Package 'markeR'

November 12, 2026

Title An R Toolkit for Evaluating Gene Signatures as Phenotypic Markers

Version 1.2.0

Description markeR is an R package that provides a modular and extensible framework for the systematic evaluation of gene sets as
phenotypic markers using transcriptomic data. The package is designed to support both quantitative analyses and visual exploration of
gene set behaviour across experimental and clinical phenotypes. It implements multiple methods, including score-based and enrichment
approaches, and also allows the exploration of expression behaviour of individual genes. In addition, users can assess the
similarity of their own gene sets against established collections (e.g., those from MSigDB), facilitating biological interpretation.

License Artistic-2.0

biocViews GeneExpression, Transcriptomics, Visualization, Software, GeneSetEnrichment, Classification

Encoding UTF-8

Language en-GB

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Additional_repositories https://bioconductor.org/packages/release/bioc

Imports circlize, edgeR, ComplexHeatmap, ggh4x, ggplot2, ggpubr, grid, gridExtra, pROC, RColorBrewer, reshape2, rstatix, scales, stats, utils, fgsea, limma, ggrepel, effectsize, msigdbr, tibble

Suggests devtools, markdown, renv, testthat, BiocManager, knitr, rmarkdown, roxygen2, mockery, covr, magick, BiocStyle

Config/testthat/edition 3

Depends R (>= 4.5.0)

URL https://diseasetranscriptomicslab.github.io/markeR/, https://github.com/DiseaseTranscriptomicsLab/markeR

BugReports https://github.com/DiseaseTranscriptomicsLab/markeR/issues

VignetteBuilder knitr

Config/Needs/website rmarkdown

git_url https://git.bioconductor.org/packages/markeR

git_branch RELEASE_3_23

git_last_commit 55addc4

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Rita Martins-Silva [aut, cre] (ORCID: <https://orcid.org/0000-0002-1067-7993>), Alexandre Kaizeler [aut, ctb] (ORCID: <https://orcid.org/0000-0002-9117-6073>), Nuno Luís Barbosa-Morais [aut, led, ths] (ORCID: <https://orcid.org/0000-0002-1215-0538>)

Maintainer Rita Martins-Silva <rita.silva@medicina.ulisboa.pt>

Contents

AUC_Scores . . . 3
calculateDE . . . 6
CalculateScores . . . 8
CalculateScores_logmedian . . . 10
CalculateScores_Ranking . . . 11
CalculateScores_ssGSEA . . . 12
CalculateScores_ssGSEA_bidirectional . . . 13
CalculateScores_ssGSEA_unidirectional . . . 14
calculateScore_logmedian_bidirectional . . . 15
calculateScore_logmedian_unidirectional . . . 16
CohenD_allConditions . . . 17
CohenD_IndividualGenes . . . 18
CohenF_allConditions . . . 20
cohen_d . . . 22
colRanking . . . 22
compute_cohens_f_pval . . . 23
compute_cohen_d . . . 23
compute_stat_tests . . . 24
CorrelationHeatmap . . . 26
counts_example . . . 28
create_contrast_column . . . 29
ExpressionHeatmap . . . 30
flatten_results . . . 32

FPR_Simulation 32
generate_all_contrasts 35
genesets_example 36
geneset_similarity 37
getRanking 39
GSEA_VariableAssociation 40
Heatmap_Cohen 42
identify_variable_type 44
IndividualGenes_Violins 45
markeR 47
metadata_example 48
plotCombinedGSEA 49
plotGSEAenrichment 50
plotNESlollipop 51
plotPCA 54
PlotScores 56
PlotScores_Categorical 61
PlotScores_Numeric 64
plotVolcano 66
remove_division 69
ROCandAUCplot 70
ROCAUC_Scores_Calculate 72
ROC_Scores 73
runGSEA 75
Score_VariableAssociation 77
ssGSEA_alternative 79
VariableAssociation 81
VisualiseIndividualGenes 84
Volcano_Cohen 87
wrap_title 88

Index **89**

`AUC_Scores` *Generate Heatmaps for AUC Scores using ggplot2*

Description

This function computes AUC scores for multiple gene signatures and scoring methods, and generates a heatmap for each gene signature. The heatmap displays the AUC scores, with the contrasts as rows and methods as columns. The heatmaps are then arranged in a grid layout.

Usage

```
AUC_Scores(
  data,
  metadata,
  gene_sets,
  method = c("logmedian", "ssGSEA", "ranking", "all"),
  mode = c("simple", "medium", "extensive"),
  variable,
  nrow = NULL,
  ncol = NULL,
  limits = NULL,
  widthTitle = 22,
  titlesize = 12,
  ColorValues = c("#F9F4AE", "#B44141"),
  title = NULL
)
```

Arguments

data	A data frame of gene expression data with genes as rows and samples as columns. Row names should contain gene names and column names sample identifiers.
metadata	A data frame of sample metadata. The first column must contain sample identifiers matching those in data.
gene_sets	A named list of gene sets.
method	A character string specifying the scoring method(s) ("logmedian", "ssGSEA", "ranking", or "all").
mode	A string specifying the level of detail for contrasts. Options are: • "simple": Pairwise comparisons (e.g., A - B). • "medium": Pairwise comparisons plus comparisons against the mean of other groups. • "extensive": All possible groupwise contrasts, ensuring balance in the number of terms on each side.
variable	A string specifying the grouping variable in metadata used for computing AUC comparisons.
nrow	Optional. An integer specifying the number of rows in the heatmap grid. If NULL, the number of rows is computed automatically.
ncol	Optional. An integer specifying the number of columns in the heatmap grid. If NULL, the number of columns is computed automatically.
limits	Optional. A numeric vector of length 2 specifying the color scale limits (e.g., c(min, max)). If NULL, the limits are determined from the data.
widthTitle	An integer specifying the width used for wrapping gene set signature names in the heatmap titles. Default is 22.
titlesize	An integer specifying the text size for each of the heatmap titles. Default is 12.
ColorValues	A character vector specifying the colors for the gradient fill in the heatmaps. Default is c("#F9F4AE", "#B44141").
title	Title for the grid of plots.

Details

The function first calculates AUC scores for each gene signature using `ROCAUC_Scores_Calculate`. The resulting matrices are converted to a long format so that each cell in the heatmap can display the AUC value. A title for each heatmap is dynamically created. The heatmaps are then adjusted to display axis text and ticks only for the left-most column and bottom row, and combined into a grid layout. If neither `nrow` nor `ncol` are specified, the layout is automatically determined to best approximate a square grid.

Value

A list with two elements:

plt A combined heatmap arranged in a grid using `ggpubr::ggarrange`.

data A list containing the AUC scores for each gene signature, as computed by `ROCAUC_Scores_Calculate`.

Examples

```
# Example data
data <- as.data.frame(abs(matrix(rnorm(1000), ncol = 10)))
rownames(data) <- paste0("Gene", 1:100)  # Name columns as Gene1, Gene2, ..., Gene10
colnames(data) <- paste0("Sample", 1:10)  # Name rows as Sample1, Sample2, ..., Sample100

# Metadata with sample ID and condition
metadata <- data.frame(
  SampleID = colnames(data),  # Sample ID matches the colnames of the data
  Condition = rep(c("A", "B"), each = 5)  # Two conditions (A and B)
)

# Example gene set
gene_sets <- list(Signature1 = c("Gene1", "Gene2", "Gene3"),
                  Signature2 = c("Gene2","Gene4","Gene10"),
                  Signature3 = c("Gene6","Gene46","Gene13"))  # Example gene sets

AUC_Scores(
    data = data,
    metadata = metadata,
    gene_sets = gene_sets,
    method = "ssGSEA",
    variable = "Condition",
    nrow = 1,
    ncol = NULL,
    limits = c(0, 1),
    widthTitle = 30,
    titlesize = 14,
    ColorValues = c("#F9F4AE", "#B44141")
  )


AUC_Scores(
  data = data,
  metadata = metadata,
```

```
  gene_sets = gene_sets,
  method = "all",
  variable = "Condition",
  nrow = 1,
  ncol = NULL,
  limits = c(0, 1),
  widthTitle = 30,
  titlesize = 14,
  ColorValues = c("#F9F4AE", "#B44141")
)
```

`calculateDE` *Calculate Differential Gene Expression Statistics using limma*

Description

This function computes differential gene expression statistics for each gene using a linear model via the limma package. Users may supply a custom design matrix directly via the `design` argument, or specify a model formula (`lmexpression`) (e.g., `~0 + X` or `~X`) or variables from `metadata` to build the design matrix. When contrasts are supplied, they are applied using `limma::makeContrasts` and `limma::contrasts.fit`. Alternatively, when using `lmexpression` or a supplied `design`, specific coefficient indices may be provided via `coefs` to extract the corresponding gene-level statistics.

Usage

```
calculateDE(
  data,
  metadata = NULL,
  variables = NULL,
  modelmat = NULL,
  contrasts = NULL,
  ignore_NAs = FALSE
)
```

Arguments

`data`	A numeric matrix of gene expression values with genes as rows and samples as columns. Row names must correspond to gene identifiers. Data should *not* be transformed (i.e., not log2 transformed).
`metadata`	A data frame containing sample metadata used to build the design matrix (unless a design is provided directly).
`variables`	A character vector specifying the variable(s) from `metadata` to use in the default linear model. Ignored if `lmexpression` or `design` is provided.
`modelmat`	(Optional) A user-supplied design matrix. If provided, this design is used directly and `lmexpression` and `variables` are ignored. The order of samples in the design matrix should match the order in data.

`contrasts` A character vector specifying contrasts to be applied (e.g., `c("A-B")`). If multiple contrasts are provided, the function returns a list of DE results (one per contrast). *Required* if `lmexpression` is NULL, optional otherwise. If not provided, the average expression profile of each condition will be returned instead of differential gene expression.

`ignore_NAs` Boolean (default: FALSE). Whether to ignore NAs in the metadata. If TRUE, rows with any NAs will be removed before analysis, leading to a loss of data to be fitted in the model. Only applicable if `variables` is provided.

Details

The function fits a linear model with `limma::lmFit` and applies empirical Bayes moderation with `limma::eBayes`. Depending on the input:

- If a design matrix is provided via `design`, that design is used directly.
- Otherwise, a design matrix is constructed using the `variables` argument (with no intercept).
- If contrasts are provided, they are applied using `limma::makeContrasts` and `limma::contrasts.fit`.
- If no contrasts are provided, the function returns all possible coefficients fitted in the linear model.

Value

A list of data-frames of differential expression statistics

Examples

```
# Simulate non-negative gene expression data (counts)
set.seed(123)
expr <- matrix(rpois(1000, lambda = 20), nrow = 100, ncol = 10)
rownames(expr) <- paste0("gene", 1:100)
colnames(expr) <- paste0("sample", 1:10)

# Simulate metadata with a group variable
metadata <- data.frame(
 sample = colnames(expr),
 Group = rep(c("A", "B"), each = 5)
)

# Differential expression for Group A vs Group B using variables
de_var <- calculateDE(
  data = expr,
  metadata = metadata,
  variables = "Group",
  contrasts = "A-B"
)
head(de_var[["A-B"]])

# Build equivalent design matrix manually
design <- model.matrix(~0 + Group, data = metadata)
colnames(design) <- c("A","B")
```

```
# Differential expression using the design matrix directly
de_mat <- calculateDE(
  data = expr,
  metadata = metadata,
  modelmat = design,
  contrasts = "A-B"
)
head(de_mat[["A-B"]])
```

`CalculateScores` *Calculate Gene Signature Scores using Score-Based Approaches*

Description

This function calculates a gene signature score for each sample based on one or more predefined gene sets (signatures).

Usage

```
CalculateScores(
  data,
  metadata,
  gene_sets,
  method = c("ssGSEA", "logmedian", "ranking", "all")
)
```

Arguments

`data`	A data frame of normalized (non-transformed) counts where each row is a gene and each column is a sample. The row names should contain gene names, and the column names should contain sample identifiers. **(Required)**
`metadata`	A data frame describing the attributes of each sample. Each row corresponds to a sample and each column to an attribute. The first column of `metadata` should be the sample identifiers (i.e., the column names of `data`). Defaults to `NULL` if no metadata is provided.
`gene_sets`	Gene set input. **(Required)** If using **unidirectional** gene sets, provide a named list where each element is a vector of gene names representing a gene signature. The names of the list elements should correspond to the labels for each signature. If using **bidirectional** gene sets, provide a named list where each element is a data frame. The names of the list elements should correspond to the labels for each signature, and each data frame should contain the following structure: • The **first column** should contain gene names. • The **second column** should indicate the expected direction of enrichment (1 for upregulated genes, -1 for downregulated genes).

`method` A character string indicating the scoring method to use. Options are `"ssGSEA"`, `"logmedian"`, `"ranking"`, or `"all"` (to compute scores using all methods). Defaults to `"logmedian"`.

Details

This function calculates a gene signature score for each sample based on one or more predefined gene sets (signatures). Four methods are available:

Uses the single-sample Gene Set Enrichment Analysis (ssGSEA) method to compute an enrichment score for each signature in each sample. This method uses an adaptation from the the `gsva()` function from the `GSVA` package to compute an enrichment score, representing the absolute enrichment of each gene set in each sample.

`ssGSEA` `logmedian` Computes, for each sample, the score as the sum of the normalized (log2-median-centered) expression values of the signature genes divided by the number of genes in the signature.

`ranking` Computes gene signature scores for each sample by ranking the expression of signature genes in the dataset and normalizing the score based on the total number of genes.

`all` Computes gene signature scores using all three methods (`ssGSEA`, `logmedian`, and `ranking`). The function returns a list containing the results of each method.

Value

If a single method is chosen, a data frame containing the calculated scores for each gene signature, including metadata if provided. If `method = "all"`, a list is returned where each element corresponds to a scoring method and contains the respective data frame of scores.

sample The sample identifier (matching the column names of the input data).

score The calculated gene signature score for the corresponding sample.

(metadata) Any additional columns from the `metadata` data frame provided by the user, if available.

Examples

```
# Simulate positive gene expression data (genes as rows, samples as columns)
set.seed(42)
expr <- as.data.frame(matrix(rexp(60, rate = 0.2), nrow = 6, ncol = 10))
rownames(expr) <- paste0("Gene", 1:6)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate metadata for samples
metadata <- data.frame(
  sample = colnames(expr),
  Group = rep(c("A", "B"), each = 5)
)

# Define two simple gene sets
gene_sets <- list(
  Signature1 = c("Gene1", "Gene2", "Gene3"),
  Signature2 = c("Gene4", "Gene5", "Gene6")
```

```
)

# Calculate logmedian scores
scores_logmedian <- CalculateScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
  method = "logmedian"
)
head(scores_logmedian)

# Calculate all score types
scores_all <- CalculateScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
  method = "all"
)
lapply(scores_all, head)
```

`CalculateScores_logmedian` *Calculate Gene Signature Scores using Log-Median Approach*

Description

Computes log2-median-centered scores for each sample based on gene signature expression.

Usage

```
CalculateScores_logmedian(data, metadata = NULL, gene_sets)
```

Arguments

`data`	A data frame of normalized counts where each row is a gene and each column is a sample.
`metadata`	A data frame containing sample metadata (optional). If provided, the resulting scores will be merged with metadata.
`gene_sets`	A named list representing gene sets. **(Required)**

- **Unidirectional gene sets:** Each element should be a vector of gene names representing a signature. The names of the list elements serve as labels for the signatures.
- **Bidirectional gene sets:** Each element should be a data frame with two columns:
  - First column: gene names.
  - Second column: expected direction of enrichment (1 for upregulated, -1 for downregulated).

Value

A list of data frames containing log-median scores for each signature. If `metadata` is provided, it is merged with the scores.

Examples

```
## Not run:
data <- matrix(rnorm(1000), nrow = 100, ncol = 10)
colnames(data) <- paste0("Sample_", 1:10)
rownames(data) <- paste0("Gene_", 1:100)
gene_sets <- list(
  Signature_A = sample(rownames(data), 10),
  Signature_B = data.frame(Gene = sample(rownames(data), 10), Direction =
  sample(c(1, -1), 10, replace = TRUE))
)
scores <- CalculateScores_logmedian(data, gene_sets = gene_sets)

## End(Not run)
```

`CalculateScores_Ranking`

Calculate Gene Signature Scores using Ranking Approach

Description

Computes gene signature scores for each sample by ranking the expression of signature genes in the dataset and normalizing the score based on the total number of genes.

Usage

```
CalculateScores_Ranking(data, metadata = NULL, gene_sets)
```

Arguments

`data`	A data frame where rows represent genes, columns represent samples, and values correspond to gene expression levels. **(Required)**
`metadata`	A data frame containing sample metadata. The first column must contain sample names. **(Optional)**
`gene_sets`	A named list of gene sets. **(Required)** For unidirectional gene sets, provide a named list where each element is a vector of gene names. For bidirectional gene sets, provide a named list where each element is a data frame with two columns: • The first column: gene names. • The second column: expected direction (1 for upregulated, -1 for downregulated).

Details

- The function first validates inputs and extracts relevant genes from the dataset.
- For **unidirectional** signatures, it computes rankings based on gene expression levels.
- For **bidirectional** signatures, it computes separate rankings for upregulated and downregulated genes, then calculates a final score by subtracting downregulated rankings from upregulated rankings.
- The final scores are normalized by dividing by the total number of genes.
- This metric is not suitable to compare absolute values between different gene sets, i.e. should be used only for relative comparisons between samples when using the same gene set.

Value

A named list of data frames, where each data frame contains:

- `sample`: Sample name.
- `score`: Normalized ranking score for the given gene signature.
- Additional metadata columns (if `metadata` is provided).

