Package 'diffwrap'

August 30, 2026

Type Package

Title Differential Expression Analysis of RNA-Seq Data

Version 0.6-3

Date 2026-08-10

Description Functions for differential expression analysis of read counts from messenger RNA (mRNA) sequencing (RNA-Seq) data or micro RNA (miRNA) expression values generated by the Comprehensive Analysis Pipeline for microRNA Sequencing (CAP-miRSeq) 'expression_reports.sh' script. The workflow follows the 'edgeR'-'limma' expression data analysis pipeline providing options for different approaches, such as ``pure'' 'edgeR', voom or paired samples. The functions in the package generate text files with differential expression lists, optionally annotated with information from 'biomart', expression summary plots as well as several quality control (QC) plots. The main function, diffExpr(), is a convenience wrapper performing all steps automatically based on sensible defaults. Methods are described in Robinson, McCarthy and Smyth (2010) <doi:10.1093/bioinformatics/btp616>, Ritchie et al. (2015) <doi:10.1093/nar/gkv007>, Law et al. (2014) <doi:10.1186/gb-2014-15-2-r29> and Sun et al. (2014) <doi:10.1186/1471-2164-15-423>.

Maintainer Vidal Fey <vidal.fey@gmail.com>

Depends R (>= 4.0.0)

Imports plyr, ggplot2, edgeR, limma, RColorBrewer, convertid (>= 0.3.4), pheatmap, ggrepel, data.table, magrittr, methods, Hmisc, ltm, openxlsx, purrr, dplyr, venn, VennDiagram, grid, scales

Suggests testthat (>= 3.0.0), withr, futile.logger, rappdirs, knitr, quarto, dendextend, rmarkdown, AnnotationDbi, org.Hs.eg.db, org.Mm.eg.db, clusterProfiler, gprofiler2, topGO, igraph, scatterplot3d, readxl, WriteXLS, biomaRt

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder quarto

NeedsCompilation no

Author Vidal Fey [aut, cre],
Meeri Pekkarinen [aut],
Reija Hieta [aut],
Bogdan Iancu [aut],
Adrien Janssens [aut]

Config/roxygen2/version 8.0.0

Repository CRAN

Date/Publication 2026-08-30 09:10:19 UTC

Contents

`correlogram_pheatmap` *Function to create a correlogram pheatmap, i.e., a plot to check randomness in the data set.*

Description

Function to create a correlogram pheatmap, i.e., a plot to check randomness in the data set.

Usage

```
correlogram_pheatmap(
  expr.mat,
  clinical.mat,
  scale.fl = "none",
  legend.fl = TRUE,
  row.clust = TRUE,
  col.clust = TRUE,
  signif.stars.fl = FALSE,
  cell.size = 8,
  font.size = 11,
  color.blind.pal = "PuOr",
  color.extremes = c("#3182BD", "#E6550D"),
  anno.color = NULL,
  main.correl = "Correlogram",
  sample.correl = FALSE
)
```

Arguments

`expr.mat`	differential expression matrix in (genes, samples) format
`clinical.mat`	matrix with clinical annotation values in (clinical category, samples) format
`scale.fl`	character indicating if values should be centred and scaled in either the row direction or the column direction, or none (values ("row","column","none"), default = none)

legend.fl	logical to determine if legend should be drawn or not (default = TRUE)
row.clust	boolean values determining if rows should be clustered
col.clust	boolean values determining if cols should be clustered and corresponding heatmap output, assuming clinical.mat is provided the colors of the heatmap, otherwise min-max breaks are used by default
signif.stars.fl	boolean determining whether significance stars of p-values are shown in the correlogram (default = FALSE)
cell.size	double determining the width and height of the cell and the row/col font size (default = 8)
font.size	double determining the font size (default = 10)
color.blind.pal	string determining the RColorBrewer color blind palette (default = "PuOr"); other option can be visualized with the following command: brewer.pal.info[brewer.pal.info$colorblind,]
color.extremes	character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.
anno.color	list of named character vectors giving the colours used in the heatmap annotation bars. See 'annotation_colors' in [pheatmap()]. Automatically generated if NULL (default).
main.correl	Character; main title of the Correlogram
sample.correl	Logical;

Details

Calculates Pearson's rank correlation coefficients for all possible pairs of the input matrix and plots the resulting correlation matrix, either with significance stars and coloured according to correlation or without stars and coloured according to clinical annotation.

Value

Returns a pheatmap plot object used in the diffr_pheatmap() function.

Author(s)

Bogdan Iancu - Genevia Technologies Oy

---

diffExpr *Main wrapper for executing the entire pipeline from reading in expression data such as count files to producing text files and graphs*

Description

diffExpr is a convenience wrapper performing all steps automatically. Most sub-functions are exported and can be called by the user, as well, if desired. These functions may be applicable to different kinds of data/input, rely, however, on the conventions set for this package.

Usage

```
diffExpr(
  expr.dat,
  samp.info,
  control,
  design = NULL,
  samples = NULL,
  sample.plot.names = NULL,
  groups = NULL,
  pairs = NULL,
  block = FALSE,
  contrasts = NULL,
  out.dir = NULL,
  analysis.name = NULL,
  filter.strict = TRUE,
  filter.min.samp = NULL,
  fit.voom = FALSE,
  fit.voom.fun = "voomLmFit",
  fit.use.weights = FALSE,
  fit.norm.method = c("tmm", "quantile"),
  fit.disp = c("gene", "trend", "common"),
  fit.bayes.trend = FALSE,
  fit.bayes.robust = FALSE,
  fit.quasi.likelihood = NA,
  p.thr = 0.05,
  fdr.thr = 0.05,
  logfc.thr = 1,
  numlab = 25,
  point.lab = TRUE,
  de.plot.base.size = 16,
  biom.use = FALSE,
  biom.data.set = "hsapiens_gene_ensembl",
  biom.mart = "ensembl",
  biom.host = "https://www.ensembl.org",
  biom.timeout = 10,
  biom.filter = "ensembl_gene_id",
 biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description", "entrezgene_id"),
  biom.force.ensg = FALSE,
  biom.cache = NULL,
  biom.use.cache = FALSE,
  biom.sym.col = "hgnc_symbol",
  biom.rm.dups = FALSE,
  qc.top.n = 500,
  qc.gene.selection = "common",
  qc.pc = c(1, 2, 3),
  qc.type = c("both", "uncorrected", "pseudo-corrected"),
  qc.ellipse = TRUE,
  qc.ellipse.groups = NULL,
```

```
  qc.ellipse.legend = NA,
  qc.label.samples = TRUE,
  qc.point.size = 2,
  qc.label.size = 5,
  qc.circle = TRUE,
  qc.varname.size = 0,
  qc.var.axes = FALSE,
  hm.topn = 100,
  hm.p.thr = 0.05,
  hm.fdr.thr = 0.05,
  hm.logfc.thr = 1,
  hm.split.expr = FALSE,
  hm.pal.blind = "PuOr",
  hm.pal.n = 11,
  hm.pal.extremes = c("#3182BD", "#E6550D"),
  hm.pal.length = NULL,
  hm.anno.color = NULL,
  hm.anno.name = "Sample Class",
  enr.do = TRUE,
 enr.methods = c("clusterProfilerGO", "clusterProfilerKEGG", "gProfileR", "topGO"),
  enr.plot = FALSE,
  enr.plot.fdr.thr = fdr.thr,
  enr.plot.logfc.thr = logfc.thr,
  enr.plot.num.terms = 5,
  out.plots = TRUE,
  out.tables = TRUE,
  out.filtered.tables = TRUE,
  verbose = TRUE,
  log.file = NULL,
  dry.run = FALSE
)
```

Arguments

`expr.dat`	`character` or `list`. String or vector or list of input file paths, or matrix of count values
`samp.info`	`data.frame`. samp.info object containing information of the project's sample sheet
`control`	`character`. Name of the control group
`design`	`matrix`. design matrix
`samples`	`character`. Name of the column in 'samp.info' containing sample names. If 'samp.info' is not supplied vector of sample names.
`sample.plot.names`	`character`. Optional name of a column with "nice" sample names for plotting.
`groups`	`character`. Name of the column in 'samp.info' containing grouping information. If 'samp.info' is not supplied vector of groups.

pairs — character. Name of the column in 'samp.info' containing paired sample information.

block — logical. Are the samples not independent? See Details section.

contrasts — character. Vector of contrasts to be made. If not provided, all possible contrasts will be made. This specifies group name pairs to be compared in the format expected by makeContrasts(), i.e., "group2-group1".

out.dir — character. Path to the output directory. This argument is required: all result tables, plots and the run log are written below it. There is deliberately no default, so that no files are ever created in the working directory unintentionally. Use, e.g., out.dir = tempdir() to try the pipeline out without keeping the results.

analysis.name — character. Name of the analysis. If not provided, a default name will be generated.

filter.strict — logical. For miRNA analysis: only keep a miRNA if there are > 5 reads per million in at least half of the samples?

filter.min.samp — integer. Number of samples in which a feature needs to be covered by at least one read per million. Defaults to the size of the smallest group of replicates. See *details*.

fit.voom — logical. Should the voom function be used? Defaults to FALSE.

fit.voom.fun — character. The voom function to be used. Should be one of 'limma::voom', 'limma::voomWithQualityWeights' or 'edgeR::voomLmFit'. Defaults to edgeR::voomLmFit. See 'Details'.

fit.use.weights — logical. Should sample-specific quality weights be estimated?

fit.norm.method — character. The normalisation method to be used. Defaults to "tmm".

fit.disp — character. The dispersion method to be used. Defaults to "gene".

fit.bayes.trend — logical. Should an intensity-trend be allowed for the prior variance? Passed to 'limma::eBayes'.

fit.bayes.robust — logical. Should the estimation of df.prior and var.prior be robustified against outlier sample variances? Passed to 'limma::eBayes'.

fit.quasi.likelihood — Logical; should quasi-likelihood methods be used? See *Details* section. Defaults to NA, which will determine the method based on the number of replicate samples. If more than 4 replicates are present, the likelihood ratio test is used, otherwise the quasi-likelihood methods. If TRUE, then the quasi-likelihood methods are used, if FALSE, then the likelihood ratio test is used.

p.thr — numeric. Threshold for p-values. Defaults to 0.05.

fdr.thr — numeric. Threshold for FDR values. Defaults to 0.05.

logfc.thr — numeric. Threshold for fold-change values on the log2-scale. Defaults to 1.

numlab — numeric. Maximum number of point labels to be shown in the plot. This overrides/limits values calculated by any thresholds. Defaults to 25.

point.lab logical. Should point labels be shown in the plot? Defaults to TRUE.

de.plot.base.size
numeric. Overall text/point scale (the 'ggplot2' base_size) for the M-A and volcano plots, defaulting to 16. Raising it enlarges legend, axes, titles, point labels and points together. The default (16, up from the 'ggplot2' default of ~11) keeps text readable on the default per-contrast plot PDF, which is opened at 15x15 inches to hold the heatmaps.

biom.use logical. Should the biomart be used for gene annotation? Defaults to FALSE.

biom.data.set character. The biomart dataset to be used. Defaults to "hsapiens_gene_ensembl".

biom.mart character. The biomart to be used.

biom.host character. The host to be used for the biomart. Defaults to "www.ensembl.org".

biom.timeout numeric. Time budget in seconds for each attempt of the BioMart availability check performed at the start of a run when biom.use=TRUE. A host that does not answer within this budget is treated as unreachable, so a server that is down costs seconds rather than a full network timeout per candidate host. This applies only to the reachability probe, never to the annotation query itself, which may legitimately take much longer. Raise it if a slow but working mart is being rejected. Defaults to 10.

biom.filter character. The biomart filter to be used. Defaults to "ensembl_gene_id".

biom.attributes
character. The biomart attributes to be used. Defaults to c("ensembl_gene_id", "hgnc_symbol", "description", "entrezgene_id").

biom.force.ensg
logical. Should Ensembl Gene IDs be checked for (and stripped of) Ensembl version numbers? Defaults to FALSE.

biom.cache character. Path name giving the location of the cache getBM() uses if use.cache=TRUE. Defaults to the value in the *BIOMART_CACHE* environment variable.

biom.use.cache (logical). Should getBM() use the cache? Defaults to TRUE as in the getBM() function and is passed on to that.

biom.sym.col character. Name of the column in the query result with gene symbols

biom.rm.dups logical. Should duplicates be removed from the output of the biomart request? Defaults to FALSE.

qc.top.n integer. Passed to plotMDS() (top): number of top genes used to calculate pairwise distances. Defaults to 500.

qc.gene.selection
character. passed to plotMDS() specifying the mode to select genes for comparisons. Defaults to "common".

qc.pc numeric. Which principal components to plot. Defaults to 1:3.

qc.type character. Which type of plot to produce. Needs to be one of "both", "uncorrected", "pseudo-corrected" describing which values should be plotted. "uncorrected" will plot the input counts matrix while "pseudo-corrected" will plot pseudo counts for blocked designs (e.g., paired samples or batch factors). Defaults to "both".

qc.ellipse logical. Should an ellipse be plotted around samples belonging to the same sample group? Defaults to TRUE.

