Package 'goatea'

February 21, 2026

Type Package

Title Interactive Exploration of GSEA by the GOAT Method

Version 1.0.2

Description Geneset Ordinal Association Test Enrichment Analysis (GOATEA) provides a 'Shiny' interface with interactive visualizations and utility functions for performing and exploring automated gene set enrichment analysis using the 'GOAT' package.
'GOATEA' is designed to support large-scale and user-friendly enrichment workflows across multiple gene lists and comparisons, with flexible plotting and output options. Visualizations pre-
enrichment include interactive 'Volcano' and 'UpSet' (overlap) plots. Visualizations post-enrichment include interactive geneset dotplot, geneset treeplot, gene-effectsize heatmap, gene-geneset heatmap and 'STRING' database of protein-protein-interactions network graph.
'GOAT' reference: Frank Koopmans (2024) <doi:10.1038/s42003-024-06454-5>.

URL https://github.com/mauritsunkel/goatea

BugReports https://github.com/mauritsunkel/goatea/issues

License Apache License (>= 2)

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

biocViews GeneSetEnrichment, NetworkEnrichment, Visualization, ShinyApps, GUI, Transcriptomics, Genetics, FunctionalGenomics, DifferentialExpression, Network

Depends R (>= 4.5.0), dplyr (>= 1.1.4)

Imports goat (>= 1.0), shiny (>= 1.10.0), shinyjs (>= 2.1.0), shinyjqui (>= 0.4.1), shinydashboard (>= 0.7.2), openxlsx (>= 4.2.7.1), upsetjs (>= 1.11.1), ComplexHeatmap (>= 2.24.0), InteractiveComplexHeatmap (>= 1.12.0), EnhancedVolcano (>= 1.22.0), tidyr (>= 1.3.1), purrr (>= 1.0.2), ggplot2 (>= 3.5.1), plotly (>= 4.10.4), igraph (>= 2.1.4), visNetwork (>= 2.1.2), arrow (>= 18.1.0.1), htmltools (>= 0.5.8.1), methods (>= 4.5.0), AnnotationDbi (>= 1.69.1), DT (>= 0.33), plyr (>= 1.8.9), tibble (>= 3.2.1), rlang (>= 1.1.6), DOSE (>= 4.4.0), enrichplot (>= 1.30.4), EnhancedVolcano (>= 1.28.1)

Suggests org.Hs.eg.db, org.Mm.eg.db, org.Dm.eg.db, org.Mmu.eg.db, org.Rn.eg.db, org.Ce.eg.db, org.Pt.eg.db, org.Dr.eg.db, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

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

git_branch RELEASE_3_22

git_last_commit 4d95ac0

git_last_commit_date 2026-02-17

Repository Bioconductor 3.22

Date/Publication 2026-02-20

Author Maurits Unkel [aut, cre, fnd, cph] (ORCID: <`https://orcid.org/0000-0003-1920-6001`>)

Maintainer Maurits Unkel <`mauritsunkel@gmail.com`>

Contents

`goatea-package` *goatea: Interactive Exploration of GSEA by the GOAT Method*

Description

Geneset Ordinal Association Test Enrichment Analysis (GOATEA) provides a 'Shiny' interface with interactive visualizations and utility functions for performing and exploring automated gene set enrichment analysis using the 'GOAT' package. 'GOATEA' is designed to support large-scale and user-friendly enrichment workflows across multiple gene lists and comparisons, with flexible plotting and output options. Visualizations pre-enrichment include interactive 'Volcano' and 'UpSet' (overlap) plots. Visualizations post-enrichment include interactive geneset dotplot, geneset treeplot, gene-effectsize heatmap, gene-geneset heatmap and 'STRING' database of protein-protein-interactions network graph. 'GOAT' reference: Frank Koopmans (2024) doi:10.1038/s42003-024064545.

Author(s)

Maintainer: Maurits Unkel <`mauritsunkel@gmail.com`> (ORCID) [funder, copyright holder]

See Also

Useful links:

- `https://github.com/mauritsunkel/goatea`
- Report bugs at `https://github.com/mauritsunkel/goatea/issues`

`calculate_geneSetRatio` *Calculate geneSetRatio per gene per enrichment contrast*

Description

Get a percentage of genesets the specific gene is included

Usage

```
calculate_geneSetRatio(enrichment_results, gene_overview_df)
```

Arguments

`enrichment_results`	list of enrichment results
`gene_overview_df`	dataframe with gene-wise information

Value

numerical vector of gene set ratios

Examples

```
calculate_geneSetRatio(
list(
  A = get(load(system.file("extdata", "example_enrichment.rda", package = "goatea"))),
  B = get(load(system.file("extdata", "example_enrichment.rda", package = "goatea")))
),
get(load(system.file("extdata", "example_genes_overview.rda", package = "goatea"))))
```

`colorify` *Create and/or modify color/gradient palettes*

Description

Note for colorblind use: "Okabe-Ito"

Addition of values happens before multiplication with factors.