Examples

```
## Not run:
# Example dataset with 5 genes (rows) and 3 samples (columns)
set.seed(123)
data <- as.data.frame(matrix(runif(15, 1, 100), nrow = 5, ncol = 3))
rownames(data) <- paste0("Gene_", 1:5)
colnames(data) <- paste0("Sample_", 1:3)

# Unidirectional gene set example
gene_sets <- list(Signature1 = c("Gene_1", "Gene_3", "Gene_5"))

# Compute scores
scores <- CalculateScores_Ranking(data, gene_sets = gene_sets)
print(scores)

## End(Not run)
```

`CalculateScores_ssGSEA` *Calculate Gene Signature Scores using ssGSEA*

Description

Computes an enrichment score for each gene signature in each sample using the single-sample Gene Set Enrichment Analysis (ssGSEA).

Usage

```
CalculateScores_ssGSEA(data, metadata = NULL, gene_sets)
```

Arguments

`data`	A data frame of normalized (non-transformed) counts where each row is a gene and each column is a sample.
`metadata`	A data frame containing sample metadata (optional).
`gene_sets`	Gene set input. **(Required)** If using **unidirectional** gene sets, provide a named list where each element is a vector of gene names representing a gene signature. The names of the list elements should correspond to the labels for each signature. If using **bidirectional** gene sets, provide a named list where each element is a data frame. The names of the list elements should correspond to the labels for each signature, and each data frame should contain the following structure: • The **first column** should contain gene names. • The **second column** should indicate the expected direction of enrichment (1 for upregulated genes, -1 for downregulated genes).

Value

A list of data frames containing ssGSEA scores for each signature.

`CalculateScores_ssGSEA_bidirectional` *Calculate ssGSEA Scores for Bidirectional Gene Signatures*

Description

Computes single-sample Gene Set Enrichment Analysis (ssGSEA) scores for each sample using a bidirectional gene signature (separating upregulated and downregulated genes).

Usage

```
CalculateScores_ssGSEA_bidirectional(data, signature)
```

Arguments

`data`	A data frame of normalized (non-transformed) counts where rows are genes and columns are samples.
`signature`	A data frame with: • The **first column** containing gene names. • The **second column** (`Signal`) indicating the expected direction of enrichment (1 for upregulated genes, -1 for downregulated genes).

Details

- The input gene expression matrix (`data`) is log2-transformed before applying ssGSEA.
- Upregulated and downregulated genes are analyzed separately.
- As both upregulated and downregulated genes are present, the final score is computed as:

$$score = (score_{up} \frac{|up_genes|}{|total_genes|}) - (score_{down} \frac{|down_genes|}{|total_genes|})$$

- If no downregulated genes are present, only the upregulated score is used.
- The results are reshaped into a long-format data frame with one score per sample.

Value

A data frame containing:

- `sample`: Sample name.
- `score`: Final ssGSEA enrichment score (computed as the difference between upregulated and downregulated scores).

Examples

```
## Not run:
# Example dataset with 5 genes (rows) and 3 samples (columns)
set.seed(123)
data <- matrix(runif(15, 1, 100), nrow = 5, ncol = 3)
rownames(data) <- paste0("Gene_", 1:5)
colnames(data) <- paste0("Sample_", 1:3)

# Define a bidirectional gene signature
signature <- data.frame(Gene = c("Gene_1", "Gene_3", "Gene_5"),
                        Signal = c(1, -1, 1))

# Compute scores
scores <- CalculateScores_ssGSEA_bidirectional(data, signature = signature)
print(scores)

## End(Not run)
```

`CalculateScores_ssGSEA_unidirectional`
Calculate ssGSEA Scores for Unidirectional Gene Signatures

Description

Computes single-sample Gene Set Enrichment Analysis (ssGSEA) scores for each sample using a unidirectional gene signature.

Usage

```
CalculateScores_ssGSEA_unidirectional(data, signature)
```

Arguments

data	A data frame of normalized (non-transformed) counts where rows are genes and columns are samples.
signature	A vector of gene names representing a unidirectional gene signature.

Value

A data frame containing:

- sample: Sample name.
- score: ssGSEA enrichment score for the gene signature.

Examples

```
## Not run:
# Example dataset with 5 genes (rows) and 3 samples (columns)
set.seed(123)
data <- matrix(runif(15, 1, 100), nrow = 5, ncol = 3)
rownames(data) <- paste0("Gene_", 1:5)
colnames(data) <- paste0("Sample_", 1:3)

# Define a unidirectional gene signature
signature <- c("Gene_1", "Gene_3", "Gene_5")

# Compute scores
scores <- CalculateScores_ssGSEA_unidirectional(data, signature = signature)
print(scores)

## End(Not run)
```

---

calculateScore_logmedian_bidirectional

Calculate Log-Median Scores for Bidirectional Gene Sets

Description

Computes gene signature scores considering both upregulated and downregulated genes separately, then calculates a differential score by subtracting downregulated from upregulated scores.

Usage

```
calculateScore_logmedian_bidirectional(data, signature)
```

Arguments

`data`	A data frame of normalized counts (genes as rows, samples as columns).
`signature`	A data frame with: • The **first column** containing gene names. • The **second column** specifying enrichment direction (`1` for upregulated, `-1` for downregulated).

Value

A named vector with log-median-centered scores per sample.

`calculateScore_logmedian_unidirectional` *Calculate Log-Median Scores for Unidirectional Gene Sets*

Description

Computes log-median-centered scores for gene signatures where all genes are expected to be enriched in the same direction, or when direction is not known.

Usage

```
calculateScore_logmedian_unidirectional(data, signature)
```

Arguments

`data`	A data frame of normalized counts (genes as rows, samples as columns).
`signature`	A vector of gene names or a data frame where the first column contains gene names.

Value

A named vector with log-median-centered scores per sample.

`CohenD_allConditions` *Compute Cohen\'s d for All Gene Signatures Across Conditions*

Description

Computes Cohen\'s d effect sizes and corresponding p-values for all gene signatures using scores calculated by various methods. The function first computes gene signature scores using `CalculateScores` with the "all" option, flattens the results, and then computes pairwise comparisons for a specified grouping variable.

Usage

```
CohenD_allConditions(
  data,
  metadata,
  gene_sets,
  variable,
  mode = c("simple", "medium", "extensive"),
  p.adjust.method = "BH"
)
```

Arguments

`data`	A data frame of gene expression data, with genes as rows and samples as columns.
`metadata`	A data frame containing sample metadata. The first column should contain sample identifiers matching the column names of `data`.
`gene_sets`	A named list of gene sets. For unidirectional gene sets, each element is a vector of gene names; for bidirectional gene sets, each element is a data frame where the first column contains gene names and the second column indicates the expected direction (1 for upregulated, -1 for downregulated).
`variable`	A string specifying the grouping variable in `metadata` used to compare scores between conditions.
`mode`	A string specifying the level of detail for contrasts. Options are: • `"simple"`: Pairwise comparisons (e.g., A - B). • `"medium"`: Pairwise comparisons plus comparisons against the mean of other groups. • `"extensive"`: All possible groupwise contrasts, ensuring balance in the number of terms on each side.
`p.adjust.method`	Character string specifying the method to use for multiple testing correction. Must be one of `"BH"` (Benjamini-Hochberg, default), `"holm"`, `"hommel"`, `"bonferroni"`, `"BY"` (Benjamini-Yekutieli), `"fdr"`, or `"none"`. Passed to `p.adjust`.

Value

A named list where each element corresponds to a gene signature. Each signature element is a list with three components:

CohenD A data frame where rows are methods and columns are group contrasts (formatted as \"Group1:Group2\"), containing the computed Cohen\'s d effect sizes.

PValue A data frame with the same structure as `CohenD` containing the corresponding p-values.

padj A data frame with the same structure as `PValue` containing the corresponding p-values corrected using the BH method, for all signatures and contrasts, and by method.

Examples

```
## Not run:
 # Assume gene_data is your gene expression data frame, sample_metadata is your metadata, and
  # gene_sets is a named list of gene sets.
  results <- CohenD_allConditions(data = gene_data, metadata = sample_metadata,
                                  gene_sets = gene_sets, variable = "Condition")
  # Access Cohen's d for a specific signature:
  results$Signature_A$CohenD

## End(Not run)
```

`CohenD_IndividualGenes`

Cohen's d Heatmap Function

Description

This function computes Cohen's d for each gene based on gene expression data and sample metadata. For each gene, it compares the expression values between samples where `condition_var` equals `class` (the positive class) versus the remaining samples. The resulting effect sizes are then visualized as a heatmap.

Usage

```
CohenD_IndividualGenes(
  data,
  metadata,
  genes = NULL,
  condition_var,
  class,
  group_var = NULL,
  title = NULL,
  titlesize = 16,
  params = list()
)
```

Arguments

data	A data frame or matrix containing gene expression data, with genes as rows and samples as columns.
metadata	A data frame containing sample metadata. The first column should contain sample identifiers that match the column names of data.
genes	A character vector specifying which genes to include. If NULL (default), all genes in data are used. A warning is issued if more than 30 genes are selected.
condition_var	A character string specifying the column name in metadata representing the condition of interest. (Mandatory; no default.)
class	A character string or vector specifying the positive class label for the condition. (Mandatory; no default.)
group_var	An optional character string specifying the column name in metadata used for grouping samples. If not provided (NULL), all samples are treated as a single group.
title	An optional character string specifying a custom title for the heatmap. If not provided, a default title is generated.
titlesize	A numeric value specifying the size of the title. Default is 14.
params	A list of additional parameters for customizing the heatmap. Possible elements include: cluster_rows Logical; if TRUE (default), rows are clustered. cluster_columns Logical; if TRUE (default), columns are clustered. colors A vector of length 2 of colors to be used for the minimum and maximum values of the color scale. Defaults to c("#FFFFFF", "#21975C"), but note that the default mapping for Cohen's d is set to a divergent scale. limits A numeric vector of length 2 specifying the minimum and maximum values for the color scale. If not provided, defaults to c(-2, 2). name A character string for the legend title of the color scale. Default is "Cohen's d". row_names_gp Optional graphical parameters for row names (passed to **ComplexHeatmap**). column_names_gp Optional graphical parameters for column names (passed to **ComplexHeatmap**).

Details

This function computes Cohen's d for each gene by comparing the expression values between samples with condition_var == class and those that do not. The effect sizes are then visualized as a heatmap using **ComplexHeatmap**. When group_var is not provided, all samples are treated as a single group.

Value

Invisibly returns a list containing:

plot The **ComplexHeatmap** object of the Cohen's d heatmap.

data A data frame with the calculated Cohen's d values for each gene and group.

Examples

```
# Simulate gene expression data (genes as rows, samples as columns)
set.seed(101)
expr <- matrix(abs(rnorm(40)), nrow = 4, ncol = 10)  # 4 genes, 10 samples,
# positive values
rownames(expr) <- paste0("Gene", 1:4)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate sample metadata with a binary condition and a group variable
metadata <- data.frame(
  Sample = colnames(expr),
  Condition = rep(c("Case", "Control"), each = 5),
  Group = rep(c("A", "B"), times = 5),
  stringsAsFactors = FALSE
)

# 1. Cohen's d heatmap for all genes across groups
CohenD_IndividualGenes(
  data = expr,
  metadata = metadata,
  genes = rownames(expr),
  condition_var = "Condition",
  class = "Case",
  group_var = "Group",
  title = "Cohen's d Heatmap (Case vs Control)",
  params = list(limits = c(0, 2))
)

# 2. Cohen's d barplot (single group across all samples)
CohenD_IndividualGenes(
  data = expr,
  metadata = metadata,
  genes = rownames(expr),
  condition_var = "Condition",
  class = "Case",
  group_var = NULL,
  title = "Cohen's d Barplot (All Samples)",
  params = list(colors = c("#CCCCCC"))
)
```

CohenF_allConditions *Compute Cohen's f for All Gene Signatures Across a Categorical Variable*

Description

Computes Cohen's f effect sizes and corresponding p-values for all gene signatures using scores calculated by multiple methods. The function first computes gene signature scores using `CalculateScores`

with the "all" option, flattens the results, and fits linear models using the specified variable to estimate effect sizes.

Usage

```
CohenF_allConditions(
  data,
  metadata,
  gene_sets,
  variable,
  p.adjust.method = "BH"
)
```

Arguments

`data`	A data frame of gene expression data, with genes as rows and samples as columns.
`metadata`	A data frame containing sample metadata. The first column should contain sample identifiers matching the column names of `data`.
`gene_sets`	A named list of gene sets. For unidirectional gene sets, each element is a vector of gene names; for bidirectional gene sets, each element is a data frame where the first column contains gene names and the second column indicates the expected direction (1 for upregulated, -1 for downregulated).
`variable`	A string specifying the categorical variable in `metadata` used to model the gene signature scores.
`p.adjust.method`	Character string specifying the method to use for multiple testing correction. Must be one of `"BH"` (Benjamini-Hochberg, default), `"holm"`, `"hommel"`, `"bonferroni"`, `"BY"` (Benjamini-Yekutieli), `"fdr"`, or `"none"`. Passed to `p.adjust`.

Details

This function is designed for use with categorical variables, where the goal is to evaluate the overall group effect (e.g., using ANOVA) across multiple levels.

Value

A named list where each element corresponds to a gene signature. Each signature element is a list with three components:

CohenF A data frame where rows are scoring methods and columns are the variable used in the linear model (usually one column), containing the computed Cohen's f effect size.

PValue A data frame of the corresponding raw p-values from the linear model for each method.

padj A data frame of adjusted p-values (Benjamini-Hochberg method) across signatures and contrasts, per method.

cohen_d *Compute Cohen\'s d Effect Size*

Description

Computes the absolute Cohen\'s d effect size between two numeric vectors. This function returns the absolute value of the difference in means divided by the pooled standard deviation.

Usage

```
cohen_d(x, y)
```

Arguments

x	A numeric vector representing the values for group 1.
y	A numeric vector representing the values for group 2.

Value

A numeric value representing Cohen\'s d. Returns NA if either group has fewer than two observations or if the pooled standard deviation is zero.

colRanking *Compute Independent Column-wise Ranks of Matrix Elements*

Description

This function computes the rank of each element in every column of a numeric matrix independently. For each column, the smallest element receives a rank of 1, the second smallest a rank of 2, and so on.

Usage

```
colRanking(x, ties.method = "average")
```

Arguments

x	A numeric matrix.
ties.method	A character string specifying the method used for tie-breaking. Options include "average", "first", "random", "max", or "min". The default is "average".

Value

A numeric matrix of the same dimensions as x where each column contains the ranks of the corresponding column's elements.

compute_cohens_f_pval *Compute Cohen's f and p-value for a given model and predictor*

Description

This function calculates the Cohen's f effect size and the corresponding p-value for a given linear model or ANOVA model based on the predictor variable type (numeric or categorical).

Usage

```
compute_cohens_f_pval(model, type)
```

Arguments

model	A linear model (lm) or ANOVA model (aov) fitted to the data.
type	A string indicating whether the predictor is numeric or categorical. Options are "Numeric" or "Categorical".

Details

Cohen's f effect size is computed from the eta-squared (η^2) value. For numeric predictors (continuous variables), the p-value is obtained from the t-test in `summary(lm(...))`. For categorical predictors (binary or multi-level), the p-value is obtained from the F-test in `anova(lm(...))`.

Value

A named vector with two elements:

- `Cohen_f`: The Cohen's f effect size value.
- `P_Value`: The p-value from the statistical test.

compute_cohen_d *Compute Pairwise Cohen\'s d and P-Values*

Description

Computes Cohen\'s d effect sizes and corresponding p-values for all pairwise comparisons of a grouping variable in a data frame.

Usage

```
compute_cohen_d(
  dfScore,
  variable,
  quantitative_var = "score",
  mode = c("simple", "medium", "extensive")
)
```

Arguments

dfScore	A data frame containing at least one numeric column and a grouping variable. Output from flatten_results.
variable	A string specifying the name of the categorical grouping column in dfScore.
quantitative_var	A string specifying the name of the numeric column (default is "score").

Value

A data frame with the following columns:

Group1 The first group in the pair.

Group2 The second group in the pair.

CohenD The computed Cohen\'s d effect size for the comparison.

PValue The p-value from a t-test comparing the two groups.

---

compute_stat_tests *Compute Statistical Tests for Variable Associations with a Target Variable*

Description

Performs statistical tests to assess the relationship between predictor variables and a target variable, selecting appropriate methods based on variable types. Returns a list of data frames containing metric values and p-values.

Usage

```
compute_stat_tests(
  df,
  target_var,
  cols = NULL,
  numeric = "pearson",
  categorical_bin = "t.test",
  categorical_multi = "anova",
  p.adjust.method = "BH"
)
```

Arguments

df	A data frame containing the target variable and predictors.
target_var	A string specifying the dependent variable.
cols	Optional. A character vector of predictor variables. If NULL, all variables except target_var are used.

`numeric` The correlation method for numeric predictors. Options: `"pearson"` (default), `"spearman"`, `"kendall"`.

`categorical_bin` The statistical test for binary categorical variables. Options: `"t.test"` (default) or `"wilcoxon"`.

`categorical_multi` The statistical test for multi-level categorical variables. Options: `"anova"` (default) or `"kruskal-wallis"`.

`p.adjust.method` Character string specifying the method to use for multiple testing correction. Must be one of `"BH"` (Benjamini-Hochberg, default), `"holm"`, `"hommel"`, `"bonferroni"`, `"BY"` (Benjamini-Yekutieli), `"fdr"`, or `"none"`. Passed to `p.adjust`.

Details

Variable Classification:

- **Numeric**: Continuous numeric or integer variables with more than 10 unique values.
- **Categorical Bin**: Binary categorical variables (factors, characters, or integers with exactly 2 unique values).
- **Categorical Multi**: Categorical variables with more than 2 unique values (up to 10 levels recommended). A warning is issued for categorical variables with more than 10 unique values.