qc.ellipse.groups
character The name of the column in 'samp.info' with group names for ellipse drawing. If NULL (default) will use the groups column. If 'samp.info' is not supplied vector of groups.

qc.ellipse.legend
logical. Should a legend be added for ellipses in PCA plots? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

qc.label.samples
logical. Should points in appropriate QC plots be labelled. So far, applies only to PCA ggplot. Defaults to TRUE.

qc.point.size numeric. Size of points in appropriate QC plots. So far, applies only to PCA ggplot. Defaults to 2.

qc.label.size numeric. Font size used for point labels in appropriate QC plots. So far, applies to PCA ggplot and M-A plots. Defaults to 5.

qc.circle logical. Draw a correlation circle around points representing correlating samples? Only applies when prcomp was called with scale = TRUE and when var.scale = 1. Defaults to TRUE.

qc.varname.size
numeric. Size of the text for variable names. Defaults to 0.

qc.var.axes logical. Draw arrows for the variables? Defaults to FALSE.

hm.topn numeric. Number of top values to be plotted. Defaults to 100.

hm.p.thr, hm.fdr.thr, hm.logfc.thr
numeric. Significance and fold-change thresholds used specifically for selecting genes shown in the heatmaps, kept separate from 'p.thr', 'fdr.thr' and 'logfc.thr' (which control the plots and tables) because heatmaps usually read best with a stricter gene set. Default to 0.05, 0.05 and 1.

hm.split.expr logical. Should the top up- and top down-regulated genes be displayed at equal numbers (50/50), if they meet the significance threshold (regardless of the actual significance)? Defaults to FALSE.

hm.pal.blind string determining the RColorBrewer color blind palette (default = "PuOr"); other option can be visualized with the following command: brewer.pal.info[brewer.pal.info$colorblind,]

hm.pal.n desired length of the number of different colours in 'color.blind.pal'. Will also be used as length of the numeric vector of probabilities in 'quantile_breaks()' (see ?quantile); defaults to 11

hm.pal.extremes
character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.

hm.pal.length integer setting the desired length of the colour palette to be used in the heatmap

hm.anno.color list of named character vectors giving the colours used in the heatmap annotation bars. See 'annotation_colors' in [pheatmap()]. Automatically generated if NULL (default).

`hm.anno.name` character string used as the column annotation legend title. If 'anno.color' is not NULL and of length 1 the slot name will be used if existing.

`enr.do` `logical`. Whether or not to call enrichment wrapper. Defaults to `TRUE`.

`enr.methods` `character`. One or more of the following: c("clusterProfilerGO", "clusterProfilerKEGG", "gProfileR", "topGO"). By default, uses them all.

`enr.plot` `logical`. Whether or not to draw a network plot for the enrichment results. Defaults to `FALSE`.

`enr.plot.fdr.thr` `numeric`. FDR threshold used in the enrichment plot. This may be useful to tweak to produce a more informative plot. Defaults to 0.05.

`enr.plot.logfc.thr` `numeric`. FC threshold on the log2-scale used in the enrichment plot. Defaults to 1.

`enr.plot.num.terms` `integer`. Number of terms shown in the plot. Defaults to 5.

`out.plots` `logical`. Should plots be produced? Defaults to `TRUE`.

`out.tables` `logical`. Should lists of differentially expressed genes be produced? Defaults to `TRUE`.

`out.filtered.tables` `logical`. Should lists of differentially expressed genes be produced for filtered data? Defaults to `TRUE`.

`verbose` `logical` or `character`. Controls console output. `TRUE` (default) prints major workflow steps, `FALSE` prints nothing, and `"all"` mirrors the full detail of the log file to the console. Console output is written to `stdout` and is controlled solely by this argument. Irrespective of this setting, the complete log is always written to '`log.file`'.

`log.file` `character`. Path to the run log file. If `NULL` (default), a log file named *<analysis.name>_diffwrap.log* is created in '`out.dir`'. No log file is written if `dry.run=TRUE`.

`dry.run` `logical`. If `TRUE`, the function will not create any output files or directories.

Details

For experimental designs involving comparisons within as well as between subjects inter-subject needs to be computed. In this case, the column specified in the 'pairs' argument must assign the subjects to the treatment/tissue/etc groups. For example, if we have two treatments the effects of which are to be observed in each two tissues, this design would apply. The 'pairs' factor is passed to the functions 'duplicateCorrelation()' and 'lmFit()'. The 'block' argument is used to specify whether the comparisons are to be made within AND between subjects or in the case of technical replicates, i.e., if the samples are not independent, in other words, correlated. That correlation is addressed by means of the 'duplicateCorrelation()' function in the limma package. If 'block' is set to `TRUE`, the (selected) 'voom' function is enforced. As of version 0.4, the 'edgeR::voomLmFit()' function is incorporated, which replaces 'voom()', 'lmFit()' and 'voomWithQualityWeights()'. voomLmFit()' ensures unbiased estimation of the residual variances and automates the estimation of sample weights and intrablock correlations. In edgeR, it is recommended to remove features without at least 1 read per million in n of the samples, where n is the size of the smallest group of replicates (determined from the 'groups' vector). The 'min.samp' argument is used to specify the number of

samples in which a feature needs to be covered by at least one read per million. Quasi-likelihood pipeline: While the likelihood ratio test is a more obvious choice for inferences with GLMs, the QL F-test is preferred as it reflects the uncertainty in estimating the dispersion for each gene. It provides more robust and reliable error rate control when the number of replicates is small. The QL dispersion estimation and hypothesis testing is done by using the functions glmQLFit() and glmQLFTest().

Value

A list of all relevant objects generated in the course of the workflow:

- if 'voom=TRUE', voomed counts, otherwise DGEList object with TMM-normalisation factors
- if 'voom=TRUE', lmFit object, otherwise DGEList object with estimated dispersion
- if 'voom=TRUE', eBayes output of contrasted groups, otherwise glmFit object
- annotated topTable/topTags output

However, the function is first and foremost called for its side effects of generating results tables and plots.

Argument groups

The arguments are grouped by name prefix:

- no prefix: core experiment definition, plus the primary differential-expression thresholds (`p.thr`, `fdr.thr`, `logfc.thr`, `numlab`, `point.lab`);
- `filter.*`: count pre-filtering;
- `fit.*`: model fitting (voom / edgeR options);
- `biom.*`: 'biomart' gene annotation;
- `qc.*`: quality-control plots (MDS / PCA);
- `hm.*`: heatmaps, including their own significance thresholds and colour palette;
- `enr.*`: enrichment analysis;
- `out.*`, `verbose`, `log.file`, `dry.run`: output and run control.

Examples

```
out.dir <- file.path(tempdir(), "diffwrap_demo")
dir.create(out.dir, showWarnings = FALSE)
res <- diffExpr(expr.dat = diffwrap_counts,
                samp.info = diffwrap_samp_info,
                samples = "SampleName", groups = "Group",
                control = "control", analysis.name = "demo",
                out.dir = out.dir, enr.do = FALSE)
names(res$contrasts)
```

`diffr_expr_generate_cleaned_de_table_output` *Helper function to generate an output table with only most relevant columns*

Description

Helper function to generate an output table with only most relevant columns

Usage

```
diffr_expr_generate_cleaned_de_table_output(
  contrast,
  annotated.normcnt,
  out.dir,
  samp.name.and.group.key,
  analysis.name = NULL,
  filtered.lists = TRUE,
  fdr.thr = 0.05,
  logfc.thr = 1
)
```

Arguments

`contrast`	Character; a contrast group name pair in the typical format, i.e., "group2-group1"
`annotated.normcnt`	data frame; annotated output table generated by performing pair-wise comparisons and corresponding statistical tests. From this the normalised counts are extracted.
`out.dir`	Character; output directory for final tables. Required; no default is used so that nothing is written to the working directory unintentionally.
`samp.name.and.group.key`	`data.frame` with sample names (matching the count columns of `annotated.normcnt`) as row names and a `group` column giving each sample's group. Used to drop count columns of samples outside the contrast in multi-group designs; may be `NULL`.
`analysis.name`	Character used in the output file name.
`filtered.lists`	Logical; should the DGE table be filtered before saving it to a file?
`fdr.thr`	Numeric; the FDR threshold for filtering.
`logfc.thr`	Numeric; the fold-change threshold on the log2-scale for filtering.

Value

No return value. Called for its side effect of writing a reduced and optionally significance-filtered differential expression table to a tab-separated file below '`out.dir`'.

Examples

```
set.seed(1)
key <- data.frame(group = rep(c("control", "treated"), each = 4),
                  row.names = paste0("S", 1:8))
d3 <- data.frame(ID = paste0("g", 1:50), gene_symbol = paste0("G", 1:50),
                 logFC = rnorm(50), FDR = runif(50))
diffr_expr_generate_cleaned_de_table_output(
  contrast = "treated-control", annotated.normcnt = d3,
  samp.name.and.group.key = key, out.dir = tempdir(),
  analysis.name = "demo")
```

`diffr_pheatmap` *Function to create a heatmap from differential gene expression values*

Description

Function to create a heatmap from differential gene expression values

Usage

```
diffr_pheatmap(
  expr.mat,
  clinical.mat,
  scale.fl = "none",
  legend.fl = TRUE,
  row.clust = TRUE,
  col.clust = TRUE,
  biserial.fl = FALSE,
  quantile.breaks.fl = FALSE,
  signif.stars.fl = FALSE,
  cell.size = 8,
  font.size = 10,
  color.blind.pal = "PuOr",
  n.pal.cols = 11,
  color.extremes = c("#3182BD", "#E6550D"),
  palette.length = NULL,
  anno.color = NULL,
  main = NULL,
  add.main = NULL,
  filt.info = NULL
)
```

Arguments

`expr.mat`	differential expression matrix in (genes, samples) format
`clinical.mat`	matrix with clinical annotation values in (clinical category, samples) format

`scale.fl`	character indicating if values should be centred and scaled in either the row direction or the column direction, or none (values ("row","column","none"), default = none)
`legend.fl`	logical to determine if legend should be drawn or not (default = TRUE)
`row.clust`	boolean values determining if rows should be clustered
`col.clust`	boolean values determining if cols should be clustered
`biserial.fl`	= boolean values determining if biserial correlation is calculated and corresponding heatmap output, assuming clinical.mat is provided
`quantile.breaks.fl`	boolean values determining if quantile breaks are used to change the colors of the heatmap, otherwise min-max breaks are used by default
`signif.stars.fl`	boolean determining whether significance stars of p-values are shown in the correlogram (default = FALSE)
`cell.size`	double determining the width and height of the cell and the row/col font size (default = 8)
`font.size`	double determining the font size (default = 10)
`color.blind.pal`	string determining the color-blind-friendly palette; must be one of RColorBrewer's diverging palettes (default = "PuOr"); other option can be visualized with the following command: brewer.pal.info[brewer.pal.info$colorblind,]
`n.pal.cols`	desired length of the number of different colours in 'color.blind.pal'. Will also be used as length of the numeric vector of probabilities in 'quantile_breaks()' (see ?quantile); must be within 3:11; defaults to 11
`color.extremes`	character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.
`palette.length`	integer setting the desired length of the colour palette to be used in the heatmap
`anno.color`	list of named character vectors giving the colours used in the heatmap annotation bars. See 'annotation_colors' in [pheatmap()]. Automatically generated if NULL (default).
`main`	`character`. Main plot title.
`add.main`	`character`. Optional text added in parenthesis to the plot title.
`filt.info`	`character`. Optional information on the filtering strategy added in parenthesis to the plot title.

Details

The plot is produced as a side effect of function 'pheatmap_plots()' by printing the individual plot objects generated by 'diffr_pheatmap()'.

Value

Returns a list of pheatmap plot objects used in the pheatmap_plots() function.

