sort.bed` *Sorter function for .bed files.*

Description

Sorts .bed files by chromosome and position.

Usage

```
## S3 method for class 'bed'
sort(
  bed,
  bed.header = F,
  sep = "\t",
  return.bed = T,
  export.bed = F,
  export.file.name = paste(getwd(), "sorted.bed", sep = "/"),
  export.header = F
)
```

Arguments

`bed`	Two options are possible: - String with the path to a .bed file; - Data.frame corresponding to a bed file format (all the columns and their names will be kept).
`bed.header`	Logic value to define whether the .bed file contains an header or not. By default `FALSE`.
`sep`	String containing the separator character for a .bed file. By default `"\t"`.
`return.bed`	Logic value to define if to return the bed as a data.frame. By default `TRUE`. Only unique rows are kept.
`export.bed`	Logic value to define if to export the bed file. By default `FALSE`. Only unique rows are kept.
`export.file.name`	String to define the path to the file to be exported, if required. By default `"<working.directory>/sorted.bed"`.
`export.header`	Logic value to define whether the header should be exported in the sorted bed file. By default `FALSE`.

Details

The function keeps only unique rows.
To get more information about the bed file format see the following page:
https://genome.ucsc.edu/FAQ/FAQformat.html#format1.

Value

If required, returns a data.frame corresponding to the sorted .bed file.

`store_packages` *Stores the information of installed packages in a .rda file.*

Description

Saves the list of all the installed packages in a .rda file. This file can be used to restore the packages from a computer to another or after installation of a new R version by the function `restore_packages` of this package.

Usage

```
store_packages(output_directory = getwd())
```

Arguments

`output_directory`
: Path to the directory in which export the .rda file. By default `<working.directory>`.

Value

Nothing is returned. An .rda file will be exported at the `output_directory` indicated.

`substract.bw` *Combination of two or more list in a unique one.*

Description

Combines two or more lists in a single one keeping the elements names

Usage

```
substract.bw(
  bw1,
  bw2,
  wd = getwd(),
  return.substracted.bw = F,
  export.substracted.bw = T,
  substracted.bw.file = paste(getwd(), "subtraction.bw", sep = "/")
)
```

Arguments

bw1	Full path to the first bigWig (the second one will be substracted to this one).
bw2	Full path to the second bigWig (it will be substracted to the first one).
return.substracted.bw	Logic value to define whether return the resulting bigWig as GRanges object. By default FALSE.
export.substracted.bw	Logic value to define whether export the resulting bigWig. By default TRUE.
substracted.bw.file	String for the path of the resulting bigwig file to be exported. By default <working.directory>/subtraction.bw.

Value

If required a subtraction bigWig is returned as GRanges object. The resulting bigWig can be also directly exported.

---

update_pkgs *function to automatically update the R packages.*

Description

Automatically updates the R packages from CRAN and BioConductor repositories.

Usage

```
update_pkgs(ask = FALSE)
```

Arguments

ask	Logical indicating whether to ask the user to select packages before they are downloaded and installed, or the character string "graphics", which brings up a widget to allow the user to (de-)select from the list of packages which could be updated. (The latter value only works on systems with a GUI version of select.list, and is otherwise equivalent to ask = TRUE). By default FALSE.

Value

Nothing. The packages will be updated.

Examples

```
update_pkgs()
```

volcano *Volcano plot generator for RNA-seq data.*

Description

Generates a volcano plot in order to visualize the differentially expressed genes. The plot is highly customizable.

Usage

```
volcano(
  log2FC_data,
  padj_data,
  FC_t = 1.5,
  p_t = 0.05,
  FC_unresponsive_rigth = 1.1,
  FC_unresponsive_left = 1/FC_unresponsive_rigth,
  x_ends = NULL,
  y_min = 0,
  y_max = NULL,
  left_label = "UP",
  right_label = "DOWN",
  unresponsive_label = "NoResp",
  null_label = "NULL",
  names = as.character(c(1:length(log2FC_data))),
  left_names = FALSE,
  right_names = FALSE,
  padding = FALSE,
  names_size = 10,
  print_plot = F,
  left_color = "#00BA38",
  right_color = "#F8766D",
  unresponsive_color = "#00A5CF",
  null_color = "gray30",
  point_size = 0.5,
  legend = TRUE,
  legend_title = "Expression status",
  x_label = "log2(fold change expression)",
  y_label = "-log10(p-value adjusted)",
  title = "Volcano plot",
  sub_title = NULL,
  add_threshold_lines = T,
  threshold_line_color = "gray70",
  threshold_line_type = "dotted",
  font_family = "Helvetica",
  font_size = 12
)
```

