Package 'Rseb'

September 18, 2023

Type Package

Title An R-package for NGS data managing and visualization

Version 0.3.2

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 (>= 4.0.0)

Imports BiocManager, AnnotationFilter, Biostrings, biomaRt, colorspace, diffloop, enrichplot, EnsDb.Hsapiens.v75, EnsDb.Hsapiens.v86, EnsDb.Mmusculus.v79, GenomicRanges, GO.db, IRanges, RColorBrewer, rtracklayer, S4Vectors, cowplot, data.table, eulerr, ggplot2 (>= 3.3.3), ggbio, ggforce, ggrepel, ggpubr, ggpmisc, grDevices, matrixStats, jpeg, plyr, pryr, dplyr, tidyr, purrr, readxl, robustbase, R.utils, scales, stringr, tools, devtools, rvcheck, curl, prettydoc, knitr, rmarkdown, stats, openssl

biocViews

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

URL https://sebastian-gregoricchio.github.io/Rseb/
https://github.com/sebastian-gregoricchio/Rseb/
https://sebastian-gregoricchio.github.io/

BugReports https://github.com/sebastian-gregoricchio/Rseb/issues

R topics documented:

`actualize` `Rseb` *updates verification*

Description

It verifies if `Rseb` is up-to-date and installs it when required.

Usage

```
actualize(
  update = TRUE,
  verbose = TRUE,
  force = FALSE,
  build.manual = TRUE,
  build.vignettes = TRUE
)
```

Arguments

`update`	Logical value to define whether update the `Rseb` package. By default `TRUE`.
`verbose`	Logical value to define whether print messages. By default `TRUE`.
`force`	Logical value to define whether to force the installation of `Rseb` even though already up-to-date. Parameter passed to `devtools::install_github()`. By default `FALSE`.
`build.manual`	Logic value to define whether to build the manual. By default `TRUE`.
`build.vignettes`	Logic value to define whether to build the vignettes. By default `TRUE`.

Details

This function will check for internet availability.

Value

Warnings and/or messages. Installation of the latest version of `Rseb` if required.

`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 alpha-numeric 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))
```

closest.regions *Find closets regions to reference regions.*

Description

This tools return the closest upstream and downstream regions from a reference region.

Usage

```
closest.regions(
  reference.regions,
  reference.regions.table.name = "referenceRegions",
  target.regions,
  export.table.file = NULL,
  return.table = TRUE,
  collapse.regions = FALSE,
  verbose = TRUE
)
```

Arguments

`reference.regions`
: A full path to a bed file or a data.frame in at least BED3 format with the regions to use as reference.

`reference.regions.table.name`
: A string with the name to use for the group reference regions. By default `"referenceRegions"`.

`target.regions`
: A full path to a bed file or a data.frame in at least BED3 format with the regions to uses as targets.

`export.table.file`
: A string with the full path for the file in which the table should be exported. By default `NULL`: not export.

`return.table`
: Logical value to define whether the output table should be returned. By default `TRUE`.

`collapse.regions`
: Logical value to define whether the partially overlapping regions should be collapsed or not. By default `FALSE`.

`verbose`
: Logical value to define whether messages should be printed. By default `TRUE`.

Value

The function returns a data.frame composed of a triplicated chr-start-end-name table for reference.region, upstream.region and downstream.region, respectively.

---

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)
```

`CNV.data` *CNV data results example*

Description

Simulation of Copy Number Variation (CNV) analysis on a cohort of patients.

Usage

```
CNV.data
```

Format

A data frame with 25 rows and 9 variables:

`geneName` hypothetical gene symbols
`patient_1 ... patient_N` hypothetical patients ID

Source

Simulated data

`collapse.bed` *Merger of overlapping peaks in a provided .bed file.*

Description

Merge overlapping peaks in a provided .bed file.

Usage

```
collapse.bed(
  bed,
  maximal.distance = 0,
  keep.strandness = FALSE,
  only.one.strand = NULL,
  score.operation = "mean",
  bed.header = FALSE,
  sep = "\t",
  return.bed = TRUE,
  export.file.name = NULL,
  export.header = FALSE,
  verbose = TRUE
)
```

Arguments

bed	Two options are possible: - String with the path to a .bed file; - data.frame corresponding to a bed file format (only the first 6 columns, BED6, will be kept).
maximal.distance	Maximal distance between regions allowed for regions to be merged. By default 0.
keep.strandness	Logic value to indicate whether to force to only merge regions that are in the same strand. By default FALSE, disabled. Subordinated to not NULL value for 'only.one.strand' option.
only.one.strand	Atomic string to indicate whether to force merge for one specific strand only. It must be indicated the wished strand (e.g., '+', '-', '.'). Regions in the other strand/s will be kept without any modification. By default NULL.
score.operation	Applicable only if the regions contain scores. Atomic string to indicate the operation to apply to the scores of merged regions. Possible choices: 'mean', 'median', 'sum'. By default "mean".
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.file.name	Optional: string to define the path to the file to be exported, if required. By default NULL, not exported.
export.header	Logic value to define whether the header should be exported in the sorted bed file. By default FALSE.
verbose	Logic value to indicate whether messages should be printed or not. By default TRUE.

Details

The function pre-sorts the bed and keeps only unique rows and only up to 6 columns (chr, start, end, name, score, strand).
The names of the regions (if available) of merged regions corresponds to the concatenation of all original region's name.
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 collapsed .bed file.

color.gradient *Gradient colors generation and assignment*

Description

Give a vector of colors generates a finite number of shadows that will be assigned to a numeric vector depending on the value of each element.

Usage

```
color.gradient(values, colors = c("blue", "white", "red"), bins = 100)
```

Arguments

values	A numeric vector containing the values to which a color must be assigned (NAs and NaN will be converted to 0).
colors	A string vector with the colors, in the wished order, that have to be used to generated the shadows. By default `c("blue","white","red")`.
bins	An atomic integer value to define the total number of bins/steps in which the gradient should be dived.

Value

A vector containing the assigned colors corresponding to each element of `values`.

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 and plot.density.summary. 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 = paste0("/home/", Sys.getenv("USERNAME"),
    "/anaconda3/bin/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 `"/home/USERNAME/anaconda3/bin/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.

`deeptools.matrix` *deepTools matrix example*

Description

List result of the function read.computeMatrix.file used to read a matrix.gz file generated by deepTools computeMatrix function.

Usage

```
deeptools.matrix
```

Format

A named list of 3 variables:

`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

Source

Simulated data

`density.matrix` *Density matrix builder*

Description

A function (completely in R) that generates a matrix given a list of regions (.bed files) and signals (.bigWig files) alternative (even though more time consuming) to computeMatrix.deeptools. The output can be passed as it is to the functions plot.density.profile, plot.density.summary and, plot.density.differences.

Usage

```
## S3 method for class 'matrix'
density(
  mode,
  regions.list,
  samples.list,
  region.names = NULL,
  sample.names = NULL,
  sort.regions.coordinates = FALSE,
  reference.point = "center",
  reference.point.label = NULL,
  upstream = 500,
  downstream = 500,
  body.length = 1000,
  missing.data.as.zero = FALSE,
  bin.size = 10,
  binning.operation = "mean",
  stranded = FALSE
)
```

Arguments

`mode`	A string indicating the method for the matrix computation: • `scale-regions` all regions in the BED file are stretched or shrunken to the length (in bases) indicated by the user (`body.length`); • `reference-point` the matrix will be performed on the range -upstream+downstream from the indicated reference point (center, TSS, TES).
`regions.list`	A string vector with a list of full paths to bed files or list of data.frames in at least BED3 format (eg. generated by build.bed).
`samples.list`	A string vector with a list of full paths to bigWig files.
`region.names`	A string vector with the names of the regions. If `NULL` or of length lower than the number of regions the names will be assigned using the basename of the file if a path is provided otherwise "region_<order number>". By default `NULL`.

`sample.names` A string vector with the names of the samples. If `NULL` or of length lower than the number of samples the names will be assigned using the basename of the file. By default `NULL`.

`sort.regions.coordinates`
Logical value to define whether the output matrix should contain the regions sorted by genomic location for each region group (sorted by sort.bed). By default `FALSE`.

`reference.point`
The reference point for the matrix generation could be either the region start (`"TSS"`), the region end (`"TES"`) or the `"center"` of the region. By default `"center"`.