Author(s)

Bogdan Iancu - Genevia Technologies Oy

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
expr <- edgeR::cpm(counts, log = TRUE)[1:30, ]
clin <- data.frame(Group = si$Groups, row.names = si$SampleNames)
hm <- diffr_pheatmap(expr, clin)
```

---

diffr_venn *Function to produce a Venn diagram of differentially expressed gene tables*

Description

Function to produce a Venn diagram of differentially expressed gene tables

Usage

```
diffr_venn(list.comp.tables, join.vec = "gene_symbol", .log = FALSE)
```

Arguments

list.comp.tables	list of DE tables, preferably a list of data.frames
join.vec	vector to perform the join operation on; corresponds to column names in the DE tables. Defaults to "gene_symbol", the column present in every diffwrap DE table. A longer vector (e.g. also including "ensembl_gene_id") joins on more annotation columns, but note that every column named here must exist in each input table.
.log	Logical; should logging be done?

Details

The actual plot is produced in the main plotting function by means of 'grid::grid.draw()' using the plot object as input.

Value

A list containing the Venn diagram grid object and the intersected (or joined) input tables.

Author(s)

Bogdan Iancu - Genevia Technologies Oy

Examples

```
mk <- function(sig) data.frame(gene_symbol = paste0("G", sig), fdr = 0.01)
tabs <- list(A = mk(1:20), B = mk(10:30))
v <- diffr_venn(tabs, join.vec = "gene_symbol")
names(v)
```

`diffwrap` *Differential expression analysis of RNA-Seq data*

Description

This package provides functionality for differential expression analysis of read counts from mRNA sequencing data or miRNA expression values generated by the CAP-miRSEQ 'expression_reports.sh' script. The workflow follows the 'edgeR'-'limma' expression data analysis pipeline providing options for different approaches, such as "pure" 'edgeR', 'voom' or paired samples. The functions in the package generate text files with differential expression lists, optionally annotated with information from 'biomart', expression summary plots as well as several QC plots.

Details

Package:	diffwrap
Type:	Package
Initial version:	0.1-0
Created:	2015-08-27
License:	GNU GPL v3
LazyLoad:	yes

The package contains utilities and methods for differential expression analysis of RNA-Seq data. It comes with a wrapper function, "diffExpr", performing all steps of the analysis. Most sub-functions are exported and can be called by the user, as well, if desired. These functions may be applicable to different kinds of data/input, rely, however, on the conventions set for this package.

Acknowledgements

This work was largely supported by university-level strategic and profiling area funding schemes at University of Turku and Tampere University. Downstream analysis functionality including all pathway analyses and visualisations as well as some DGE visualisations and helper functions were created or further developed at Genevia Technologies Oy.

Author(s)

Vidal Fey <vidal.fey@gmail.com>, Meeri Pekkarinen <meeri.pekkarinen@gmail.com>, Reija Hieta <reija.hieta@geneviatechnologies.com>, Bogdan Iancu <bogdan.iancu@abo.fi>, Adrien Janssens <Adrien.RFM.Janssens@proton.me> Maintainer: Vidal Fey <vidal.fey@gmail.com>

`diffwrap_counts` *Simulated RNA-Seq read counts for the package examples*

Description

A small simulated count matrix used throughout the examples, tests and the vignette. The data are **not** real measurements: they were generated from a negative binomial (gamma-Poisson) model so that the package can be demonstrated and tested without any external data or network access.

Usage

```
diffwrap_counts
```

Format

An `integer` matrix with 405 rows (400 genes plus 5 summary rows) and 8 columns (samples `S01`-`S08`). Row names are gene identifiers, column names sample names.

Details

The experiment consists of eight samples in two groups of four (`"control"` and `"treated"`). Each of the four subjects (`P1`-`P4`) contributes one control and one treated sample, so the data support paired and blocked analyses as well as the simple unpaired comparison. A subject-specific offset is built in, giving the paired designs a real effect to remove.

Sixty of the 400 genes are truly differentially expressed (30 up-, 30 down-regulated) with absolute log2 fold changes between 1.2 and 3. The biological dispersion is 0.15 (biological coefficient of variation of about 0.39) and library sizes are roughly 2-3 million reads.

Forty genes are given deliberately low expression so that they straddle the default `strict` filtering threshold of `diff_expr_filter_counts()` (more than 5 counts per million in at least half of the samples, which corresponds to about 13 counts here); 28 of them are removed by that filter.

The matrix additionally carries the five `htseq-count` summary rows (`__no_feature`, `__ambiguous`, `__too_low_aQual`, `__not_aligned` and `__alignment_not_unique`) that `diff_expr_filter_counts()` is expected to strip. They are retained on purpose so that the filtering step can be demonstrated.

Gene identifiers are well-formed but fictitious Ensembl gene IDs; they do not correspond to real genes and will not return annotation from 'biomart'.

Source

Simulated by '`data-raw/make_example_data.R`' from '`inst/extdata/example_counts.tsv`'.

See Also

`diffwrap_samp_info` for the matching sample sheet.

Examples

```
data(diffwrap_counts)
dim(diffwrap_counts)
head(diffwrap_counts[, 1:4])
# the htseq-count summary rows that get filtered out:
diffwrap_counts[grep("^__", rownames(diffwrap_counts)), 1:4]
```

`diffwrap_samp_info` *Sample sheet accompanying the simulated example counts*

Description

The sample information table matching `diffwrap_counts`, in the layout expected by `diffExpr()` and `diff_expr_get_samp_info()`.

Usage

```
diffwrap_samp_info
```

Format

A `data.frame` with 8 rows and 4 columns:

SampleName `character`. Sample identifier, matching the column names of `diffwrap_counts`.

Group `character`. Experimental group, `"control"` or `"treated"`.

Subject `character`. Subject identifier (`P1`-`P4`); pass to `pairs` for paired or blocked designs.

PlotName `character`. Human-readable sample label for plots.

Details

The `Subject` column pairs each control sample with a treated sample from the same subject and is the column to pass to the `pairs` argument. `PlotName` supplies prettier labels for plotting and is the column to pass to `sample.plot.names`.

Source

Simulated by '`data-raw/make_example_data.R`' from '`inst/extdata/example_samp_info.tsv`'.

See Also

`diffwrap_counts` for the matching count matrix.

Examples

```
data(diffwrap_samp_info)
diffwrap_samp_info
table(diffwrap_samp_info$Group, diffwrap_samp_info$Subject)
```

diff_expr_3d_scatterplot *Function to generate a 3D scatterplot*

Description

Function to generate a 3D scatterplot

Usage

```
diff_expr_3d_scatterplot(
  PCA,
  samp.name = NULL,
  groups,
  grp.nam = NULL,
  PC = c(1, 2, 3),
  main = NULL
)
```

Arguments

PCA	List of class prcomp or just a list having a component x that contains the rotated variables from prcomp.
samp.name	Optional sample names to be used in the plot, given as character vector.
groups	Sample groups for plot annotation as character vector or factor.
grp.nam	Legend title.
PC	Integer vector of length three specifying the principal components to be plotted.
main	Plot title.

Value

No return value. Called for its side effect of drawing a three-dimensional PCA scatterplot on the current graphics device.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
pca <- diff_expr_PCA(edgeR::cpm(counts, log = TRUE), n = 100)
diff_expr_3d_scatterplot(pca, groups = groups)
```

`diff_expr_biomart` *Function to retrieve additional information from biomart*

Description

Function to retrieve additional information from biomart

Usage

```
diff_expr_biomart(
  d3,
  biom.data.set = "hsapiens_gene_ensembl",
  biom.mart = "ensembl",
  host = "https://www.ensembl.org",
  biom.filter = "ensembl_gene_id",
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description"),
  biom.cache = NULL,
  use.cache = FALSE,
  sym.col = "hgnc_symbol",
  rm.dups = FALSE,
  force.ensg = FALSE,
  verbose = FALSE
)
```

Arguments

`d3`	data frame to be annotated with biomart; annotated output table generated by performing pair-wise comparisons and corresponding statistical tests.
`biom.data.set`	`character` of length one. Biomart data set to use.
`biom.mart`	`character` vector. Biomart to use (uses the first element of the vector), defaults to "ensembl".
`host`	`character` of length one. Host URL.
`biom.filter`	`character` of length one. Name of biomart filter, i.e., type of query ids, defaults to "ensembl_gene_id".
`biom.attributes`	`character` vector. Biomart attributes, i.e., type of desired result(s); make sure query id type is included!
`biom.cache`	`character`. Path name giving the location of the cache `getBM()` uses if `use.cache=TRUE`. Defaults to the value in the *BIOMART_CACHE* environment variable.
`use.cache`	(`logical`). Should `getBM()` use the cache? Defaults to `TRUE` as in the `getBM()` function and is passed on to that.
`sym.col`	`character`. Name of the column in the query result with gene symbols.
`rm.dups`	`logical`. Should duplicated input IDs (‘`biom.filter`’) be removed from the result?

force.ensg	logical. Should Ensembl Gene IDs be checked for (and stripped of) Ensembl version numbers? Defaults to FALSE.
verbose	logical. Passed to convert.bm()/getBM() to show query and chunk progress for large BioMart queries. Defaults to FALSE.

Value

A data.frame: the input table 'd3' merged with the annotation retrieved from biomart. The column holding gene symbols is renamed to gene_symbol.

Examples

```
# Needs network access to Ensembl BioMart. The query is wrapped in tryCatch() so that an
# unreachable or slow server reports the problem instead of failing the example.
if (requireNamespace("biomaRt", quietly = TRUE)) {
  d3 <- data.frame(ID = c("ENSG00000141510", "ENSG00000012048"))
  ann <- tryCatch(diff_expr_biomart(d3, biom.data.set = "hsapiens_gene_ensembl"),
                  error = function(e) {
                    message("BioMart not reachable: ", conditionMessage(e))
                    NULL
                  })
  if (!is.null(ann)) nrow(ann)
}
```

---

diff_expr_dendro_plot *Function to generate dendrogram plots based on hierarchical clustering*

Description

Function to generate dendrogram plots based on hierarchical clustering

Usage

```
diff_expr_dendro_plot(
  counts,
  groups,
  grp.nam = NULL,
  main = NULL,
  col.grps = FALSE
)
```

Arguments

counts	Counts matrix.
groups	Sample groups for plot annotation as character vector or factor.
grp.nam	Legend title.

main	Plot title.
col.grps	Logical indicating whether to colour the dendrogram by groups.

Value

No return value. Called for its side effect of drawing a hierarchical clustering dendrogram on the current graphics device.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
diff_expr_dendro_plot(edgeR::cpm(counts, log = TRUE), groups = groups)
```

---

diff_expr_extract_contrasts

Function to extract contrasts and generate top tables and plots

---

Description

Function to extract contrasts and generate top tables and plots

Usage

```
diff_expr_extract_contrasts(
  contrasts = NULL,
  fit,
  fit2 = NULL,
  normcnt,
  out.l,
  do.voom = TRUE,
  quasi.likelihood = TRUE,
  out.dir,
  analysis.name = NULL,
  biomart = FALSE,
  biom.data.set = "hsapiens_gene_ensembl",
  biom.mart = "ensembl",
  host = "https://www.ensembl.org",
  biom.filter = "ensembl_gene_id",
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description"),
  biom.force.ensg = FALSE,
  biom.cache = NULL,
  use.cache = FALSE,
  sym.col = "hgnc_symbol",
  rm.dups = FALSE,
```

```
  p.thr = 0.05,
  fdr.thr = 0.05,
  logfc.thr = 1,
  numlab = 25,
  point.lab = TRUE,
  heatmap.topn = 100,
  hm.p.thr = 0.05,
  hm.fdr.thr = 0.05,
  hm.logfc.thr = 1,
  heatmap.split.expr = FALSE,
  color.blind.pal = "PuOr",
  n.pal.cols = 11,
  color.extremes = c("#3182BD", "#E6550D"),
  palette.length = NULL,
  anno.color = NULL,
  anno.name = "Sample Class",
  heatmap.main = NULL,
  font.size = 5,
  plots = TRUE,
  lists = TRUE,
  filtered.lists = TRUE,
  samp.info = NULL,
  samples = NULL,
  groups = NULL,
  sample.plot.names = NULL,
  de.plot.base.size = 16
)
```

Arguments

contrasts	Contrast matrix as generated by makeContrasts(). If NULL, the default, the design matrix is expected to describe simple pair-wise comparisons, i.e., with the contrasts already included in the matrix. Consequently, in that case, no contrasts will be fitted.
fit	An MArrayLM object produced by lmFit.
fit2	An MArrayLM object produced by eBayes.
normcnt	Matrix; depth-adjusted reads per million or "voomed" counts, i.e., a numeric matrix of normalized expression values on the log2 scale.
out.l	List; list of result objects generated upstream in the workflow to add output from this function to.
do.voom	Logical; should data be "voomed"?
quasi.likelihood	Logical; should quasi-likelihood methods be used? See *Details* section. If TRUE, the default, then the quasi-likelihood methods are used, if FALSE, then the likelihood ratio test is used. Please note the main wrapper function diffExpr() will set this parameter.

out.dir	Character; output directory for final tables. Required; no default is used so that nothing is written to the working directory unintentionally.
analysis.name	Character used in the output file name.
biomart	Logical; should biomart annotation be added?
biom.data.set	character of length one. Biomart data set to use.
biom.mart	character vector. Biomart to use (uses the first element of the vector), defaults to "ensembl".
host	character of length one. Host URL.
biom.filter	character of length one. Name of biomart filter, i.e., type of query ids, defaults to "ensembl_gene_id".
biom.attributes	character vector. Biomart attributes, i.e., type of desired result(s); make sure query id type is included!
biom.force.ensg	logical. Should Ensembl Gene IDs be checked for (and stripped of) Ensembl version numbers? Defaults to FALSE.
biom.cache	character. Path name giving the location of the cache getBM() uses if use.cache=TRUE. Defaults to the value in the *BIOMART_CACHE* environment variable.
use.cache	(logical). Should getBM() use the cache? Defaults to TRUE as in the getBM() function and is passed on to that.
sym.col	character. Name of the column in the query result with gene symbols.
rm.dups	logical. Should duplicated input IDs ('biom.filter') be removed from the result?
p.thr	Numeric; P-Value threshold used in plots.
fdr.thr	Numeric; FDR threshold used in plots and for generating cleaned output tables.
logfc.thr	Numeric; FC threshold on the log2-scale used in plots and for generating cleaned output tables.
numlab	numeric. Maximum number of labels per plot. Overrides numbers calculated based on 'p.thr' and 'fdr.thr'.
point.lab	logical. Should points be labelled, at all?
heatmap.topn	numeric. Number of top values to be plotted. Defaults to 100.
hm.p.thr, hm.fdr.thr, hm.logfc.thr	numeric. Significance and fold-change thresholds used specifically when selecting genes for the heatmaps. These are kept separate from the plot/table thresholds ('p.thr', 'fdr.thr', 'logfc.thr') because heatmaps usually read best with a stricter gene set. Default to 0.05, 0.05 and 1.
heatmap.split.expr	logical. Should the top up- and top down-regulated genes be displayed at equal numbers (50/50), if they meet the significance threshold (regardless of the actual significance)? Defaults to FALSE.
color.blind.pal	string determining the RColorBrewer color blind palette (default = "PuOr"); other option can be visualized with the following command: brewer.pal.info[brewer.pal.info$colorblind,]

n.pal.cols	desired length of the number of different colours in 'color.blind.pal'. Will also be used as length of the numeric vector of probabilities in 'quantile_breaks()' (see ?quantile); defaults to 11
color.extremes	character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.
palette.length	integer setting the desired length of the colour palette to be used in the heatmap
anno.color	list of named character vectors giving the colours used in the heatmap annotation bars. See 'annotation_colors' in [pheatmap()]. Automatically generated if NULL (default).
anno.name	character string used as the column annotation legend title. If 'anno.color' is not NULL and of length 1 the slot name will be used if existing.
heatmap.main	character. Additional information printed in the heatmap title.
font.size	numeric. Size of point labels in M-A plots.
plots	Logical; should plots be generated?
lists	Logical; should output tables be written to files?
filtered.lists	Logical; should the DGE table be filtered before saving it to a file?
samp.info	data.frame. samp.info object containing information of the project's sample sheet.
samples	character. Name of the column in 'samp.info' containing sample names. If 'samp.info' is not supplied vector of sample names.
groups	character. Name of the column in 'samp.info' containing grouping information. If 'samp.info' is not supplied vector of groups.
sample.plot.names	character. Optional name of a column with "nice" sample names for plotting.
de.plot.base.size	numeric. Overall text/point scale (‘ggplot2‘ base_size, default 16) passed to the M-A and volcano plots. Raise it so labels and legends stay readable on the large per-contrast PDF (opened at 15x15 inches).

Value

The input list 'out.l', extended with a contrasts element holding one annotated result data.frame per contrast and, if plots were requested, with the MAplots, volcanoPlots and heatmapPlots elements. Called also for its side effects of writing result tables, plot files and Venn section spreadsheets below 'out.dir'.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
d <- edgeR::calcNormFactors(edgeR::DGEList(counts, group = groups))
design <- diff_expr_make_design(si, groups)
contrasts <- diff_expr_make_contrasts(design, groups)
fit <- diff_expr_fit(counts, d, design, do.voom = FALSE, quasi.likelihood = TRUE)
```

```
normcnt <- edgeR::cpm(fit$d2, log = TRUE)
out <- diff_expr_extract_contrasts(contrasts = contrasts, fit = fit$fit,
                                   normcnt = normcnt, out.l = list(),
                                   do.voom = FALSE, out.dir = tempdir(),
                                   analysis.name = "demo", plots = FALSE,
                                   samp.info = si, samples = "SampleNames",
                                   groups = groups)
names(out$contrasts)
```