Palette names are stripped of whitespace and lowered for name matching. All RColorBrewer and Viridis palettes are included.

All grDevices plotting functions are provided as palettes, simply use colors = "rainbow", "heat", "terrain", "topo" or "cm".

Usage

```
colorify(
  n = NULL,
  colors = character(0),
  colors_lock = NULL,
  colors_names = character(0),
  colors_breakpoints = numeric(0),
  gradient_n = n,
  gradient_space = c("rgb", "Lab"),
  gradient_interpolate = c("linear", "spline"),
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
  sv = 0,
  lv = 0,
  rv = 0L,
  gv = 0L,
  bv = 0L,
  alpha = 1,
  rev = FALSE,
  plot = FALSE,
  export = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`n`	default: NULL, else integer, amount of colors to create, if palette selected and more colors requested they will be generated
`colors`	character (vector), combination of selecting palette(s) by name (options: see display_palettes()), and/or vector of R color names and/or color hexcodes
`colors_lock`	default: rep(FALSE, length(colors), numerical or logical index of colors (not) to be modified, if logical length != colors it will be cut or filled with TRUE/FALSE, prefix with '!' for logical vectors and '-' for numerical vectors to get inverse, see examples. If gradient_n %% length(colors) == 0, i.e. if gradient_n divisive by amount of colors without rest, set repeat given locking pattern
`colors_names`	default: character(0), else character vector of color names
`colors_breakpoints`	default: numeric(0), else numeric vector of breakpoints to colorRamp in between
`gradient_n`	default: n, else integer, amount of colors to output as gradient, after completing palette for n colors
`gradient_space`	default: "rgb", else "Lab", see ?grDevices::colorRamp()
`gradient_interpolate`	default: "linear", else "spline", see ? grDevices::colorRamp()
`hf`	hue factor, default: 1, multiply values by factor, proportional to base value of 1
`sf`	saturation factor, default: 1, multiply values by factor, proportional to base value of 1
`lf`	lightness/brightness factor, default: 1, multiply values by factor, proportional to base value of 1
`rf`	red factor, default: 1, multiply values by factor, proportional to base value of 1
`gf`	green factor, default: 1, multiply values by factor, proportional to base value of 1
`bf`	blue factor, default: 1, multiply values by factor, proportional to base value of 1
`hv`	hue value, default: 0, add value to values, linear from base value of 0
`sv`	saturation value, default: 0, add value to values, linear from base value of 0
`lv`	lightness/brightness value, default: 0, add value to values, linear from base value of 0
`rv`	red value, default: 0, add value to values, linear from base value of 0
`gv`	green value, default: 0, add value to values, linear from base value of 0
`bv`	blue value, default: 0, add value to values, linear from base value of 0
`alpha`	numeric, sets color alpha values
`rev`	default: FALSE, if TRUE, reverse order of colors
`plot`	default: FALSE, if TRUE plot pie chart of color palette
`export`	default: FALSE, if TRUE: export = getwd(), if export = "string/", save hexcodes, rgb, and hsl values to export/colorify.csv
`verbose`	default: TRUE, else FALSE - to log status messages
`...`	additional arguments to pass on

Details

Either generate theoretically maximally different colors, select an available R grDevices palette and/or modify the colors of the given gradient/palette

Value

vector of color hexcodes

Examples

```
colorify(10, plot = TRUE)
```

`colorify_map` *Colorify colorRamp between colors mapping to breakpoint values*

Description

Note that breakpoints and colors will be ordered ascendingly by breakpoints values

Usage

```
colorify_map(colors, breakpoints, ...)
```

Arguments

`colors`	hexcolor character vector
`breakpoints`	numeric vector matching colors per value
`...`	to pass arguments to grDevices::colorRamp

Value

function with colors and breaks attributes, can be called as function(c(values)) to return hexcolor-codes

---

`display_palettes` *Display R grDevices palettes*

Description

Use colorify() to select and modify the palettes, see its documentation. Note that discrete palettes with maximum n colors will be repeated in plotting.

Any numeric i_palettes over maximum amount of palettes are not displayed.

Contains all Viridis palettes, excluding Turbo.

Usage