`reference.point.label`
A single string with the label for the reference point that could be used for the plots.

`upstream` Distance, in bases (bp), upstream of the reference-point, in "reference-point" mode, or the region start, in "scale-regions" mode. By default `500`.

`downstream` Distance, in bases (bp), downstream of the reference-point, in "reference-point" mode, or the region start, in "scale-regions" mode. By default `500`.

`body.length` Distance, in bases (bp), to which all regions will be fit. By default: `1000`.

`missing.data.as.zero`
A logical value to define whether missing data (NAs) should be treated as zeros. By default `FALSE`.

`bin.size` Length, in bases (bp), of the non-overlapping bins for averaging the score over the regions length. By default `10`.

`binning.operation`
A single string to define the type of statistic that should be used over the bin size range. The options are: "mean", "median", "sum". By default `"mean"`.

`stranded` Logical value to indicate whether the strand of the region should be taken into account. When `TRUE`, the order of the bigWig score for the given region will be reversed. Default `FALSE`.

Value

The function returns a named list containing:

- `metadata` data.frame with the parameters used to build the matrix;
- `matrix.data` data.frame with the computed scores;
- `original.file.path` with the string: "Matrix generated by Rseb::density.matrix()".

This list can be passed as it is to the functions plot.density.profile, plot.density.summary and, plot.density.differences.

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.

Value

Prints the plot

References

https://magesblog.com/

Examples

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

`evaluate.heterogeneity` *Evaluate genomic heterogeneity among samples.*

Description

This tools evaluates what is the fraction of peaks covered by each sample provided in a peaks dataset obtained by merging all the peaks together or provided by the user. The peaks in the reference dataset are ranked by number of samples in which are present and average score all over the samples. This function uses the deeptools function multiBigwigSummary.

Usage

```
evaluate.heterogeneity(
  bigWig.list,
  peak.list,
  labels = sub(pattern = "(.*)\\..*$", replacement = "\\1", basename(bigWig.list)),
  reference.peaks = NULL,
  distribution.line.color = "#1c30a3",
  distribution.line.size = 1,
  distribution.line.type = 1,
  distribution.n.vertical.divisions = NULL,
  distribution.as.percentage = F,
  heatmap.color = "#1c30a3",
  heatmap.zMax = NA,
  heatmap.log1p.scale = TRUE,
  bar.color = distribution.line.color,
  widths.proportion = c(0.25, 1),
  heights.proportion = c(1, 1),
  min.percentage.reference = 0,
  min.percentage.test = 0,
  min.bases.overlap = 1,
  multiBigWigSummary.path = "multiBigWigSummary"
)
```

Arguments

`bigWig.list`	A string vector with bigwig paths (same order than paths).
`peak.list`	A list of GRanges objects (not GRglist) or data.frames or a string vector with the path to bed files (same order than bigwigs).
`labels`	The labels to use for the samples (same order than bigwigs/peaks). Default: the basename of the bigWig.list.
`reference.peaks`	Default: `NULL`, the peaks of all samples provided are merged and collapsed together.

`distribution.line.color`
: Color to use for the distribution line. Default: `"#1c30a3"` (dark blue).

`distribution.line.size`
: Line size of the distribution plot. Default: `1`.

`distribution.line.type`
: Line type of the distribution plot. Default: `1`.

`distribution.n.vertical.divisions`
: Number of sectors in which divide the distribution plot (vertical lines will be plotted). Default: `NULL` (no divisions).

`distribution.as.percentage`
: Logical value to define whether the distribution plot should show percentage of sample coverage rather than number of samples. Default: `FALSE`.

`heatmap.color`
: Color to use for the heatmaps; a gradient from this color to white will be used. Default: `"#1c30a3"` (dark blue).

`heatmap.zMax`
: Maximum of the heatmap scale. Default: `NA`.

`heatmap.log1p.scale`
: Logic value to define whether the heatmap scale should display log1p values. Default: `TRUE`.

`bar.color`
: Color to use for the barplot showing the fraction of reference peaks present in each sample. Default is to use the 'distribution.line.color'.

`widths.proportion`
: Two-elements numeric vector to be passed to `plot_grid` rel_width.

`heights.proportion`
: Two-elements numeric vector to be passed to `plot_grid` rel_height.

`min.percentage.reference`
: Numeric value within 0-100 to define which percentage of 'reference' dataset must overlap with a 'sample'. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: `0`.

`min.percentage.test`
: Numeric value within 0-100 to define which percentage of 'sample' datasets must overlap with a region in the 'reference' one. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: `0`.

`min.bases.overlap`
: Integer, greater than 0, value to indicate the minimal number of bases to consider as minimum overlap between two regions. Non integer values will be rounded at integer, while number lower that 1 will be coerced to 1. Default value: `1`.

`multiBigWigSummary.path`
: Path/command to run deeptools multiBigWigSummary tool. Default: `"multiBigWigSummary"`.

Details

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

Value

The function returns a list containing:

- `metadata`: a list with the following elements
  - `sample.id`: vector with the labels of the samples
  - `bigWig.file`: string vector with the path to each bigwig file
  - `bed.file`: list of peaks associated to each sample
  - `n.peaks`: vector with number of peaks in each sample
- `count.matrix`: data.frame with presence (1) or absence (0) of each peak per each sample;
- `score.matrix`: data.frame with average score at each peak per each sample;
- `plot.list`: list of single separated plots: counts.distribution, fraction.peaks.per.sample, scores.heatmap;
- `multiplot`: the multiplot generated from the plot.list.

`floating.ceiling` *Ceiling to floating values*

Description

Computes the ceiling of the given value but with any number of digits (to the closest floating number of given digits).

Usage

```
floating.ceiling(num, digits = 1)
```

Arguments

`num`	A single number or a numeric vector.
`digits`	A single integer indicating the maximum number of digits required.

Value

A floored number or numeric vector.

floating.floor *Flooring to floating values*

Description

Computes the floor of the given value but with any number of digits (to the closest floating number of given digits).

Usage

```
floating.floor(num, digits = 1)
```

Arguments

num	A single number or a numeric vector.
digits	A single integer indicating the maximum number of digits required.

Value

A floored number or numeric vector.

genomic.tracks *Genomic tracks plotter*

Description

The functions allows to plot different types of genomic data (bigWig, bed, bedpe) at a specific genomic region. It is possible to highlight specific regions and the gene annotations are plotted automatically at the bottom of all the tracks.

Usage

```
genomic.tracks(
  tracks,
  genomic.region,
  genome,
  track.labels = NULL,
  track.labels.fontzise = 5,
  track.labels.position = c(-0.1, 0),
  track.colors = "#000000",
  grouping = NULL,
  gene.annotation.color = "darkblue",
  expand.bed = TRUE,
  arcs.direction = "down",
  fraction.arc.base = 0.025,
```

```
  highlight.bed = NULL,
  highlight.color = "yellow",
  highlight.transparency = 0.15,
  missing.data.as.zero.bw = FALSE,
  smooth.bigWig.signal = TRUE,
  smooth.bigWig.loess.span = 0.05,
  plot.bigWig.area = TRUE,
  bigWig.range.label.size = 2.5,
  score.bed.shadow = FALSE,
  height.ratios = NULL,
  width.ratios = c(1, 5)
)
```

Arguments

`tracks`	A vector indicating the list of full paths of the files/tracks/signals to plot. Supported formats: bed/bd/narrowPeak/broadPeak, bw/bigWig/bigwig, bedpe.
`genomic.region`	An atomic string indicating the genomic region into which restrict the final plot in the format 'chr1:1234-5678'.
`genome`	An atomic string indicating the genome to use for the annotations. Allowed values are: • `hg19`: loads an 'EnsDb' object from the library `EnsDb.Hsapiens.v75`; • `hg38`: loads an 'EnsDb' object from the library `EnsDb.Hsapiens.v86`; • `mm10`: loads an 'EnsDb' object from the library `EnsDb.Mmusculus.v79`; • *custom 'EnsDb' object*: provide an 'EnsDb' object manually generated; visit the page https://bioconductor.org/packages/release/bioc/vignettes/ensembldb/inst/doc/ensembldb.html#102_building_annotation_packages for more information.
`track.labels`	A vector indicating the labels to use for each track (genome annotation track excluded). By default `NULL`: the file base-name will be used.
`track.labels.fontzise`	A numerical value to indicate the font size of the track labels. Default value `5`.
`track.labels.position`	A two-element numeric vector passed to `xlim` function for the the definition of the frame size of the track labels. Default value `c(-0.1,0)`.
`track.colors`	A string vector indicating the color to use for each track (genome annotation track excluded). If only one value is provided it will be used for all the tracks. Default value `"#000000"` ("black").
`grouping`	A single numerical vector or a list of numeric vectors. Each list's element indicates the indexes corresponding to the tracks (1 = first track, 2 = second track, etc) for which the y-axes should be normalized. Each element will be taken into account in the order. Default value `NULL`.
`gene.annotation.color`	A string indicating the color to use for the genome annotation track.
`expand.bed`	A logical value to define whether overlapping regions in a bed should be plotted on different levels. Default `TRUE`.