Arguments

log2FC_data	Numeric vector containing the log2(FoldChange) values of each gene.
padj_data	Numeric vector of p-values. Use of adjusted p-values is recommended.
FC_t	Value of the threshold to use for the fold change expression to define differentially expressed genes, expressed as linear value. By default 1.5 and by consequence 1/1.5.
p_t	Value of the threshold to use for the p-values to define differentially expressed genes, expressed as linear value. By default 0.05.
FC_unresponsive_rigth	Value of the threshold to use for the fold change expression to define unresponsive genes when FC > 1, expressed as linear value. By default 1.1. If NULL it will be calculated symmetrically from FC_NoResp_left as 1/FC_NoResp_left.
FC_unresponsive_left	Value of the threshold to use for the fold change expression to define unresponsive genes when FC < 1, expressed as linear value. By default 1/FC_unresponsive_rigth. If NULL it will be calculated symmetrically from FC_NoResp_rigth as 1/FC_NoResp_rigth.
x_ends	Numeric positive value to define manually the range of the X-axis: it will be calculated as c(-x_ends,x_ends), for this reason the plot will be symmetrical. By default NULL, the range is assigned automatically and the plot can be asymmetrical.
y_min	Numeric value for the minimum value of the Y-axis. By default 0. Set it to NULL for automatic computation.
y_max	Numeric value for the maximum value of the Y-axis. By default NULL.
left_label	String to indicate the label to use for the set of genes in the left side of the graph (those with FoldChange < 1/FC_t and p.value < p_t. By default "UP".
right_label	String to indicate the label to use for the set of genes in the right side of the graph (those with FoldChange > FC_t and p.value < p_t. By default "DOWN".
unresponsive_label	String to indicate the label to use for the set of unresponsive genes (those with FC_unresponsive_left < FoldChange < FC_unresponsive_rigth and p.value > p_t. By default "NoResp".
null_label	String to indicate the label to use for the set of null genes (those with 1/FC_t < FoldChange < FC_t and p.value < p_t. By default "NULL".
names	String vector with the names to be plotted if required, eg. gene names. By default as.character(c(1:length(log2FC_data))).
left_names	Logic value to indicate if to print the set of differentially expressed genes in the left side of the graph (those with FoldChange < 1/FC_t and p.value < p_t. By default FALSE.
right_names	Logic value to indicate if to print the set of differentially expressed genes in the right side of the graph (those with FoldChange > FC_t and p.value < p_t. By default FALSE.

padding — Logic value to indicate if to plot the padding around the names of genes. By default FALSE.

names_size — Numeric value to define de size of the point names size. By default 10.

print_plot — Logic value to define whether to print the volcano plot once created. By default FALSE.

left_color — String to indicate the color to use for the set of genes in the left side of the graph (those with FoldChange < 1/FC_t and p.value < p_t. By default "#00BA38", a green.

right_color — String to indicate the color to use for the set of genes in the right side of the graph (those with FoldChange > FC_t and p.value < p_t. By default "#F8766D", a pink/red.

unresponsive_color — String to indicate the color to use for the set of unresponsive genes (those with FC_unresponsive_left < FoldChange < FC_unresponsive_rigth and p.value > p_t. By default "#00A5CF", a light blue.

null_color — String to indicate the color to use for the set of null genes (those with 1/FC_t < FoldChange < FC_t and p.value < p_t. By default "gray30", a dark gray.

point_size — Numeric value to define de size of the points. By default 0.5.

legend — Logic value to define if to print the legend. By default TRUE.

legend_title — A string to indicate the label of the legend title. By default "Expression status".

x_label — A string to indicate the X-axis label. By default "log2(fold change expression)".

y_label — A string to indicate the Y-axis label. By default "-log10(p-value adjusted)".

title — A string to indicate the title of the plot. By default "Volcano plot".

sub_title — A string to indicate the subtitle of the plot. By default NULL, no subtitle is written.

add_threshold_lines — Logic value to define if lines for the thresholds, both FC and p.value, should be plotted. By default TRUE.

threshold_line_color — String to define the color of the threshold lines. By default "gray70"

threshold_line_type — String or numeric value to define the threshold lines type. Both numeric and string standard R codes are accepted. By default "dotted", equivalent to 2.

font_family — String to define the font family to use in the plot writings. By default "Helvetica".

font_size — Numeric value to define the font size. By default 12.

Value

A plot in ggplot2 format.

Index