Statistical Tests Applied:

- **Numeric Predictors**: Pearson, Spearman, or Kendall correlation.
- **Categorical Bin Predictors**: T-test or Wilcoxon rank-sum test.
- **Categorical Multi Predictors**: ANOVA (default) or Kruskal-Wallis test. If ANOVA is used, Tukey's HSD post-hoc test is performed for multiple comparisons.

The function automatically detects variable types and applies the appropriate test. If a categorical variable has more than 10 unique levels, a warning is issued. If an invalid statistical test is requested, the function stops with an error message.

Value

A named list (one entry per variable being analysed) where each element is a data frame with:

- **Metric**: The test statistic (correlation coefficient, t-statistic, ANOVA F-value, etc.).
- **p-value**: The significance value of the test.
- For **Categorical Multi**, multiple rows are included for pairwise comparisons (Tukey HSD results).

Examples

```
## Not run:
df <- data.frame(
  score = c(80, 85, 90, 95, 100),
  age = c(25, 30, 35, 40, 45),
  gender = c("Male", "Female", "Male", "Female", "Male"),
```

```
  group = factor(c("A", "B", "A", "B", "C"))
)

results <- compute_stat_tests(df, target_var = "score")
print(results)

## End(Not run)
```

`CorrelationHeatmap` *CorrelationHeatmap: Generate correlation heatmaps with optional grouping*

Description

This function generates correlation heatmaps using the `ComplexHeatmap` package. It allows users to compute correlation matrices for a set of genes and visualize them in a heatmap. If a grouping variable is provided (`separate.by`), multiple heatmaps are created, each corresponding to a different level of the grouping variable.

Usage

```
CorrelationHeatmap(
  data,
  metadata = NULL,
  genes,
  separate.by = NULL,
  method = c("pearson", "spearman", "kendall"),
  colorlist = list(low = "blue", mid = "white", high = "red"),
  limits_colorscale = NULL,
  widthTitle = 16,
  title = NULL,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  detailedresults = FALSE,
  legend_position = c("right", "top"),
  titlesize = 20,
  show_row_names = TRUE,
  show_column_names = TRUE
)
```

Arguments

`data`	A numeric counts data frame where rows correspond to genes and columns to samples.
`metadata`	A data frame containing metadata. Required if `separate.by` is specified. The first column should be the sample ID.
`genes`	A character vector of gene names to be included in the correlation analysis.

`separate.by` A character string specifying a column in `metadata` to separate heatmaps by (e.g., "Condition").

`method` Correlation method: `"pearson"` (default), `"spearman"`, or `"kendall"`.

`colorlist` A named list specifying the colors for the heatmap (`low`, `mid`, `high`), corresponding to the limits of the colorscale.

`limits_colorscale` A numeric vector of length 3 defining the limits for the color scale (default: min, 0, max).

`widthTitle` Numeric value controlling the width of the plot title. Default is `16`.

`title` A string specifying the main title of the heatmap(s).

`cluster_rows` Logical; whether to cluster rows (default = `TRUE`).

`cluster_columns` Logical; whether to cluster columns (default = `TRUE`).

`detailedresults` Logical; if `TRUE`, additional analysis results are stored in the output list (default = `FALSE`).

`legend_position` Character; position of the legend (`"right"` - default

- or `"top"`).

`titlesize` Numeric; font size of the heatmap title (default = `20`).

`show_row_names` A character string specifying whether row names (genes) should be displayed.

`show_column_names` A character string specifying whether column names (samples) should be displayed.

Value

A list containing:

`data` Correlation matrices for each condition (or a single matrix if `separate.by = NULL`).

`plot` The generated heatmap object(s).

`aux` A list containing additional analysis results if `detailedresults = TRUE`.

If `separate.by` is specified, the `aux` list contains one element per condition. Each element is a list with:

- `method`: The correlation method used.
- `corrmatrix`: The computed correlation matrix for that condition.
- `metadata`: The subset of metadata corresponding to the condition.
- `heatmap`: The `ComplexHeatmap` object before being drawn.

If `separate.by = NULL` (single heatmap case), the `aux` list contains:

- `method`: The correlation method.
- `corrmatrix`: The computed correlation matrix.

Examples

```
# Simulate gene expression data (genes as rows, samples as columns)
set.seed(1)
expr <- as.data.frame(matrix(runif(60, min = 0, max = 10), nrow = 6, ncol = 10))
rownames(expr) <- paste0("Gene", 1:6)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate metadata with a group variable
metadata <- data.frame(
 SampleID = colnames(expr),
 Condition = rep(c("A", "B"), each = 5)
)

# Basic heatmap for selected genes
res <- CorrelationHeatmap(
 data = expr,
 genes = rownames(expr)
)

# Heatmap separated by condition
res_sep <- CorrelationHeatmap(
 data = expr,
 metadata = metadata,
 genes = rownames(expr),
 separate.by = "Condition"
)
```

---

counts_example *Gene Expression Counts for Marthandan et al. (2016) RNA-Seq Data*

Description

A numeric matrix containing filtered and normalized (non log-transformed) gene expression data from the Marthandan et al. (2016) study (GEO accession GSE63577).

Usage

```
data(counts_example)
```

Format

A numeric matrix with rows as genes (gene symbols) and columns as samples (sample IDs).

Details

Raw FASTQ files were downloaded using `fasterq-dump` (v2.11.0) and processed in a reproducible conda environment (Python v3.11.5). Quality control was conducted using FastQC (v0.12.1) and summarised with MultiQC (v1.14). Pseudo-alignment to the RefSeq transcriptome (NCBI release 109) was performed using kallisto (v0.44.0). Genes with low expression (mean count < 70 in all conditions) were filtered out. Count normalization factors were calculated with `edgeR::calcNormFactors`.

Intermediate time points for HFF and MRC5 cell lines were excluded, resulting in a final dataset with 45 high-quality samples across proliferative, quiescent, and senescent conditions.

For illustration and package size reduction, genes with variance in the bottom 10% across samples were removed, retaining the 90% most variable genes in the dataset.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63577

References

Marthandan S, Priebe S, Baumgart M, Groth M et al. Similarities in Gene Expression Profiles during In Vitro Aging of Primary Human Embryonic Lung and Foreskin Fibroblasts. *Biomed Res Int* 2015;2015:731938. PMID: 26339636

Marthandan S, Baumgart M, Priebe S, Groth M et al. Conserved Senescence Associated Genes and Pathways in Primary Human Fibroblasts Detected by RNA-Seq. *PLoS One* 2016;11(5):e0154531. PMID: 27140416

`create_contrast_column` *Create Contrast Column in Metadata*

Description

This function extracts and processes contrast groups from a given contrast string, then assigns contrast labels to a metadata subset based on the variable of interest.

Usage

```
create_contrast_column(metadata, variable_name, contrast)
```

Arguments

`metadata`	A data frame containing sample metadata.
`variable_name`	A character string specifying the column name in `metadata` that represents the variable of interest.
`contrast`	A character string representing the contrast in the form "(A + B) - (C + D)" (e.g.).

Value

A subset of `metadata` with an added `cohentest` column, indicating group membership based on the contrast.

`ExpressionHeatmap` *ExpressionHeatmap: Generate an expression heatmap with customizable sample annotations and separate legend positions*

Description

This function creates a heatmap of Z-score scaled gene expression using the `ComplexHeatmap` package. Genes are displayed as rows and samples as columns. A color annotation bar is added on top based on specified metadata columns. The user can control the position of the heatmap color scale (scale_position) and the annotation legend (legend_position) independently.

Usage

```
ExpressionHeatmap(
  data,
  metadata = NULL,
  genes,
  annotate.by = NULL,
  annotation_colors = NULL,
  colorlist = list(low = "blue", mid = "white", high = "red"),
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  title = NULL,
  titlesize = 20,
  scale_position = c("right", "top", "bottom"),
  legend_position = c("top", "right", "bottom"),
  show_row_names = TRUE,
  show_column_names = FALSE
)
```

Arguments

`data`	A numeric expression matrix where rows correspond to genes and columns to samples.
`metadata`	A data frame containing metadata for the samples. It must contain a column named `"Sample"` with sample IDs matching the column names of `data`.
`genes`	A character vector of gene names to include in the heatmap.
`annotate.by`	A character vector of metadata column names to be used for sample annotations (e.g., `c("Condition", "Batch")`). If provided, a color bar is added on top.
`annotation_colors`	Optional. A named list where each element corresponds to an annotation variable and provides a named vector mapping each unique level to a color. If not provided, default Brewer palettes are used.

`colorlist` A named list specifying the colors for the heatmap (for scaled expression) with elements `low`, `mid`, and `high`. Default is `list(low = "blue", mid = "white", high = "red")`.

`cluster_rows` Logical; whether to cluster rows (default = `TRUE`).

`cluster_columns` Logical; whether to cluster columns (default = `TRUE`). If `FALSE`, the columns are reordered based on the values in `annotate.by`.

`title` A string specifying the main title of the heatmap.

`titlesize` Numeric; font size of the heatmap title (default = 20).

`scale_position` A character string specifying the position of the heatmap color scale. Options are `"right"` (default), `"top"`, or `"bottom"`. The scale legend will adopt a vertical orientation if on the right and horizontal if on top or bottom.

`legend_position` A character string specifying the position of the annotation legend. Options are `"top"` (default), `"right"`, or `"bottom"`.

`show_row_names` A character string specifying whether row names (genes) should be displayed.

`show_column_names` A character string specifying whether column names (samples) should be displayed.

Value

Invisibly returns a list with:

data Scaled expression matrix (Z-scores).

plot Generated ComplexHeatmap object.

Examples

```
# Simulate gene expression data (genes as rows, samples as columns)
set.seed(1)
expr <- matrix(rnorm(25), nrow = 5, ncol = 5)
rownames(expr) <- paste0("Gene", 1:5)
colnames(expr) <- paste0("Sample", 1:5)

# Simulate metadata for samples
metadata <- data.frame(
  Sample = colnames(expr),
  Condition = rep(c("A", "B"), length.out = 5),
  Batch = rep(c("X", "Y"), length.out = 5),
  stringsAsFactors = FALSE
)

# Define annotation colors for the metadata variables
annotation_colors <- list(
  Condition = c(A = "orange", B = "purple"),
  Batch = c(X = "green", Y = "blue")
)
```

```
# Generate the expression heatmap
ExpressionHeatmap(
  data = expr,
  metadata = metadata,
  genes = rownames(expr),
  annotate.by = c("Condition", "Batch"),
  annotation_colors = annotation_colors,
  cluster_columns = FALSE,
  title = "Demo Expression Heatmap",
  scale_position = "right",
  legend_position = "top",
  titlesize = 14
)
```

`flatten_results` *Flatten a Nested List of Results into a Data Frame*

Description

Converts a nested list (where the first level is a method, the second level is a gene signature, and the third level is a data frame) into a single data frame. Additional columns for method and signature are added to the data frame.

Usage

```
flatten_results(nested_list)
```

Arguments

`nested_list`	A nested list with structure: `list(method = list(signature = data.frame(...)))`.

Value

A data frame combining all the nested data frames, with added columns `method` and `signature`.

`FPR_Simulation` *FPR Simulation Plot*

Description

This function simulates false positive rates (FPR) by generating simulated gene signatures and comparing the observed effect size values (Cohen's d or f) of the original signatures to those from simulated signatures. The effect size is computed using three scoring methods (`ssGSEA`, `logmedian`, and `ranking`), and the results are visualized as violin plots with overlaid observed values.

Usage

```
FPR_Simulation(
  data,
  metadata,
  original_signatures,
  Variable,
  gene_list = NULL,
  number_of_sims = 100,
  title = NULL,
  widthTitle = 30,
  titlesize = 12,
  pointSize = 2,
  labsize = 10,
  mode = c("none", "simple", "medium", "extensive"),
  ColorValues = NULL,
  ncol = NULL,
  nrow = NULL,
  p.adjust.method = "BH"
)
```

Arguments

`data`	A data frame or matrix of gene expression values (genes as rows, samples as columns).
`metadata`	A data frame containing metadata for the samples (columns of `data`).
`original_signatures`	A named list of gene signatures. Each element can be either: • A vector of gene names (unidirectional), or • A data frame with columns `"Gene"` and `"Signal"` for bidirectional signatures.
`Variable`	A column in `metadata` indicating the variable of interest for grouping or regression. This can be categorical or numeric.
`gene_list`	A character vector of gene names from which simulated signatures are generated by sampling. Default is all genes in `data`.
`number_of_sims`	Integer. Number of simulated gene signatures to generate per original signature.
`title`	Optional title for the overall plot.
`widthTitle`	Integer. Max width for wrapping the title text (default: 30).
`titlesize`	Numeric. Font size for the title text (default: 12).
`pointSize`	Numeric. Size of the points representing simulations (default: 2).
`labsize`	Numeric. Font size for axis labels (default: 10).
`mode`	A string specifying the level of detail for contrasts. Options are: • `"simple"`: Performs the minimal number of pairwise comparisons between individual group levels (e.g., A - B, A - C). Default.

- "medium": Includes comparisons between one group and the union of all other groups (e.g., A - (B + C + D)), enabling broader contrasts beyond simple pairs.
- "extensive": Allows for all possible algebraic combinations of group levels (e.g., (A + B) - (C + D)), supporting flexible and complex contrast definitions.
- "none": Comparing all levels of Variable (default)

ColorValues Named vector of colors for plot points, typically Original and Simulated. If NULL, default colors are used.

ncol Integer. Number of columns for arranging signature plots in a grid layout. If NULL, layout is auto-calculated.

nrow Integer. Number of rows for arranging signature plots in a grid layout. If NULL, layout is auto-calculated.

p.adjust.method
Character string specifying the method to use for multiple testing correction. Must be one of "BH" (Benjamini-Hochberg, default), "holm", "hommel", "bonferroni", "BY" (Benjamini-Yekutieli), "fdr", or "none". Passed to p.adjust.

Details

The function supports both categorical and numeric variables:

- For **categorical variables**, Cohen's *d* is used and contrasts are defined by the mode parameter, if mode!=none.
- For **numeric variables**, Cohen's *f* is used to quantify associations through linear modeling.

For each original gene signature, a number of simulated signatures are created by sampling genes from gene_list. Each simulated signature is scored using three methods, and its effect size is computed relative to the variable of interest. The resulting distributions are shown as violins, overlaid with the observed value from the original signature. A red dashed line marks the 95th percentile of the simulated distribution per method.

The function internally uses CohenD_allConditions() and CohenF_allConditions() depending on variable type.

Value

Invisibly returns a list containing:

plot A combined ggplotusing ggarrange; one violin plot is generated per signature and contrast. Observed values are highlighted and compared to the simulated distribution. Significance (adjusted p-value <= 0.05) is indicated by point shape.

data A list of data frames, one for each signature, containing the original and simulated effect sizes.

Examples

```
# Simulate gene expression matrix (genes as rows, samples as columns)
set.seed(444)
expr <- as.data.frame(matrix(abs(rnorm(60)), nrow = 6, ncol = 10))
rownames(expr) <- paste0("Gene", 1:6)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate sample metadata with a categorical variable
metadata <- data.frame(
  sample = colnames(expr),
  Condition = rep(c("A", "B"), each = 5),
  stringsAsFactors = FALSE
)

# Define two gene signatures (as character vectors)
signatures <- list(
  Sig1 = c("Gene1", "Gene2", "Gene3"),
  Sig2 = c("Gene4", "Gene5")
)

# Run FPR simulation (with fewer sims for speed in example)
FPR_Simulation(
  data = expr,
  metadata = metadata,
  original_signatures = signatures,
  Variable = "Condition",
  number_of_sims = 20,
  title = "FPR Simulation Example",
  pointSize = 3
)
```

---

generate_all_contrasts

Generate All Possible Unique Contrasts Between Groups

Description

This function creates statistical contrasts between levels of a categorical variable. Users can choose the level of complexity:

- `"simple"`: Pairwise comparisons (e.g., A - B).
- `"medium"`: Pairwise comparisons plus comparisons against the mean of other groups.
- `"extensive"`: All possible groupwise contrasts, ensuring balance in the number of terms on each side.

Usage

```
generate_all_contrasts(levels, mode = "simple")
```

Arguments

levels	A character vector of unique group levels.
mode	A string specifying the level of detail for contrasts. Options are "simple" (pairwise only), "medium" (pairwise + vs. mean of others), or "extensive" (all possible balanced groupwise contrasts). Default is "extensive".

Value

A character vector of unique contrast expressions.

Examples

```
## Not run:
levels <- c("A", "B", "C", "D")
generate_all_contrasts(levels, mode = "simple")    # Pairwise only
generate_all_contrasts(levels, mode = "medium")    # Pairwise + mean comparisons
generate_all_contrasts(levels, mode = "extensive") # All balanced contrasts

## End(Not run)
```

---

genesets_example *Example Gene Sets for Cellular Senescence*

Description

Example Gene Sets for Cellular Senescence

Usage

```
data(genesets_example)
```

Format

A named list of length 3:

Literature_Senescence Character vector of gene symbols. A small, curated gene set of commonly reported senescence markers, with directionality (+1 or -1).

REACTOME_Senescence Character vector of gene symbols. The REACTOME_CELLULAR_SENESCENCE from MSigDB database No directionality.

HernandezSegura A data frame with columns gene and direction. A gene set from Hernandez-Segura et al. (2017), with directionality (+1 or -1).