`arcs.direction` A string indicating the direction on which arcs should be plotted for bedpe files. Available options `"up"` or `"down"`. Default value `"down"`.

`fraction.arc.base`
A numerical value indicating the fraction of total plot height to be used as arc base thickness. By default `0.025` (2.5% of the track height).

`highlight.bed` Either a string indicating the full path to a bed file or a data.frame in BED3 format (chr, start, end) containing regions that should be highlighted in the plot. Regions included in the genomic range will be automatically selected. By default `NULL`.

`highlight.color`
A string indicating the color to use for the regions to highlight in the plot. By default `'yellow'`.

`highlight.transparency`
A numerical value indicating the transparency (alpha) to use for the highlighted regions. Default value `0.15`.

`missing.data.as.zero.bw`
A logical value to define wthere missing data in the bigWigs should be converted to zeros. Default `FALSE`.

`smooth.bigWig.signal`
Logical value to indicate whether the bigWig signals should be smoothed (by loess x ~ y function). By default `TRUE`.

`smooth.bigWig.loess.span`
Numerical value to indicate the span value for the loess function used to smooth bigWig signals. By default `0.05`.

`plot.bigWig.area`
Logical value to indicate whether the bigWig profile should be filled or not. If `FALSE` only the signal outline will be plotted. By default `TRUE`.

`bigWig.range.label.size`
A numerical value to indicate the font size of the bigWig signal range. Default value `2.5`.

`score.bed.shadow`
Logical value to define whether the filling intensity of the bed segments should reflect the score of each signal. By default `FALSE`.

`height.ratios` Numerical vector of relative track heights, passed to 'rel_heights' parameter of `cowplot::plot_grid()`. For example, in a two-row grid, rel_heights = c(2, 1) would make the first column twice as wide as the second column. Value `1` indicates that all the tracks should have the same size. By default `NULL`, automatic ratios will be computed by this function.

`width.ratios` Numerical vector of relative labels vs tracks widths, passed to 'rel_widths' parameter of `cowplot::plot_grid()`. For example, in a two-column grid, rel_widths = c(2, 1) would make the first column twice as wide as the second column. Value `1` indicates that all the tracks should have the same size. By default `c(1,5)` (1 label : 5 tracks).

Value

The function returns a named list containing:

- configuration: data.frame with the parameters used to build the plot(s);
- highlighted.region: data.frame with the regions used for the highlighting;
- single.track.list: a named list containing each single track plot used for the creation of the multi.track.plot;
- single.label.plot.list: a named list containing each single track label plot used for the creation of the multi.track.plot;
- multi.track.plot: the assembled multi.track labelled plot.

---

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")
```

`get.single.base.score.bw` *Single base bigWig score selector*

Description

Function to get the score from a bigWig for each base in a given genomic region.

Usage

```
get.single.base.score.bw(
  region,
  bigWig,
  missing.data.as.zero = TRUE,
  reverse.score = FALSE
)
```

Arguments

`region`	An atomic string indicating the genomic region into which restrict the final plot in the format 'chr1:1234-5678'.
`bigWig`	Full path to a bigWig file.
`missing.data.as.zero`	A logical value to define whether missing data (NAs) should be treated as zeros. By default `TRUE`.
`reverse.score`	A logical value to indicate whether the score order should be inverted. Default `TRUE`.

Value

The output is a numeric vector containing the score for each base at a given position.

`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_number

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,
  delay.interval = 10,
  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. "mm9", "mm10", "hg19", "hg38", ... By default NULL.
fivePrime	Numeric value to define of how many base-pairs (bp) expand from full gene position at it's 5'-end, default 1000bp.
threePrime	Numeric value to define of how many base-pairs (bp) expand 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. By default 1000.
delay.interval	Sets a delay (sleep) time in milliseconds. The sleep interval is invoked between successive commands. By default 10. helps to give the time to IGV to adapt the view before the snap (such as the autoscale).
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.

`incucyte` *Incucyte analysis tool*

Description

This tool generates confluency plots over time starting from incucyte data.

Usage

```
incucyte(
  metadata,
  raw.data,
  comparisons = "all",
  error = "SEM",
  normalization.method = "none",
  start.hour = 0,
  plot.days = FALSE,
  show.error.ribbon = TRUE,
  show.error.bars = FALSE,
  show.points = TRUE,
  show.lines = TRUE,
  show.legend = TRUE,
  same.y.scale = TRUE,
  group.order = NULL,
  error.transparency = 0.25,
  point.size = 1,
  line.type = 1,
  line.smooth.span = 0.25,
  colors = NULL,
  skip.head.lines.in.data = 1
)
```

Arguments

`metadata`	Path to a table or a data.frame with at least two columns: 'well.ID' and 'group'. The 'well.id' indicates the ID of the wells/columns in the rawData table; the 'group' indicates the name of the group at which each column belongs (wells belonging to the same column will be averaged together). A third column 'color'

can be used to indicate the color; notice that each group should have the same color.

`raw.data` Path to the rawData table or a data.frame. First column must be 'Elapsed' (the time in hours) and the other columns the wells (e.g., A1, A2, B5, B8, ...).

`comparisons` List of vectors containing the groups to be used in each comparison: one vector per comparison. E.g.: list(c("group_A", "group_B"), c("group_B", "group_C", "group_E")). Default: `"all"`.

`error` String to indicate the error type. Possible choices: 'SEM', 'SD'. Default: `SEM`.

`normalization.method`
String to indicate the normalization method. Possible choices: "none", "division", "subtraction". Default: `none`. Division: for each group the values are divided by the first timepoint.

`start.hour` Number of hours to be excluded from the calculations. Default: `0`.

`plot.days` Logical value to indicate whether to plotted days instead of hours. Default: `FALSE`.

`show.error.ribbon`
Logical value to indicate whether to show the error ribbon. Default: `TRUE`.

`show.error.bars`
Logical value to indicate whether to show the error bars. Default: `FALSE`.

`show.points` Logical value to indicate whether to show the individual average points at each time point. Default: `TRUE`.

`show.lines` Logical value to indicate whether to show the interpolation line. Default: `TRUE`.

`show.legend` Logical value to indicate whether to show the color legend. Default: `TRUE`.

`same.y.scale` Logical value to indicate whether the y-axis should show the same limit range. Default: `TRUE`.

`group.order` String vector with the specific order to use for the groups indicated in the metadata (factor levels). Default: `NULL`.

`error.transparency`
Number between 0 and 1 indicating the transparency (alpha) to use for the error ribbon. Default: `0.25`.

`point.size` Number for the point size. Default: `1`.

`line.type` A vector indicating the line type to use for each group (both numeric and string values accepted). Default: `1` (applied to all the groups).

`line.smooth.span`
Numeric value of the 'span' for the smoothing of the interpolation line. Default: `0.25`.

`colors` A vector indicating the color to use for each group (any R color format is accepted). Default: `NULL` (random colors are generated).

`skip.head.lines.in.data`
Number of lines to be skipped at the beginning of the raw.data file. Default: `1`.

Value

The function returns a list containing:

- `metadata`: table used for the metadata;
- `raw.data`: tables used as raw.data;
- `analyzed.data`: data.frame with the analyzed data (means, n, SEM, SD, groups, ...);
- `normalized.data`: data.frame with the normalized data used for the plotting;
- `plot.list`: names list of plots, one plot for each group plus 'all' comparison;
- `multiplot`: a plot with all the plots in the plot.list .

`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 = FALSE,
  e = FALSE,
  s = FALSE,
  S = FALSE,
  split = FALSE,
  sorted = FALSE,
  g = NULL,
  srun = FALSE,
  intersect.bedtools.command = paste0("/home/", Sys.getenv("USERNAME"),
    "/anaconda3/bin/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 — Logic value defining if the fraction (parameter f) is required to be 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 — Logic value defining if the fraction (parameter f) must 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 "/home/USERNAME/anaconda3/bin/

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")
```

`intersect.regions` *Genomic regions overlapper*

Description

A tool to define overlaps between bed files/regions derived from different formats. The function allows the overlap in stranded mode and can considered a specific minimal percentage of overlap between regions.