```
display_palettes(n = 10, i_palettes = seq_len(1000), border = FALSE)
```

Arguments

`n`	integer, amount of colors to display
`i_palettes`	default: numeric vector as index/range for choosing palettes, or a combination of 'rcolorbrewer', 'viridis', 'rainbow' (grDevices Palettes) to show specific palettes
`border`	default: FALSE, if TRUE show color rectangle borders

Value

named vector with source and name of palettes, 'hcl' for grDevices::hcl.pals() and 'pal' for grDevices::palette.pals()

`example_enrichment` *An example enrichment*

Description

A simulated example of an enrichemnt for testing or demonstration purposes.

Format

`enrichment`**:**

A data frame with 10 rows and 17 columns:

source origin

source_version org.Xx.eg.db

id DB.001

name geneset name 1, geneset name 2

parent_id DB.010, DB.020

ngenes_input 10, 30

ngenes 10, 30

genes 10000, 10001

ngenes_signif 10, 30

score_type effectsize

pvalue 0.05, 1

zscore -Inf, 0, Inf ...

Source

generated with `data-raw/example_data.R`

`example_genelist` *An example genelist*

Description

A simulated example of a genelist for testing or demonstration purposes.

Format

`genelist`**:**

A data frame with 100 rows and 4 columns:

symbol Gene_1, Gene_2

gene 10000, 10001

pvalue 0.05, 1

effectsize 2.5, 0 ...

Source

generated with `data-raw/example_data.R`

`example_genesets` *Example genesets*

Description

A simulated example of a geneset for testing or demonstration purposes.

Format

`genesets`**:**

A data frame with 100 rows and 4 columns:

source origin

source_version org.Xx.eg.db

id DB.001

name geneset name 1, geneset name 2

parent_id DB.010, DB.020

genes 10000, 10001

ngenes 10, 30 ...

Source

generated with `data-raw/example_data.R`

`example_genes_overview` *An example genes overview*

Description

A simulated example of a gene overview for testing or demonstration purposes.

Format

`genes overview`**:**

A data frame with 100 rows and ~11 columns:

gene 10000, 10001

symbol Gene_1, Gene_2

sample_efsi 2.5, 0

sample_pval 0.05, 1

sample_perc 0, 100

genelist_overlap "", "A", "B", "AB"

sample_geneSetRatio 0, 50, 100 ...

Source

generated with `data-raw/example_data.R`

`example_ppi_data` *An example ppi data*

Description

A simulated example of a ppi dataframe for testing or demonstration purposes.

Format

`ppi data`**:**

A data frame with 15 rows and 5 columns:

from_symbol gene_A, gene_B

to_symbol gene_A, gene_B

combined_score 0, 1000

from gene_A_ID, gene_B_ID

to gene_A_ID, gene_B_ID ...

Source

generated with `data-raw/example_data.R`

file_extension *Get file extension*

Description

Get file extension

Usage

```
file_extension(x)
```

Arguments

x	string filepath

Value

string file extension

Examples

```
file_extension('filename.ext')
```

filter_enrichment *Filter enrichment*

Description

Search and filter and sort or summarize (compiled) enrichment output.

Usage

```
filter_enrichment(
  df,
  genes_input = "",
  genes_any_all = c("any", "all"),
  terms_query = "",
  terms_query_all_any = c("any", "all"),
  terms_antiquery = "",
  terms_antiquery_all_any = c("any", "all"),
  min_ngenes = 0,
  min_ngenes_input = 0,
  min_ngenes_signif = 0,
  min_abs_zscore = 0,
  min_pvalue_adjust = 0,
  max_ngenes = 1e+06,
  max_ngenes_input = 1e+06,
  max_ngenes_signif = 1e+06,
  max_abs_zscore = 1e+06,
  max_pvalue_adjust = 1
)
```

Arguments

`df`	enrichment output dataframe
`genes_input`	default: UI input/character vector of genes to select df terms for
`genes_any_all`	default: 'any', else 'all', use to define to take only specific terms containing any or all associated genes
`terms_query`	dfeault: UI input/character vector of keywords to match (grepl) term names
`terms_query_all_any`	default: 'any', else 'all', defines if terms should match any or all of the query keywords given
`terms_antiquery`	dfeault: UI input/character vector of keywords to NOT match (grepl) term names
`terms_antiquery_all_any`	default: 'any', else 'all', defines if terms should NOT match any or all of the query keywords given
`min_ngenes`	default: 0, set higher to filter terms with less n genes
`min_ngenes_input`	default: 0, else set higher to filter terms with less n input genes
`min_ngenes_signif`	default: 0, set higher to filter terms with less n significant genes
`min_abs_zscore`	default: 0, set higher to filter terms with less absolute zscore
`min_pvalue_adjust`	default: 0, set higher to filter terms with lower multiple testing corrected p-value
`max_ngenes`	default: 0, set lower to filter terms with more n genes
`max_ngenes_input`	default: 0, else set lower to filter terms with more n input genes
`max_ngenes_signif`	default: 0, set lower to filter terms with more n significant genes
`max_abs_zscore`	default: 0, set lower to filter terms with more absolute zscore
`max_pvalue_adjust`	default: 1, set lower to filter terms with higher adjusted p-value for multiple correction

Value

filtered dataframe

Examples