`diff_expr_filter_counts` *Function to filter counts*

Description

Function to filter counts

Usage

```
diff_expr_filter_counts(counts, samp.info, strict = TRUE, min.samp = NULL)
```

Arguments

`counts`	Count matrix.
`samp.info`	`data.frame`. samp.info object containing information of the project's sample sheet.
`strict`	Logical; only keep a miRNA if there are > 5 reads per million in at least half of the samples?
`min.samp`	Integer; Number of samples in which a feature needs to be covered by at least one read per million. Defaults to the size of the smallest group of replicates. See *details*.

Details

In edgeR, it is recommended to remove features without at least 1 read per million in n of the samples, where n is the size of the smallest group of replicates (determined from the 'groups' vector).

Value

A `matrix` of counts with weakly expressed features and the non-informative `htseq-count` summary rows removed.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_read_counts(diffwrap_counts, si)
filtered <- diff_expr_filter_counts(counts, si, strict = TRUE)
c(before = nrow(counts), after = nrow(filtered))
```

`diff_expr_fit` *Function to compute linear model fit and optionally apply 'voom' beforehand*

Description

Function to compute linear model fit and optionally apply 'voom' beforehand

Usage

```
diff_expr_fit(
  counts,
  d,
  design,
  do.voom = TRUE,
  voom.fun = edgeR::voomLmFit,
  norm.method = c("quantile", "tmm"),
  use_weights = FALSE,
  quasi.likelihood = TRUE,
  bayes.trend = FALSE,
  bayes.robust = FALSE,
  pairs = NULL,
  pairs_col = NULL,
  block = FALSE,
  contrasts = NULL,
  disp = "tagwise.dispersion"
)
```

Arguments

`counts`	Count matrix.
`d`	Passed to `voom.fun`: a numeric matrix containing raw counts, or an ExpressionSet containing raw counts, or a DGEList object.
`design`	Numeric design matrix.
`do.voom`	Logical; should data be "voomed"?
`voom.fun`	Voom function: one of "voom", "voomWithQualityWeights" or voomLmFit.
`norm.method`	Character; one of "quantile", "tmm".
`use_weights`	`logical`. Should sample-specific quality weights be estimated?

quasi.likelihood
: Logical; should quasi-likelihood methods be used? See *Details* section. If TRUE, the default, then the quasi-likelihood methods are used, if FALSE, then the likelihood ratio test is used. Please note the main wrapper function diffExpr() will set this parameter.

bayes.trend
: Logical passed to eBayes(): should an intensity-dependent trend be allowed for the prior variance? If FALSE then the prior variance is constant. Alternatively, trend can be a row-wise numeric vector, which will be used as the covariate for the prior variance.

bayes.robust
: Logical passed to eBayes(): should the estimation of df.prior and var.prior be robustified against outlier sample variances?

pairs
: Factor of identifiers specifying paired samples for paired or other block designs, or batch effects.

pairs_col
: character. Name of the column in 'samp.info' containing paired sample information.

block
: logical. Are the comparisons to be made within AND between subjects? See Details section.

contrasts
: Contrast matrix as generated by makeContrasts(). If NULL, the default, the design matrix is expected to describe simple pair-wise comparisons, i.e., with the contrasts already included in the matrix. Consequently, in that case, no contrasts will be fitted.

disp
: Character; one of "tagwise.dispersion", "trended.dispersion", "bin.dispersion"

Details

For experimental designs involving comparisons within as well as between subjects inter-subject needs to be computed. In this case, the column specified in the 'pairs' argument must assign the subjects to the treatment/tissue/etc groups. For example, if we have two treatments the effects of which are to be observed in each two tissues, this design would apply. The 'pairs' factor is passed to the functions duplicateCorrelation() and lmFit(). Quasi-likelihood pipeline: While the likelihood ratio test is a more obvious choice for inferences with GLMs, the QL F-test is preferred as it reflects the uncertainty in estimating the dispersion for each gene. It provides more robust and reliable error rate control when the number of replicates is small. The QL dispersion estimation and hypothesis testing is done by using the functions glmQLFit() and glmQLFTest().

Value

A named list. If do.voom=TRUE the elements are v (the EList of voom-transformed counts), fit (the MArrayLM linear model fit) and fit2 (the MArrayLM fit after eBayes(), with contrasts applied if these were supplied). Otherwise they are d (the input DGEList), d2 (the DGEList with estimated dispersions) and fit (the DGEGLM generalised linear model fit).

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
d <- edgeR::calcNormFactors(edgeR::DGEList(counts, group = groups))
```

```
design <- diff_expr_make_design(si, groups)
fit <- diff_expr_fit(counts, d, design, do.voom = FALSE, quasi.likelihood = TRUE)
class(fit$fit)
```

`diff_expr_get_samp_info` *Function to standardize samp.info sample information data frame*

Description

Function to standardize samp.info sample information data frame

Usage

```
diff_expr_get_samp_info(
  samp.info,
  samples,
  groups,
  ellipse.mapping.groups = NULL
)
```

Arguments

`samp.info`	`data.frame`. samp.info object containing information of the project's sample sheet.
`samples`	`character`. Name of the column in 'samp.info' containing sample names. If 'samp.info' is not supplied vector of sample names.
`groups`	`character`. Name of the column in 'samp.info' containing grouping information. If 'samp.info' is not supplied vector of groups.
`ellipse.mapping.groups`	Optional additional grouping for ellipse drawing. Overrides sample groups. Use for selective highlighting of user-defined sample groups.

Value

A `data.frame` holding the sample sheet standardised to the conventions of this package: the sample and grouping columns renamed to `SampleNames` and `Groups` (and, where supplied, the ellipse grouping column to `Ellipse`), all coerced to factors with unused levels dropped, and the rows ordered by sample name.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info,
                              samples = "SampleName", groups = "Group")
head(si)
```

`diff_expr_ggplot_mds` *Function to generate a MDS plot using 'ggplot2'*

Description

Function to generate a MDS plot using 'ggplot2'

Usage

```
diff_expr_ggplot_mds(
  counts,
  samp.name,
  groups,
  grp.nam = NULL,
  pairs = NULL,
  pairs.name = NULL,
  gene.selection = "common",
  dim.plot = c(1, 2),
  main = NULL
)
```

Arguments

`counts`	Counts matrix.
`samp.name`	Optional sample names to be used in the plot, given as character vector.
`groups`	Sample groups for plot annotation as character vector or factor.
`grp.nam`	Legend title.
`pairs`	Factor of identifiers specifying paired samples for paired or other block designs, or batch effects.
`pairs.name`	Legend title for paired or block design variables in MDS ggplot.
`gene.selection`	Character passed to `plotMDS()` specifying the mode to select genes for comparisons.
`dim.plot`	Integer vector of length two passed to `plotMDS()` specifying the principal components to be plotted.
`main`	Plot title.

Value

A `ggplot` object containing the multidimensional scaling plot.

See Also

[plotMDS()]

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
lcpm <- edgeR::cpm(counts, log = TRUE)
g <- diff_expr_ggplot_mds(lcpm, samp.name = colnames(lcpm), groups = groups)
class(g)
```

`diff_expr_make_contrasts`

Function to make contrast matrix

Description

Function to make contrast matrix

Usage

```
diff_expr_make_contrasts(
  design,
  groups,
  pairs = NULL,
  block = FALSE,
  contrasts = NULL
)
```

Arguments

`design`	Numeric design matrix.
`groups`	`character`. Vector of group names.
`pairs`	`character`. Name of the column in 'samp.info' containing paired sample information.
`block`	`logical`. Are the samples not independent? See Details section.
`contrasts`	Character vector specifying group name pairs to be compared in the format expected by `makeContrasts()`, i.e., "group2-group1".

Details

The 'block' argument is used to specify whether the comparisons are to be made within AND between subjects or in the case of technical replicates, i.e., if the samples are not independent, in other words, correlated. @seealso [makeContrasts()]

Value

A contrast `matrix` as produced by `makeContrasts`, or `NULL` when no contrast matrix is needed because the comparisons are already inherent to the design matrix.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
groups <- stats::relevel(si$Groups, ref = "control")
design <- diff_expr_make_design(si, groups)
diff_expr_make_contrasts(design, groups)
```

`diff_expr_make_design` *Function to create design matrix*

Description

Function to create design matrix

Usage

```
diff_expr_make_design(
  samp.info,
  groups,
  pairs = NULL,
  block = FALSE,
  use_weights = FALSE
)
```

Arguments

`samp.info`	`data.frame`. samp.info object containing information of the project's sample sheet. vector of sample names.
`groups`	`character`. Name of the column in 'samp.info' containing grouping information. If 'samp.info' is not supplied vector of groups.
`pairs`	`character`. Name of the column in 'samp.info' containing paired sample information.
`block`	`logical`. Are the samples not independent? See Details section.
`use_weights`	`logical`. Are sample-specific quality weights used? See Details section. (Placeholder for future versions)

Details

The 'block' argument is used to specify whether the comparisons are to be made within AND between subjects or in the case of technical replicates, i.e., if the samples are not independent, in other words, correlated. If sample-specific quality weights are to be estimated by means of 'voomWithQualityWeights()' or 'voomLmFit()' and 'sample.weights' set to TRUE, 'use_weights' will be TRUE, enforcing a design matrix containing an 'intercept' column, i.e., where the columns reflect contrasts. The choice of the design matrix type impacts the estimated weights due to the effect of the intercept on the residual variance per sample in more complex designs, e.g., involving blocking factors, interactions or continuous covariates. The recommendation by the limma authors is to use the default design matrix, i.e., with intercept. With simple designs, the type of design matrix is not relevant. NOTE: This argument is not yet functional but a mere place-holder for future versions allowing for readily implemented more complex designs.

Value

A design matrix as produced by model.matrix: a means model without intercept (~0 + groups) for unpaired or blocked designs, or an additive model with intercept (~pairs + groups) when 'pairs' enters the model as a fixed effect.

See Also

[model.matrix()]

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
groups <- stats::relevel(si$Groups, ref = "control")
diff_expr_make_design(si, groups)
```

---

diff_expr_ma_plot *Function to generate a M-A plot using 'ggplot2'*

Description

Function to generate a M-A plot using 'ggplot2'

Usage

```
diff_expr_ma_plot(
  dat,
  contr,
  id = NULL,
  sym.col = "gene_symbol",
  p.thr = 0.05,
  fdr.thr = 0.05,
  logfc.thr = 1,
  numlab = 25,
  out.dir,
  analysis.name = NULL,
  point.lab = TRUE,
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description"),
  font.size = 5,
  lists = TRUE,
  base.size = 16
)
```

Arguments

dat	data frame; annotated output table generated by performing pair-wise comparisons and corresponding statistical tests.

contr	Character; contrast to be processed. This is either a column name of a contrast matrix or a character conforming to the same format.
id	character. Name of the column with the IDs used for labeling. Used as fallback for sym.col.
sym.col	character. Name of the column with the IDs used for labeling. Takes priority over id.
p.thr	Numeric; P-Value threshold for filtering gene labels. Only genes (points) with a value below this threshold will be labeled.
fdr.thr	Numeric; FDR threshold for filtering gene labels (see p.thr).
logfc.thr	Numeric; FC threshold on the log2-scale for filtering gene labels.
numlab	numeric. Maximum number of labels per plot. Overrides numbers calculated based on 'p.thr' and 'fdr.thr'.
out.dir	Character; output directory for final tables. Required; no default is used so that nothing is written to the working directory unintentionally.
analysis.name	Character used in the output file name.
point.lab	logical. Should points be labelled, at all?
biom.attributes	character. Vector of column names to be retrieved from biomart.
font.size	Size of point labels in M-A plots.
lists	Logical; should output tables be written to files?
base.size	numeric. Overall text scale ('ggplot2' base_size, default 16) applied via theme_gray; controls legend, axis and title text. Independent of font.size, which sets the point-label size. Raise it for readability on a large device (e.g. the 15x15 inch pipeline PDF).