Usage

```
intersect.regions(
  reference.regions,
  test.regions,
  min.percentage.reference = 0,
  min.percentage.test = 0,
  min.bases.overlap = 1,
  sort.overlaps = FALSE,
  stranded = FALSE,
  return.as.data.frame = TRUE
)
```

Arguments

`reference.regions`
: A single value or a list of regions to be used as 'reference'. The values accepted are: a. a character with the full path to a bed file, b. a data.frame in at least BED3 format, c. a GRanges object in at least BED3 format. If a list of elements is provided all the regions will be merged in a unique combined list and only completely identical regions will be remove to avoid duplicates. Combination of different formats is allowed.

`test.regions`
: A single value or a list of regions to be used as 'test'. The values accepted are: a. a character with the full path to a bed file, b. a data.frame in at least BED3 format, c. a GRanges object in at least BED3 format. If a list of elements is provided all the regions will be merged in a unique combined list and only completely identical regions will be remove to avoid duplicates. Combination of different formats is allowed.

`min.percentage.reference`
: A numeric value in 0-100 to define which percentage of a region in the 'reference' dataset must overlap with a region in the 'test' one. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: `0`.

`min.percentage.test`
: A numeric value in 0-100 to define which percentage of a region in the 'test' dataset must overlap with a region in the 'reference' one. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: `0`.

`min.bases.overlap`
: Integer, greater than 0, value to indicate the minimal number of bases to consider as minimum overlap between two regions. Non integer values will be rounded at integer, while number lower that 1 will be coerced to 1. Default value: `1`.

`sort.overlaps`
: Logic value to define whether the output should be sorted or not. Default value: `FALSE`.

`stranded`
: A logical value to define whether the analyses should be performed by strand: regions in one strand will be overlapped only with regions of the same strand. The strand symbols considered are '+' and '-', any other symbol will considered in a unique separated category. Default value: `FALSE`.

`return.as.data.frame`
: Logical value to define whether the output list should contain data.frames instead of GRanges objects. Default value: `TRUE`.

Value

The function returns a list of data.frames/GRanges objects containing:

- `overlaps.reference`: XX;
- `non.overlaps.reference`: XXX;
- `overlaps.testt`: VV;
- `non.overlaps.test`: XX.

`is.color` *is.color*

Description

Function to define if each element of a string vector is an R-supported color string.

Usage

```
is.color(x)
```

Arguments

`x` A string vector.

Value

A logical vector of the same length of `x`.

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 = FALSE,
  print.versions = TRUE,
  return.versions = FALSE,
  session.file = NULL
)
```

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`.
`session.file`	If a string to a path is provided, a .txt file with session and versions info will be exported. Default `NULL`, no exported files.

Value

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

plot.density.differences *Plot the distribution of overall NGS density at specific regions from deepTools matrices.*

Description

Computes the score of each element in a list of regions and generates violins plots with percentiles and the mean (optional) for each sample/region. It uses as input a score matrix computed by deeptools's computeMatrix function or by computeMatrix.deeptools and density.matrix functions from this package.

Usage