```
filter_enrichment(
get(load(system.file("extdata", "example_enrichment.rda", package = "goatea"))),
min_ngenes = 15)
```

`get_base_folder` *Set base folder*

Description

sets/gets given folder path if provided else checks in order:

- path.expand("~") (tilde (~) expands to HOME folder path)
- Sys.getenv("R_USER") (set on R session start)
- Sys.getenv("USERPROFILE") (Windows specific)

Usage

```
get_base_folder(folder_path = NULL)
```

Arguments

`folder_path`	character, default NULL, else existing directory

Value

character folder path

Examples

```
get_base_folder()
```

`get_ppigraph` *Get PPI igraph*

Description

Uses Leiden clustering on modularity for community detection. Leiden was chosen as default as expected PPI data is not inherently hierarchical, which is why modularity optimalization is used on the graph topology. Expected PPI data comes from genes/proteins (of interest) selected from gene set enrichment analysis or differential expression analysis. Using clustering from terms is not possible, as genes can be in multiple terms. Leiden also scales well to large graphs, has consistent clustering outcomes and provides some inherent guarantees by its method, e.g. locally optimal assignment.

Usage

```
get_ppigraph(ppi_data, vertex_clustering = NULL)
```

Arguments

`ppi_data`	dataframe, PPI by aliases/ids in columns 'from' and 'to'
`vertex_clustering`	NULL, else numerical vector of cluster IDs

Value

igraph object of PPI data

References

Traag, V.A., Waltman, L. & van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9, 5233 (2019). https://doi.org/10.1038/s41598-019-41695-z

Examples

```
get_ppigraph(
  get(load(system.file("extdata", "example_ppi_data.rda", package = "goatea")))
)
```

`get_string_ppi` *Get STRING database Protein-Protein Interactions*

Description

STRING documentation: https://string-db.org/cgi/help?sessionId=baEZCS5u1RdM

Protocol used for downloading STRING files is https

Usage

```
get_string_ppi(
  aliases,
  score_threshold = 0L,
  organism = 9606L,
  network_type = "full",
  link_data = "combined_only",
  folder = tempdir(),
  version = "latest",
  versions = NULL
)
```

Arguments

`aliases`	character, vector with protein/gene symbols/aliases
`score_threshold`	integer, default: 0, to get all PPI, ranges between `[0-1000]`, 200 for low, 400 for medium and 700 for high/stringent scoring PPI
`organism`	integer, default: 9606 (Homo Sapiens), see `?goat::load_genesets_go_bioconductor` taxid parameter for possible organism taxIDs
`network_type`	character, default: 'full', else 'physical' for only STRING documented physical interactions
`link_data`	character, default: 'combined_only', else 'full' or 'detailed', see STRING documentation

`folder`	character, default: tempdir(), else given folder path for where to download STRING files, converted to .parquet for compression and query efficiency, if tempdir() the temporary directory with the downloaded files are removed after the R session
`version`	character, default: 'latest', else a version to check availability, e.g. "12.0", if version not available the available versions are printed
`versions`	NULL, else character vector with versions to choose from with version

Value

dataframe (tibble) with protein-protein interactions (symbols and STRING IDs) and STRING combined score

Examples

```
get_string_ppi(c("TP53", "EGFR", "BRCA1", "MTOR", "MYC", "SOX2"))
```

`get_terms_by_keywords` *Get term names by searching with (partial) keywords*

Description

Get term names by searching with (partial) keywords

Usage

```
get_terms_by_keywords(patterns, terms, pos_neg = "pos", all_any = "all")
```

Arguments

`patterns`	keywords to match (grepl) term names
`terms`	character vector to be grepl searched
`pos_neg`	return positive matches or negate matches
`all_any`	need all or any patterns to match search terms

Value

character vector with matching terms by patterns

Examples

```
get_terms_by_keywords('circa', c('circadian rhythm', 'no match', 'circadian clock'))
```

`get_visNetwork` *Get visNetwork graph*

Description

Gets visNetwork graph with ppigraph, and optionally genes overview, metadata

Usage

```
get_visNetwork(ppigraph, genes_overview = NULL, sample_name = NULL)
```

Arguments

`ppigraph`	igraph object, get from `get_ppigraph()`
`genes_overview`	(optional) dataframe, default: NULL, else metadata dataframe for ppigraph proteins/genes aliases
`sample_name`	(optional) character, default: NULL, else sample name found in genes_overview columns

Value

list of visNetwork nodes and edges and given ppigraph

Examples