Value

A named list of ggplot objects with the elements FDR and Pval, holding the M-A plot with points highlighted by false discovery rate and by p-value, respectively.

Examples

```
set.seed(1)
d3 <- data.frame(gene_symbol = paste0("G", 1:100),
                 logFC = rnorm(100), PValue = runif(100), FDR = runif(100),
                 AveExpr = rnorm(100, 5))
rownames(d3) <- d3$gene_symbol
g <- diff_expr_ma_plot(d3, contr = "treated-control", sym.col = "gene_symbol",
                       out.dir = tempdir(), lists = FALSE)
names(g)
```

`diff_expr_mds_plot` *Wrapper around 'limma::plotMDS' to generate a MDS plot*

Description

Wrapper around 'limma::plotMDS' to generate a MDS plot

Usage

```
diff_expr_mds_plot(
  d,
  groups,
  n = 500,
  sample.plot.names = NULL,
  analysis.name = NULL,
  do.pdf = FALSE,
  out.dir
)
```

Arguments

`d`	Passed to `plotMDS()` (`x`): Any data object that can be coerced to a matrix of log-expression values, for example an ExpressionSet or an EList. Rows represent genes or genomic features while columns represent samples.
`groups`	Factor of sample groups for colouring and legend.
`n`	Passed to `plotMDS()` (`top`): Integer; number of top genes used to calculate pairwise distances.
`sample.plot.names`	Passed to `plotMDS()` (`labels`): character vector of sample names or labels. Defaults to colnames(d).
`analysis.name`	Character used in the plot title and the output file name if the plot is saved to a PDF
`do.pdf`	Logical indicating whether a PDF should be produced.
`out.dir`	Character; path where to save PDF. Required; no default is used so that nothing is written to the working directory unintentionally.

Value

No return value. Called for its side effect of drawing a multidimensional scaling plot, optionally into a PDF file below '`out.dir`'.

See Also

[plotMDS()]

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
d <- edgeR::calcNormFactors(edgeR::DGEList(counts, group = groups))
diff_expr_mds_plot(d, groups = groups, do.pdf = FALSE, out.dir = tempdir())
```

`diff_expr_PCA` *Function to do PCA using 'stats::prcomp'*

Description

Function to do PCA using 'stats::prcomp'

Usage

```
diff_expr_PCA(counts, n = 500, scale. = FALSE)
```

Arguments

`counts`	Counts matrix.
`n`	Number of rows to be selected from the sorted variance matrix (by default, the top 500 rows are selected from the matrix sorted in decreasing order).
`scale.`	A logical value passed to `prcomp`; should the variables be scaled to have unit variance?

Value

An object of class `prcomp` as returned by `prcomp`, holding the principal component decomposition of the (optionally variance-filtered) count matrix.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
pca <- diff_expr_PCA(edgeR::cpm(counts, log = TRUE), n = 100)
pca$sdev[1:3]
```

`diff_expr_PCA_ggbiplot` *Function to generate a PCA biplot using 'ggbiplot.n', a version of 'ggbiplot' from https://github.com/vqv/ggbiplot.*

Description

Function to generate a PCA biplot using 'ggbiplot.n', a version of 'ggbiplot' from https://github.com/vqv/ggbiplot.

Usage

```
diff_expr_PCA_ggbiplot(
  PCA,
  groups,
  grp.nam = NULL,
  ellipse = TRUE,
  circle = TRUE,
  varname.size = 0,
  var.axes = FALSE,
  main = NULL,
  fix.aspect = FALSE,
  tweak = FALSE,
  ...
)
```

Arguments

`PCA`	List of class `prcomp` or just a list having a component `x` that contains the rotated variables from `prcomp`.
`groups`	Sample groups for plot annotation as character vector or factor.
`grp.nam`	Legend title.
`ellipse`	Logical indicating whether to draw an ellipse around the sample groups.
`circle`	`logical`. Draw a correlation circle? (only applies when prcomp was called with scale = TRUE and when var.scale = 1)
`varname.size`	`double`. size of the text for variable names
`var.axes`	`logical`. draw arrows for the variables?
`main`	Plot title.
`fix.aspect`	`logical`. Should the aspect ratio of the x- and y-axes be kept constant for different plot sizes?
`tweak`	`logical`. Should the plot theme be tweaked? Will apply values set in axes.title.size, legend.text.size and legend.title.size. Defaults to TRUE.
`...`	Arguments passed to `ggbiplot.n()`.

Value

A ggplot object containing the PCA biplot. The plot is returned rather than drawn, so it has to be printed to appear on a device.

See Also

[ggbiplot.n()]

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
pca <- diff_expr_PCA(edgeR::cpm(counts, log = TRUE), n = 100)
g <- diff_expr_PCA_ggbiplot(pca, groups = groups)
class(g)
```

---

diff_expr_PCA_ggplot *Function to generate an ordinary two-dimensional PCA plot using 'ggplot2'*

Description

Function to generate an ordinary two-dimensional PCA plot using 'ggplot2'

Usage

```
diff_expr_PCA_ggplot(
  PCA,
  samp.name = NULL,
  groups,
  grp.nam = NULL,
  PC = c(1, 2),
  main = NULL,
  ellipse = TRUE,
  ellipse.mapping.groups = NULL,
  ellipse.grp.nam = NULL,
  label.samples = TRUE,
  geom.point.size = 2,
  label.font.size = 5,
  plot.ellipse.legend = NA,
  do.plot = TRUE
)
```

Arguments

PCA	List of class prcomp or just a list having a component x that contains the rotated variables from prcomp.
samp.name	Optional sample names to be used in the plot, given as character vector.
groups	Sample groups for plot annotation as character vector or factor.
grp.nam	Legend title.
PC	Integer vector of length two specifying the principal components to be plotted.
main	Plot title.
ellipse	Logical indicating whether to draw an ellipse around the sample groups.
ellipse.mapping.groups	Optional additional grouping for ellipse drawing. Overrides sample groups. Use for selective highlighting of user-defined sample groups.
ellipse.grp.nam	Not implemented.
label.samples	Logical; should the points be annotated with sample name labels?
geom.point.size	Numeric passed to geom_point giving the point size.
label.font.size	Numeric passed to geom_text_repel giving the label font size.
plot.ellipse.legend	Logical; should the ellipse legend be plotted. NA, the default, will plot it if any aesthetics are mapped.
do.plot	Logical; should the plot be printed to the graphics device? Defaults to TRUE.

Value

A ggplot object containing the labelled PCA scatterplot.

See Also

[stat_ellipse()]

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
pca <- diff_expr_PCA(edgeR::cpm(counts, log = TRUE), n = 100)
g <- diff_expr_PCA_ggplot(pca, samp.name = NULL, groups = groups, do.plot = FALSE)
class(g)
```

`diff_expr_pseudo_counts` *Function to calculate pseudo counts representing batch-corrected normalised but untransformed values*

Description

Function to calculate pseudo counts representing batch-corrected normalised but untransformed values

Usage

```
diff_expr_pseudo_counts(
  d,
  design,
  pairs = "pairs",
  disp = "tagwise.dispersion",
  do.cpm = TRUE
)
```

Arguments

`d`	Passed to `edgeR` functions: matrix of counts or a DGEList object.
`design`	numeric design matrix
`pairs`	`character`. Name of column with identifiers specifying paired samples for paired or other block designs, or batch effects. Defaults to "pairs" as this is the prefix added by 'diff_expr_make_design()'. When used outside the package's scope the user must supply the correct prefix.
`disp`	`character`. one of "tagwise.dispersion", "trended.dispersion", "bin.dispersion"
`do.cpm`	`logical`. should the pseudo counts be transformed to CPMs?

Value

A `matrix` of pseudo counts in which the effect of the blocking variable has been removed; on the log2 counts per million scale if `do.cpm=TRUE` and on the count scale otherwise.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
si$Subject <- diffwrap_samp_info$Subject[order(diffwrap_samp_info$SampleName)]
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
d <- edgeR::calcNormFactors(edgeR::DGEList(counts, group = groups))
design <- diff_expr_make_design(si, groups, pairs = "Subject")
pc <- diff_expr_pseudo_counts(d = d, design = design, pairs = "pairs")
dim(pc)
```

diff_expr_pval_hist_plot *Function to generate a histogram of the P-Value distribution*

Description

Function to generate a histogram of the P-Value distribution

Usage

```
diff_expr_pval_hist_plot(d3)
```

Arguments

d3	data.frame. Data frame containing a P-Value column to generate a histogram. The column name will be determined by pattern matching.

Value

Invisibly, an object of class histogram as returned by hist. Called for its side effect of drawing the p-value distribution.

Examples

```
d3 <- data.frame(ID = paste0("g", 1:200), PValue = runif(200))
diff_expr_pval_hist_plot(d3)
```

diff_expr_QC_plots *Main wrapper function for QC plots*

Description

Main wrapper function for QC plots

Usage

```
diff_expr_QC_plots(
  counts,
  samp.info,
  control,
  out.l,
  grp.nam = NULL,
  PC = c(1, 2, 3),
  sample.plot.names = NULL,
  ellipse = TRUE,
  ellipse.mapping.groups = NULL,
```

```
  ellipse.grp.nam = NULL,
  label.samples = TRUE,
  geom.point.size = 2,
  label.font.size = 5,
  plot.ellipse.legend = NA,
  circle = TRUE,
  varname.size = 0,
  var.axes = FALSE,
  pairs = NULL,
  pairs.name = NULL,
  gene.selection = "common",
  n = 500,
  type = NULL,
  analysis.name = NULL,
  out.dir
)
```

Arguments

counts	Counts matrix.
samp.info	data.frame. samp.info object containing information of the project's sample sheet.
control	character. Name of the control group
out.l	list. Object returned by diffExpr() containing all output components from the analysis.
grp.nam	Legend title.
PC	Integer vector of length two or three specifying the principal components to be plotted.
sample.plot.names	Passed to plotMDS() (labels): character vector of sample names or labels. Defaults to colnames(d).
ellipse	Logical indicating whether to draw an ellipse around the sample groups.
ellipse.mapping.groups	Optional additional grouping for ellipse drawing. Overrides sample groups. Use for selective highlighting of user-defined sample groups.
ellipse.grp.nam	Not implemented.
label.samples	Logical; should the points be annotated with sample name labels?
geom.point.size	Numeric passed to geom_point giving the point size.
label.font.size	Numeric passed to geom_text_repel giving the label font size.
plot.ellipse.legend	Logical; should the ellipse legend be plotted. NA, the default, will plot it if any aesthetics are mapped.

circle	logical. Draw a correlation circle? (only applies when prcomp was called with scale = TRUE and when var.scale = 1)
varname.size	double. size of the text for variable names
var.axes	logical. draw arrows for the variables?
pairs	Factor of identifiers specifying paired samples for paired or other block designs, or batch effects.
pairs.name	Legend title for paired or block design variables in MDS ggplot.
gene.selection	Character passed to plotMDS() specifying the mode to select genes for comparisons.
n	Passed to plotMDS() (top): Integer; number of top genes used to calculate pairwise distances.
type	Character; one of "both", "uncorrected", "pseudo-corrected" describing which values should be plotted. "uncorrected" will plot the input counts matrix while "pseudo-corrected" will plot pseudo counts for blocked designs (e.g., paired samples or batch factors).
analysis.name	Character used in the plot title and the output file name if the plot is saved to a PDF
out.dir	Character; path where to save PDF. Required; no default is used so that nothing is written to the working directory unintentionally.

Value

The input list 'out.l', with a QCplots element added (or extended) holding the generated quality control plots as ggplot objects, named after the plot type and the value of 'type'.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
lcpm <- edgeR::cpm(counts, log = TRUE)
out.l <- diff_expr_QC_plots(counts = lcpm, samp.info = si, control = "control",
                            out.l = list(), grp.nam = "Group",
                            sample.plot.names = colnames(lcpm),
                            analysis.name = "demo", out.dir = tempdir())
names(out.l$QCplots)
```

---

diff_expr_read_counts *Function to read counts as produced by htseq-count*

Description

Function to read counts as produced by htseq-count

Usage

```
diff_expr_read_counts(expr.dat, samp.info, miRSEQ = FALSE)
```

Arguments

expr.dat	character or list. String or vector or list of input file paths, or matrix of count values. Allowed are individual text files with counts for one sample each, with gene IDs in the first and counts in the second column or a single counts matrix file containing read counts for all samples with rows corresponding to genes (genomic features) and columns to samples. Negative values or NAs are not allowed and gene IDs are expected in the first column. If miRSEQ=TRUE this expects the output from CAP-miRSEQ summary script which is also a counts matrix.
samp.info	data.frame. samp.info object containing information of the project's sample sheet.
miRSEQ	logical. Is the input data the output from CAP-miRSEQ summary script?