```
## S3 method for class 'density.differences'
plot(
  matrix.file,
  missing.data.as.zero = NULL,
  sample.names = NULL,
  region.names = NULL,
  signal.type = "mean",
  error.type = "sem",
  subset.range = NULL,
  inverted.comparisons = F,
  stat.method = "wilcox.test",
  stat.paired = T,
 stat.p.levels = list(cutpoints = c(0, 1e-04, 0.001, 0.01, 0.05, 1), symbols = c("****",
    "***", "**", "*", "ns")),
  area.line.width = 0.5,
  area.fill.area = T,
  area.plot.zero.line = T,
  area.y.identical.auto = T,
  area.y.ticks.interval = NULL,
  area.y.digits = 1,
  correlation.log2 = T,
  correlation.plot.correlation = T,
  correlation.correlation.method = "lm",
  correlation.show.equation = T,
  correlation.correlation.line.width = 0.75,
  correlation.correlation.line.color = "purple",
  correlation.correlation.line.type = 1,
  correlation.correlation.line.SE = T,
  correlation.correlation.formula = "y ~ x",
  correlation.add.rug = T,
  correlation.x.identical.auto = T,
  correlation.y.identical.auto = T,
  correlation.x.ticks.interval = NULL,
  correlation.y.ticks.interval = NULL,
  correlation.x.digits = 1,
  correlation.y.digits = 1,
  points.size = 0.5,
  transparency = 0.25,
  axis.line.width = 0.5,
  text.size = 12,
  legend.position = c(0.2, 0.85),
 colors = c(Sample1 = "#F8766D", Sample2 = "#00A5CF", `No difference` = "#00BA38"),
  n.row.multiplot = 1,
  by.row = T
)
```

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 or density.matrix.

`missing.data.as.zero` Logical 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/compared. Available parameters are "mean", "median" and "sum". By default "mean".

`error.type` String indicating the type of error to be computed and that will be available in the output data.table. Available parameters are "sem" and "sd", standard error mean and standard deviation respectively. By default "sem". Parameter considered only when `show.mean = TRUE`).

`subset.range` A numeric vector indicating the range to which restrict the analyses (eg. c(-150, 250)). In the case of "scale-region" mode, the range is represented by (-upstream | 0 | body_length | body_length+downstream).By default `NULL`: the whole region is considered.

`inverted.comparisons` Logical value to indicate whether to invert the order of the pair-comparisons. By default `FALSE`.

`stat.method` A single string defining the method to use for the statistical comparisons. By default `"wilcox.test"`. Available options: "t.test" "wilcox.test".

`stat.paired` Logical value to define if the statistical comparisons should be performed paired. By default `TRUE`. Notice that to allow a paired comparison the number of data should be the same in the two groups compared, so in the most of the cases non applicable to the comparisons between two regions. Used only in `"t.test"` and `"wilcox.test"` methods.

`stat.p.levels` A list containing the p-values levels/thresholds in the following format (default): `list(cutpoints = c(0,0.0001,0.001,0.01,0.05,1),symbols = c("****","***","**","*","ns")`
In other words, we use the following convention for symbols indicating statistical significance:

- ns: p > 0.05
- * p <= 0.05
- ** p <= 0.01
- *** p <= 0.001
- **** p <= 0.0001

area.line.width
: Numeric value to define width of the line connecting the points in the area.plots. By default 0.5.

area.fill.area
: Logical value to indicate whether to fill the area under the line in the area.plot. By default TRUE.

area.plot.zero.line
: Logical value to define whether to plot a dashed gray vertical line in correspondence of the 0 of each area.plot. By default TRUE.

area.y.identical.auto
: Logical value to define whether use the same Y-axis range for all the area.plots automatically depending on their values. By default TRUE.

area.y.ticks.interval
: A number indicating the interval/bin spacing two ticks on the Y-axis of area.plots. By default NULL: ticks are assigned automatically.

area.y.digits
: Numeric value defining the number of digits to use for the Y-axis values of area.plots. By default 1 (eg. 1.5).

correlation.log2
: Logical value to define whether the correlation.plots should show the log2 value of the score. By default TRUE.

correlation.plot.correlation
: Local value to indicate whether to plot the correlation curve on the correlation.plot. By default TRUE.

correlation.correlation.method
: Atomic string describing the method to use to compute the regression curve, eg. "lm", "glm", "gam", "loess", "rlm". By default 'lm'.

correlation.show.equation
: = T

correlation.correlation.line.width
: Numeric value to define correlation line width for all correlation.plots. By default 0.75.

correlation.correlation.line.color
: Numeric value to define correlation line width for all correlation.plots. By default "purple".

correlation.correlation.line.type
: A numeric or character value to define the correlation line type. Both numeric and string codes are accepted. By default "solid".

correlation.correlation.line.SE
: Logical value to indicate whether to plot the standard error (SE) of the correlation curve in the correlation.plot. By default TRUE.

correlation.correlation.formula
: Atomic string indicating the formula to use to compute the correlation curve. By default "y ~ x".

correlation.add.rug
: Logical value to indicate whether to add a rug representation (1-d plot) of the data to the correlation.plot. By default TRUE.

`correlation.x.identical.auto`
: Logical value to define whether use the same X-axis range for all the correlation.plots automatically depending on their values. By default `TRUE`.

`correlation.y.identical.auto`
: Logical value to define whether use the same Y-axis range for all the correlation.plots automatically depending on their values. By default `TRUE`.

`correlation.x.ticks.interval`
: A number indicating the interval/bin spacing two ticks on the X-axis of correlation.plots. By default `NULL`: ticks are assigned automatically.

`correlation.y.ticks.interval`
: A number indicating the interval/bin spacing two ticks on the Y-axis of correlation.plots. By default `NULL`: ticks are assigned automatically.

`correlation.x.digits`
: Numeric value defining the number of digits to use for the X-axis values of correlation.plots. By default `1` (eg. 1.5).

`correlation.y.digits`
: Numeric value defining the number of digits to use for the Y-axis values of correlation.plots. By default `1` (eg. 1.5).

`points.size`
: A numeric value defining the size of the points in both area and correlation plot. By default `0.5`.

`transparency`
: A numeric value to define the fraction of transparency of the fill area in the area.plot and the SE in the correlation plot (0 = transparent, 1 = full). By default `0.25`.

`axis.line.width`
: Numeric value to define the axes and ticks line width for all plots. By default `0.5`.

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

`legend.position`
: Any ggplot supported value for the legend position (eg. "none, "top", "bottom", "left", "right", c(fraction.x, fraction.y)). By default `c(0.2,0.85)`.

`colors`
: Vector of 3 elements to define the points and area colors ('Sample1', 'Sample2' and, 'No difference' values respectively). If only one value is provided it will applied to all the samples. If the number of values is less then 3, the default color set will be used. All supported R.colors values are accepted. By default `c("Sample1" = "#F8766D","Sample2" = "#00A5CF","No difference" = "#00BA38")`.

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

`by.row`
: Logical value to define whether the plots should be arranged by row. By default `TRUE`.

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 function returns a list containing:

- `data.table` with the computed values with all groups and all samples;
- `metadata` table with the information obtained from the matrix_file.gz;
- `comparison.table.list` with a list of tables for each group with a table per each comparison containing the original data and the compared values (differences);
- `comparison.statistics.table` with a table with all the statistical comparisons;
- `area.plot.byGroup.list` with a list per group with a all the area.plots of each comparison;
- `correlation.plot.byGroup.list` with a list per group with a all the correlation.plots of each comparison;
- `area.multiplot.list` with an area.multiplot per each group;
- `correlation.multiplot.list` with an correlation.multiplot per each group.

`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 and density.matrix functions 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.interval = NULL,
```

```
  y.digits = 1,
  axis.line.width = 0.5,
  text.size = 12,
  legend.position = c(0.2, 0.85),
  plot.vertical.lines = T,
  write.reference.points = T,
 colors = c("#00A5CF", "#F8766D", "#AC88FF", "#E08B00", "#00BA38", "#BB9D00", "#FF61C9",
    "gray30"),
  n.row.multiplot = 1,
  multiplot.export.file = NULL,
  real.width.single.plot = 2.9,
  real.height.single.plot = 3.5,
  by.row = TRUE,
  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 or density.matrix.
plot.by.group	Logical value to define whether plot by group of regions or by sample. By default TRUE.
missing.data.as.zero	Logical 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	Logical 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 or string vector to define the X-axis label for all the plots. By default `NULL`, the label will be defined automatically.

`y.lab` Single string or string vector 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.interval` A number indicating the interval/bin spacing two ticks on the Y-axis. By default `NULL`: ticks are assigned automatically. Active only when `y.identical.auto = TRUE` and `y.lim != NULL`.

`y.digits` A numeric value to define the number of digits to use for the y.axis values. By default 1 (eg. 1.5).

`axis.line.width` Numeric value to define the axes and ticks line width for all plots. By default `0.5`.

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

`legend.position` Any ggplot supported value for the legend position (eg. "none", top", "bottom", "left", "right", c(fraction.x, fraction.y)). By default `c(0.2,0.85)`.

`plot.vertical.lines` Logical 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` Logical 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"`

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

`multiplot.export.file`
: If a string with the name of a PDF file is provided the multiplot will be exported. By default `NULL`.

`real.width.single.plot`
: Numeric value, in inches, to define the real width of each plot in the multiplot exported, if required. By default 2.9 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 3.5 inches.

`by.row`
: Logical value to define whether the plots should be arranged by row. By default `TRUE`.

`print.multiplot`
: Logical 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 function 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, multiplot.export.file = "/path.to/multiplot.pdf",
  real.width.single.plot = 2.5, real.height.single.plot = 3,
  print.multiplot = FALSE)
```

plot.density.profile.smooth *Plot of NGS density signal at specific regions from deepTools matrices (signal smoothing version).*

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 and density.matrix functions from this package (signal smoothing version). The error on the line cannot be plotted in this case. See also plot.density.profile.

Usage

```
## S3 method for class 'density.profile.smooth'
plot(
  matrix.file,
  plot.by.group = T,
  missing.data.as.zero = NULL,
  sample.names = NULL,
  region.names = NULL,
  signal.type = "mean",
  smooth.span = 0.1,
  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.interval = NULL,
  y.digits = 1,
  axis.line.width = 0.5,
  text.size = 12,
  legend.position = c(0.2, 0.85),
  plot.vertical.lines = T,
  write.reference.points = T,
 colors = c("#00A5CF", "#F8766D", "#AC88FF", "#E08B00", "#00BA38", "#BB9D00", "#FF61C9",
    "gray30"),
  n.row.multiplot = 1,
  multiplot.export.file = NULL,
  real.width.single.plot = 2.9,
  real.height.single.plot = 3.5,
  by.row = TRUE,
  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 or density.matrix.
plot.by.group	Logical value to define whether plot by group of regions or by sample. By default TRUE.
missing.data.as.zero	Logical 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".
smooth.span	Numerical value to indicate the span value for the loess function used to smooth bigWig signals. By default 0.1.
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 or string vector to define the X-axis label for all the plots. By default NULL, the label will be defined automatically.
y.lab	Single string or string vector 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.interval`
: A number indicating the interval/bin spacing two ticks on the Y-axis. By default `NULL`: ticks are assigned automatically. Active only when `y.identical.auto = TRUE` and `y.lim != NULL`.

`y.digits`
: A numeric value to define the number of digits to use for the y.axis values. By default 1 (eg. 1.5).

`axis.line.width`
: Numeric value to define the axes and ticks line width for all plots. By default `0.5`.

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

`legend.position`
: Any ggplot supported value for the legend position (eg. "none", top", "bottom", "left", "right", c(fraction.x, fraction.y)). By default `c(0.2,0.85)`.

`plot.vertical.lines`
: Logical 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`
: Logical 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"`

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

`multiplot.export.file`
: If a string with the name of a PDF file is provided the multiplot will be exported. By default `NULL`.

`real.width.single.plot`
: Numeric value, in inches, to define the real width of each plot in the multiplot exported, if required. By default 2.9 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 3.5 inches.

`by.row`
: Logical value to define whether the plots should be arranged by row. By default `TRUE`.

`print.multiplot`
: Logical 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 function 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.smooth(
  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, multiplot.export.file = "/path.to/multiplot.pdf",
  real.width.single.plot = 2.5, real.height.single.plot = 3,
  print.multiplot = FALSE)
```

---

plot.density.summary *Plot the distribution of overall NGS density at specific regions from deepTools matrices.*

Description

Computes the score of each element in a list of regions and generates violins plots with percentiles and the mean (optional) for each sample/region. It uses as input a score matrix computed by deeptools's computeMatrix function or by computeMatrix.deeptools and density.matrix functions from this package.