```
ppi_graph <- get_ppigraph(
  get(load(system.file("extdata", "example_ppi_data.rda", package = "goatea")))
)
get_visNetwork(ppi_graph)
```

`goatea_server` *Server for goatea package*

Description

Server for goatea package

Usage

```
goatea_server(input, output, session, css_colors)
```

Arguments

`input`	Shiny input elements handling
`output`	Shiny input elements handling
`session`	Shiny handling reactivity in app
`css_colors`	see app.R, user set manual colors for the GOATEA UI

Value

Shiny server function

goatea_ui *UI for GOATEA package*

Description

UI for GOATEA package

Usage

```
goatea_ui()
```

Value

Shiny UI function

hexcolor2rgba *Hex code colors to rgba format*

Description

Hex code colors to rgba format

Usage

```
hexcolor2rgba(hexcolors, alpha = NULL)
```

Arguments

`hexcolors`	character (vector), hexcode colors (e.g. #FFFFFF)
`alpha`	numeric in range `[0-1]`, default: NULL to use full opacity or given opacity (AA) in hexcolors (#RRGGBBAA)

Value

colors in rgba format

Examples

```
colors <- colorify(5)
hexcolor2rgba(colors)
hexcolor2rgba(colors, alpha = .5)
colors <- gsub('FF$', 75, colors)
hexcolor2rgba(colors)
hexcolor2rgba(colors, alpha = .5)
```

`palette_name_mapping` *Palette original name mapping*

Description

All ColorBrewer palettes overlap with grDevices palettes Viridis palettes, except "Magma", overlap with grDevices palettes

Usage

```
palette_name_mapping(palette)
```

Arguments

`palette`	string: name of palette, will be lower()ed and stripped of whitespace

Value

original palette name

`plot_ComplexHeatmap` *Plot ComplexHeatmap*

Description

Plot ComplexHeatmap from enrichment analysis results and corresponding genelist

Usage

```
plot_ComplexHeatmap(
  enrichment_result,
  genelist,
  genes = NULL,
  cluster_method = "single",
  n_cluster = 1,
  n_top_terms = NA,
  n_top_genes = NA,
  genelist_overlap = NULL,
  plot = FALSE
)
```

Arguments

`enrichment_result`	dataframe containing enrichment analysis results. Must include `name` (gene set names) and `symbol` (listed genes associated with gene sets)
`genelist`	dataframe with gene-level statistics, including at least `symbol`, `pvalue`, `effectsize`, and `signif` columns
`genes`	character, default: NULL, if genes given, these are prioritized for visualization

`cluster_method`	default: 'single', else one of hclust methods
`n_cluster`	default: 1, integer, number of hierarchical clusters to define
`n_top_terms`	default: NULL, if integer, plot only top genesets (recommended for visual clarity: 70)
`n_top_genes`	default: NULL, if integer, plot only top genes (recommended for visual clarity: 150)
`genelist_overlap`	(Optional) dataframe with gene overlap information, including `symbol` and `genelist_overlap`, see run_genelists_overlap()
`plot`	default: FALSE, if TRUE, display drawn ComplexHeatmap

Value

A **ComplexHeatmap** object displaying genesets (rows) and genes (columns), potentially clustered based on their binary associations. The heatmap includes:

- Row annotations: Gene set size, p-value, and average effect size.
- Column annotations: Gene p-values, effect sizes, and optional overlap categories.
- Customized row/column labels highlighting significant elements.
- A color-mapped heatmap showing clustering results.

Examples

```
plot_ComplexHeatmap(
 get(load(system.file("extdata", "example_enrichment.rda", package = "goatea")))[seq.int(1, 3), ],
  get(load(system.file("extdata", "example_genelist.rda", package = "goatea"))),
  n_cluster = 3,
  n_top_genes = 10
)
```

`plot_EnhancedVolcano` *Plot EnhancedVolcano*

Description

Plot EnhancedVolcano

Usage