Value

A matrix of raw counts with features in rows and samples in columns, restricted to the samples listed in 'samp.info' and ordered as they are there.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_read_counts(diffwrap_counts, si)
dim(counts)
```

---

diff_expr_volcano_plot

Function to generate a Volcano plot using 'ggplot2'

Description

'diff_expr_volcano_plot' generates two Volcano plots highlighting genes that are differentially expressed beyond custom thresholds for significance (set by parameters 'p.thr' and fdr.thr') and differential expression level (set by parameter 'logfc.thr').

Usage

```
diff_expr_volcano_plot(
  d3,
  id,
  sym.col = "gene_symbol",
  main = NULL,
  p.thr = 0.05,
  fdr.thr = 0.05,
```

```
  logfc.thr = 1,
  numlab = 25,
  point.lab = TRUE,
  base.size = 16
)
```

Arguments

d3	data.frame. Data frame containing all necessary columns to generate a Volcano plot with gene labels (at least p-values, FDR values, log-ratios and gene symbols or other IDs)
id	character. Name of the gene ID column. Can be the same as 'sym.col' but usually refers to an additional column with, e.g., Ensembl Gene IDs.
sym.col	character. Name of column with gene symbols, e.g., HGNC Symbols.
main	character. Main plot title. (Will be complemented with additional information, e.g., 'FDR' when labelling according to and FDR threshold.)
p.thr	numeric. Plotted values with a P-Value below this threshold will be labelled in the P-Value plot.
fdr.thr	numeric. Plotted values with a FDR below this threshold will be labelled in the FDR plot.
logfc.thr	numeric. Plotted ('abs'olute) values above this threshold will have bigger dots.
numlab	numeric. Maximum number of labels per plot. Takes precedence to numbers calculated based on 'p.thr' and 'fdr.thr'.
point.lab	logical. Should points be labelled, at all?
base.size	numeric. Overall text/point scale passed on to prepare_volcano_of_given_property (the 'ggplot2' base_size, default 16). Raise it to enlarge legend, axes, labels and points together when printing to a large device (e.g. the 15x15 inch pipeline PDF).

Value

A named list of ggplot objects with the elements FDR and Pval, holding the volcano plot with points highlighted by false discovery rate and by p-value, respectively. Both plots are additionally printed, so that they are captured when the function is called with an open plotting device.

Examples

```
set.seed(1)
d3 <- data.frame(gene_symbol = paste0("G", 1:100),
                 logFC = rnorm(100), PValue = runif(100), FDR = runif(100))
g <- diff_expr_volcano_plot(d3, id = "gene_symbol", sym.col = "gene_symbol")
names(g)
```

`format_ensembl_ids_annotated_to_term` *Helper function for formatting the gene ID column of enrichment data frame.*

Description

Helper function for formatting the gene ID column of enrichment data frame.

Usage

```
format_ensembl_ids_annotated_to_term(result, species, which.split = ",")
```

Arguments

`result`	Data.frame; a data frame with Ensembl Gene IDs in one (only one) column
`species`	Character of length one; name of the species the IDs refer to. Only "human" and "mouse" are supported.
`which.split`	Character; separator used in the ID column of `result`. See *details*.

Details

Enrichment tools report all genes annotated to a certain term in one string, separated by comma or similar. The function splits each row into individual IDs before converting and later re-collapses converted IDs.

Value

The input enrichment result with the Ensembl gene identifiers in the gene column replaced by the corresponding gene symbols.

`get_hm_breaks` *Function to define breaks to be used for changing the palette of the heatmap.*

Description

Function to define breaks to be used for changing the palette of the heatmap.

Usage

```
get_hm_breaks(
  expr.mat,
  scale.fl = "row",
  palette.length = 100,
  quantile.breaks.fl = FALSE,
  n = 11
)
```

Arguments

expr.mat	differential gene expression matrix or data frame in (genes, samples) format
scale.fl	character indicating if values should be centred and scaled in either the row direction or the column direction, or none (values ("row","column","none"), default = none)
palette.length	integer setting the desired length of the colour palette to be used in the heatmap
quantile.breaks.fl	boolean values determining if quantile breaks are used to change the colors of the heatmap, otherwise min-max breaks are used by default
n	desired length of the numeric vector of probabilities (see ?quantile); defaults to the number of different colours in the palette

Value

a numeric vector containing the breaks to be used in the heatmap (see 'breaks' in [pheatmap()]).

See Also

[quantile()]

---

get_hm_colors *Function to compute colour palettes to be used in the heatmap.*

Description

Function to compute colour palettes to be used in the heatmap.

Usage

```
get_hm_colors(
  palette.length = 100,
  color.extremes = c("#3182BD", "#E6550D"),
  breaks = NA,
  color.blind.pal = NULL,
  n.pal.cols = 11
)
```

Arguments

palette.length	integer setting the desired length of the colour palette to be used in the heatmap
color.extremes	character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.
breaks	numeric vector of breaks to be used for compute the colour palette; defaults to NA which means no breaks are used and later computed by 'pheatmap()'.

color.blind.pal
: string determining the RColorBrewer color blind palette (default = "PuOr"); other option can be visualized with the following command: 'brewer.pal.info[brewer.pal.info$colorblind,]

n.pal.cols
: integer giving the number of different colours in 'color.blind.pal'.

Value

a character vector containing the colour codes to be used in the heatmap.

See Also

[colorRampPalette()]

---

ggbiplot.n *Make a biplot of PCA output data using ggplot2.*

Description

Make a biplot of PCA output data using ggplot2.

Usage

```
ggbiplot.n(
  pcobj,
  choices = 1:2,
  scale = 1,
  pc.biplot = TRUE,
  obs.scale = 1 - scale,
  var.scale = scale,
  groups = NULL,
  grp.nam = NULL,
  ellipse = FALSE,
  ellipse.prob = 0.68,
  labels = NULL,
  labels.size = 3,
  alpha = 1,
  var.axes = TRUE,
  circle = FALSE,
  circle.prob = 0.69,
  varname.size = 3,
  varname.adjust = 1.5,
  varname.abbrev = FALSE,
  point.size = 1,
  axes.title.size = 1,
  legend.text.size = 1,
  legend.title.size = 1,
  ellipse.lwd = 1,
```

```
  main = NULL,
  fix.aspect = TRUE,
  tweak = TRUE,
  tidy = TRUE,
  ...
)
```

Arguments

pcobj	an object returned by prcomp() or princomp()
choices	length-two numeric. which PCs to plot; default: 1:2
scale	length-one numeric. covariance biplot (scale = 1) [default], form biplot (scale = 0). When scale = 1, the inner product between the variables approximates the covariance and the distance between the points approximates the Mahalanobis distance.
pc.biplot	logical. for compatibility with biplot.princomp()
obs.scale	length-one numeric. scale factor to apply to observations; default: 1-scale
var.scale	length-one numeric. scale factor to apply to variables; default: scale
groups	factor. optional factor variable indicating the groups that the observations belong to. If provided the points will be colored according to groups
grp.nam	character. optional name of the grouping variable to be used as legend title
ellipse	logical. draw a normal data ellipse for each group?
ellipse.prob	double. size of the ellipse in Normal probability
labels	character. optional vector of labels for the observations
labels.size	double. size of the text used for the labels
alpha	double. alpha transparency value for the points (0 = transparent, 1 = opaque)
var.axes	logical. draw arrows for the variables?
circle	logical. Draw a correlation circle? (only applies when prcomp was called with scale = TRUE and when var.scale = 1)
circle.prob	double. size of the circle in Normal probability
varname.size	double. size of the text for variable names
varname.adjust	double. adjustment factor the placement of the variable names, >= 1 means farther from the arrow
varname.abbrev	logical. whether or not to abbreviate the variable names
point.size	double. expansion factor for point size; uses rel() internally
axes.title.size	double. expansion factor for axes title sizes; uses rel() internally
legend.text.size	double. expansion factor for legend text size; uses rel() internally
legend.title.size	double. expansion factor for legend title size; uses rel() internally
ellipse.lwd	double. expansion factor for ellipse line width; uses rel() internally

main	character. Plot title. If NULL (default) no title will be added.
fix.aspect	logical. Should the aspect ratio of the x- and y-axes be kept constant for different plot sizes?
tweak	logical. Should the plot theme be tweaked? Will apply values set in axes.title.size, legend.text.size and legend.title.size. Defaults to TRUE.
tidy	logical. If TRUE, theme_minimal will be applied.
...	currently not in use

Value

The final plot object returned by ggplot.

Examples

```
si <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
pca <- diff_expr_PCA(edgeR::cpm(counts, log = TRUE), n = 100)
g <- ggbiplot.n(pca, groups = groups)
class(g)
```

---

make_pheatmap_anno_color

Function to create the annotation colour list used in the heatmap

---

Description

Function to create the annotation colour list used in the heatmap

Usage

```
make_pheatmap_anno_color(clinical.mat)
```

Arguments

clinical.mat	matrix with clinical annotation values in (clinical category, samples) format

Value

A named list of named character vectors of colours, one element per annotation variable, in the form expected by the annotation_colors argument of pheatmap.

pheatmap_plots *Function to generate heatmap of gene expression values*

Description

Function to generate heatmap of gene expression values

Usage

```
pheatmap_plots(
  d3,
  id,
  sym.col = "gene_symbol",
  samp.info,
  samples,
  groups,
  sample.plot.names = NULL,
  main = NULL,
  add.main = NULL,
  color.blind.pal = "PuOr",
  n.pal.cols = 11,
  color.extremes = c("#3182BD", "#E6550D"),
  palette.length = NULL,
  anno.color = NULL,
  anno.name = "Sample Class",
  p.thr = 0.05,
  fdr.thr = 0.05,
  logfc.thr = 1,
  topn = 100,
  split.expr = FALSE
)
```

Arguments

d3	data.frame. Data frame containing all necessary columns to generate a heatmap with gene labels (at least p-values, FDR values, log-ratios and gene symbols or other IDs)
id	character. Name of the gene ID column. Can be the same as 'sym.col' but usually refers to an additional column with, e.g., Ensembl Gene IDs.
sym.col	character. Name of column with gene symbols, e.g., HGNC Symbols.
samp.info	data.frame. samp.info object containing information of the project's sample sheet
samples	character. Name of the column in 'samp.info' containing sample names. If 'samp.info' is not supplied vector of sample names.
groups	Factor of sample groups for colouring and legend.

`sample.plot.names`
: `character`. Optional name of a column with "nice" sample names for plotting. Need to be in the same order as sample column names!

`main`
: `character`. Main plot title. (Will be complemented with additional information, e.g., 'FDR' when labelling according to and FDR threshold.)

`add.main`
: `character`. Additional information printed in the heatmap title.

`color.blind.pal`
: string determining the color-blind-friendly palette; must be one of RColorBrewer's diverging palettes (default = "PuOr"); other option can be visualized with the following command: brewer.pal.info[brewer.pal.info$colorblind,]

`n.pal.cols`
: desired length of the number of different colours in 'color.blind.pal'. Will also be used as length of the numeric vector of probabilities in 'quantile_breaks()' (see ?quantile); must be within 3:11; defaults to 11

`color.extremes`
: character vector of length 2 giving the two extremes of a user-defined colour palette varying from the first hue to the second via white.

`palette.length`
: integer setting the desired length of the colour palette to be used in the heatmap

`anno.color`
: list of named character vectors giving the colours used in the heatmap annotation bars. See 'annotation_colors' in [pheatmap()]. Automatically generated if NULL (default).

`anno.name`
: character string used as the column annotation legend title. If 'anno.color' is not NULL and of length 1 the slot name will be used if existing.

`p.thr`
: `numeric`. Plotted values with a P-Value below this threshold will be labelled in the P-Value plot.

`fdr.thr`
: `numeric`. Plotted values with a FDR below this threshold will be labelled in the FDR plot.

`logfc.thr`
: `numeric`. FC threshold on the log2-scale. Only used if 'split.expr' is TRUE. Values above this threshold will be retained. Defaults to 1.

`topn`
: `numeric`. Number of top values to be plotted. Defaults to 100.

`split.expr`
: `logical`. Should the top up- and top down-regulated genes be displayed at equal numbers (50/50), if they meet the significance threshold (regardless of the actual significance)? Defaults to `FALSE`.

Details

The actual heatmap is produced as a side effect by printing the individual plot components.

This is the convenience entry point used by `diffExpr()` to build a contrast's heatmaps; it is exported so the same heatmaps can be regenerated from a stored result table (e.g. `res$contrasts[[<contrast>]]`) with different thresholds or colours, without re-running the pipeline. The lower-level work horse it calls is `diffr_pheatmap`.

Value

Returns a named `list` of heatmap objects: an element `fdr` (built from the FDR-filtered genes) and/or `pval` (used only when nothing passes the FDR threshold), each a list holding the `regular` heatmap and the gene/sample `correlogram` objects. Also called for its side effect of drawing those objects on the active graphics device.

Examples

