Package 'GeneTonic'

November 12, 2026

Title Enjoy Analyzing And Integrating The Results From Differential Expression Analysis And Functional Enrichment Analysis

Version 3.6.0

Date 2025-09-23

Description This package provides functionality to combine the existing pieces of the transcriptome data and results, making it easier to generate insightful observations and hypothesis. Its usage is made easy with a Shiny application, combining the benefits of interactivity and reproducibility e.g. by capturing the features and gene sets of interest highlighted during the live session, and creating an HTML report as an artifact where text, code, and output coexist. Using the GeneTonicList as a standardized container for all the required components, it is possible to simplify the generation of multiple visualizations and summaries.

Depends R (>= 4.0.0)

Imports AnnotationDbi, backbone, bs4Dash (>= 2.0.0), circlize, colorspace, colourpicker, ComplexHeatmap, ComplexUpset, dendextend, DESeq2, dplyr, DT, dynamicTreeCut, expm, ggforce, ggplot2 (>= 3.5.0), ggrepel, ggridges, GO.db, graphics, grDevices, grid, igraph, matrixStats, methods, mosdef (>= 1.1.0), plotly, RColorBrewer, rintrojs, rlang, rmarkdown, S4Vectors, scales, shiny, shinyAce, shinycssloaders, shinyWidgets, stats, SummarizedExperiment, tidyr, tippy, tools, utils, viridis, visNetwork

Suggests knitr, BiocStyle, htmltools, clusterProfiler, macrophage, org.Hs.eg.db, magrittr, testthat (>= 2.1.0)

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/federicomarini/GeneTonic

BugReports https://github.com/federicomarini/GeneTonic/issues

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

biocViews GUI, GeneExpression, Software, Transcription, Transcriptomics, Visualization, DifferentialExpression, Pathways, ReportWriting, GeneSetEnrichment, Annotation, GO, ShinyApps

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

git_branch RELEASE_3_23

git_last_commit 57c6b8d

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Federico Marini [aut, cre] (ORCID: <https://orcid.org/0000-0003-3252-7758>), Annekathrin Nedwed [aut] (ORCID: <https://orcid.org/0000-0002-2475-4945>), Edoardo Filippi [ctb] (ORCID: <https://orcid.org/0009-0003-9858-0137>)

Maintainer Federico Marini <marinif@uni-mainz.de>

Contents

.check_pandoc . . . 3
checkup_GeneTonic . . . 4
checkup_gtl . . . 5
check_colors . . . 7
cluster_markov . . . 8
create_jaccard_matrix . . . 9
create_kappa_matrix . . . 10
create_upsetdata . . . 11
deprecated . . . 12
describe_gtl . . . 13
deseqresult2df . . . 14
distill_enrichment . . . 14
editor_to_vector_sanitized . . . 16
enhance_table . . . 17
enrichment_map . . . 19
enrichr_output_macrophage . . . 21
export_for_iSEE . . . 22
export_to_sif . . . 23
fgseaRes . . . 23
geneinfo_2_html . . . 24
GeneTonic . . . 25
GeneTonic-pkg . . . 27
GeneTonicList . . . 28
gene_plot . . . 30
get_aggrscores . . . 32
get_expression_values . . . 33

ggs_backbone 34
ggs_graph 37
gostres_macrophage 39
go_2_html 40
gs_alluvial 40
gs_dendro 42
gs_fuzzyclustering 44
gs_heatmap 46
gs_horizon 49
gs_mds 51
gs_radar 53
gs_scores 55
gs_scoresheat 57
gs_simplify 59
gs_summary_heat 60
gs_summary_overview 61
gs_summary_overview_pair 63
gs_upset 65
gs_volcano 67
happy_hour 69
map2color 72
overlap_coefficient 73
overlap_jaccard_index 74
res_macrophage_IFNg_vs_naive 74
shake_davidResult 75
shake_enrichResult 75
shake_enrichrResult 77
shake_fgseaResult 78
shake_gprofilerResult 78
shake_gsenrichResult 79
shake_topGOtableResult 81
signature_volcano 81
styleColorBar_divergent 84
summarize_ggs_hubgenes 85
topgoDE_macrophage_IFNg_vs_naive 87

Index **88**

`.check_pandoc` *Check whether* pandoc *and* pandoc-citeproc *are available*

Description

Check whether pandoc and pandoc-citeproc are available

Usage

```
.check_pandoc(ignore_pandoc)
```

Arguments

ignore_pandoc	Logical. If TRUE, just give a warning if one of pandoc or pandoc-citeproc is not available. If FALSE, an error is thrown.