References

Hernandez-Segura A, de Jong TV, Melov S, Guryev V, Campisi J, Demaria M. Unmasking Transcriptional Heterogeneity in Senescent Cells. *Curr Biol.* 2017 Sep 11;27(17):2652-2660.e4. doi: 10.1016/j.cub.2017.07.033. Epub 2017 Aug 30. PMID: 28844647; PMCID: PMC5788810.

geneset_similarity *Plot Signature Similarity via Jaccard Index or Fisher's Odds Ratio*

Description

Visualizes similarity between user-defined gene signatures and either other user-defined signatures or MSigDB gene sets, using either the Jaccard index or Fisher's Odds Ratio. Produces a heatmap of pairwise similarity metrics.

Usage

```
geneset_similarity(
  signatures,
  other_user_signatures = NULL,
  collection = NULL,
  subcollection = NULL,
  metric = c("jaccard", "odds_ratio"),
  universe = NULL,
  or_threshold = 1,
  pval_threshold = 0.05,
  limits = NULL,
  title_size = 12,
  color = "#B44141",
  neutral_color = "white",
  cold_color = "#4173B4",
  title = NULL,
  jaccard_threshold = 0,
  msig_subset = NULL,
  width_text = 20,
  na_color = "grey90"
)
```

Arguments

signatures	A named list of character vectors representing reference gene signatures.
other_user_signatures	Optional. A named list of character vectors representing other user-defined signatures to compare against.
collection	Optional. MSigDB collection name (e.g., "H" for hallmark, "C2" for curated gene sets). Use msigdbr::msigdbr_collections() for the available options.
subcollection	Optional. Subcategory within an MSigDB collection (e.g., "CP:REACTOME"). Use msigdbr::msigdbr_collections() for the available options.
metric	Character. Either "jaccard" or "odds_ratio".
universe	Character vector. Background gene universe. Required for odds ratio.
or_threshold	(only if method == "odds_ratio" only) Numeric. Minimum Odds Ratio required for a gene set to be included in the plot. Default is 1.

`pval_threshold` (only if method == "odds_ratio" only) Numeric. Maximum adjusted p-value required for a gene set to be included in the plot. Default is 0.05.

`limits` Numeric vector of length 2. Limits for color scale. If `NULL`, is automatically set to c(0,1) for Jaccard or the range of OR for odds ratio.

`title_size` Integer specifying the font size for the plot title. Default is `12`.

`color` Character. The color for the maximum of the scale. Default is `red`.

- If `method = "jaccard"`, the scale goes from `neutral_color` to `color`.
- If `method = "odds_ratio"` and any OR >= 1, the scale ends at `color`.
- If `method = "odds_ratio"` and all OR <= 1, `color` is not used; instead, the scale runs from `cold_color` (minimum) to `neutral_color` (OR = 1, if present; otherwise `neutral_color` is the maximum).

`neutral_color` Character. The neutral reference color. Default is `white`.

- If `method = "jaccard"`, this is the minimum of the scale.
- If `method = "odds_ratio"` and any OR >= 1, this corresponds to OR = 1 if such values exist; otherwise it is the minimum of the scale.
- If `method = "odds_ratio"` and all OR <= 1, this corresponds to OR = 1 if such values exist; otherwise it is the maximum of the scale (with `cold_color` as the minimum).

`cold_color` Character. The color for values below OR = 1 (only used when `method = "odds_ratio"`). Default is `blue`.

- If `method = "odds_ratio"` and any OR < 1, the scale runs from `cold_color` (minimum) to `neutral_color` (OR = 1 if present; otherwise `neutral_color` is the maximum).
- Ignored if `method = "jaccard"` or if all OR >= 1.

`title` Optional. Custom title for the plot. If `NULL`, the title defaults to `"Signature Overlap"`.

`jaccard_threshold`
(only if method == "jaccard" only) Numeric. Minimum Jaccard index required for a gene set to be included in the plot. Default is `0`.

`msig_subset` Optional. Character vector of MSigDB gene set names to subset from the specified collection. Useful to restrict analysis to a specific set of pathways. If supplied, other filters will apply only to this subset. Use "collection = "all" to mix gene sets from different collections.

`width_text` Integer. Character wrap width for labels.

`na_color` Character. Color for NA values in the heatmap. Default is `"grey90"`.

Value

Invisibly returns a list containing:

`plot` The **ggplot2** object of the similarity heatmap.

`data` The data frame object containing the similarity scores per pair of gene sets.

Examples

```
# Create two simple gene signatures
sig1 <- c("TP53", "BRCA1", "MYC", "EGFR", "CDK2")
sig2 <- c("ATXN2", "FUS", "MTOR", "CASP3")
signatures <- list(SignatureA = sig1, SignatureB = sig2)

# Compare the signatures using the Jaccard index
plt <- geneset_similarity(
  signatures = signatures,
  metric = "jaccard",
  collection = "H",
  jaccard_threshold = 0.01
)

# Print the plot (will show a small heatmap)
print(plt)


# Odds ratio example (requires universe)
gene_universe <- unique(c(
  sig1, sig2,
  msigdbr::msigdbr(species = "Homo sapiens", category = "C2")$gene_symbol
))

plt_or <- geneset_similarity(
  signatures = signatures,
  metric = "odds_ratio",
  universe = gene_universe,
  collection = "H"
)
print(plt_or)
```

---

getRanking *Get Gene Expression Ranking*

Description

Computes the rank sum of a given gene set within a sample based on its expression level.

Usage

```
getRanking(data, sample, geneset)
```

Arguments

data	A data frame where rows represent genes, columns represent samples, and values correspond to expression levels.
sample	A character string specifying the sample name (column in data).
geneset	A vector of gene names to be ranked.

Details

- The function orders gene expression levels from lowest to highest.
- It then determines the rank of each gene in `geneset` and returns the sum of these ranks.
- If some genes are missing, they are omitted from the ranking calculation.

Value

The sum of the ranks of the genes found in the sample.

Examples

```
## Not run:
# Example dataset with 5 genes and 3 samples
set.seed(123)
data <- as.data.frame(matrix(runif(15, 1, 100), nrow = 5, ncol = 3))
rownames(data) <- paste0("Gene_", 1:5)
colnames(data) <- paste0("Sample_", 1:3)

# Define gene set
geneset <- c("Gene_1", "Gene_3", "Gene_5")

# Compute ranking for Sample_1
rank_score <- getRanking(data, "Sample_1", geneset)
print(rank_score)

## End(Not run)
```

GSEA_VariableAssociation

GSEA Variable Association

Description

This function assesses the association between gene expression (or another molecular score) and metadata variables using differential expression (DE) analysis and Gene Set Enrichment Analysis (GSEA). It generates all possible contrasts for categorical variables and uses linear modeling for continuous variables.

Usage

```
GSEA_VariableAssociation(
  data,
  metadata,
  cols,
  stat = NULL,
  mode = c("simple", "medium", "extensive"),
  gene_set,
```

```
  nonsignif_color = "grey",
  signif_color = "red",
  saturation_value = NULL,
  sig_threshold = 0.05,
  widthlabels = 18,
  labsize = 10,
  titlesize = 14,
  pointSize = 5,
  ignore_NAs = FALSE,
  printplt = TRUE,
  p.adjust.method = "BH"
)
```

Arguments

`data`	A matrix or data frame containing gene expression data, where rows represent genes and columns represent samples.
`metadata`	A data frame containing sample metadata with at least one column corresponding to the variables of interest.
`cols`	A character vector specifying the metadata columns (variables) to analyse.
`stat`	Optional. The statistic to use for ranking genes before GSEA. If `NULL`, it is automatically determined based on the gene set: • `"B"` for gene sets with **no known direction** (vectors). • `"t"` for **unidirectional** or **bidirectional** gene sets (data frames). • If provided, this argument overrides the automatic selection.
`mode`	A string specifying the level of detail for contrasts. Options are: • `"simple"`: Performs the minimal number of pairwise comparisons between individual group levels (e.g., A - B, A - C). Default. • `"medium"`: Includes comparisons between one group and the union of all other groups (e.g., A - (B + C + D)), enabling broader contrasts beyond simple pairs. • `"extensive"`: Allows for all possible algebraic combinations of group levels (e.g., (A + B) - (C + D)), supporting flexible and complex contrast definitions.
`gene_set`	A named list defining the gene sets for GSEA. **(Required)** • If using **unidirectional** gene sets, provide a list where each element is a vector of gene names representing a signature. • If using **bidirectional** gene sets, provide a list where each element is a data frame: • The **first column** should contain gene names. • The **second column** should indicate the expected direction of enrichment (`1` for upregulated, `-1` for downregulated).
`nonsignif_color`	A string specifying the color for the middle of the adjusted p-value gradient. Default is `"white"`. Lower limit correspond to the value of `sig_threshold`.

signif_color A string specifying the color for the low end of the adjusted p-value gradient until the value chosen for significance (sig_threshold). Default is "red".

saturation_value A numeric value specifying the lower limit of the adjusted p-value gradient, below which the color will correspond to signif_color. Default is the results' minimum, unless that value is above the sig_threshold; in that case, it is 0.001.

sig_threshold A numeric value specifying the threshold for significance visualization in the plot. Default: 0.05.

widthlabels An integer controlling the maximum width of contrast labels before text wrapping. Default: 18.

labsize An integer controlling the axis text size in the plot. Default: 10.

titlesize An integer specifying the plot title size. Default: 14.

pointSize Numeric. The size of points in the lollipop plot (default is 5).

ignore_NAs Boolean (default: FALSE). Whether to ignore NAs in the metadata when fitting the linear model. If TRUE, rows with any NAs will be removed before analysis, leading to a loss of data to be fitted in the model.

printplt Boolean specifying if plot is to be printed. Default: TRUE.

p.adjust.method Character string specifying the method to use for multiple testing correction. Must be one of "BH" (Benjamini-Hochberg, default), "holm", "hommel", "bonferroni", "BY" (Benjamini-Yekutieli), "fdr", or "none". Passed to p.adjust.

Value

A list with two elements:

- data: A data frame containing the GSEA results, including normalized enrichment scores (NES), adjusted p-values, and contrasts.
- plot: A ggplot2 object visualizing the GSEA results as a lollipop plot.

Heatmap_Cohen *Generate Heatmaps for Cohen's d Effect Sizes using ggplot2*

Description

This function computes Cohen's d effect sizes and corresponding p-values for multiple gene signatures and produces individual heatmaps. Each heatmap displays cell text showing the Cohen's d value along with its p-value. The heatmaps are then arranged in a grid layout.

Usage

```
Heatmap_Cohen(
  cohenlist,
  nrow = NULL,
  ncol = NULL,
  limits = NULL,
  widthTitle = 22,
  titlesize = 12,
  ColorValues = NULL,
  title = NULL
)
```

Arguments

cohenlist	A named list where each element corresponds to a gene signature. Output of CohenD_allConditions. Each signature element is a list with three components: **CohenD** A data frame where rows are methods and columns are group contrasts (formatted as "Group1:Group2"), containing the computed Cohen\'s d effect sizes. **PValue** A data frame with the same structure as CohenD containing the corresponding p-values. **padj** A data frame with the same structure as PValue containing the corresponding p-values corrected using the BH method, for all signatures and contrasts, and by method.
nrow	Optional. An integer specifying the number of rows in the heatmap grid. If NULL, the number of rows is computed automatically.
ncol	Optional. An integer specifying the number of columns in the heatmap grid. If NULL, the number of columns is computed automatically.
limits	Optional. A numeric vector of length 2 specifying the color scale limits (e.g., c(min, max)). If NULL, the limits are determined from the data.
widthTitle	An integer specifying the width used for wrapping gene set signature names in the heatmap titles. Default is 22.
titlesize	An integer specifying the text size for each of the heatmap titles. Default is 12.
ColorValues	A character vector specifying the colors for the gradient fill in the heatmaps. Default is c("#F9F4AE", "#B44141").
title	Title for the grid of plots.

Details

The function first calculates Cohen\'s d effect sizes and corresponding p-values for each gene signature using CohenD_allConditions (assumed to be defined elsewhere in the package). The resulting matrices are converted to a long format so that each cell in the heatmap can display the Cohen\'s d value and its associated p-value (formatted as Cohen\'s d (p-value)).

The heatmaps are then adjusted to display axis text and ticks only for the left-most column and bottom row, and combined into a grid layout. If neither nrow nor ncol are specified, the layout is automatically determined to best approximate a square grid.

Value

A list with two elements:

plt A combined heatmap arranged in a grid using `ggpubr::ggarrange`.

data A list containing the Cohen\'s d effect sizes and p-values for each gene signature, as computed by `CohenD_allConditions`.

See Also

`CohenD_allConditions`, `CohenF_allConditions`

`identify_variable_type` *Identify Variable Types*

Description

Determines the type of each variable in a given data frame. Variables are classified as "Numeric", "Categorical Bin" (binary categorical), or "Categorical Multi" (multi-level categorical). Warnings are issued if categorical variables have more than 10 unique values.

Usage

```
identify_variable_type(df, cols = NULL)
```

Arguments

`df`	A data frame containing the variables to classify.
`cols`	A character vector of column names to consider.

Value

A named character vector where names correspond to column names and values indicate the variable type: "Numeric", "Categorical Bin", or "Categorical Multi".

Examples

```
## Not run:
df <- data.frame(
  age = c(25, 30, 35, 40),
  gender = c("Male", "Female", "Female", "Male"),
  score = c(80, 85, 90, 95)
)
identify_variable_type(df)

## End(Not run)
```

IndividualGenes_Violins *Generate Violin Plots for Individual Genes*

Description

This function creates violin plots of gene expression data with jittered points and optional faceting. It allows visualization of individual gene expression distributions across sample groups.

Usage

```
IndividualGenes_Violins(
  data,
  metadata = NULL,
  genes,
  GroupingVariable,
  plot = TRUE,
  ncol = NULL,
  nrow = NULL,
  divide = NULL,
  invert_divide = FALSE,
  ColorValues = NULL,
  pointSize = 2,
  ColorVariable = NULL,
  title = NULL,
  widthTitle = 16,
  y_limits = NULL,
  legend_nrow = NULL,
  xlab = NULL,
  colorlab = NULL
)
```

Arguments

data	A data frame containing gene expression values with row names as gene names and column names as sample IDs. **(Required)**
metadata	An optional data frame containing sample metadata. The first column must match the sample IDs from data. **(Optional)**
genes	A character vector of gene names to be plotted. **(Required)**
GroupingVariable	A character string specifying the column in metadata used for grouping samples on the x-axis. **(Required)**
plot	A logical value indicating whether to print the plot. If FALSE, only the output list is returned. Default is TRUE. **(Optional)**

ncol	An optional numeric value specifying the number of columns in the facet grid. If not provided, it is computed automatically. Only applicable if divide is NULL. **(Optional)**
nrow	An optional numeric value specifying the number of rows in the facet grid. If not provided, it is computed automatically. Only applicable if divide is NULL. **(Optional)**
divide	An optional character string specifying a column in metadata to be used for facetting, besides faceting by genes. **(Optional)**
invert_divide	A logical value indicating whether to invert the facet layout, when divide is being used. Default is FALSE, corresponding to genes in the rows. **(Optional)**
ColorValues	An optional named vector mapping unique values of ColorVariable to specific colors. If NULL, a default Brewer palette ("Paired") is used. **(Optional)**
pointSize	A numeric value specifying the size of the points in the plot. Default is 2. **(Optional)**
ColorVariable	A character string specifying a metadata column used for coloring points. Default is NULL. **(Optional)**
title	A character string specifying the title of the plot. Default is NULL. **(Optional)**
widthTitle	A numeric value specifying the maximum width of the title before inserting line breaks. **(Optional)**
y_limits	A numeric vector of length 2 specifying the limits of the y-axis. If NULL (default), the y-axis is adjusted automatically. **(Optional)**
legend_nrow	A numeric value specifying the number of rows in the legend. Default is NULL. **(Optional)**
xlab	A character string specifying the x-axis label. If NULL, it defaults to GroupingVariable. **(Optional)**
colorlab	A character string specifying the legend title for colors. Default is an empty string. **(Optional)**

Details

The function processes the gene expression data, filters for the specified genes, and transforms expression values using log2(). A violin plot with jittered points is generated using ggplot2. A median summary is added as a crossbar. If divide is provided, facets are created using ggh4x::facet_grid2(). Color customization is available via ColorVariable and ColorValues.

Value

A list containing:

plot	A ggplot2 object representing the facetted violin plots.
data	A data frame used for plotting, including transformed expression values (log2) and metadata.

Examples

```
# Example dataset
data <- data.frame(
  A = c(10, 20, 30),
  B = c(5, 15, 25),
  C = c(2, 12, 22)
)
rownames(data) <- c("Gene1", "Gene2", "Gene3")

metadata <- data.frame(
  sample = c("A", "B", "C"),
  Group = c("Control", "Treatment", "Control")
)

genes <- c("Gene1", "Gene2")

IndividualGenes_Violins(data, metadata, genes, "Group")
```

`markeR` *markeR: An R Toolkit for Evaluating Gene Signatures as Phenotypic Markers*

Description

The **markeR** package provides tools for evaluating gene signatures across phenotypes in transcriptomics datasets (especially bulk RNA-seq). It implements scoring and enrichment approaches, alongside intuitive visualizations and performance metrics.

Key features:

- Score-based signature quantification (e.g., median-centered, ssGSEA, ranking)
- Enrichment analysis using GSEA
- Visualization of gene expression, scores, and enrichment results
- Assessment of gene set similarity

Author(s)

Maintainer: Rita Martins-Silva <rita.silva@medicina.ulisboa.pt> (ORCID)

Authors:

- Alexandre Kaizeler (ORCID) [contributor]
- Nuno Luís Barbosa-Morais (ORCID) [lead, thesis advisor]

See Also

For more information on using the markeR package, check out the markeR Documentation. You can also visit the GitHub Repository for the latest updates and source code.