```
plot_EnhancedVolcano(
  genelist,
  effectsize_threshold = 1,
  pvalue_threshold = 0.05,
  background_color = "black",
  foreground_color = "white",
  interactive = FALSE,
  legend_labels = c("NS", "FC", "P", "FC & P"),
  x_label = "effectsize (FC)",
  y_label = "-log10(pvalue) (P)",
  title = "Volcano plot",
  subtitle = "EnhancedVolcano",
  caption = paste0("N genes: ", nrow(genelist)),
  label_size = 3,
  legend_label_size = 14,
  axes_label_size = 18,
  point_size = 2
)
```

Arguments

`genelist`	UI value/list of tibbles/dataframes
`effectsize_threshold`	numeric, default: 1, threshold for showing significance on effectsize axis
`pvalue_threshold`	numeric, default: 0.05, threshold for showing significance on pvalue axis
`background_color`	default: ’black’, else character hexcolor or colorname
`foreground_color`	default: ’white’, else character hexcolor or colorname
`interactive`	default: FALSE, else TRUE
`legend_labels`	character vector, default: c(’NS’, ’FC’, ’P’, ’FC & P’), plot legend labels
`x_label`	character, default: ’effectsize (FC)’, plot x-axis label
`y_label`	character, default: "’-log10(pvalue) (P)’, plot y-axis label
`title`	character, default: ’Volcano plot’, plot title
`subtitle`	character, default: ’EnhancedVolcano’, plot subtitle
`caption`	character, default: paste0("N genes: ", nrow(genelist)), plot caption
`label_size`	numeric, default: 3, plot variable label size
`legend_label_size`	numeric, default: 14, plot legend label size
`axes_label_size`	numeric, default: 18, plot x- and y-axis lable sizes
`point_size`	numeric, default: 2, plot point size

Value

plotly or ggplot2 object

Examples

```
plot_EnhancedVolcano(
  get(load(system.file("extdata", "example_genelist.rda", package = "goatea")))
)
```

plot_genelists_overlap_upsetjs

Visualize genelists gene overlap in an interactive UpSet plot

Description

UpSetJS examples: https://upset.js.org/integrations/r/articles/combinationModes.html#distinct-intersection-mode

Usage

```
plot_genelists_overlap_upsetjs(
  genelists,
  mode = "distinct",
  interactive = FALSE,
  main.color = "black",
  highlight.color = "green"
)
```

Arguments

genelists	UI value/list of tibbles/dataframes
mode	string, default: 'intersect', else 'distinct' or 'union' - how to overlap the listed genes
interactive	default: FALSE, else TRUE
main.color	default: 'white' else character hexcolor or colorname
highlight.color	default: 'green' else character hexcolor or colorname

Value

upset plot

Examples

```
plot_genelists_overlap_upsetjs(list(
  A = get(load(system.file("extdata", "example_genelist.rda", package = "goatea"))),
  B = get(load(system.file("extdata", "example_genelist.rda", package = "goatea")))
))
```

plot_gene_effectsize_ComplexHeatmap

Plot gene2effectsize ComplexHeatmap

Description

Plot gene2effectsize ComplexHeatmap

Usage

```
plot_gene_effectsize_ComplexHeatmap(
  genes,
  genes_overview,
  rows_dendrogram = TRUE,
  cols_dendrogram = TRUE,
  plot_n_genes = 50
)
```

Arguments

`genes`	character, genes to visualize
`genes_overview`	dataframe, containing columns: 'symbol', 'SAMPLE_efsi' and 'SAMPLE_pval'
`rows_dendrogram`	default: FALSE, TRUE to cluster rows and show dendrogram
`cols_dendrogram`	default: FALSE, TRUE to cluster columns and show dendrogram
`plot_n_genes`	integer, default: 50, NULL to plot all genes

Value

ComplexHeatmap object

Examples

```
plot_gene_effectsize_ComplexHeatmap(
  c('gene_1', 'gene_2', 'gene_3', 'gene_4', 'gene_5'),
  get(load(system.file("extdata", "example_genes_overview.rda", package = "goatea")))
)
```

`plot_splitdot` *Plot splitdot plot*

Description

Plot splitdot plot

Usage

```
plot_splitdot(enrichment, topN = NA)
```

Arguments

`enrichment`	GOAT enrichment result
`topN`	default: NA to plot all, else integer to plot topN terms by adjusted pvalue

Value

ggplot2 object

Examples

```
plot_splitdot(
  get(load(system.file("extdata", "example_enrichment.rda", package = "goatea")))
)
```

plot_termtree *Plot semantic similarity termtree*

Description

Plot semantic similarity termtree

Usage

```
plot_termtree(
  genelist,
  map_organism = 9606,
  effectsize_threshold = 1,
  Nterms = NA,
  Nwords = 5,
  Nclusters = 3
)
```

Arguments

genelist	GOAT current genelist from selected enrichment sample
map_organism	integer, default: 9606 (human) - input organism ID that will be mapped to org.Xx.eg.db
effectsize_threshold	numerical, default: 1 - genelist effectsize threshold
Nterms	integer, default: NA to plat all terms, integer sets amount of terms to plot
Nwords	integer, default: 5, sets N summarized words per cluster
Nclusters	integer, default: 1, sets N clusters of terms

Value

ggtree/gg/ggplot object

Examples