Details

Credits to the original implementation proposed by Charlotte Soneson, upon which this function is **heavily** inspired.

Value

No value is returned. If `pandoc` or `pandoc-citeproc` are missing, either warning or error messages are triggered.

---

checkup_GeneTonic *Checking the input objects for GeneTonic*

Description

Checking the input objects for GeneTonic, whether these are all set for running the app

Usage

```
checkup_GeneTonic(dds, res_de, res_enrich, annotation_obj, verbose = FALSE)
```

Arguments

dds	A `DESeqDataSet` object, normally obtained after running your data through the `DESeq2` framework.
res_de	A `DESeqResults` object. As for the `dds` parameter, this is also commonly used in the `DESeq2` framework.
res_enrich	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
annotation_obj	A `data.frame` object, containing two columns, `gene_id` with a set of unambiguous identifiers (e.g. ENSEMBL ids) and `gene_name`, containing e.g. HGNC-based gene symbols.
verbose	Logical, to control level of verbosity of the messages generated

Details

Some suggestions on the requirements for each parameter are returned in the error messages.

Value

Invisible NULL

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

checkup_GeneTonic(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)
# if all is fine, it should return an invisible NULL and a simple message
```

---

checkup_gtl *Checking the* gtl *input object for GeneTonic*

Description

Checking the gtl ("GeneTonic list") input object for GeneTonic, with the correct content and format expected

Usage

```
checkup_gtl(gtl, verbose = FALSE)
```

Arguments

gtl	A DESeqDataSet object, normally obtained after running your data through the DESeq2 framework. This list should contain • in the dds slot: A DESeqDataSet object • in the res_de: A DESeqResults object • in the res_enrich: A data.frame object, storing the result of the functional enrichment analysis • in the annotation_obj: A data.frame object, containing two columns, gene_id with a set of unambiguous identifiers (e.g. ENSEMBL ids) and gene_name, containing e.g. HGNC-based gene symbols.
verbose	Logical, to control level of verbosity of the messages generated

Details

Some suggestions on the requirements for the gtl are returned in the error messages.

Value

Invisible NULL

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
```

```
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gtl <- list(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)

checkup_gtl(gtl)
# if all is fine, it should return an invisible NULL and a simple message
```

`check_colors` *Check colors*

Description

Check correct specification of colors

Usage

```
check_colors(x)
```

Arguments

x	A vector of strings specifying colors

Details

This is a vectorized version of `grDevices::col2rgb()`

Value

A vector of logical values, one for each specified color - `TRUE` if the color is specified correctly

Examples

```
# simple case
mypal <- c("steelblue", "#FF1100")
check_colors(mypal)
mypal2 <- rev(
  scales::alpha(
    colorRampPalette(RColorBrewer::brewer.pal(name = "RdYlBu", 11))(50), 0.4
  )
```

```
)
check_colors(mypal2)
# useful with long vectors to check at once if all cols are fine
all(check_colors(mypal2))
```

`cluster_markov` *Markov Clustering (MCL) for community detection*

Description

This function implements the Markov Clustering (MCL) algorithm for finding community structure, in an analogous way to other existing algorithms in `igraph`.

Usage

```
cluster_markov(
  g,
  add_self_loops = TRUE,
  loop_value = 1,
  mcl_expansion = 2,
  mcl_inflation = 2,
  allow_singletons = TRUE,
  max_iter = 100,
  return_node_names = TRUE,
  return_esm = FALSE
)
```

Arguments

`g`	The input graph object
`add_self_loops`	Logical, whether to add self-loops to the matrix by setting the diagonal to `loop_value`
`loop_value`	Numeric, the value to use for self-loops
`mcl_expansion`	Numeric, cluster expansion factor for the Markov clustering iteration - defaults to 2
`mcl_inflation`	Numeric, cluster inflation factor for the Markov clustering iteration - defaults to 2
`allow_singletons`	Logical; if `TRUE`, single isolated vertices are allowed to form their own cluster. If set to `FALSE`, all clusters of size = 1 are grouped in one cluster (to be interpreted as background noise).
`max_iter`	Numeric value for the maximum number of iterations for the Markov clustering
`return_node_names`	Logical, if the graph is named and set to `TRUE`, returns the node names.
`return_esm`	Logical, controlling whether the equilibrium state matrix should be returned

Details

This implementation has been driven by the nice explanations provided in

- https://sites.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation2.pdf
- https://medium.com/analytics-vidhya/demystifying-markov-clustering-aeb6cdabbfc7
- https://github.com/GuyAllard/markov_clustering (python implementation)