Usage

```
## S3 method for class 'density.summary'
plot(
  matrix.file,
  plot.by.group = T,
  missing.data.as.zero = NULL,
  sample.names = NULL,
  region.names = NULL,
  signal.type = "mean",
  linear = F,
  error.type = "sem",
  show.mean = T,
```

```
  mean.error.type = "se",
  mean.color = "blue",
  mean.symbol.shape = 20,
  mean.symbol.size = 1,
  show.stat.multiplot = T,
  stat.method = "wilcox.test",
  stat.paired = F,
  stat.labels.format = "p.signif",
  stat.hide.ns = T,
 stat.p.levels = list(cutpoints = c(0, 1e-04, 0.001, 0.01, 0.05, 1), symbols = c("****",
    "***", "**", "*", "ns")),
  title = NULL,
  x.lab = NULL,
  y.lab = NULL,
  x.labs.angle = 0,
  dodge.width = 1,
  border.width = 0.5,
  border.color = "#000000",
  transparency = 0.5,
  subset.range = NULL,
  y.lim = NULL,
  y.identical.auto = T,
  y.ticks.interval = NULL,
  y.digits = 1,
  axis.line.width = 0.5,
  text.size = 12,
  legend.position = c(0.2, 0.85),
 colors = c("#00A5CF", "#F8766D", "#AC88FF", "#E08B00", "#00BA38", "#BB9D00", "#FF61C9",
    "gray30"),
  n.row.multiplot = 1,
  multiplot.export.file = NULL,
  real.width.single.violinplot = 1,
  real.height.single.violinplot = 3.5,
  by.row = TRUE,
  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 or density.matrix.
`plot.by.group`	Logical value to define whether plot by group of regions or by sample. By default `TRUE`.
`missing.data.as.zero`	Logical 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".

`linear` Logical value to define whether the plots should show the score in linear scale. By default `FALSE`.

`error.type` String indicating the type of error to be computed and that will be available in the output data.table. Available parameters are "sem" and "sd", standard error mean and standard deviation respectively. By default "sem". Parameter considered only when `show.mean = TRUE`).

`show.mean` Logical value to define whether the mean value should be shown as a symbol on the plots. By default `TRUE`.

`mean.error.type`
String indicating the type of error for the mean to be computed. Available parameters are "se", "sd" and, "none". Respectively standard error, standard deviation, and no error plotted. By default "se". Parameter considered only when `show.mean = TRUE`).

`mean.color` A single string expressing an R-supported color for the mean symbol. By default `"blue"`.

`mean.symbol.shape`
A numeric value or string defining the shape for the mean symbol. By default `20`.

`mean.symbol.size`
A numeric value defining the size of the mean symbol. By default `1`.

`show.stat.multiplot`
Logical value to define if to add to the plot the statistical comparisons of the means for the groups present in the multiplot. By default `TRUE`. All possibile comparisons will be performed.

`stat.method` A single string defining the method to use for the statistical comparisons. By default `"wilcox.test"`. Available options: "t.test" "wilcox.test".

`stat.paired` Logical value to define if the statistical comparisons should be performed paired. By default `"FALSE"`. Notice that to allow a paired comparison the number of data should be the same in the two groups compared, so in the most of the cases non applicable to the comparisons between two regions. Used only in `"t.test"` and `"wilcox.test"` methods.

`stat.labels.format`
A single string indicating the format of the p-value to show for the statistical comparisons. By default `"p.signif"`. Available options: "p.format" (normal p-value), "p.signif" (significance stars), "p.adj" (p-value adjusted).

`stat.hide.ns` Logical value indicating if the NS ("Not Significant") comparisons should be shown or not. By default `TRUE`.

`stat.p.levels` A list containing the p-values levels/thresholds in the following format (default): `list(cutpoints = c(0,0.0001,0.001,0.01,0.05,1),symbols = c("****","***","**","*","ns")`
In other words, we use the following convention for symbols indicating statistical significance:

- ns: p > 0.05
- * p <= 0.05
- ** p <= 0.01
- *** p <= 0.001
- **** p <= 0.0001

`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 or string vector to define the X-axis label for all the plots. By default `NULL`, the label will be defined automatically.

`y.lab` Single string or string vector to define the Y-axis label for all the plots. By default `NULL`, the label will be defined automatically.

`x.labs.angle` A single numeric value indicating the degrees of rotation of the category labels in the X-axis. By default `0`, horizontal without rotation.

`dodge.width` Numeric value defining the width of each single violin plot. By default `1`.

`border.width` Numeric value to define the border width for all the violin plots. By default `0.5`.

`border.color` A single string indicating the color to use for the border of the violin plots. By default `"#000000"` (full black).

`transparency` A numeric value to define the fraction of transparency of the plots fill (0 = transparent, 1 = full). By default `0.5`.

`subset.range` A numeric vector indicating the range to which restrict the analyses (eg. c(-150, 250)). In the case of "scale-region" mode, the range is represented by (-upstream | 0 | body_length | body_length+downstream).By default `NULL`: the whole region is considered.

`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.interval` A number indicating the interval/bin spacing two ticks on the Y-axis. By default `NULL`: ticks are assigned automatically. Active only when `y.identical.auto = TRUE` and `y.lim != NULL`.

`y.digits` A numeric value to define the number of digits to use for the y.axis values. By default 1 (eg. 1.5).

`axis.line.width`
: Numeric value to define the axes and ticks line width for all plots. By default `0.5`.

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

`legend.position`
: Any ggplot supported value for the legend position (eg. "none, "top", "bottom", "left", "right", c(fraction.x, fraction.y)). By default `c(0.2,0.85)`.

`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"`

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

`multiplot.export.file`
: If a string with the name of a PDF file is provided the multiplot will be exported. By default `NULL`.

`real.width.single.violinplot`
: Numeric value, in inches, to define the real width (not precise) of each single violin plot in the multiplot exported, if required. By default 1 inch.

`real.height.single.violinplot`
: Numeric value, in inches, to define the real height (not precise) of each single violin plot in the multiplot exported, if required. By default 3.5 inches.

`by.row`
: Logical value to define whether the plots should be arranged by row. By default `TRUE`.

`print.multiplot`
: Logical value to define whether to print the multiplot once generated. 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 function returns a list containing:

- `data.table` with the computed values used for the plot;
- `metadata` table with the information obtained from the matrix_file.gz;
- `plot.list` with a plot for each list element;
- `density.profile` with the density profile of the mean signal generated by plot.density.profile corresponding to the regions/samples for which the summary multiplot have been generated;
- `multiplot` with the image of all the plots together;
- `summary.plot.samples` with a plot showing the scores of all regions per each sample;

- summary.plot.regions with a plot showing the scores of all samples per each region;
- means.comparisons table with the statistical means comparisons (when show.stat.multiplot = TRUE, otherwise a string is returned).

---

plot.gsea *GSEA plotter*

Description

Function to plot GSEA results (see clusterprofiler).

Usage