```
si     <- diff_expr_get_samp_info(diffwrap_samp_info, "SampleName", "Group")
counts <- diff_expr_filter_counts(diff_expr_read_counts(diffwrap_counts, si), si)
groups <- stats::relevel(si$Groups, ref = "control")
d      <- edgeR::calcNormFactors(edgeR::DGEList(counts, group = groups))
design <- diff_expr_make_design(si, groups)
contr  <- diff_expr_make_contrasts(design, groups)
fit    <- diff_expr_fit(counts, d, design, do.voom = FALSE, quasi.likelihood = TRUE)
## a per-contrast table like the ones diffExpr() stores in res$contrasts
de <- edgeR::glmQLFTest(fit$fit, contrast = contr[, 1])
d3 <- merge(edgeR::cpm(fit$d2, log = TRUE),
            as.data.frame(edgeR::topTags(de, n = Inf)), by = "row.names")
names(d3)[1] <- "ID"; d3$gene_symbol <- d3$ID
## regenerate the heatmap at a stricter cut-off and a different palette, no refit
hm <- pheatmap_plots(d3, id = "ID", samp.info = si, samples = "SampleNames",
                     groups = groups, fdr.thr = 0.01, topn = 30, color.blind.pal = "RdBu")
```

---

plot_enrichment_network

Function for making network visualisation based on enrichment result, DE gene table and thresholding. Can take the input tables either as data frames or Excel files

---

Description

Function for making network visualisation based on enrichment result, DE gene table and thresholding. Can take the input tables either as data frames or Excel files

Usage

```
plot_enrichment_network(
  enrichment.result,
  DE.result,
  plot.filename,
  show.terms = 5,
  logfc.thr = NULL,
  fdr.thr = NULL,
  pdf.width = 11,
  pdf.height = 8,
  legend.cex.main = 0.8,
  legend.cex.text = 0.7
)
```

Arguments

`enrichment.result`
: data.frame; output of the enrichment tool. In general, a data frame with all columns needed for the plot.

DE.result — data.frame; fold-change table with corresponding statistics, e.g., the output of topTable() for a particular contrast.

plot.filename — character; name of the output image file. The file extension can be omitted and will be added internally.

show.terms — integer; Number of enrichment terms to be plotted, i.e., the first Integer rows of enrichment.result.

logfc.thr — numeric. Fold-change threshold on the log2-scale.

fdr.thr — numeric. Threshold for adjusted p-values (applied in both filtering of DE-genes and in enrichment results). Default 0.05

pdf.width, pdf.height — Numeric; width and height of the PDF graphics region in inches.

legend.cex.main, legend.cex.text — Numeric; text sizes of legend title and text, given as character expansion (magnification) relative to the default.

Value

No return value. Called for its side effect of writing a network visualisation of the enrichment result to the file given by 'plot.filename'.

Examples

```
# builds a network plot from an enrichment result and writes it to 'plot.filename'
if (requireNamespace("igraph", quietly = TRUE)) {
  enr <- data.frame(Description = c("Pathway A", "Pathway B"),
                    genes = c("G1,G2,G3", "G3,G4,G5"))
  de <- data.frame(gene_symbol = paste0("G", 1:5),
                   logFC = c(2.5, -1.8, 1.2, -2.1, 0.9),
                   FDR = c(0.001, 0.002, 0.01, 0.003, 0.02))
  plot_enrichment_network(enr, de,
                          plot.filename = file.path(tempdir(), "network.pdf"))
}
```

---

prepare_scale_for_legend

Helper function for enrichment visualisations: prepare plot legend y coordinates and labels

---

Description

Helper function for enrichment visualisations: prepare plot legend y coordinates and labels

Usage

```
prepare_scale_for_legend(
  scale.minimum,
  scale.maximum,
  int.values.for.ticks = NULL
)
```

Arguments

`scale.minimum`	Numeric; minimum y coordinates for text labels
`scale.maximum`	Numeric; maximum y coordinates for text labels
`int.values.for.ticks`	Not implemented.

Details

The coordinates are used in a network graph and minimum and maximum scale values correspond to minimum and maximum fold-changes, by default.

Value

A named `list` with the elements `scale.y.coordinates` (the vertical positions of the legend tick labels) and `scale.labels` (the corresponding label values).

Examples

```
prepare_scale_for_legend(scale.minimum = -3, scale.maximum = 4)
```

`prepare_volcano_of_given_property` *Helper function that returns a volcano plot.*

Description

Builds a single volcano plot for one significance measure. Points are coloured by an 'EnhancedVolcano'-style, colour-blind safe (Okabe-Ito) four-level scheme distinguishing genes that pass neither threshold ("not sign."), the fold-change threshold only, the significance threshold only, or both. Only the doubly-significant genes are eligible for text labels.

Usage

```
prepare_volcano_of_given_property(
  data.df,
  property.to.plot = c("fdr", "p"),
  property.column,
  property.thr = 0.05,
  logfc.thr = 1,
```

```
  main = NULL,
  numlab = 25,
  point.lab = TRUE,
  sym.col = "gene_symbol",
  pretty.breaks = FALSE,
  base.size = 16
)
```

Arguments

data.df	data.frame. Data frame containing all necessary columns to generate a Volcano plot with gene labels (at least p-values, FDR values, log-ratios and gene symbols or other IDs)
property.to.plot	character. Statistic to be plotted, one of "fdr" or "p". Matched via match.arg.
property.column	character. Name of the actual column holding the statistic to be plotted.
property.thr	numeric. Significance threshold; genes below it (in 'property.column') are considered significant. Defaults to 0.05.
logfc.thr	numeric. Absolute log2 fold-change threshold; genes beyond it, together with the significance threshold, form the highlighted ("both") set that is eligible for labelling. Defaults to 1.
main	character. Main plot title. Defaults to NULL.
numlab	numeric. Maximum total number of gene labels per plot (default 25). Labels are taken from the doubly-significant genes and split between up- and down-regulated in proportion to the skew, with a guaranteed minimum per side so the minor side is never left empty. The chosen split is logged.
point.lab	logical. Should points be labelled, at all?
sym.col	character. Name of column with gene symbols, e.g., HGNC Symbols.
pretty.breaks	logical. Should the breaks on the axes be pretty? Uses 'scales::breaks_extended'.
base.size	numeric. Overall text/point scale (the 'ggplot2' base_size, default 16). Directly used by legend, axis and title text; point-label, threshold-annotation and point sizes are derived from it. Useful when the plot is printed to a large device (e.g. the 15x15 inch pipeline PDF), where the default 'ggplot2' sizes are too small.

Value

A ggplot object containing the volcano plot for the requested significance measure.

Examples

```
set.seed(1)
d3 <- data.frame(gene_symbol = paste0("G", 1:100),
                 logFC = rnorm(100), PValue = runif(100))
g <- prepare_volcano_of_given_property(d3, property.to.plot = "p",
                                       property.column = "PValue",
```

```
                                        property.thr = 0.05, logfc.thr = 1)
class(g)
```

`quantile_breaks` *Function to define quantile breaks to be used for changing the palette of the heatmap.*

Description

Function to define quantile breaks to be used for changing the palette of the heatmap.

Usage

```
quantile_breaks(xs, n = 20)
```

Arguments

`xs`	numeric vector to calculate quantiles for
`n`	desired length of the numeric vector of probabilities (see ?quantile)

Value

A `numeric` vector of unique quantile break points, of length at most 'n'.

Note

Function by: Kamil Slowikowski, https://github.com/slowkow/slowkow.com/blob/master/_rmd/2017-02-16-heatmap-tutorial.R,

See Also

[quantile()]

`reorderFactors` *Function which reorders the levels of a column of a data frame specified as a factor*

Description

Function which reorders the levels of a column of a data frame specified as a factor

Usage

```
reorderFactors(df, column = "my_column_name", desired_level_order)
```

Arguments

df	data frame to be processed
column	name of the column to be reordered
desired_level_order	vector of factor levels in the desired order

Details

The desired order is a vector containing the levels of the factor in the desired order.

Value

The input data.frame with the levels of the specified factor column reordered.

Note

Written by https://stackoverflow.com/users/1701600/boern.

---

run.topGO *Function to run GO term enrichment analysis using the 'topGO' package.*

Description

Function to run GO term enrichment analysis using the 'topGO' package.

Usage

```
run.topGO(
  background,
  foreground,
  ontologies = c("BP"),
  organism,
  ID_type = "ENSEMBL",
  pAdjustMethod = "BH"
)
```

Arguments

background	path to the background set of genes
foreground	path to the foreground set of genes
ontologies	character string specifying the ontology of interest (BP,MF,CC), default = "BP"
organism	the organism database used, eg. for Human: "org.Hs.eg.db"
ID_type	character; the type of gene ID used in the input data, default = "ENSEMBL"
pAdjustMethod	character; the method used for p-value adjustment, default = "BH"

Details

The columns in the final table produced by topGO (their description is fetched from topGo documentation): - Annotated : number of genes in org.Hs.eg.db which are annotated with the GO-term. - Significant : number of genes belonging to your input which are annotated with the GO-term. - Expected : show an estimate of the number of genes a node of size Annotated would have if the significant genes were to be randomly selected from the gene universe. - pvalues : pvalue obtained after the test

Column 'p.adj.weight01' represents the adjusted p-value for the weight01 algorithm.

Value

A data frame with enriched terms and p-values from Fisher's exact test, using different algorithms: elim, classic, weight01.

Author(s)

Bogdan Iancu - Genevia Technologies Oy

Examples

```
# needs org.Hs.eg.db; builds a genome-wide GO universe, so it is slow
if (requireNamespace("topGO", quietly = TRUE) &&
    requireNamespace("org.Hs.eg.db", quietly = TRUE) &&
    requireNamespace("AnnotationDbi", quietly = TRUE)) {
  fg <- c("ENSG00000141510", "ENSG00000012048", "ENSG00000139618")
  # a small background keeps the example quick; a real analysis uses the full gene universe,
  # e.g. AnnotationDbi::keys(org.Hs.eg.db::org.Hs.eg.db, keytype = "ENSEMBL")
  bg <- unique(c(fg, utils::head(
    AnnotationDbi::keys(org.Hs.eg.db::org.Hs.eg.db, keytype = "ENSEMBL"), 2000)))
  go <- tryCatch(run.topGO(background = bg, foreground = fg, ontologies = "BP",
                           organism = "org.Hs.eg.db", ID_type = "ENSEMBL"),
                 error = function(e) {
                   message("topGO enrichment could not be run: ", conditionMessage(e))
                   NULL
                 })
}
```

---

runEnrichmentAnalyses *Wrapper for executing various enrichment analyses*

Description

runEnrichmentAnalyses enables the auto-run of some over-representation analysis (ORA) and gene set enrichment analysis (GSEA) functions for the output of diffExpr main wrapper. Functions in various R-packages (clusterProfiler, topGO, gProfileR) are integrated. Currently supports human or mouse!

Usage