More info on the MCL: https://micans.org/mcl/index.html, and https://micans.org/mcl/sec_description1.html

Value

This function returns a `communities` object, containing the numbers of the assigned membership (in the slot `membership`). Please see the `igraph::communities()` manual page for additional details

References

van Dongen, S.M., Graph clustering by flow simulation (2000) PhD thesis, Utrecht University Repository - https://dspace.library.uu.nl/handle/1874/848

Enright AJ, van Dongen SM, Ouzounis CA, An efficient algorithm for large-scale detection of protein families (2002) Nucleic Acids Research, Volume 30, Issue 7, 1 April 2002, Pages 1575–1584, https://doi.org/10.1093/nar/30.7.1575

Examples

```
library("igraph")
g <- make_full_graph(5) %du% make_full_graph(5) %du% make_full_graph(5)
g <- add_edges(g, c(1, 6, 1, 11, 6, 11))
cluster_markov(g)
V(g)$color <- cluster_markov(g)$membership
plot(g)
```

`create_jaccard_matrix` *Compute the overlap matrix for enrichment results*

Description

Compute the overlap matrix for enrichment results, based on the Jaccard Index between each pair of sets

Usage

```
create_jaccard_matrix(
  res_enrich,
  gtl = NULL,
  n_gs = nrow(res_enrich),
  gs_ids = NULL,
  return_sym = FALSE
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to see the formatting requirements.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included (from the top ranked ones). Defaults to the number of rows of res_enrich
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included, additionally to the ones specified via n_gs. Defaults to NULL.
return_sym	Logical, whether to return the symmetrical matrix or just the upper triangular - as needed by enrichment_map(), for example.

Value

A matrix with the kappa scores between gene sets

See Also

gs_mds(), enrichment_map()

Examples

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)

jmat <- create_jaccard_matrix(res_enrich[1:200, ])
dim(jmat)
```

---

create_kappa_matrix *Compute the kappa matrix for enrichment results*

Description

Compute the kappa matrix for enrichment results, as a measure of overlap

Usage

```
create_kappa_matrix(
  res_enrich,
  gtl = NULL,
  n_gs = nrow(res_enrich),
  gs_ids = NULL
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to see the formatting requirements.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included (from the top ranked ones). Defaults to the number of rows of res_enrich
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included, additionally to the ones specified via n_gs. Defaults to NULL.

Value

A matrix with the kappa scores between gene sets

See Also

gs_mds()

Examples

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)

kmat <- create_kappa_matrix(res_enrich[1:200, ])
dim(kmat)
```

---

create_upsetdata *Create a geneset upset dataset*

Description

Create a data frame that can be fed to the upset function

Usage

```
create_upsetdata(res_enrich, use_ids = FALSE)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to see the formatting requirements.
use_ids	Logical - whether to use the gs_identifiers as names, or the values provided as gs_description. Defaults to FALSE, using the full descriptions

Value

A data.frame to be used in `ComplexUpset::upset()`

Examples

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)

create_upsetdata(res_enrich[1:20, ])
dim(create_upsetdata(res_enrich[1:20, ]))

create_upsetdata(res_enrich[1:5, ], use_ids = TRUE)
```

`deprecated` *Deprecated functions in GeneTonic*

Description

Functions that are on their way to the function afterlife. Their successors are also listed.

Arguments

...	Ignored arguments.

Details

The successors of these functions are likely coming after the rework that led to the creation of the `mosdef` package. See more into its documentation for more details.

Value

All functions throw a warning, with a deprecation message pointing towards its descendent (if available).

Transitioning to the mosdef framework

- `deseqresult2df()`, now replaced by the more consistent `mosdef::deresult_to_df()`. The only change in its functionality concerns the parameter names
- `gene_plot()` has now been moved to `mosdef::gene_plot()`, where it generalizes with respect to the container of the DE workflow object. In a similar fashion, `get_expression_values()` is replaced by `mosdef::get_expr_values()`. Both functions lose the `gtl` parameter, which was anyways not really exploited throughout the different calls in the package
- `go_2_html()` and `geneinfo_2_html()` have been replaced by the more information-rich (and flexible) `mosdef::go_to_html()` and `mosdef::geneinfo_to_html()`, respectively. No change is expected for the end user