```
plot_termtree(get(load(system.file("extdata", "example_genelist.rda", package = "goatea"))))
```

`process_string_input` *Process Shiny area input string*

Description

Process Shiny area input string

Usage

```
process_string_input(string_input)
```

Arguments

`string_input`	siny string

Value

processed string - no whitespace, enters, only letters and numbers

Examples

```
process_string_input("test string \n")
```

`read_validate_genelist` *Read and validate a table with genes (that should be tested in overrepresentation-analysis) for compatibility with this R package#'*

Description

if 'pvalue' is not in the genelist columns, it is set and defaulted to 1 for visualization purposes if 'effectsize' is not in the genelist columns, it is set and defaulted to 0 for visualization purposes

Usage

```
read_validate_genelist(
  file,
  remove_non_numerical_ids = TRUE,
  remove_duplicated = TRUE,
  remove_Rik_genes = TRUE,
  remove_Gm_genes = TRUE,
  map_organism = NULL
)
```

Arguments

`file`	full filepath to gene tibble in .csvs/.xlsx/.tsv
`remove_non_numerical_ids`	boolean, default TRUE, if non-numerical in gene column, remove
`remove_duplicated`	boolean, default TRUE, removes duplicated gene symbols/ids
`remove_Rik_genes`	boolean, default TRUE, grepl("Rik$") search and remove Riken non-canonical mouse genes
`remove_Gm_genes`	boolean, default TRUE, grepl("^Gm") search and remove Gm non-canonical mouse genes
`map_organism`	default: NULL, if numeric taxid, used for selecting org.Xx.eg.db to map gene symbols to gene column via AnnotationDbi::mapIds(keytype = 'ALIAS') - if mapped to NA the genes are removed - need to download org.Xx.eg.db manually! Symbols are set toupper() to match formatting. Protein symbols could be used too.

- 9606 = Human (Homo sapiens) (org.Hs.eg.db)
- 9544 = Rhesus monkey (Macaca mulatta) (org.Mmu.eg.db)
- 10090 = Mouse (Mus musculus) (org.Mm.eg.db)
- 10116 = Rat (Rattus norvegicus) (org.Rn.eg.db)
- 7227 = Fruit fly (Drosophila melanogaster) (org.Dm.eg.db)
- 6239 = Worm (Caenorhabditis elegans) (org.Ce.eg.db)

Value

tibble dataframe with columns: symbol (string), gene (string as integer ID), pvalue (numeric), effestsize (numeric)

Examples

```
file_path <- system.file("extdata", "example_genelist.csv", package = "goatea")
read_validate_genelist(file = file_path)
```

`rename_gene_overview` *Rename the gene overview*

Description

Rename the gene overview

Usage

```
rename_gene_overview(names, genes_overview)
```

Arguments

`names`	names to rename gene overview
`genes_overview`	UI given genes overview dataframe (rv_genelists_overlap$gene_overview)

Value

genes overview renamed

`run_genelists_overlap` *Create gene overview through overlapping genelists information by overlapping significant genes*

Description

Create gene overview through overlapping genelists information by overlapping significant genes

Usage

```
run_genelists_overlap(genelists)
```

Arguments

`genelists`	UI value/list of tibbles/dataframes

Value

tibble/dataframe with (annotated) genes and p-value/effectsize info for each genelist, concluding with overlapping genelists by significant genes

Examples

```
run_genelists_overlap(list(
  A = get(load(system.file("extdata", "example_genelist.rda", package = "goatea"))),
  B = get(load(system.file("extdata", "example_genelist.rda", package = "goatea")))
))
```

`run_geneset_enrichment`
Perform geneset enrichment testing using any supported method

---

Description

See original documentation at test_genesets

Usage

