Package ‘Rseb’

December 14, 2020

Type Package

Title A package for NGS data managing and visualization

Version 0.1.0

Author Sebastian GREGORICCHIO

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

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

License Artistic-2.0

Depends R

biocViews BiocManager, Biostrings, biomaRt, GO.db, rtracklayer

Imports 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:

`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)))
```

---

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 colors of the lines. If only one value is provided or the number of values are less then 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

Helps to the generation of 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,
  help = 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`.
`help`	Logical value to indicate whether display the help. By default `FALSE`.

Details

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.

`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.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 signals computed by deeptools's computeMatrix function.
Creates a plot starting from the matrix.gz generated by deepTools.

Usage

```
plot.density.profile(
  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`	Input matrix_file.gz generated by deeptools's function `computeMatrix` from deepTools.
`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 `FLASE` 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 colors of the lines. If only one value is provided or the number of values are less then the required ones only the first value will be used. 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 get 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))
```

---

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

```
sort.bed(
  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",
  ...
)
```

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"`.

Value

A plot in ggplot2 format.

Index