- map2color() and styleColorBar_divergent() are now moved to the mosdef::map_to_color() and mosdef::styleColorBar_divergent(), again with no visible changes for the end user
- The internally defined functions .link2amigo(), .link2ncbi(), .link2genecards() and .link2gtex() are now replaced by the equivalent functions in mosdef: mosdef::create_link_GO(), mosdef::create_link_NCBI(), mosdef::create_link_GeneCards() and mosdef::create_link_GO(). Notably, the mosdef package expanded on the concept of automatically generated buttons, taking this to the extreme of efficiency with the mosdef::buttonifier() function

Author(s)

Federico Marini

Examples

```
# try(deseqresult2df())
```

---

describe_gtl *Describe a GeneTonic list*

Description

Obtain a quick textual overview of the essential features of the components of the GeneTonic list object

Usage

```
describe_gtl(gtl)
```

Arguments

gtl	A GeneTonic-list object, containing in its named slots the required dds, res_de, res_enrich, and annotation_obj

Value

A character string, that can further be processed (e.g. by message() or cat(), or easily rendered inside Shiny's renderText elements)

deseqresult2df *Generate a table from the* DESeq2 *results*

Description

Generate a tidy table with the results of DESeq2

Usage

```
deseqresult2df(res_de, FDR = NULL)
```

Arguments

res_de	A DESeqResults object.
FDR	Numeric value, specifying the significance level for thresholding adjusted p-values. Defaults to NULL, which would return the full set of results without performing any subsetting based on FDR.

Value

A tidy data.frame with the results from differential expression, sorted by adjusted p-value. If FDR is specified, the table contains only genes with adjusted p-value smaller than the value.

Examples

```
data(res_de_macrophage, package = "GeneTonic")
head(res_macrophage_IFNg_vs_naive)
res_df <- mosdef::deresult_to_df(res_macrophage_IFNg_vs_naive)
head(res_df)
```

distill_enrichment *Distill enrichment results*

Description

Distill the main topics from the enrichment results, based on the graph derived from constructing an enrichment map

Usage

```
distill_enrichment(
  res_enrich,
  res_de,
  annotation_obj,
  gtl = NULL,
  n_gs = nrow(res_enrich),
  cluster_fun = "cluster_markov"
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis.
res_de	A DESeqResults object. As for the dds parameter, this is also commonly used in the DESeq2 framework.
annotation_obj	A data.frame object, containing two columns, gene_id with a set of unambiguous identifiers (e.g. ENSEMBL ids) and gene_name, containing e.g. HGNC-based gene symbols.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be used.
cluster_fun	Character, referring to the name of the function used for the community detection in the enrichment map graph. Could be one of "cluster_markov", "cluster_louvain", or "cluster_walktrap", as they all return a communities object.

Value

A list containing three objects:

- the distilled table of enrichment, distilled_table, where the new meta-genesets are identified and defined, specifying e.g. the names of each component, and the genes associated to these.
- the distilled graph for the enrichment map, distilled_em, with the information on the membership
- the original res_enrich, augmented with the information of the membership related to the meta-genesets

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
```

```
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

distilled <- distill_enrichment(res_enrich,
  res_de,
  annotation_obj,
  n_gs = 100,
  cluster_fun = "cluster_markov"
)
colnames(distilled$distilled_table)
distilled$distilled_em
```

`editor_to_vector_sanitized` *Extract vectors from editor content*

Description

Extract vectors from the shinyAce editor content, also removing comments and whitespaces from text.

Usage

```
editor_to_vector_sanitized(txt)
```

Arguments

`txt`	A single character text input.

Value

A character vector representing valid lines in the text input of the editor.

enhance_table *Visually enhances a functional enrichment result table*

Description

Creates a visual summary for the results of a functional enrichment analysis, by displaying also the components of each gene set and their expression change in the contrast of interest

Usage

```
enhance_table(
  res_enrich,
  res_de,
  annotation_obj,
  gtl = NULL,
  n_gs = 50,
  gs_ids = NULL,
  chars_limit = 70,
  plot_style = c("point", "ridgeline"),
  ridge_color = c("gs_id", "gs_score"),
  plot_title = NULL
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation. information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed.
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be displayed.
chars_limit	Integer, number of characters to be displayed for each geneset name.
plot_style	Character value, one of "point" or "ridgeline". Defines the style of the plot to summarize visually the table.
ridge_color	Character value, one of "gs_id" or "gs_score", controls the fill color of the ridge lines. If selecting "gs_score", the z_score column must be present in the enrichment results table - see get_aggrscores() to do that.
plot_title	Character string, used as title for the plot. If left NULL, it defaults to a general description of the plot and of the DE contrast.

Value

A ggplot object

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)
enhance_table(res_enrich,
  res_de,
  anno_df,
  n_gs = 10
)

# using the ridge line as a style, also coloring by the Z score
res_enrich_withscores <- get_aggrscores(
  res_enrich,
  res_de,
  anno_df
)
enhance_table(res_enrich_withscores,
  res_de,
  anno_df,
  n_gs = 10,
```

```
  plot_style = "ridgeline",
  ridge_color = "gs_score"
)
```