`metadata_example` *Metadata for Marthandan et al. (2016) RNA-Seq Study*

Description

A data frame containing metadata for samples from the Marthandan et al. (2016) study (GEO code GSE63577).

Usage

```
data(metadata_example)
```

Format

A data frame with 45 rows and 6 columns:

sampleID Unique sample identifier.

DatasetID Identifier for the dataset (e.g., "Marthandan2016").

CellType Cell type, e.g. "Fibroblast".

Condition Experimental condition ("Senescent" or "Proliferative").

SenescentType Mechanism of senescence (e.g., "Telomere shortening" for senescent samples, "none" for proliferative).

Treatment Treatment or age descriptor (e.g., "PD72 (Replicative senescence)" for senescent samples, "young" for proliferative).

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63577

References

Marthandan S, Priebe S, Baumgart M, Groth M et al. Similarities in Gene Expression Profiles during In Vitro Aging of Primary Human Embryonic Lung and Foreskin Fibroblasts. Biomed Res Int 2015;2015:731938. PMID: 26339636

Marthandan S, Baumgart M, Priebe S, Groth M et al. Conserved Senescence Associated Genes and Pathways in Primary Human Fibroblasts Detected by RNA-Seq. PLoS One 2016;11(5):e0154531. PMID: 27140416

plotCombinedGSEA *Plot Combined GSEA Results*

Description

This function creates a scatter plot visualizing multiple GSEA (Gene Set Enrichment Analysis) results across different contrasts. Each point represents a pathway, where:

- The x-axis corresponds to the Normalized Enrichment Score (NES).
- The y-axis corresponds to the significance level (-log10 adjusted p-value).
- The color represents different pathways.
- The shape represents different contrasts.
- A dashed horizontal line marks the chosen significance threshold.

Usage

```
plotCombinedGSEA(
  GSEA_results,
  sig_threshold = 0.05,
  PointSize = 4,
  widthlegend = 16
)
```

Arguments

GSEA_results	A named list of data frames, where each data frame contains GSEA results for a contrast. Each data frame should have the columns: NES (Normalized Enrichment Score), padj (adjusted p-value), and pathway (pathway name). Output from runGSEA.
sig_threshold	Numeric, default = 0.05. Adjusted p-value threshold for significance. A dashed horizontal line is drawn at this threshold.
PointSize	Numeric, default = 4. Size of the plotted points.
widthlegend	Numeric, default = 16. Controls the width of pathway labels in the legend.

Value

A ggplot2 object displaying the combined GSEA results.

Examples

```
# Example GSEA results (mock data)
GSEA_results <- list(
"Contrast1" = data.frame(
NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 1:3),
```

```
  stat_used = c("t", "B", "B")
),
"Contrast2" = data.frame(
  NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 4:6),
  stat_used = c("t", "B", "B")
)
)


# Generate the plot
plotCombinedGSEA(GSEA_results, sig_threshold = 0.05, PointSize = 4)
```

`plotGSEAenrichment` *Plot GSEA Enrichment Results*

Description

This function generates enrichment plots for gene sets using the `fgsea::plotEnrichment()` function. It supports both individual plots (returned as a list) and a grid layout using `ggpubr::ggarrange()`.

Usage

```
plotGSEAenrichment(
  GSEA_results,
  DEGList,
  gene_sets,
  widthTitle = 24,
  grid = FALSE,
  nrow = NULL,
  ncol = NULL,
  titlesize = 12
)
```

Arguments

`GSEA_results`	A named list of data frames containing GSEA results for each contrast. Each data frame should have a column named `pathway` specifying the gene set, and columns `NES` and `padj` for results. Output from `runGSEA`.
`DEGList`	A named list of data frames containing differentially expressed genes (DEGs) for each contrast. Each data frame must include a column named `t` with t-statistics for ranking genes. Output from `calculateDE`.
`gene_sets`	A named list of gene sets, where each entry is either: • A vector of gene names (unidirectional gene set) • A data frame with two columns: gene names and direction (+1 for enriched and -1 for depleted).

widthTitle	Integer. The maximum width (in characters) for wrapping plot titles. Default is 24.
grid	Logical. If TRUE, plots are arranged in a grid using ggpubr::ggarrange(). Default is FALSE.
nrow	Integer. Number of rows for the grid layout (used only if grid = TRUE). If NULL, it is auto-calculated.
ncol	Integer. Number of columns for the grid layout (used only if grid = TRUE). If NULL, it is auto-calculated.
titlesize	Integer. Font size for plot titles. Default is 12.

Value

If grid = FALSE, returns a named list of ggplot objects (each plot corresponding to a contrast-signature pair). If grid = TRUE, returns a single ggplot object with all enrichment plots arranged in a grid.

Examples

```
# Example GSEA results (mock data, missing columns if running by runGSEA)

GSEA_results <- list(
"Contrast1" = data.frame(
NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 1:3),
  stat_used = c("t", "B", "B")
),
"Contrast2" = data.frame(
  NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 4:6),
  stat_used = c("t", "B", "B")
)
)


# Generate the plot
plot <- plotCombinedGSEA(GSEA_results, sig_threshold = 0.05, PointSize = 7)
print(plot)
```

---

plotNESlollipop *Create a Lollipop Plot for GSEA Results*

Description

This function generates a lollipop plot to visualize Gene Set Enrichment Analysis (GSEA) results. Pathways are shown on the y-axis, while the Normalized Enrichment Score (NES) is shown on the x-axis. The color of the lollipops represents the adjusted p-values (padj), with a custom color gradient. It supports multiple contrasts and can combine individual plots into a grid layout.

Usage

```
plotNESlollipop(
  GSEA_results,
  signif_color = "red",
  nonsignif_color = "white",
  sig_threshold = 0.05,
  saturation_value = NULL,
  pointSize = 5,
  grid = FALSE,
  nrow = NULL,
  ncol = NULL,
  widthlabels = 18,
  title = NULL,
  titlesize = 12
)
```

Arguments

GSEA_results	A named list of data frames, each containing the GSEA results for a specific contrast. Output from runGSEA. Each data frame must include the following columns: **pathway** A character vector of pathway names. **NES** A numeric vector of Normalized Enrichment Scores for the pathways. **padj** A numeric vector of adjusted p-values for the pathways.
signif_color	A string specifying the color for the low end of the adjusted p-value gradient until the value chosen for significance (sig_threshold). Default is "red".
nonsignif_color	A string specifying the color for the middle of the adjusted p-value gradient. Default is "white". Lower limit correspond to the value of sig_threshold.
sig_threshold	A numeric value that sets the midpoint for the color scale. Typically used for the significance threshold. Default is 0.05.
saturation_value	A numeric value specifying the lower limit of the adjusted p-value gradient, below which the color will correspond to signif_color. Default is the results' minimum, unless that value is above the sig_threshold; in that case, it is 0.001.
pointSize	Numeric. The size of points in the lollipop plot (default is 5).
grid	A logical value indicating whether to arrange individual plots into a grid layout. If TRUE, the function combines all plots into a grid. Default is FALSE.
nrow	A numeric value specifying the number of rows to arrange the plots into if grid = TRUE. If NULL, the function calculates this automatically. Default is NULL.
ncol	A numeric value specifying the number of columns to arrange the plots into if grid = TRUE. If NULL, the function calculates this automatically. Default is NULL.
widthlabels	A numeric value specifying the maximum width for pathway names. If a pathway name exceeds this width, it will be wrapped to fit. Default is 18.

title	A character string for the title of the combined plot (used only when grid = TRUE). Default is NULL.
titlesize	A numeric value specifying the font size for the title (used only when grid = TRUE). Default is 12.

Details

The function creates a lollipop plot for each contrast in the GSEA_results list. Each plot includes:

- A vertical segment for each pathway, where the x-coordinate represents the NES and the y-coordinate represents the pathway.
- A colored point at the end of each segment, where the color represents the adjusted p-value (padj), mapped using a custom color gradient.

If a pathway's padj value exceeds the maximum value in padj_limit, the corresponding pathway is colored using the high_color. Additionally, missing values (NA) for padj are assigned the high_color by setting na.value = high_color. Pathway names are wrapped using the wrap_title function to fit within the specified width (widthlabels).

Value

If grid = FALSE, a list of ggplot objects is returned, each corresponding to a contrast. If grid = TRUE, a single ggplot object is returned, representing the combined grid of plots.

Examples

```
# Example GSEA results (mock data, missing columns if running by runGSEA)

GSEA_results <- list(
"Contrast1" = data.frame(
NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 1:3),
  stat_used = c("t", "B", "B")
),
"Contrast2" = data.frame(
  NES = rnorm(3),
  padj = runif(3),
  pathway = paste("Pathway", 4:6),
  stat_used = c("t", "B", "B")
)
)


# Generate individual plots without grid
plot_list <- plotNESlollipop(GSEA_results)
plot_list

# Generate combined grid of plots with custom title
combined_plot <- plotNESlollipop(GSEA_results, grid = TRUE,
title = "GSEA Results Overview", titlesize = 14)
```

```
combined_plot
```

---

plotPCA *Principal Component Analysis (PCA) Plot*

Description

This function performs PCA on a given dataset and visualizes the results using ggplot2. It allows users to specify genes of interest, customize scaling and centering, and color points based on a metadata variable.

Usage

```
plotPCA(
  data,
  metadata = NULL,
  genes = NULL,
  scale = FALSE,
  center = TRUE,
  PCs = list(c(1, 2)),
  ColorVariable = NULL,
  ColorValues = NULL,
  pointSize = 5,
  legend_nrow = 2,
  legend_position = c("bottom", "top", "right", "left"),
  ncol = NULL,
  nrow = NULL
)
```

Arguments

`data`	A numeric matrix or data frame where rows represent genes and columns represent samples.
`metadata`	A data frame containing sample metadata. The first column should contain sample names. Default is NULL.
`genes`	A character vector specifying genes to be included in the PCA. Default is NULL (uses all genes).
`scale`	Logical; if TRUE, variables are scaled before PCA. Default is FALSE.
`center`	Logical; if TRUE, variables are centered before PCA. Default is TRUE.
`PCs`	A list specifying which principal components (PCs) to plot. Default is `list(c(1,2))`.
`ColorVariable`	A character string specifying the metadata column used for coloring points. Default is NULL.
`ColorValues`	A vector specifying custom colors for groups in `ColorVariable`. Default is NULL.

pointSize	Numeric; sets the size of points in the plot. Default is 5.
legend_nrow	Integer; number of rows in the legend. Default is 2.
legend_position	Character; position of the legend ("bottom", "top", "right", "left"). Default is "bottom".
ncol	Integer; number of columns in the arranged PCA plots. Default is determined automatically.
nrow	Integer; number of rows in the arranged PCA plots. Default is determined automatically.

Details

The function performs PCA using `prcomp()` and visualizes the results using `ggplot2`. If a metadata data frame is provided, it ensures the sample order matches between data and metadata.

Value

A list with two elements:

- `plt`: A ggplot2 or ggarrange object displaying the PCA plot.
- `data`: A data frame containing PCA-transformed values and sample metadata (if available).

Examples

```
# Example dataset
set.seed(123)
data <- abs(matrix(rnorm(1000), nrow=50, ncol=20))
colnames(data) <- paste0("Sample", 1:20)
rownames(data) <- paste0("Gene", 1:50)

metadata <- data.frame(Sample = colnames(data),
                       Group = rep(c("A", "B"), each = 10))

# Basic PCA plot
plotPCA(data, metadata, ColorVariable = "Group", pointSize = 10)

set.seed(42)
n_genes <- 100
n_samples <- 10

# Group A: samples 1-5, lower mean
group_A <- matrix(rlnorm(n_genes * 5, meanlog = 1, sdlog = 0.3), nrow = n_genes)

# Group B: samples 6-10, higher mean
group_B <- matrix(rlnorm(n_genes * 5, meanlog = 2, sdlog = 0.3), nrow = n_genes)

# Combine
data <- cbind(group_A, group_B)
colnames(data) <- paste0("Sample", 1:n_samples)
rownames(data) <- paste0("Gene", 1:n_genes)
```

```
# Metadata
metadata <- data.frame(Sample = colnames(data),
                       Group = rep(c("A", "B"), each = 5))

# Plot PCA
plotPCA(data, metadata, ColorVariable = "Group", pointSize = 10)
```

PlotScores *Plot gene signature scores using various methods.*

Description

Computes and visualizes gene signature scores using one or more methods, returning plots such as scatter plots, violin plots, heatmaps, or volcano plots depending on inputs.

Usage

```
PlotScores(
  data,
  metadata,
  gene_sets,
  method = c("ssGSEA", "logmedian", "ranking", "all"),
  ColorVariable = NULL,
  Variable = NULL,
  ColorValues = NULL,
  ConnectGroups = FALSE,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  widthTitle = 20,
  titlesize = 12,
  limits = NULL,
  legend_nrow = NULL,
  pointSize = 4,
  xlab = NULL,
  labsize = 10,
  compute_cohen = TRUE,
  cond_cohend = NULL,
  pvalcalc = FALSE,
  mode = c("simple", "medium", "extensive"),
  widthlegend = 22,
  sig_threshold = 0.05,
  cohen_threshold = 0.5,
  colorPalette = "Set3",
  cor = c("pearson", "spearman", "kendall"),
  p.adjust.method = "BH"
)
```

Arguments

data	A data frame of Normalised (non-transformed) counts where each row is a gene and each column is a sample. Row names should contain gene names, and column names should contain sample identifiers. **(Required)**
metadata	A data frame with sample-level attributes. Each row corresponds to a sample, with the first column containing sample IDs that match `colnames(data)`. **Required if** `method = "all"` **or if metadata-derived groupings or colors are used.**
gene_sets	A named list of gene sets to score. For unidirectional gene sets, provide a list of character vectors. For bidirectional gene sets, provide a list of data frames with two columns: gene names and direction (1 = up, -1 = down). **(Required)**
method	Scoring method to use. One of `"ssGSEA"`, `"logmedian"`, `"ranking"`, or `"all"` (default = `"logmedian"`). The `"all"` option triggers a full analysis returning both heatmap and volcano plots. Other values return single-score plots depending on `Variable` type.
ColorVariable	Name of a metadata column to color points by. Used in **single-method mode** (`"ssGSEA"`, etc.). Ignored in `"all"` mode.
Variable	Metadata column to define groups or numeric comparisons. This is **required if** `method = "all"` (used to compute and compare effect sizes). If `NULL` and `method != "all"`, density plots of each signature score across samples are shown (no grouping or comparison).
ColorValues	Optional. A named vector or list of colors used to control the coloring of plot elements across different methods and variable types. Behavior depends on the combination of `method` and `Variable`. If `method != "all"`, then: • If `Variable` is `NULL`, a single color will be applied in density plots (default: `"#ECBD78"`). • If `Variable` is categorical, a named vector should map each level of `Variable` (or `ColorVariable`) to a specific color. This overrides the palette specified by `colorPalette`. • If `Variable` is numeric, a single color is applied to all points in the scatter plot (default: `"#5264B6"`). If `method == "all"`, then: • `ColorValues` can be a named list with two elements: – `heatmap`: a vector of two colors used as a diverging scale for the heatmap of effect sizes (default: `c("#F9F4AE", "#B44141")`). – `volcano`: a named vector of colors used for labeling or grouping gene signatures (e.g., in the volcano plot). If not provided, defaults will be used for both components. In all cases, `ColorValues` takes precedence over the default `colorPalette` setting if specified.
ConnectGroups	Logical. If `TRUE`, connects points by sample ID across conditions (used for categorical variables and `method != "all"`).

ncol	Number of columns for facet layout (used in both heatmaps and score plots).
nrow	Number of rows for facet layout (used in both heatmaps and score plots).
title	Plot title (optional).
widthTitle	Width allocated for title (affects alignment).
titlesize	Font size for plot title.
limits	Y-axis limits (numeric vector of length 2).
legend_nrow	Number of rows for plot legend (used in single-method plots).
pointSize	Numeric. Size of points in **score plots** (violin or scatter), used when plotting individual sample scores for both categorical and numeric variables, including when method = "all".
xlab	Label for x-axis (optional; defaults to Variable).
labsize	Font size for axis and facet labels.
compute_cohen	Logical. Whether to compute Cohen's effect sizes in **score plots** (method != "all"). This only applies when method != "all"; ignored otherwise. If the variable is categorical and cond_cohend is specified, computes **Cohen's d** for the specified comparison. If the variable is categorical and cond_cohend is not specified, it computes **Cohen's d** if there are exactly two groups, or **Cohen's f** if there are more than two groups. If the variable is numeric, computes **Cohen's f** regardless of cond_cohend.
cond_cohend	Optional. List of length 2 with the two groups being used to compute effect size. The values in each entry should be levels of Variable. Used with compute_cohen = TRUE.
pvalcalc	Logical. If TRUE, computes p-values between groups.
mode	A string specifying the contrast mode when method = "all". Determines the complexity and breadth of comparisons performed between group levels. Options are: "simple" performs the minimal number of pairwise comparisons between individual group levels (e.g., A - B, A - C). This is the default. "medium" includes comparisons between one group and the union of all other groups (e.g., A - (B + C + D)), enabling broader contrasts beyond simple pairs. "extensive" allows for all possible algebraic combinations of group levels (e.g., (A + B) - (C + D)), supporting flexible and complex contrast definitions.
widthlegend	Width of the legend in **volcano plots** (used only if method = "all") and violin score plots.
sig_threshold	P-value cutoff shown as a **guide line** in volcano plots. Only applies when method = "all".
cohen_threshold	Effect size threshold shown as a **guide line** in volcano plots. Used only when method = "all".
colorPalette	Name of an RColorBrewer palette used to assign colors in plots. Applies to all methods. Default is "Set3". If ColorValues is provided, it overrides this palette. If Variable is NULL and method != "all" (i.e., for density plots), a default color "#ECBD78" is used. If method = "all" (i.e., for heatmaps and volcano plots), a default diverging color scale is used: c("#F9F4AE", "#B44141"), unless ColorValues is manually specified.

cor
: Correlation method for numeric variables. One of "pearson" (default), "spearman", or "kendall". Only applies when the variable is numeric and method != "all".

p.adjust.method
: Character string specifying the method to use for multiple testing correction. Must be one of "BH" (Benjamini-Hochberg, default), "holm", "hommel", "bonferroni", "BY" (Benjamini-Yekutieli), "fdr", or "none". Passed to p.adjust. Only if method == "all".