```
runEnrichmentAnalyses(
  diffr.wrapper.output,
  analysis.name = "enrichment",
  use.background.from.diffr.output = TRUE,
  use.pval.in.DE.filtering.if.no.sign.fdrs = FALSE,
  out.dir,
  species = "human",
  p.thr = 0.05,
  fdr.thr = 0.05,
  logfc.thr = 1,
  do.plot = FALSE,
  plot.fdr.thr = fdr.thr,
  plot.logfc.thr = logfc.thr,
  plot.num.terms = 5,
 enrichment.methods = c("clusterProfilerGO", "clusterProfilerKEGG", "gProfileR",
    "topGO"),
 clusterProfilerGO.params = list(analysis.approach = "ORA", do.similarity.filtering =
  FALSE, min.gene.set.size = 10, max.gene.set.size = 1000, ontology = "BP", min.overlap
    = 2, p.adjust.method = "BH"),
 clusterProfilerKEGG.params = list(analysis.approach = "ORA", min.gene.set.size = 10,
  max.gene.set.size = 1000, ontology = "BP", min.overlap = 2, p.adjust.method = "BH"),
 gProfileR.params = list(data.sources = "GO:BP", show.only.significant = TRUE,
    measure_underrepresentation = FALSE, evidence_codes = TRUE, domain_scope =
    "annotated", highlight = TRUE),
  topGO.params = list(ontologies.used = c("BP"), org = "hsapiens")
)
```

Arguments

diffr.wrapper.output	list. Nested list system produced by diffrExpr-wrapper. Has to contain element "contrasts" that contains contrast-specific expression tables
analysis.name	character. Descriptive character-tag used in output file names
use.background.from.diffr.output	logical. Whether the (ORA) analyses are run with experiment-specific background obtained from pre-filtered expression matrix or with the default background of the functions (genome)
use.pval.in.DE.filtering.if.no.sign.fdrs	logical Sometimes no DE genes with significant adjusted p-value is found. In such cases, should uncorrected p-values be used in order to get at least some results
out.dir	character. Root directory for the resulting subdirectories. Must contain subfolders for contrasts. Required; no default is used so that nothing is written to the working directory unintentionally.
species	character. Currently valid options are "human" or "mouse".

p.thr numeric. Threshold for un-adjusted p-values (applied in both filtering of DE-genes and in enrichment results, when relevant (i.e. no significant fdr-entries are found)). Default 0.05

fdr.thr numeric. Threshold for adjusted p-values (applied in both filtering of DE-genes and in enrichment results). Default 0.05

logfc.thr numeric. Threshold for the log2-fold-change. Defaults to 1.

do.plot logical. Whether or not to draw a network plot for the enrichment results. Defaults to FALSE.

plot.fdr.thr numeric. FDR threshold used in the enrichment plot. This may be useful to tweak to produce a more informative plot. Defaults to 0.05.

plot.logfc.thr numeric. FC threshold on the log2-scale used in the enrichment plot. Defaults to 1.

plot.num.terms integer. Number of terms shown in the plot. Defaults to 5.

enrichment.methods
character. Enrichment methods to be run. One or more of the following: c("clusterProfilerGO", "clusterProfilerKEGG", "gProfileR", "topGO")

clusterProfilerGO.params
list. Method-specific parameters for clusterProfilerGO. One or more of the following (default values shown and used for all such elements not given in the call): analysis.approach="ORA", do.similarity.filtering=F,min.gene.set.size=10,max.gene.set.size=1000, ontology="BP", min.overlap=2,p.adjust.method="BH". Analysis approach can be "ORA" or "KEGG". If do.similarity.filtering is set to TRUE, clusterProfiler::simplify() is run.

clusterProfilerKEGG.params
list.

gProfileR.params
list.

topGO.params list.

Value

A list of all relevant objects generated in the course of the enrichment analyses

Examples

```
# needs annotation packages; run on real data with mappable gene IDs
if (requireNamespace("clusterProfiler", quietly = TRUE) &&
    requireNamespace("org.Hs.eg.db", quietly = TRUE) &&
    requireNamespace("AnnotationDbi", quietly = TRUE)) {
  out.dir <- file.path(tempdir(), "diffwrap_demo")
  dir.create(out.dir, showWarnings = FALSE)
  res <- diffExpr(diffwrap_counts, diffwrap_samp_info, samples = "SampleName",
                  groups = "Group", control = "control", analysis.name = "demo",
                  out.dir = out.dir, enr.do = FALSE)
  # wrapped in tryCatch(): enrichment depends on annotation databases and, for some
  # methods, on remote services, neither of which may be available on a check machine
  enr <- tryCatch(
```

```
    runEnrichmentAnalyses(res, analysis.name = "demo", out.dir = out.dir,
                          species = "human", enrichment.methods = "clusterProfilerGO"),
    error = function(e) {
      message("Enrichment could not be run: ", conditionMessage(e))
      NULL
    })
}
```

`run_clusterProfiler_GO` *Runs clusterProfiler GO enrichment function for a DEG list or for a ranked gene list.*

Description

Runs clusterProfiler GO enrichment function for a DEG list or for a ranked gene list.

Usage

```
run_clusterProfiler_GO(
  input_genes,
  background_genes = "",
  file_name = NULL,
  ordered_query = FALSE,
  id_type = "ENSEMBL",
  ontology = "BP",
  OrgDb = "org.Hs.eg.db",
  pvalueCutoff = 0.05,
  min_set_size = 10,
  max_set_size = 1000,
  min_overlap = 2,
  pAdjustMethod = "BH",
  similarity_filtering = FALSE,
  rng.seed = NULL
)
```

Arguments

`input_genes`	A character vector of gene IDs (ORA) or a named, ordered vector of fold changes of ALL genes with gene IDs as names (GSEA).
`background_genes`	A character vector of background gene IDs. If not specified, by default uses all human genes annotated to term domain.
`file_name`	A character string used as a file name.
`ordered_query`	If set to TRUE, computes GSEA-style p-values for an input ranked gene list.

id_type	By default uses "ENSEMBL". Can be any ID type that is supported by the corresponding OrgDb.
ontology	A vector of ontology types to use. One of the following: GO:BP = GO Biological Process GO:MF = GO Molecular Function GO:CC = GO Cellular Component
OrgDb	Organism annotation package, by default uses "org.Hs.eg.db".
pvalueCutoff	Adjusted p-value cut-off.
min_set_size	Minimum size of the functional category, uses 10 by default.
max_set_size	Maximum size of the functional category, uses 1000 by default.
min_overlap	Minimum size of the overlap (intersection) between query and functional category, smaller intersections are excluded. By default uses 2.
pAdjustMethod	The algorithm used for multiple testing correction, one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". By default uses "BH".
similarity_filtering	Similarity filtering method, either FALSE (default) or TRUE (uses simplify function).
rng.seed	Optional integer. If supplied, `set.seed()` is called with this value before the permutation-based gene set enrichment analysis, making the result reproducible. Defaults to `NULL`, i.e. the random number generator is left untouched, so that the function does not alter the state of the user's session. Note that this is distinct from the `seed` argument of `clusterProfiler::gseGO()`/`gseKEGG()`, which controls that package's own internal seeding.

Value

A table listing statistically significant enrichment results according to threshold set in pvalueCutoff The table is also saved in xlsx format with user-specified name.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  genes <- c("ENSG00000141510", "ENSG00000012048", "ENSG00000139618")
  # wrapped in tryCatch(): the result depends on the installed annotation database
  go <- tryCatch(run_clusterProfiler_GO(input_genes = genes, ontology = "BP",
                                        OrgDb = "org.Hs.eg.db", id_type = "ENSEMBL"),
                 error = function(e) {
                   message("GO enrichment could not be run: ", conditionMessage(e))
                   NULL
                 })
}
```

`run_clusterProfiler_KEGG` *Runs clusterProfiler KEGG enrichment function for a DEG list or for a ranked gene list.*

Description

Runs clusterProfiler KEGG enrichment function for a DEG list or for a ranked gene list.

Usage

```
run_clusterProfiler_KEGG(
  input_genes,
  background_genes = "",
  file_name = NULL,
  ordered_query = FALSE,
  id_type = "kegg",
  organism = "hsa",
  pvalueCutoff = 0.05,
  min_set_size = 10,
  max_set_size = 1000,
  min_overlap = 2,
  pAdjustMethod = "BH",
  rng.seed = NULL
)
```

Arguments

`input_genes`	A character vector of Entrez gene IDs (ORA) or a named, ordered vector of fold changes of ALL genes with Entrez IDs as names (GSEA).
`background_genes`	A character vector of background gene IDs. If not specified, by default uses all human genes annotated to term domain.
`file_name`	A character string used as a file name.
`ordered_query`	If set to TRUE, runs gene set enrichment analysis for an input ranked gene list.
`id_type`	By default uses "kegg", which is Entrez ID for eukaryotes and Locus ID for prokaryotes. Other options: 'ncbi-geneid', 'ncbi-proteinid' or 'uniprot'.
`organism`	"hsa" by default. Supported organism listed in 'http://www.genome.jp/kegg/catalog/org_list.html'.
`pvalueCutoff`	Adjusted p-value cut-off.
`min_set_size`	Minimum size of the functional category, uses 10 by default.
`max_set_size`	Maximum size of the functional category, uses 1000 by default.
`min_overlap`	Minimum size of the overlap (intersection) between query and functional category, smaller intersections are excluded. By default uses 2.

pAdjustMethod — The algorithm used for multiple testing correction, one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". By default uses "BH".

rng.seed — Optional integer. If supplied, `set.seed()` is called with this value before the permutation-based gene set enrichment analysis, making the result reproducible. Defaults to `NULL`, i.e. the random number generator is left untouched, so that the function does not alter the state of the user's session. Note that this is distinct from the `seed` argument of `clusterProfiler::gseGO()`/`gseKEGG()`, which controls that package's own internal seeding.

Value

A table listing statistically significant enrichment results according to threshold set in pvalueCutoff The table is also saved in xlsx format with user-specified name.

Examples

```
# enrichKEGG downloads pathway data from the KEGG web service, so this needs network access.
# Wrapped in tryCatch() so that an unreachable service reports the problem instead of
# failing the example.
if (requireNamespace("clusterProfiler", quietly = TRUE) &&
    requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  # KEGG works on Entrez identifiers
  kegg <- tryCatch(run_clusterProfiler_KEGG(input_genes = c("7157", "672", "675"),
                                            organism = "hsa", id_type = "kegg"),
                   error = function(e) {
                     message("KEGG service not reachable: ", conditionMessage(e))
                     NULL
                   })
}
```

`run_gprofiler` *Runs gprofiler function for a DEG list or for a ranked gene list.*

Description

Runs gprofiler function for a DEG list or for a ranked gene list.

Usage

```
run_gprofiler(
  input_genes,
  organism = "hsapiens",
  background_genes = "",
  file_name = NULL,
  ordered_query = FALSE,
  multi_query = FALSE,
  show_only_significant = FALSE,
```

```
  evidence_codes = FALSE,
  exclude_iea = FALSE,
  measure_underrepresentation = FALSE,
  max_p_value = 0.5,
  correction_method = "fdr",
  domain_scope = "annotated",
  data_sources = NULL,
  highlight = FALSE
)
```

Arguments

input_genes	A character vector of gene IDs, e.g. Ensembl or HGNC. Can be any ID type that has been linked to genes in the Ensembl database, and also a mixed vector of IDs.
organism	By default uses "hsapiens".
background_genes	A character vector of background gene IDs. If not specified, by default uses all human genes annotated to term domain.
file_name	A character string used as a file name without file extension. If not NULL output will be saved to an Excel file. Should be given as a full path, since a bare name would write to the current working directory. Defaults to NULL, in which case no file is written. Currently not in use.
ordered_query	If set to TRUE, computes GSEA-style p-values for an input ranked gene list.
multi_query	In case of multiple gene lists, returns comparison table of these lists. If enabled, the result data frame has columns named 'p_values', 'gconvert_sizes', 'intersection_sizes' with vectors showing values in the order of input queries.
show_only_significant	Shows only significant results based on the set padjCutoff, default TRUE.
evidence_codes	If set to TRUE, includes evidence codes to the results. Note that this can decrease performance and make the query slower. In addition, a column 'intersection' is created that contains the gene id-s that intersect between the query and term. This parameter does not work if 'multi_query' is set to TRUE.
exclude_iea	If TRUE, excludes electronic GO annotations (with evidence code IEA).
measure_underrepresentation	If TRUE, measures under-representation.
max_p_value	Adjusted p-value cut-off. Shows only terms with p-value under this cut-off if showOnlySignificant = TRUE.
correction_method	The algorithm used for multiple testing correction, one of "gSCS", "fdr", "bonferroni". By default uses "fdr".
domain_scope	How to define statistical domain, one of "annotated", "known", "custom" or "custom_annotated".
data_sources	A vector of data sources to use. One or more of the following: GO:BP = GO Biological Process GO:MF = GO Molecular Function GO:CC = GO Cellular

Component KEGG = Kyoto Encyclopedia of Genes and Genomes Pathway Database REAC = Reactome Pathway Database TF = TRANSFAC Database, putative transcription factor binding sites MI = miRTarBase, microRNA-Target Interactions CORUM = Database of manually annotated protein complexes in human, mouse and rat HPA = Human Protein Atlas, protein expression in normal tissues HP = Human Phenotype Ontology, human disease gene annotations OMIM = Online Mendelian Inheritance in Man, an online catalog of human genes and genetic disorders By default, uses all data sources.

highlight — If set to TRUE, returns a TRUE-FALSE column called 'highlighted' to indicate driver terms in GO.

Value

A table listing statistically significant enrichment results according to threshold set in padjCutoff. The table is also saved in xlsx format with user-specified name.

Examples

```
# Needs network access to the g:Profiler service. Wrapped in tryCatch() so that an
# unreachable server reports the problem instead of failing the example.
if (requireNamespace("gprofiler2", quietly = TRUE)) {
  gp <- tryCatch(run_gprofiler(input_genes = c("ENSG00000141510", "ENSG00000012048"),
                               organism = "hsapiens"),
                 error = function(e) {
                   message("g:Profiler not reachable: ", conditionMessage(e))
                   NULL
                 })
}
```

Index