```
run_geneset_enrichment(
  genesets,
  genelist,
  method = "goat",
  score_type = "effectsize",
  padj_method = "BH",
  padj_sources = TRUE,
  padj_cutoff = 0.01,
  padj_min_signifgenes = 0L,
  ...
)
```

Arguments

`genesets`	tibble with genesets, must contain columns 'source', 'source_version', 'id', 'name', 'genes', 'ngenes', 'ngenes_signif'
`genelist`	tibble with genes, must contain column 'gene' and 'test'. gene = character column, which are matched against list column 'genes' in genesets tibble. test = boolean column (you can set all to FALSE if not performing Fisher-exact or hypergeometric test downstream)
`method`	method for overrepresentation analysis. Options: "goat", "hypergeometric", "fisherexact", "fisherexact_ease", "gsea", "idea"
`score_type`	string, default: "effectsize", alternatively set to "pvalue", "effectsize_up", "effectsize_down", "effectsize_abs"
`padj_method`	first step of multiple testing correction; method for p-value adjustment, passed to stats::p.adjust() via padjust_genesets(), e.g. set "BH" to compute FDR adjusted p-values (default) or "bonferroni" for a more stringent procedure
`padj_sources`	second step of multiple testing correction; apply Bonferroni adjustment to all p-values according to the number of geneset sources that were tested. Boolean parameter, set TRUE to enable (default) or FALSE to disable
`padj_cutoff`	cutoff for adjusted p-value, signif column is set to TRUE for all values lesser-equals
`padj_min_signifgenes`	if a value larger than zero is provided, this will perform additional post-hoc filtering; after p-value adjustment, set the pvalue_adjust to NA and signif to FALSE for all genesets with fewer than padj_min_signifgenes 'input genes that were significant' (ngenes_signif column in genesets table). So this does not affect the accuracy of estimated p-values, in contrast to prefiltering genesets prior to p-value computation or adjusting p-values
`...`	further parameters are passed to the respective stats method

Value

the input genesets, with results stored in columns 'pvalue', 'pvalue_adjust', 'signif' and 'zscore'

Examples

```
run_geneset_enrichment(
  get(load(system.file("extdata", "example_genesets.rda", package = "goatea"))),
  get(load(system.file("extdata", "example_genelist.rda", package = "goatea")))
)
```

---

scale_values_between *Scale values between given min/max*

Description

Scale values between given min/max

Usage

```
scale_values_between(
  values,
  old_min = min(values),
  old_max = max(values),
  new_min = 0,
  new_max = 100
)
```

Arguments

`values`	numeric (vector)
`old_min`	numeric, default: min(values), else set as current expected minimum of values
`old_max`	numeric, default: max(values), else set as current expected maximum of values
`new_min`	numeric, default: 0, else set to wanted new minimum value
`new_max`	numeric, default: 100, else set to wanted new maximum value

Value

scaled numeric values

Examples

```
scale_values_between(c(1,3,1,4,1,6,1,6,5,7))
```

`set_significant_N_genes` *Set significant and number of genes*

Description

Set significant and number of genes

Usage

```
set_significant_N_genes(
  genelist,
  significance_by = "pvalue_effectsize",
  pvalue_threshold = 0.05,
  effectsize_threshold = 1,
  keep_max_n_genes = FALSE,
  keep_max_n_genes_by = "pvalue"
)
```

Arguments

`genelist`	list, loaded genelist with goatea::read_validate_genelist()
`significance_by`	string, default: 'pvalue_effectsize', else 'pvalue' or 'effectsize' to set gene significance to TRUE/FALSE in 'signif' column
`pvalue_threshold`	numeric, default: 0.05, to set gene significance based on pvalue
`effectsize_threshold`	numeric, default: 1, to set gene significance based on effectsize
`keep_max_n_genes`	boolean, default: TRUE, filter down by pvalue to max n genes allowed by goat (max(goat::goat_nulldistributions$N))
`keep_max_n_genes_by`	string, default: 'pvalue', else 'effectsize', order genes based on lowest pvalues or highest absolute effect sizes

Value

genelist with added 'signif' column with TRUE/FALSE values

Examples

```
set_significant_N_genes(
  get(load(system.file("extdata", "example_genelist.rda", package = "goatea")))
)
```

`wrap_hovertip` *Wrap Shiny UI element with a hoverable tooltip contained in html div tags*

Description

Wrap Shiny UI element with a hoverable tooltip contained in html div tags

Usage

```
wrap_hovertip(ui_element, hovertip)
```

Arguments

`ui_element`	Shiny UI element to wrap with hovertext
`hovertip`	text that will show as hover popup

Value

tags$div element around given Shiny UI element

Examples

```
wrap_hovertip(shiny::actionButton('id_example', 'example'), 'example')
```

`wrap_loader` *Wrap Shiny UI element with a loading spinner contained in html div tags*

Description

Wrap Shiny UI element with a loading spinner contained in html div tags

Usage

```
wrap_loader(id, ui_element)
```

Arguments

`id`	string: id of loader, used with show/hide in server side
`ui_element`	wrapped Shiny UI element

Value

html div element wrapped around given Shiny UI element

Examples

```
wrap_loader('id_example', shiny::actionButton('id_example', 'example'))
```

`%>%` *Pipe operator*

Description

See `%>%` for details.

Usage

```
lhs %>% rhs
```

Arguments

`lhs`	A value or the dplyr placeholder.
`rhs`	A function call using the dplyr semantics.

Value

The result of calling `rhs(lhs)`.

Index