Details

Behavior based on method:

For "all", the function requires metadata and Variable. It computes scores using all available methods and returns a heatmap of Cohen's effect sizes and a volcano plot showing effect size vs p-value across gene signatures. Additional parameters include mode to define how contrasts between groups are constructed, sig_threshold and cohend_threshold which add guide dashed lines to the volcano plot (do not affect point coloring), widthlegend controlling width of the volcano plot legend, and pointSize controlling dot size for signature points in the volcano plot. ColorValues can be a named list with heatmap (two-color gradient for effect sizes) and signatures (named vector of colors for gene signatures in the volcano plot).

For "ssGSEA", "logmedian", or "ranking", the type of Variable determines the plot. If categorical, violin plots with optional group comparisons are produced. If numeric, scatter plots with correlation are produced. If Variable is NULL, density plots for each signature across all samples are produced. Additional arguments include ColorVariable and ColorValues for coloring control, colorPalette (overridden by ColorValues if present), ConnectGroups to link samples by ID for categorical Variable, cor to specify correlation method for numeric Variable, pvalcalc to enable group-wise p-value calculations for categorical variables, compute_cohen to calculate effect sizes when applicable, and cond_cohend to focus Cohen's d calculation on a specific comparison.

Behavior based on Variable type:

If Variable is numeric, scatter plots are output (in single-method mode) with computed correlation (cor). Parameters compute_cohen, cond_cohend, and pvalcalc are ignored. Color is uniform (default: "#5264B6") unless overridden via ColorValues. Cohen's f effect size estimation (compute_cohen = TRUE) and significance if pvalcalc is TRUE.

If Variable is categorical, violin plots are output (in single-method mode) supporting p-value comparisons (pvalcalc = TRUE), optional connection lines (ConnectGroups = TRUE), and Cohen's effect size estimation (compute_cohen = TRUE) with significance (pvalcalc is TRUE). If cond_cohend is specified, computes Cohen's d for that comparison. If not specified, computes Cohen's d if 2 groups or Cohen's f if more than 2 groups. Colors are matched to factor levels using ColorValues or colorPalette.

If Variable is NULL and method != "all", density plots of signature scores are produced. A single fill color is used (default "#ECBD78" or from ColorValues).

Value

Depending on method:

If method = "all", returns a list with heatmap and volcano ggplot objects.

If method is a single method, returns a single ggplot object (scatter or violin plot depending on variable type).

Examples

```
# Simulate positive gene expression data (genes as rows, samples as columns)
set.seed(42)
expr <- as.data.frame(matrix(rexp(60, rate = 0.2), nrow = 6, ncol = 10))  # values > 0
rownames(expr) <- paste0("Gene", 1:6)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate metadata for samples with categorical and numeric variables
metadata <- data.frame(
  sample = colnames(expr),
  Group = rep(c("A", "B"), each = 5),
  Age = seq(30, 75, length.out = 10)
)

# Define two simple gene sets
gene_sets <- list(
  Signature1 = c("Gene1", "Gene2", "Gene3"),
  Signature2 = c("Gene4", "Gene5", "Gene6")
)

# 1. Categorical variable: Violin plot (logmedian)
PlotScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
  method = "logmedian",
  Variable = "Group"
)

# 2. Numeric variable: Scatter plot (logmedian)
PlotScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
  method = "logmedian",
  Variable = "Age"
)

# 3. No variable: Density plot (logmedian)
PlotScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
  method = "logmedian"
)

# 4. All methods, categorical variable: Heatmap and volcano
# (Returns a list with $heatmap and $volcano elements)
all_plots <- PlotScores(
  data = expr,
  metadata = metadata,
  gene_sets = gene_sets,
```

```
  method = "all",
  Variable = "Group"
)
# Print the heatmap and volcano plot if desired
print(all_plots$heatmap)
print(all_plots$volcano)
```

PlotScores_Categorical *Plot Gene Set Scores by Group or Continuous Variable*

Description

This function computes and visualizes gene set enrichment scores using various methods, optionally comparing across groups or numeric variables. It supports categorical and numeric comparisons, statistical testing, Cohen's d effect sizes, and visualizations such as heatmaps and volcano plots.

Usage

```
PlotScores_Categorical(
  data,
  metadata,
  gene_sets,
  method = c("ssGSEA", "logmedian", "ranking"),
  ColorVariable = NULL,
  GroupingVariable = NULL,
  ColorValues = NULL,
  ConnectGroups = FALSE,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  widthTitle = 10,
  titlesize = 12,
  limits = NULL,
  legend_nrow = NULL,
  pointSize = 2,
  xlab = NULL,
  labsize = 10,
  compute_cohen = TRUE,
  cond_cohend = NULL,
  pvalcalc = FALSE,
  mode = c("simple", "medium", "extensive"),
  widthlegend = 22,
  cohen_threshold = 0.6,
  colorPalette = "Set3"
)
```

Arguments

`data`	A data frame of Normalised (non-transformed) counts where each row is a gene and each column is a sample. Row names should contain gene names, and column names should contain sample identifiers. **(Required)**
`metadata`	A data frame describing the attributes of each sample, where each row corresponds to a sample and each column to an attribute. The first column should contain sample identifiers (i.e., the column names of `data`). **(Required if method = "all")**
`gene_sets`	Gene set input. **(Required)** • **Unidirectional gene sets**: Provide a named list where each element is a vector of gene names representing a gene signature. • **Bidirectional gene sets**: Provide a named list where each element is a data frame with two columns: – The **first column** contains gene names. – The **second column** indicates the expected direction of enrichment (1 for upregulated genes, -1 for downregulated genes).
`method`	A character string indicating the scoring method to use. Options are `"ssGSEA"`, `"logmedian"` or `"ranking"`. Defaults to `"logmedian"`.
`ColorVariable`	Optional. Name of the metadata column to use for point color in plots.
`GroupingVariable`	Optional. Name of the metadata column to use for group comparison.
`ColorValues`	Optional. Named vector of colors to use for each group in `ColorVariable` or `GroupingVariable`.
`ConnectGroups`	Logical. If TRUE, connects points of the same sample across conditions.
`ncol`	Number of columns in the facet layout of the plot.
`nrow`	Number of rows in the facet layout of the plot.
`title`	Optional. Main title of the plot.
`widthTitle`	Numeric. Width of the title area (for alignment purposes).
`titlesize`	Numeric. Font size of the title text.
`limits`	Optional numeric vector of length 2 specifying y-axis limits.
`legend_nrow`	Optional. Number of rows in the plot legend.
`pointSize`	Numeric. Size of the points in the plots.
`xlab`	Optional. Label for the x-axis.
`labsize`	Numeric. Font size for axis and facet labels.
`compute_cohen`	Logical. If TRUE, computes Cohen's d effect sizes between groups.
`cond_cohend`	Optional. Specify a condition or comparison subset for calculating Cohen's d.
`pvalcalc`	Logical. If TRUE, computes p-values for group comparisons.
`mode`	Character string indicating comparison complexity. Options: `"simple"`, `"medium"`, `"extensive"`.
`widthlegend`	Numeric. Width of the legend area in volcano plots.
`cohen_threshold`	Numeric. Cohen's d threshold to highlight effect size in volcano plots (default = 0.6).
`colorPalette`	Character. Name of RColorBrewer palette for coloring (default = `"Set3"`).

Details

Four methods are available:

- **ssGSEA**: Uses the single-sample Gene Set Enrichment Analysis (ssGSEA) method to compute an enrichment score for each signature in each sample using an adaptation of the `gsva()` function from the `GSVA` package.
- **logmedian**: Computes the score as the sum of the Normalised (log2-median-centered) expression values of the signature genes divided by the number of genes in the signature.
- **ranking**: Computes gene signature scores for each sample by ranking the expression of signature genes in the dataset and normalizing the score based on the total number of genes.
- **all**: Computes gene signature scores using all three methods (`ssGSEA`, `logmedian`, and `ranking`). Returns a heatmap summarizing Cohen's d for all metric combinations of the variables of interest.

Depending on the method and the type of variable (categorical, numeric, or `NULL`), the function produces different plots:

- **If** `method = "all"` and the variable is **categorical**, a heatmap of Cohen's d or F statistics and a volcano plot showing contrasts between all groups of that variable are produced.
- **If** `method = "all"` and the variable is **numeric**, a heatmap of Cohen's f and a volcano plot are produced.
- **If** `method != "all"` and the variable is **categorical**, a violin plot for each signature is generated.
- **If** `method != "all"` and the variable is `NULL`, a density plot of the score distribution is displayed.
- **If** `method != "all"` and the variable is **numeric**, a scatter plot is created to show the relationship between the scores and the numeric variable.
- **If** `method = "all"` and the variable is **categorical**, the function returns a heatmap of Cohen's d or F statistics and a volcano plot showing contrasts between all groups of that variable.
- **If** `method = "all"` and the variable is **numeric**, a heatmap of Cohen's f and a volcano plot will be produced.
- **If** `method != "all"` and the variable is **categorical**, a violin plot for each signature will be displayed.
- **If** `method != "all"` and the variable is `NULL`, a density plot of the score distribution will be displayed.
- **If** `method != "all"` and the variable is **numeric**, a scatter plot will be generated to show the relationship between the scores and the numeric variable.

Value

A `ggplot` or a `ggpubr::ggarrange` object depending on the input and parameters:

- If `GroupingVariable` is `NULL`, returns a faceted grid of density plots (one per gene set).
- If `GroupingVariable` is provided and `method != "all"`, returns a faceted grid of violin plots overlaid with jittered sample points and median bars, optionally annotated with Cohen's d or f and p-values.
- Each individual plot corresponds to one gene set score computed using the selected method.

PlotScores_Numeric *Plot Gene Signature Scores with Continuous Variables*

Description

This function visualizes gene signature scores using scatter plots and regression lines across a continuous metadata variable. Signature scores are computed per sample using one of three methods: "ssGSEA", "logmedian", or "ranking". Optionally, the effect size (Cohen's f) and p-value for the association between the signature score and the continuous variable can be computed and displayed.

Usage

```
PlotScores_Numeric(
  data,
  metadata,
  gene_sets,
  method = c("ssGSEA", "logmedian", "ranking"),
  Variable = NULL,
  ColorValues = NULL,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  widthTitle = 10,
  titlesize = 12,
  limits = NULL,
  pointSize = 2,
  xlab = NULL,
  labsize = 10,
  compute_cohen = TRUE,
  pvalcalc = FALSE,
  colorPalette = "Set3",
  cor = c("pearson", "spearman", "kendall")
)
```

Arguments

data	A data frame of Normalised (non-transformed) gene expression counts. Rows are genes, columns are samples. Row names should be gene names, and column names should match sample identifiers in metadata.
metadata	A data frame where each row corresponds to a sample and contains sample-level attributes (e.g., clinical or experimental metadata). Must include a column matching the sample IDs in data.
gene_sets	A list of gene sets (signatures). Each element is either a character vector of gene names or a data frame with gene names and enrichment direction (1 for upregulated, -1 for downregulated).

method	Scoring method to use. One of "ssGSEA", "logmedian", or "ranking". Default is "logmedian".
Variable	Name of the continuous variable in metadata to use on the x-axis for scoring association.
ColorValues	(Optional) A named vector defining the color for the plotted points. If NULL, defaults to a preset color.
ncol, nrow	Number of columns and rows in the facet grid layout. If NULL, computed automatically.
title	Optional string for the overall title of the plot grid.
widthTitle	Maximum character width for titles before inserting line breaks. Default is 10.
titlesize	Numeric value for the font size of plot titles. Default is 12.
limits	Optional numeric vector of length 2 to define y-axis limits.
pointSize	Size of the plotted points. Default is 2.
xlab	Optional label for the x-axis. If NULL, defaults to the name of Variable.
labsize	Font size for axis labels. Default is 10.
compute_cohen	Logical. If TRUE (default), computes Cohen's f effect size for the association between signature score and the continuous variable.
pvalcalc	Logical. If TRUE, includes the p-value in the plot subtitle. Default is FALSE.
colorPalette	Name of the RColorBrewer palette for coloring. Default is "Set3". Currently unused but kept for consistency.
cor	Character string indicating the correlation method to be used in ggpubr::stat_cor(). Options are "pearson" (default), "kendall", or "spearman".

Details

For each gene signature, the function:

- Computes a signature score per sample using the selected method.
- Plots the score against a continuous metadata variable (Variable).
- Adds a regression line and optionally computes and displays Cohen's f effect size and p-value.
- Returns a faceted grid of ggplots, arranged by ncol and nrow.

This version of the function is specifically tailored for use with continuous variables.

Value

A ggplot2 object or a multi-plot figure showing scatter plots for each gene signature, with linear regression lines and optional statistical annotations.

plotVolcano *Volcano Plots from Differential Expression Results*

Description

This function creates a composite volcano plot grid from a list of differential expression results., or a single volcano if no genes to highlight are provided and no more than one contrast is used. For each contrast (provided in DEResultsList) and gene signature (from the genes argument), a volcano plot is generated using the specified x and y statistics. By default, if invert = FALSE and more than one gene signature is provided (i.e. the names in genes are not "ALL" or "genes"), the plots are arranged with gene signatures in rows and contrasts in columns. When invert = TRUE, the arrangement is reversed (signatures in columns and contrasts in rows). If only one gene signature is provided, an automatic grid is computed.

Usage

```
plotVolcano(
  DEResultsList,
  genes = NULL,
  N = NULL,
  x = "logFC",
  y = "-log10(adj.P.Val)",
  pointSize = 2,
  color = "#6489B4",
  highlightcolor = "#05254A",
  highlightcolor_upreg = "#038C65",
  highlightcolor_downreg = "#8C0303",
  nointerestcolor = "#B7B7B7",
  threshold_y = NULL,
  threshold_x = NULL,
  xlab = NULL,
  ylab = NULL,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  labsize = 10,
  widthlabs = 20,
  invert = FALSE
)
```

Arguments

DEResultsList	A named list of data frames containing differential expression results for each contrast. Each data frame should have row names corresponding to gene names and include columns for the x and y statistics. Output from calculateDE.

genes — Optional. A list of gene signatures to highlight. Each element may be a data frame (in which case its first column is extracted) or a vector of gene names. If NULL, no genes will be highlighted.

N — Optional. An integer specifying the number of top (and bottom) genes to annotate with text labels.

x — Character. The column name in the differential expression results to use for the x-axis (default is "logFC").

y — Character. The column name to use for the y-axis (default is "-log10(adj.P.Val)"). When using this default, threshold values for threshold_y should be provided in non-log scale (e.g., 0.05).

pointSize — Numeric. The size of points in the volcano plots (default is 2).

color — Character. The color used to highlight interesting genes based on thresholds (default is "#6489B4").

highlightcolor — Character. The color used to highlight genes belonging to the specified gene signatures (default is "#05254A"), if direction is not known or not specified.

highlightcolor_upreg — Character. The color used to highlight upregulated genes belonging to the specified gene signatures (default is "#038C65").

highlightcolor_downreg — Character. The color used to highlight downregulated genes belonging to the specified gene signatures (default is "#8C0303").

nointerestcolor — Character. The color for non-interesting genes (default is "#B7B7B7").

threshold_y — Numeric. A threshold value for the y-axis statistic. If y is "-log10(adj.P.Val)", the value should be provided as a non-log value (e.g., 0.05) and will be transformed internally.

threshold_x — Numeric. A threshold value for the x-axis statistic.

xlab — Optional. A label for the x-axis; if NULL, the value of x is used.

ylab — Optional. A label for the y-axis; if NULL, the value of y is used.

ncol — Optional. The number of columns for arranging plots in the grid. Only applicable if genes is NULL.

nrow — Optional. The number of rows for arranging plots in the grid.

title — Optional. A main title for the entire composite plot.

labsize — Numeric. The font size for label annotations (default is 10). The title size will be this value + 4.

widthlabs — Numeric. The width parameter to pass to the wrap_title() function for wrapping long labels (default is 20).

invert — Logical. If FALSE (default), the grid is arranged with gene signatures in rows and contrasts in columns. If TRUE, the arrangement is inverted (gene signatures in columns and contrasts in rows).

Details

This function generates a volcano plot for each combination of gene signature (from genes) and contrast (from DEResultsList). It uses the specified x and y statistics to plot points via ggplot2. Non-interesting genes are plotted using nointerestcolor, while genes in the specified gene signature (if not "ALL") are highlighted using highlightcolor. Optionally, the top and bottom N genes can be annotated with text labels (using ggrepel::geom_text_repel). Threshold lines for the x and/or y axes are added if threshold_x or threshold_y are provided. The individual plots are arranged into a grid using ggpubr::ggarrange and annotated with labels using ggpubr::annotate_figure and grid::textGrob. The custom wrap_title() function is used to wrap long labels.

Additionally, the function allows:

- Plotting of differentially expressed genes based on provided statistics (e.g., x = "logFC" and y = "-log10(adj.P.Val)").
- Coloring of non-interesting genes and highlighting genes belonging to specific gene signatures.
- Annotation of the top N genes with text labels (using ggrepel::geom_text_repel).
- Addition of threshold lines for the x and/or y axes.