```
## S3 method for class 'gsea'
plot(
  gsea.results,
  geneset.id = NULL,
  erinchment.geom = "line",
  erinchment.color = "green",
  enrichment.geom.size = 1,
  enrichment.plot.zero.line = FALSE,
  enrichment.zero.line.color = "gray",
  enrichment.zero.line.width = 0.5,
  enrichment.annotations.vjust.offset = 0,
  geneset.segments.width = 0.3,
  geneset.segments.color = "black",
  rank.max.color = "indianred",
  ranking.color = "gray",
  gradient.colors = c("Reds", "Blues"),
  title.position = "center",
  title = NA,
  image.file.name = NULL,
  image.width = 7,
  image.height = 5,
  return.all.objects = FALSE
)
```

Arguments

gsea.results	A gseaResult object as generate by clusterprofiler.
geneset.id	Numeric value or a string identifying the Nth geneSet (numeric) or a specific id (string) if geneSet in the result table. Default value: NULL, which returns the ordered list of available geneSets.
erinchment.geom	String indicating the type of graph to use to plot the enrichment scores. Possible options: 'line', 'lines', 'dot', 'dots', 'point', 'points' (case insensitive). Default: 'line'.

erinchment.color
: String indicating any R-supported color to be used for the enrichment score plot. Default: 'green'.

enrichment.geom.size
: Numeric value indicating the size of the line, or dots, used in the enrichment score plot. Default: 1.

enrichment.plot.zero.line
: Logical value to indicated whether to plot an horizontal line at 0 in the enrichment score plot. Default: FALSE.

enrichment.zero.line.color
: String indicating any R-supported color to be used for the 0-line in the enrichment score plot (active when enrichment.plot.zero.line = TRUE). Default: 'gray'.

enrichment.zero.line.width
: Numeric value indicating the line width of the 0-line in the enrichment score plot (active when enrichment.plot.zero.line = TRUE). Default: 0.5.

enrichment.annotations.vjust.offset
: Numeric value to add to the vjust (vertical positioning) of the enrichment plot annotations (P, Padj, q, NES, set size). Positive values will shift-down the annotations. Default: 0 (base line).

geneset.segments.width
: Numeric value indicating the line width of the geneSet vertical segments. Default: 0.3.

geneset.segments.color
: String indicating any R-supported color to be used for the geneSet segments. Default: 'black'.

rank.max.color
: String indicating any R-supported color to be used for the max rank dotted lines and annotation. Default: 'indianred'.

ranking.color
: String indicating any R-supported color to be used for the ranked list plot (histogram). Default: 'gray'.

gradient.colors
: Two-values string vector indicating the shadows of palettes to use for the genset gradient. Possible values: 'Blues', 'Greens', 'Greys', 'Oranges', 'Purples', 'Reds'. Default: c('Reds','Blues').

title.position
: String indicating the position of the title: 'left', 'center', 'right'. Default: 'center'.

title
: String indicating the title to use. Default: NA, this will automatically use the geneset name chosen. Use NULL to do not plot the title.

image.file.name
: String indicating the full path for the export of a pdf file of the combined plot. Default: NULL, no plot will be exported.

image.width
: Numeric value to indicate the width (in inches) to use for the exported pdf file. Active only when image.file.name is not NULL. Default: 7.

image.height
: Numeric value to indicate the height (in inches) t use for the exported pdf file. Active only when image.file.name is not NULL. Default: 5.

`return.all.objects`
: Logical value to indicate whether the function should return only the combined plot (ggplot object), or all the different panels and the combined plot in a list. Default: `FALSE` (only combined plot).

Value

Either a ggplot-object with the final combined plot, or a list with the three panels separated and the combined plot: `list(enrichment.panel,geneset.panel,rank.panel,combined.plot)`.

Examples

```
data(geneList, package = "DOSE")

msigdb_hallmarks =
  msigdbr::msigdbr(species = "Homo sapiens", category = "H") %>%
  dplyr::select(gs_name, human_entrez_gene)

gsea_H = clusterProfiler::GSEA(geneList = geneList,
                               TERM2GENE = msigdb_hallmarks,
                               minGSSize = 3,
                               maxGSSize = 800,
                               pAdjustMethod = "BH",
                               pvalueCutoff = 0.05,
                               verbose = T)

plot.gsea(gsea_H, geneset.id = "HALLMARK_ADIPOGENESIS")
plot.gsea(gsea_H, geneset.id = 28)
```

---

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))
```

---

qPCR.results.rep1 *qPCR RNA expression results example (rep1)*

Description

Simulation of appliedBiosystem qPCR results (rep1)

Usage

```
qPCR.results.rep1
```

Format

A data frame with 117 rows and 35 variables. Three of these columns are required to run qPCR.rna.exp:

`Sample Name` Name of the samples/conditions

`Target Name` The target genes to quantify

`CT` Values of the cycle detected at a given threshold

Source

Simulated data

qPCR.results.rep2 *qPCR RNA expression results example (rep2)*

Description

Simulation of appliedBiosystem qPCR results (rep2).

Usage

```
qPCR.results.rep2
```

Format

A data frame with 117 rows and 35 variables. Three of these columns are required to run qPCR.rna.exp:

`Sample Name` Name of the samples/conditions

`Target Name` The target genes to quantify

`CT` Values of the cycle detected at a given threshold

Source

Simulated data

qPCR.rna.exp *qPCR RNA expression analyses tool.*

Description

Allows to easily analyse qPCR RNA expression data, including: technical replicates verification, gene expression normalization to housekeeping genes and FoldChanges over reference sample computation.

Usage

```
qPCR.rna.exp(
  results.file,
  housekeeping.genes = NULL,
  max.delta.reps = 0.5,
  reference.sample = NULL,
  exclude.housekeeping.FC = TRUE,
  exclude.samples = NULL,
  fix.y.axis = FALSE,
  x.labels.rotation = 45,
  text.size = 3,
  results.sheet.position = 3,
```

```
  rows.to.skip = 44,
  file.header = TRUE,
  file.tail = TRUE,
  samples.order = NULL,
  ignore.reps.errors = FALSE
)
```

Arguments

results.file	String indicating the full path to the results excel file or a data.frame containing at least the following columns: 'Sample Name', 'Target Name', 'CT'.
housekeeping.genes	String vector with the list of genes that have to be used as target genes. By default NULL: an error message is printed.
max.delta.reps	Numeric value indicating the maximum difference among replicate Ct. Default value: 0.5.
reference.sample	Single string indicating the name of the sample to use as reference for the computation of the FoldChanges. By default NULL: the first sample in the order is used as reference.
exclude.housekeeping.FC	Logic value to indicate whether the housekeeping genes should be excluded in the FoldChanges plots. By default TRUE.
exclude.samples	String vector indicating the samples that should be exuded in the expression and FoldChange plots. By default NULL.
fix.y.axis	Logic value indicating whether the y-axis of the plots should be kept fixed among all the genes. By default FALSE.
x.labels.rotation	Numeric value indicating the degrees of x-axis's labels rotation. By default 45.
text.size	Numeric value to indicate the size of the text for the number above the bars. Default 3.
results.sheet.position	Numeric value indicating the position of the results sheet in the excel file. by default 3.
rows.to.skip	How many rows must be skipped before to read the excel file. By default 44.
file.header	Logic value to indicate whether the results excel file contains an header. By default TRUE.
file.tail	Logic value to indicate whether the results excel file contains extra rows at the end of the results. By default TRUE.
samples.order	A string vector indicating all the samples in order. This order will be used to order the samples in the plots. By default NULL: the reference sample will be the first, the other will be kept in the order available in the results table.
ignore.reps.errors	Logic value to define whether the difference between the Ct in replicates should be ignored: all the values are kept.

Value

The function returns a list containing:

- `original.table`: a data.frame containing the original results table;
- `reshaped.table`: a data.frame with the original results reorganized for the analyses;
- `reshaped.table.cleaned`: the reshaped data.frame upon filtering of the CT values (if required);
- `reps.validation.plot`: a plot representing a table with the differences two-by-two of the technical replicates (facet_wrapped by gene) where the cells have a red background if the difference is greater than the 'max.delta.reps' value;
- `analyzed.data`: a named list of data.frames, one for each housekeeping gene and one for the foldChange mean of all housekeeping normalization, containing the normalized expression scores and the FoldChanges over the reference sample;
- `expression.plots`: a named list of plots, one for each housekeeping gene, showing the gene expression histograms (facet_wrapped by gene);
- `foldChange.plots`: a named list of plots, one for each housekeeping gene and one for the foldChange mean of all housekeeping normalization, showing the FoldChange expression over the reference Sample (facet_wrapped by gene).

`qPCR.rna.mean.reps` *qPCR RNA expression experimental replicates mean calculator.*

Description

This function allows to generate a table and a plot result (FoldChange and normalized Expression) of the mean of different replicates of an experiment starting from analyses performed by qPCR.rna.exp.

Usage

```
qPCR.rna.mean.reps(
  reps.list,
  reference.sample = NULL,
  exclude.samples = NULL,
  exclude.housekeeping.genes = TRUE,
  plot.color = "#d1718b",
  fix.y.axis = FALSE,
  text.size = 3,
  x.labels.rotation = 45
)
```

Arguments

`reps.list`	A list of qPCR.rna.exp result objects.
`reference.sample`	Single string indicating the name of the sample to use as reference for the computation of the FoldChanges. By default `NULL`: if the input is a list of qPCR.rna.exp objects, the reference.sample is retrieved automatically. However, if the number of reference used are multiple and/or not shared among replicates, the first sample in the order is used as reference.
`exclude.samples`	String vector indicating the samples that should be exuded in the expression and FoldChange plots. By default `NULL`.
`exclude.housekeeping.genes`	Logic value to indicate whether the housekeeping genes should be excluded in the plot. By default `TRUE`.
`plot.color`	Single string to indicate the color to use for the bar plot. Default value: `#D1718B`.
`fix.y.axis`	Logic value indicating whether the y-axis of the plots should be kept fixed among all the genes. By default `FALSE`.
`text.size`	Numeric value to indicate the size of the text for the number above the bars. Default `3`.
`x.labels.rotation`	Numeric value indicating the degrees of x-axis's labels rotation. By default `45`.