Value

A composite plot (a ggplot object) arranged as a grid of volcano plots with annotated labels.

Examples

```
# (Assumes you have already created `expr`, `metadata`,
# and run `calculateDE` as shown above)
# For reference, here is the minimal workflow:
set.seed(123)
expr <- matrix(rpois(1000, lambda = 20), nrow = 100, ncol = 10)
rownames(expr) <- paste0("gene", 1:100)
colnames(expr) <- paste0("sample", 1:10)
metadata <- data.frame(
  sample = colnames(expr),
  Group = rep(c("A", "B"), each = 5)
)
de_res <- calculateDE(
  data = expr,
  metadata = metadata,
  variables = "Group",
  contrasts = "A-B"
)

# 1. Basic volcano plot (all genes)
plotVolcano(
  DEResultsList = de_res,
  genes = NULL,
  x = "logFC",
  y = "-log10(adj.P.Val)",
  pointSize = 2,
  color = "#6489B4",
```

```
  highlightcolor = "#05254A",
  nointerestcolor = "#B7B7B7",
  title = "Volcano Plot: A vs B"
)

# 2. Volcano plot highlighting a signature (e.g., top 5 upregulated genes)
sig_genes <- rownames(de_res[["A-B"]])[order(de_res[["A-B"]]$logFC,
decreasing = TRUE)[1:5]]
plotVolcano(
  DEResultsList = de_res,
  genes = list(Signature = sig_genes),
  x = "logFC",
  y = "-log10(adj.P.Val)",
  pointSize = 2,
  color = "#6489B4",
  highlightcolor = "#05254A",
  nointerestcolor = "#B7B7B7",
  title = "Volcano Plot: Highlight Signature"
)
```

`remove_division` *Remove Division Notation in Contrast Labels*

Description

This function removes division notation (e.g., `/2`, `/3`) after closing parentheses in contrast labels.

Usage

```
remove_division(contrasts)
```

Arguments

`contrasts`	A character vector containing contrast labels.

Value

A character vector with division notation removed.

ROCandAUCplot *ROC and AUC Plot Function*

Description

This function computes ROC curves and AUC values for each gene based on gene expression data and sample metadata. It can generate ROC plots, an AUC heatmap / barplot, or both arranged side-by-side.

Usage

```
ROCandAUCplot(
  data,
  metadata,
  genes = NULL,
  condition_var,
  class,
  group_var = NULL,
  plot_type = "roc",
  title = NULL,
  titlesize = 14,
  roc_params = list(),
  auc_params = list(),
  commomplot_params = list()
)
```

Arguments

data	A data frame or matrix containing gene expression data, with genes as rows and samples as columns.
metadata	A data frame containing sample metadata. The first column should contain sample identifiers that match the column names of data.
genes	A character vector specifying which genes to plot. If NULL (default), all genes in data are used. A warning is issued if more than 30 genes are selected.
condition_var	A character string specifying the column name in metadata representing the condition of interest. (Mandatory; no default.)
class	A character string or vector specifying the positive class label for the condition. (Mandatory; no default.)
group_var	An optional character string specifying the column name in metadata used for grouping samples (e.g., cell types). If not provided (NULL), all samples are treated as a single group. Should be a categorical variable.
plot_type	A character string indicating which plot(s) to generate. Accepted values are "roc" (only ROC curves), "auc" (only the AUC heatmap/barplot), or "all" (both arranged side-by-side). Default is "roc".
title	An optional character string specifying the main title of the plot.

`titlesize` A numeric value specifying the size of the title. Default is `14`.

`roc_params` A list of additional parameters for customizing the ROC plot. Possible elements include:

- `nrow` An integer specifying the number of rows in the ROC plot grid. If `NULL` (default), it is calculated automatically.
- `ncol` An integer specifying the number of columns in the ROC plot grid. If `NULL` (default), it is calculated automatically.
- `colors` A named vector of colors for the different groups. If `NULL` (default), a default color palette is generated.

`auc_params` A list of additional parameters for customizing the AUC heatmap or AUC barplot. Possible elements include:

- `cluster_rows` Logical; if `TRUE` (default), rows are clustered.
- `cluster_columns` Logical; if `TRUE` (default), columns are clustered.
- `colors` If `group_var` is used, should be a vector of length 2 of colors to be used for the minimum and maximum values of the color scale. Defaults to `c("#FFFFFF", "#21975C")`. If `group_var` is `NULL`, then should be a single color to fill the barplot. If `NULL`, defaults to `"#3B415B"`. If a vector is provided, only the first color will be used.
- `limits` A numeric vector of length 2 specifying the minimum and maximum values for the color scale. If not provided, defaults to `c(0.5, 1)`.
- `name` A character string for the legend title of the color scale. Default is `"AUC"`.
- `row_names_gp` Optional graphical parameters for row names (passed to **ComplexHeatmap**).
- `column_names_gp` Optional graphical parameters for column names (passed to **ComplexHeatmap**).

`commomplot_params` A list of parameters for customizing the layout of the combined plot when `plot_type = "all"`. Possible elements include:

- `widths` A numeric vector specifying the relative widths of the ROC and heatmap panels.
- `heights` A numeric vector specifying the relative heights of the panels.

Details

The function processes gene expression data and metadata to compute ROC curves and AUC values for each gene. Depending on the value of `plot_type`, it produces ROC plots (using **ggplot2**), an AUC heatmap (using **ComplexHeatmap**) or AUC barplot (if `group_var` is `NULL`), or both arranged side-by-side (using **gridExtra**).

Value

Invisibly returns a list containing:

`roc_plot` The **ggplot2** object of the ROC curves (if generated).

`heatmap` The **ComplexHeatmap** object (if generated).

`combined` The combined grid arrangement (if `plot_type = "all"`).

`auc_values` A data frame with the calculated AUC values.

Examples

```
# Simulate positive gene expression data (genes as rows, samples as columns)
set.seed(123)
expr <- matrix(rexp(30, rate = 0.5), nrow = 3, ncol = 10) # 3 genes, 10 samples, >0
rownames(expr) <- paste0("Gene", 1:3)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate metadata with a condition (binary class) and a grouping variable
metadata <- data.frame(
  SampleID = colnames(expr),
  Condition = rep(c("A", "B"), each = 5),
  Group = rep(c("G1", "G2"), times = 5)
)

# Run ROCandAUCplot with both ROC and AUC plots for three genes
ROCandAUCplot(
  data = expr,
  metadata = metadata,
  genes = rownames(expr),
  condition_var = "Condition",
  class = "A",
  plot_type = "all",
  title = "Example ROC/AUC Plots",
  roc_params = list(nrow=1)
)
```

ROCAUC_Scores_Calculate

Compute ROC Curves and AUC Values for Gene Signature Scores

Description

This function calculates Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) values for gene signature scores across different contrasts of a given categorical variable.

Usage

```
ROCAUC_Scores_Calculate(
  data,
  metadata,
  gene_sets,
  method = c("logmedian", "ssGSEA", "ranking", "all"),
  variable,
  mode = c("simple", "medium", "extensive")
)
```

Arguments

data	A matrix or data frame of gene expression data (genes as rows, samples as columns).
metadata	A data frame containing sample metadata, including the grouping variable.
gene_sets	A named list of gene sets, where each entry is a character vector of gene names.
method	A character string specifying the score calculation method. Options: "logmedian", "ssGSEA", "ranking", or "all".
variable	A character string specifying the categorical variable for group comparisons.#'
mode	A string specifying the level of detail for contrasts. Options are: • "simple": Pairwise comparisons (e.g., A - B). • "medium": Pairwise comparisons plus comparisons against the mean of other groups. • "extensive": All possible groupwise contrasts, ensuring balance in the number of terms on each side.

Value

A named list containing ROC curve data and AUC values for each method, signature, and contrast.

ROC_Scores *Plot ROC Curves for Gene Signature Scores*

Description

This function generates ROC curve plots for different gene signatures across multiple scoring methods.

Usage

```
ROC_Scores(
  data,
  metadata,
  gene_sets,
  method = c("logmedian", "ssGSEA", "ranking", "all"),
  variable,
  colors = c(logmedian = "#3E5587", ssGSEA = "#B65285", ranking = "#B68C52"),
  grid = TRUE,
  spacing_annotation = 0.3,
  ncol = NULL,
  nrow = NULL,
  mode = c("simple", "medium", "extensive"),
  widthTitle = 18,
  title = NULL,
  titlesize = 12
)
```

Arguments

data	A matrix or data frame of gene expression data.
metadata	A data frame containing sample metadata.
gene_sets	A named list of gene sets.
method	A character string specifying the scoring method(s) ("logmedian", "ssGSEA", "ranking", or "all").
variable	A character string specifying the categorical variable for group comparisons.
colors	A named vector specifying colors for each method. Only one color is allowed, if method != "all". Default colors are c(logmedian = "#3E5587", ssGSEA = "#B65285", ranking = "#B68C52").
grid	Logical; if TRUE, arranges plots in a grid.
spacing_annotation	numeric value specifying the spacing between labels of AUC values. Default is 0.3.
ncol	Optional numeric value specifying the number of columns in the grid layout for the combined plots. If NULL, there will be as many columns as contrasts. If this number is 1, then a near-square grid is computed.
nrow	Optional numeric value specifying the number of rows in the grid layout. If NULL, there will be as many columns as gene sets. If this number is 1, then a near-square grid is computed.
mode	A string specifying the level of detail for contrasts. Options are: • "simple": Performs the minimal number of pairwise comparisons between individual group levels (e.g., A - B, A - C). Default. • "medium": Includes comparisons between one group and the union of all other groups (e.g., A - (B + C + D)), enabling broader contrasts beyond simple pairs. • "extensive": Allows for all possible algebraic combinations of group levels (e.g., (A + B) - (C + D)), supporting flexible and complex contrast definitions.
widthTitle	Optional integer specifying the maximum width of the title before inserting line breaks. Titles break at _, -, or : where possible, or at the exact width if no such character is found. Default is 18.
title	Title for the grid of plots.
titlesize	An integer specifying the text size for each of the heatmap titles. Default is 12.

Value

A ggplot2 or ggarrange object containing the ROC curve plots.

Examples

```
# Example data
data <- as.data.frame(abs(matrix(rnorm(1000), ncol = 10)))
rownames(data) <- paste0("Gene", 1:100)  # Name columns as Gene1, Gene2, ..., Gene10
```

```
colnames(data) <- paste0("Sample", 1:10)  # Name rows as Sample1, Sample2, ..., Sample100

# Metadata with sample ID and condition
metadata <- data.frame(
  SampleID = colnames(data),  # Sample ID matches the colnames of the data
  Condition = rep(c("A", "B"), each = 5)  # Two conditions (A and B)
)

# Example gene set
gene_sets <- list(Signature1 = c("Gene1", "Gene2", "Gene3"))  # Example gene set

# Call ROC_Scores function
ROC_Scores(data, metadata, gene_sets, method = "ssGSEA", variable = "Condition")
```

---

runGSEA *Run Gene Set Enrichment Analysis (GSEA) for Multiple Contrasts*

Description

This function performs GSEA using `fgsea` for each contrast in a list of differential expression results. It automatically determines the appropriate ranking statistic based on the gene set format unless specified by the user.

Usage

```
runGSEA(
  DEGList,
  gene_sets,
  stat = NULL,
  ContrastCorrection = FALSE,
  nPermSimple = 10000,
  p.adjust.method = "BH"
)
```

Arguments

`DEGList` A named list where each element represents a contrast and contains a data frame of differential expression results.

- Each data frame must include at least the `"t"` statistic and the `"B"` statistic for each gene.
- Row names should correspond to gene identifiers.

`gene_sets` A named list where each element represents a gene set. Each gene set can be:

- A **vector of gene names** (for unidirectional gene sets).
- A **data frame** with two columns:
  - Column 1: Gene names.

  - Column 2: Expected direction (1 for upregulated genes, -1 for downregulated genes).

stat
: Optional. The statistic to use for ranking genes before GSEA. If NULL, it is automatically determined based on the gene set:
  - "B" for gene sets with **no known direction** (vectors).
  - "t" for **unidirectional** or **bidirectional** gene sets (data frames).
  - If provided, this argument overrides the automatic selection.

ContrastCorrection
: Logical, default is FALSE. If TRUE, applies an additional multiple testing correction (Benjamini-Hochberg) across all contrasts returned in the DEGList results list. This accounts for the number of contrasts tested per signature and provides more stringent control of false discovery rate across multiple comparisons. If FALSE, the function only corrects for the number of gene sets.

nPermSimple
: Number of permutations in the simple fgsea implementation for preliminary estimation of P-values. Parameter from fgsea.

p.adjust.method
: Character string specifying the method to use for multiple testing correction. Must be one of "BH" (Benjamini-Hochberg, default), "holm", "hommel", "bonferroni", "BY" (Benjamini-Yekutieli), "fdr", or "none". Passed to p.adjust.

Value

A named list where each element corresponds to a contrast. Each contrast contains a **single data frame** with GSEA results for all gene sets. P-values are corrected for multiple testing based on all contrasts. The result includes the standard fgsea output plus two additional columns:

- pathway: The name of the gene set.
- stat_used: The statistic used for ranking genes in that analysis ("t" or "B").

Examples

```
# Example input data
DEGList <- list(
 Contrast1 = data.frame(t = rnorm(100), B = rnorm(100), row.names = paste0("Gene", 1:100)),
 Contrast2 = data.frame(t = rnorm(100), B = rnorm(100), row.names = paste0("Gene", 1:100))
)

gene_sets <- list(
  UnidirectionalSet = c("Gene1", "Gene5", "Gene20"),
 BidirectionalSet = data.frame(Gene = c("Gene2", "Gene10", "Gene15"), Direction = c(1, -1, 1))
)

results <- runGSEA(DEGList, gene_sets)
print(results)
```

Score_VariableAssociation
Score Variable Association

Description

This function evaluates the association between gene expression scores and metadata variables. It uses linear modeling to get Cohen's F, and contrast-based comparisons for categorical variables to compute Cohen's D. The function generates plots summarizing the results.

Usage

```
Score_VariableAssociation(
  data,
  metadata,
  cols,
  method = c("logmedian", "ssGSEA", "ranking"),
  gene_set,
  mode = c("simple", "medium", "extensive"),
  nonsignif_color = "grey",
  signif_color = "red",
  saturation_value = NULL,
  sig_threshold = 0.05,
  widthlabels = 18,
  labsize = 10,
  title = NULL,
  titlesize = 14,
  pointSize = 5,
  discrete_colors = NULL,
  continuous_color = "#8C6D03",
  color_palette = "Set2",
  printplt = TRUE,
  p.adjust.method = "BH"
)
```

Arguments

data	A data frame or matrix containing gene expression data.
metadata	A data frame containing sample metadata with at least one column corresponding to the variables of interest.
cols	A character vector specifying metadata columns to analyse.
method	A character string specifying the scoring method ("logmedian", "ssGSEA", or "ranking").
gene_set	A named list containing one gene set for scoring.
mode	A character string specifying the contrast generation method ("simple", "medium", "extensive"). Four methods are available:

- **ssGSEA**: Uses the single-sample Gene Set Enrichment Analysis (ssGSEA) method to compute an enrichment score for each signature in each sample using an adaptation of the `gsva()` function from the `GSVA` package.
- **logmedian**: Computes the score as the sum of the normalized (log2-median-centered) expression values of the signature genes divided by the number of genes in the signature.
- **ranking**: Computes gene signature scores for each sample by ranking the expression of signature genes in the dataset and normalizing the score based on the total number of genes.

`nonsignif_color`
: A string specifying the color for non-significant results. Default: `"grey"`.

`signif_color`
: A string specifying the color for significant results. Default: `"red"`.

`saturation_value`
: A numeric value for color saturation threshold. Default: `NULL` (auto-determined).

`sig_threshold`
: A numeric value specifying the significance threshold. Default: `0.05`.

`widthlabels`
: An integer controlling contrast label wrapping. Default: `18`.

`labsize`
: An integer controlling axis text size. Default: `10`.

`titlesize`
: An integer specifying the title size. Default: `14`.

`pointSize`
: A numeric value for point size in plots. Default: `5`.

`discrete_colors`
: A named list mapping categorical variable levels to colors. Each element should be a named vector where names correspond to factor levels. Default: `NULL`.

`continuous_color`
: A string specifying the color for continuous variables. Default: `"#8C6D03"`.

`color_palette`
: A string specifying the color palette for discrete variables. Default: `"Set2"`.

`printplt`
: Boolean specifying if plot is to be printed. Default: `TRUE`.

`p.adjust.method`
: Character string specifying the method to use for multiple testing correction. Must be one of `"BH"` (Benjamini-Hochberg, default), `"holm"`, `"hommel"`, `"bonferroni"`, `"BY"` (Benjamini-Yekutieli), `"fdr"`, or `"none"`. Passed to `p.adjust`.

Value

A list with:

- `Overall`: Data frame of effect sizes and p-values for each contrasted phenotypic variable.
- `Contrasts`: Data frame of Cohen's d and adjusted p-values for contrasts between levels of categorical variables, with the resolution of contrasts determined by the mode parameter.
- `plot`: A combined visualization with three main panels: (1) lollipop plots of Cohen's f for each variable of interest, (2) distribution plots of the score by variable (density or scatter depending on variable type), and (3, if applicable) lollipop plots of Cohen's d for contrasts in categorical variables.
- `plot_contrasts`: Lollipop plots of Cohen's d effect sizes for contrasts between levels of non numerical variables (if applicable), colored by adjusted p-value (BH).

- `plot_overall`: Lollipop plot showing Cohen's f effect sizes for each variable, colored by p-value.
- `plot_distributions`: List of density or scatter plots of the score across variable levels, depending on variable type.