Value

The function returns a list containing:

- `mean.reps.data.table`: a list of data.frames, one per housekeeping gene and the mean of all housekeeping genes, containing the number of reps (n), SD and SEM for each sample-target combination for both normalized expression and FoldChanges;
- `mean.reps.exp.plots`: a list of a plots, one per housekeeping gene, showing the replicates' normalized expression over the reference Sample (facet_wrapped by gene);
- `mean.reps.FC.plots`: a list of a plots, one per housekeeping gene and the mean of all housekeeping genes, showing the replicates' mean FoldChange expression over the reference Sample (facet_wrapped by gene).

`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.

`reorder.samples.computeMatrix`

Sample reorderer computeMatrix file

Description

A tool to reorder the samples in a computeMatrix file avoiding the re-computation of the latter.

Usage

```
## S3 method for class 'samples.computeMatrix'
reorder(
  matrix.file,
  new.sample.order = NULL,
  reordered.matrix.path = gsub(".gz", "_reordered.gz", matrix.file),
  ignore.header.error = FALSE,
  verbose = TRUE
)
```

Arguments

`matrix.file`	String with the full path to a deeptools computeMatrix `.gz` file.
`new.sample.order`	String vector with the sample labels in the order in which they should appear in the matrix. By default NULL, which returns a message with the sample labels in the original order. Further, it returns the vector with the original sample order.
`reordered.matrix.path`	String with full path to for the file of the reorder matrix (.gz). By default the output name will be `<original.matrix.name>_reordered.gz`.

ignore.header.error
: Logical value to indicate whether the error of sample_label reassignment in the header should be ignored. The plotted labels can be changed during the plotting.

verbose
: Logical value to indicate whether the final message should be printed. By default TRUE.

Value

The output is a computeMatrix file (.gz format) with the samples chucks re-shuffled to be in the order provided by the user.

---

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>/restricti
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")
```

RNAseq *RNA-seq example*

Description

Dummy example of a DESeq2 result for differential expression analysis on RNA-seq data

Usage

```
RNAseq
```

Format

A data frame with 300 rows and 7 variables:

`geneName` genes symbols

`baseMean` The average of the normalized count values, dividing by size factors, taken over all samples

`log2FC` the log2 value of the Fold Change expression between two conditions

`lfcSE` log2 Fold Change standard error (SE)

`stat` Wald statistic

`pvalue` Wald test p-value

`padj` BH adjusted p-values

Source

Simulated data

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 = FALSE,
  sep = "\t",
  return.bed = TRUE,
  export.file.name = NULL,
  export.header = FALSE,
  unique.regions = TRUE,
  verbose = TRUE
)
```

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.file.name	Optional: string to define the path to the file to be exported, if required. By default NULL, not exported.
export.header	Logic value to define whether the header should be exported in the sorted bed file. By default FALSE.
unique.regions	Logic value to indicate whether the output bed must contain unique regions. By default TRUE.
verbose	Logic value to indicate whether messages should be printed or not. By default TRUE.

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, return.substracted.bw = T, substracted.bw.file = NULL)
```

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 `TRUE`.

`substracted.bw.file`
: String for the path of the resulting bigwig file to be exported. By default `NULL`, any file will be exported.

Value

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

`uniform.x.axis` *Plot X-axis uniforming*

Description

Takes a list of ggplot2 plots, compares their X-axis ranges and applies the highest/lowest limits to each plot in order to uniform all the plots. It can be used also to set the ticks step (to just change the breaks set all parameters as `FALSE`).

Usage

```
uniform.x.axis(
  plot.list,
  x.min = TRUE,
  x.max = TRUE,
  ticks.each = NULL,
  digits = 1
)
```

Arguments

`plot.list`	A single plot or a list of plots.
`x.min`	Either a logical value to define whether uniform the lower limit or a numeric value defining the lower limit. By default `TRUE`.
`x.max`	Either a logical value to define whether uniform the upper limit or a numeric value defining the upper limit. By default `TRUE`.
`ticks.each`	Numeric value to define every how much should be placed a tick. By default `NULL`, ticks will be placed automatically.
`digits`	A single integer indicating the maximum number of digits required for the rounding of the axis values. By default `1`.

Value

Returns a plot list (or a single plot when only one input plot is provided) equivalent to the input list provided by the user in which the X-axis of all the plots will be uniformed.

uniform.y.axis *Plot Y-axis uniforming*

Description

Takes a list of ggplot2 plots, compares their Y-axis ranges and applies the highest/lowest limits to each plot in order to uniform all the plots. It can be used also to set the ticks step (to just change the breaks set all parameters as FALSE).

Usage

```
uniform.y.axis(
  plot.list,
  y.min = TRUE,
  y.max = TRUE,
  ticks.each = NULL,
  digits = 1
)
```

Arguments

plot.list	A single plot or a list of plots.
y.min	Either a logical value to define whether uniform the lower limit or a numeric value defining the lower limit. By default TRUE.
y.max	Either a logical value to define whether uniform the upper limit or a numeric value defining the upper limit. By default TRUE.
ticks.each	Numeric value to define every how much should be placed a tick. By default NULL, ticks will be placed automatically.
digits	A single integer indicating the maximum number of digits required for the rounding of the axis values. By default 1.

Value

Returns a plot list (or a single plot when only one input plot is provided) equivalent to the input list provided by the user in which the Y-axis of all the plots will be uniformed.

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()
```

---

venn.overlap *VennDiagram from region overlaps*

Description

A tool to plot VennDiagrams from overlaps between bed files/regions derived from different formats. The function allows the overlap in stranded mode and can considered a specific minimal percentage of overlap between regions.

Usage

```
venn.overlap(
  region.list,
  region.names = LETTERS[1:length(region.list)],
 colors = c("#00A5CF", "#F8766D", "#AC88FF", "#E08B00", "#00BA38", "#BB9D00", "#FF61C9",
    "gray30"),
  color.transparency = 0.25,
  min.percentage.reference = 0,
  min.percentage.test = 0,
  min.bases.overlap = 1,
  input.type = "union",
  shape.type = "ellipse",
  plot.quantities = TRUE,
  stranded = FALSE
)
```

Arguments

region.list	A list of regions to be used as to compute the overlaps. The values accepted are: a. a character with the full path to a bed file, b. a data.frame in at least BED3 format, c. a GRanges object in at least BED3 format. If a list of elements is provided all the regions will be merged in a unique combined list and only completely identical regions will be remove to avoid duplicates. Combination of different formats is allowed.
region.names	String vector with the names of the regions in the order.
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"
color.transparency	Numeric floating value between 0-1 to indicate the transparency, aka alpha, of the colors.
min.percentage.reference	A numeric value in 0-100 to define which percentage of a region in the 'reference' dataset must overlap with a region in the 'test' one. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: 0.
min.percentage.test	Numeric value in 0-100 to define which percentage of a region in the 'test' dataset must overlap with a region in the 'reference' one. If the value is lower than 0 or greater than 100, will be coerced to 0 or 100 respectively. Default value: 0.
min.bases.overlap	Integer, greater than 0, value to indicate the minimal number of bases to consider as minimum overlap between two regions. Non integer values will be rounded at integer, while number lower that 1 will be coerced to 1. Default value: 1.
input.type	String with the type of input provided to the euler function. Available values are union and disjoint. Default value: union.
shape.type	String with the type of shape to use for the plot: one among ellipse and circle. Default value: ellipse.
plot.quantities	Logical value to indicate whether the quantity of each subintersection should be plotted or not. By default TRUE.
stranded	Logical value to define whether the analyses should be performed by strand: regions in one strand will be overlapped only with regions of the same strand. The strand symbols considered are '+' and '-', any other symbol will considered in a unique separated category. Default value: FALSE.

Value

The output is the Venn Diagram in an object of class eulergram/gTree/grob/gDesc.

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 = bquote("log"["2"] * "(Fold Change expression)"),
  y_label = bquote("-log"["10"] * "(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