`ssGSEA_alternative` *Alternative Implementation of Single-Sample Gene Set Enrichment Analysis (ssGSEA)*

Description

This function computes an enrichment score for each sample using an alternative single-sample Gene Set Enrichment Analysis (ssGSEA) method. It first maps gene sets to the gene indices present in the expression matrix, then ranks the genes for each sample, and finally calculates a weighted enrichment score based on the cumulative differences between in-set and out-of-set gene ranks. Source: https://rpubs.com/pranali018/SSGSEA

Usage

```
ssGSEA_alternative(
  X,
  gene_sets,
  alpha = 0.25,
  scale = TRUE,
  norm = FALSE,
  single = TRUE
)
```

Arguments

`X`	A numeric matrix of gene expression values with rows representing genes and columns representing samples. Row names should correspond to gene identifiers.
`gene_sets`	A list of gene sets, where each element is a vector of gene identifiers. The function will match these identifiers with the row names of `X`.
`alpha`	A numeric value specifying the exponent used to weight the ranking scores. Default is `0.25`.
`scale`	Logical; if `TRUE`, the cumulative difference is normalized by the total number of genes. Default is `TRUE`.
`norm`	Logical; if `TRUE`, the enrichment scores are further normalized by the absolute difference between the maximum and minimum scores. Default is `FALSE`.
`single`	Logical; if `TRUE`, the function returns the sum of the cumulative differences as the enrichment score. If `FALSE`, the maximum absolute cumulative difference is used. Default is `TRUE`.

Details

The function performs the following steps:

1. Maps each gene set to the indices of genes in `X` by matching gene identifiers.
2. Computes column-wise rankings for the gene expression matrix using a ranking method (via the `colRanking` function) with tie resolution set to `'average'`.
3. For each sample, orders the gene ranks in decreasing order.
4. For each gene set in the sample, calculates:
   - The weighted contribution (`rank_alpha`) for genes in the set raised to the power of `alpha`.
   - The cumulative distribution functions (CDFs) for genes within the gene set (`step_cdf_pos`) and those not in the gene set (`step_cdf_neg`).
   - The difference between these CDFs, optionally scaled by the number of genes if `scale = TRUE`.
   - Depending on the `single` parameter, either the sum of the differences (if `TRUE`) or the maximum absolute difference (if `FALSE`) is used as the enrichment score for that gene set.
5. Optionally normalizes the final enrichment scores by the range of values if `norm = TRUE`.

Value

A matrix of enrichment scores with rows corresponding to gene sets and columns corresponding to samples.

Examples

```
## Not run:
  # Create a sample gene expression matrix:
  X <- matrix(rnorm(1000), nrow = 100, ncol = 10)
  rownames(X) <- paste0("gene", 1:100)

  # Define example gene sets:
  gene_sets <- list(
    set1 = sample(rownames(X), 10),
    set2 = sample(rownames(X), 15)
  )

  # Compute the ssGSEA enrichment scores:
  es <- ssGSEA_alternative(X, gene_sets, alpha = 0.25, scale = TRUE,
  norm = FALSE, single = TRUE)
  print(es)

## End(Not run)
```

VariableAssociation *Variable Association Analysis*

Description

This unified function evaluates associations between gene expression and sample metadata using multiple methods: score-based (logmedian, ssGSEA, ranking) or GSEA-based association. The function returns statistical results and visualizations summarizing effect sizes and significance.

Usage

```
VariableAssociation(
  method = c("ssGSEA", "logmedian", "ranking", "GSEA"),
  data,
  metadata,
  cols,
  gene_set,
  mode = c("simple", "medium", "extensive"),
  stat = NULL,
  ignore_NAs = FALSE,
  signif_color = "red",
  nonsignif_color = "grey",
  sig_threshold = 0.05,
  saturation_value = NULL,
  widthlabels = 18,
  labsize = 10,
  titlesize = 14,
  pointSize = 5,
  discrete_colors = NULL,
  continuous_color = "#8C6D03",
  color_palette = "Set2",
  printplt = TRUE,
  p.adjust.method = "BH"
)
```

Arguments

method	Character string specifying the method to use. One of: • "logmedian" • "ssGSEA" • "ranking" • "GSEA"
data	A data frame with gene expression data (genes as rows, samples as columns).
metadata	A data frame containing sample metadata; the first column should be the sampleID.

cols	Character vector of metadata column names to analyze.
gene_set	A named list of gene sets: • For score-based methods: list of gene vectors. • For GSEA: list of vectors (unidirectional) or data frames (bidirectional).
mode	Contrast mode: "simple" (default), "medium", or "extensive".
stat	(GSEA only) Optional. Statistic for ranking genes ("B" or "t"). Auto-detected if NULL.
ignore_NAs	(GSEA only) Logical. If TRUE, rows with NA metadata are removed. Default: FALSE.
signif_color	Color used for significant associations (default: "red").
nonsignif_color	Color used for non-significant associations (default: "grey").
sig_threshold	Numeric significance cutoff (default: 0.05).
saturation_value	Lower limit for p-value coloring (default: auto).
widthlabels	Integer for contrast label width before wrapping (default: 18).
labsize	Axis text size (default: 10).
titlesize	Plot title size (default: 14).
pointSize	Size of plot points (default: 5).
discrete_colors	(Score-based only) Optional named list mapping factor levels to colors.
continuous_color	(Score-based only) Color for continuous variable points (default: "#8C6D03").
color_palette	(Score-based only) ColorBrewer palette name for categorical variables (default: "Set2").
printplt	Logical. If TRUE, plots are printed. Default: TRUE.
p.adjust.method	Character string specifying the method to use for multiple testing correction. Must be one of "BH" (Benjamini-Hochberg, default), "holm", "hommel", "bonferroni", "BY" (Benjamini-Yekutieli), "fdr", or "none". Passed to p.adjust.

Value

A list with method-specific results and ggplot2-based visualizations:

For score-based methods (logmedian**,** ssGSEA**,** ranking**):**

- Overall: Data frame of effect sizes (Cohen's f) and p-values for each metadata variable.
- Contrasts: Data frame of Cohen's d values and adjusted p-values for pairwise comparisons (based on mode).
- plot: A combined visualization including:
  - Lollipop plots of Cohen's f,
  - Distribution plots by variable (density or scatter),

  - Lollipop plots of Cohen's d for contrasts.
- `plot_contrasts`: Lollipop plots of Cohen's d effect sizes, colored by adjusted p-values (BH).
- `plot_overall`: Lollipop plot of Cohen's f, colored by p-values.
- `plot_distributions`: List of distribution plots of scores by variable.

For GSEA-based method (`GSEA`):

- `data`: A data frame with GSEA results, including normalized enrichment scores (NES), adjusted p-values, and contrasts.
- `plot`: A ggplot2 lollipop plot of GSEA enrichment across contrasts.

Examples

```
# Simulate gene expression data (genes as rows, samples as columns)
set.seed(42)
expr <- as.data.frame(matrix(rnorm(500), nrow = 50, ncol = 10))
rownames(expr) <- paste0("Gene", 1:50)
colnames(expr) <- paste0("Sample", 1:10)

# Simulate metadata (categorical and continuous)
metadata <- data.frame(
  sampleID = paste0("Sample", 1:10),
  Group = rep(c("A", "B"), each = 5),
  Age = sample(20:60, 10),
  row.names = colnames(expr)
)

# Define a toy gene set: one gene set only for discovery mode!
gene_set <- list(
  Signature1 = paste0("Gene", 1:10)
)

# Score-based association (e.g., logmedian)
res_score <- VariableAssociation(
  method = "logmedian",
  data = expr,
  metadata = metadata,
  cols = c("Group", "Age"),
  gene_set = gene_set
)
print(res_score$Overall)
print(res_score$plot)

# GSEA-based association (if GSEA_VariableAssociation is available)
# res_gsea <- VariableAssociation(
#   method = "GSEA",
#   data = expr,
#   metadata = metadata,
#   cols = "Group",
#   gene_set = gene_set
# )
```

```
# print(res_gsea$data)
print(res_score$plot)
```

`VisualiseIndividualGenes` *VisualiseIndividualGenes: Wrapper for Visualising Individual Genes in Gene Sets*

Description

This wrapper function helps explore individual gene behavior within a gene set used in `markeR`. It dispatches to specific visualisation functions based on `plot_type`, supporting various plot types: heatmaps, violin plots, correlation analysis, PCA, ROC/AUC, and effect size heatmaps.

Usage

```
VisualiseIndividualGenes(type, data, genes, metadata = NULL, ...)
```

Arguments

`type`	Character. Specifies the type of plot to generate. Must be one of: • `"violin"`: Violin plots of individual gene expression by group • `"correlation"`: Correlation heatmap of selected genes • `"expression"`: Expression heatmap of selected genes • `"roc"`: ROC plots for classification performance of individual genes • `"auc"`: AUC plots for classification performance of individual genes • `"rocauc"`: Combined ROC and AUC plots • `"cohend"`: Effect size (Cohen's D) heatmap • `"pca"`: PCA plot of selected genes
`data`	Required. Expression data matrix or data frame, with samples as rows and genes as columns.
`genes`	Required. Character vector of gene names to include in the visualisation.
`metadata`	Optional. Data frame with sample metadata, required for some plot types (e.g., violin, roc, cohend).
`...`	Additional arguments passed to the specific plotting function.

Value

The output of the specific plotting function called, usually a ggplot or ComplexHeatmap object, often wrapped in a list with additional data.

Additional required arguments (passed via ...) per `plot_type`

violin Requires `GroupingVariable` (column name in `metadata` for grouping).

roc, auc, rocauc Requires `condition_var` (metadata column with condition labels) and `class` (positive class label).

cohend Requires `condition_var` and `class` (same as roc).

correlation, expression, pca No additional mandatory arguments required.

See Also

IndividualGenes_Violins, CorrelationHeatmap, ExpressionHeatmap, CohenD_IndividualGenes, plotPCA, ROCandAUCplot

Examples

```
# Example data
set.seed(123)
expr_data <- matrix(rexp(1000, rate = 1), nrow = 50, ncol = 20)
rownames(expr_data) <- paste0("Gene", 1:50)
colnames(expr_data) <- paste0("Sample", 1:20)

sample_info <- data.frame(
  SampleID = colnames(expr_data),
  Condition = rep(c("A", "B"), each = 10),
  Diagnosis = rep(c("Disease", "Control"), each = 10),
  stringsAsFactors = FALSE
)
rownames(sample_info) <- sample_info$SampleID

selected_genes <- row.names(expr_data)[1:5]

# Violin plot
VisualiseIndividualGenes(
  type = "violin",
  data = expr_data,
  metadata = sample_info,
  genes = selected_genes,
  GroupingVariable = "Condition",
  nrow=1
)
VisualiseIndividualGenes(
  type = "correlation",
  data = expr_data,
  genes = selected_genes
)

# Expression heatmap
VisualiseIndividualGenes(
  type = "expression",
  data = expr_data,
  genes = selected_genes
)
```

```
# PCA plot
VisualiseIndividualGenes(
  type = "pca",
  data = expr_data,
  genes = selected_genes,
  metadata = sample_info,
  ColorVariable="Condition"

)

# ROC plot
VisualiseIndividualGenes(
  type = "roc",
  data = expr_data,
  metadata = sample_info,
  genes = selected_genes,
  condition_var = "Diagnosis",
  class = "Disease"

)

# AUC plot
VisualiseIndividualGenes(
  type = "auc",
  data = expr_data,
  metadata = sample_info,
  genes = selected_genes,
  condition_var = "Diagnosis",
  class = "Disease"
)

# ROC&AUC plot
VisualiseIndividualGenes(
  type = "rocauc",
  data = expr_data,
  metadata = sample_info,
  genes = selected_genes,
  condition_var = "Diagnosis",
  class = "Disease"
)

# Cohen's D plot
VisualiseIndividualGenes(
  type = "cohend",
  data = expr_data,
  metadata = sample_info,
  genes = selected_genes,
  condition_var = "Diagnosis",
  class = "Disease"
)
```

Volcano_Cohen *Volcano Plot of Cohen's d Effect Sizes and Adjusted p-values*

Description

This function computes Cohen's d effect sizes and adjusted p-values for multiple gene signatures across defined contrasts, and generates a volcano plot (Cohen's d vs -log10(padj)) using ggplot2. Each point represents a method-signature pair, faceted by contrast.

Usage

```
Volcano_Cohen(
  cohenlist,
  titlesize = 12,
  ColorValues = NULL,
  title = NULL,
  widthlegend = 22,
  pointSize = 3,
  sig_threshold = 0.05,
  cohen_threshold = 0.5,
  colorPalette = "Set3",
  nrow = NULL,
  ncol = NULL
)
```

Arguments

cohenlist	A named list from CohenD_allConditions. Each element is a list with: • CohenD: A data frame where rows are methods and columns are group contrasts (formatted as "Group1:Group2"), containing the computed Cohen's d effect sizes. • PValue: A data frame with the same structure as CohenD containing the corresponding p-values. • padj: A data frame with the same structure as PValue containing the corresponding p-values corrected using the BH method, for all signatures and contrasts, and by method.
titlesize	Integer. Size of the facet strip titles. Default is 12.
ColorValues	Character vector of colors used to distinguish signatures. If NULL, colors are automatically generated.
title	Optional title for the overall plot.
widthlegend	Integer. Width used to wrap long signature names. Default is 22.
pointSize	Numeric. Size of the points in the plot. Default is 3.
sig_threshold	Numeric. Adjusted p-value threshold for significance. Default is 0.05.

`cohen_threshold`
: Numeric. Effect size threshold. Default is 0.5.

`colorPalette`
: Character. Name of RColorBrewer palette to use if `ColorValues` is not provided. Default is "Set3".

`nrow`
: Optional numeric value specifying the number of rows in the grid layout. If `NULL`, a near-square grid is computed.

`ncol`
: Optional numeric value specifying the number of columns in the ggplot facet. If `NULL`, a near-square grid is computed.

Value

A `ggplot` object showing a faceted volcano plot of Cohen's d effect sizes across signatures and methods for each contrast.

See Also

`CohenD_allConditions`

`wrap_title` *Wrap Long Titles with Capital Letter Prioritization (when no symbols are nearby)*

Description

This function wraps long titles into multiple lines to fit a specified width, prioritizing breaks at symbols like underscores, hyphens, and colons when they are close to the wrap point. If no special symbol is found nearby, the function will break the title at the first capital letter. If neither is found, the title is broken at the specified width.

Usage

```
wrap_title(title, width = 30)
```

Arguments

`title`
: A character string representing the title to be wrapped. **(Required)**

`width`
: A numeric value specifying the maximum width for each line. The default is 30 characters. **(Optional)**

Value

A character string with the title wrapped into multiple lines. Each line will not exceed the specified width, with breaks prioritized by symbols when nearby, and capital letters used only when no symbols are present.

Index

∗ **datasets**
counts_example, 28
genesets_example, 36
metadata_example, 48
∗ **internal**
calculateScore_logmedian_bidirectional, 15
calculateScore_logmedian_unidirectional, 16
CalculateScores_logmedian, 10
CalculateScores_Ranking, 11
CalculateScores_ssGSEA, 12
CalculateScores_ssGSEA_bidirectional, 13
CalculateScores_ssGSEA_unidirectional, 14
cohen_d, 22
CohenD_allConditions, 17
CohenF_allConditions, 20
colRanking, 22
compute_cohen_d, 23
compute_cohens_f_pval, 23
compute_stat_tests, 24
create_contrast_column, 29
flatten_results, 32
generate_all_contrasts, 35
getRanking, 39
GSEA_VariableAssociation, 40
Heatmap_Cohen, 42
identify_variable_type, 44
PlotScores_Categorical, 61
PlotScores_Numeric, 64
remove_division, 69
ROCAUC_Scores_Calculate, 72
Score_VariableAssociation, 77
ssGSEA_alternative, 79
Volcano_Cohen, 87
wrap_title, 88

AUC_Scores, 3

calculateDE, 6
calculateScore_logmedian_bidirectional, 15
calculateScore_logmedian_unidirectional, 16
CalculateScores, 8
CalculateScores_logmedian, 10
CalculateScores_Ranking, 11
CalculateScores_ssGSEA, 12
CalculateScores_ssGSEA_bidirectional, 13
CalculateScores_ssGSEA_unidirectional, 14
cohen_d, 22
CohenD_allConditions, 17, *44*, *88*
CohenD_IndividualGenes, 18, *85*
CohenF_allConditions, 20, *44*
colRanking, 22
compute_cohen_d, 23
compute_cohens_f_pval, 23
compute_stat_tests, 24
CorrelationHeatmap, 26, *85*
counts_example, 28
create_contrast_column, 29

ExpressionHeatmap, 30, *85*

flatten_results, 32
FPR_Simulation, 32

generate_all_contrasts, 35
geneset_similarity, 37
genesets_example, 36
getRanking, 39
GSEA_VariableAssociation, 40

Heatmap_Cohen, 42

identify_variable_type, 44
IndividualGenes_Violins, 45, *85*

markeR, 47
markeR-package (markeR), 47
metadata_example, 48

p.adjust, *17*, *21*, *25*, *34*, *42*, *59*, *76*, *78*, *82*
plotCombinedGSEA, 49
plotGSEAenrichment, 50
plotNESlollipop, 51
plotPCA, 54, *85*
PlotScores, 56
PlotScores_Categorical, 61
PlotScores_Numeric, 64
plotVolcano, 66

remove_division, 69
ROC_Scores, 73
ROCandAUCplot, 70, *85*
ROCAUC_Scores_Calculate, 72
runGSEA, 75

Score_VariableAssociation, 77
ssGSEA_alternative, 79

VariableAssociation, 81
VisualiseIndividualGenes, 84
Volcano_Cohen, 87

wrap_title, 88