`enrichment_map` *Creates an enrichment map for the results of functional enrichment*

Description

Generates a graph for the enrichment map, combining information from `res_enrich` and `res_de`. This object can be further plotted, e.g. statically via `igraph::plot.igraph()`, or dynamically via visNetwork::visIgraph()

Usage

```
enrichment_map(
  res_enrich,
  res_de,
  annotation_obj,
  gtl = NULL,
  n_gs = 50,
  gs_ids = NULL,
  overlap_threshold = 0.1,
  scale_edges_width = 200,
  scale_nodes_size = 5,
  color_by = "gs_pvalue"
)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
`res_de`	A `DESeqResults` object.
`annotation_obj`	A `data.frame` object with the feature annotation information, with at least two columns, `gene_id` and `gene_name`.
`gtl`	A `GeneTonic`-list object, containing in its slots the arguments specified above: `dds`, `res_de`, `res_enrich`, and `annotation_obj` - the names of the list *must* be specified following the content they are expecting
`n_gs`	Integer value, corresponding to the maximal number of gene sets to be displayed
`gs_ids`	Character vector, containing a subset of `gs_id` as they are available in `res_enrich`. Lists the gene sets to be displayed.
`overlap_threshold`	Numeric value, between 0 and 1. Defines the threshold to be used for removing edges in the enrichment map - edges below this value will be excluded from the final graph. Defaults to 0.1.

scale_edges_width
: A numeric value, to define the scaling factor for the edges between nodes. Defaults to 200 (works well chained to `visNetwork` functions).

scale_nodes_size
: A numeric value, to define the scaling factor for the node sizes. Defaults to 5 - works well chained to `visNetwork` functions.

color_by
: Character, specifying the column of `res_enrich` to be used for coloring the plotted gene sets. Defaults to `gs_pvalue`.

Value

An `igraph` object to be further manipulated or processed/plotted

See Also

`GeneTonic()` embeds an interactive visualization for the enrichment map

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

em <- enrichment_map(res_enrich,
```

```
  res_de,
  anno_df,
  n_gs = 20
)

em

# could be viewed interactively with
# library("visNetwork")
# library("magrittr")
# em %>%
#   visIgraph() %>%
#   visOptions(highlightNearest = list(enabled = TRUE,
#                                      degree = 1,
#                                      hover = TRUE),
#              nodesIdSelection = TRUE)
```

`enrichr_output_macrophage` *A sample output from Enrichr*

Description

A sample output object as created from a call to Enrichr, with the interface provided by `enrichR` - using the `enrichr()` function

Details

This object has been created on the data from the `macrophage` package by analyzing downstream the differentially expressed genes when comparing IFNg treated samples vs naive samples, accounting for the different cell lines included.

Details on how this object has been created are included in the `create_gt_data.R` script, included in the `scripts` folder of the `GeneTonic` package.

References

Alasoo, et al. "Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response", Nature Genetics, January 2018 doi: 10.1038/s41588-018-0046-7.

See Also

Other pathway-analysis-results: `gostres_macrophage`, `topgoDE_macrophage_IFNg_vs_naive`

export_for_iSEE *export_for_iSEE*

Description

Combine data from a typical DESeq2 run

Usage

```
export_for_iSEE(dds, res_de, gtl = NULL)
```

Arguments

dds	A DESeq2::DESeqDataSet() object.
res_de	A DESeq2::DESeqResults() object.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting

Details

Combines the DESeqDataSet input and DESeqResults into a SummarizedExperiment object, which can be readily explored with iSEE.

A typical usage would be after running the DESeq2 pipeline and/or after exploring the functional enrichment results with GeneTonic()

Value

A SummarizedExperiment object, with raw counts, normalized counts, and variance-stabilizing transformed counts in the assay slots; and with colData and rowData extracted from the corresponding input parameters - mainly the results for differential expression analysis.

Examples

```
library("macrophage")
library("DESeq2")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# now everything is in place to launch the app
```

```
# dds_macrophage <- DESeq2::DESeq(dds_macrophage)
se_macrophage <- export_for_iSEE(dds_macrophage, res_de)
# iSEE(se_macrophage)
```

`export_to_sif` *Export to sif*

Description

Export a graph to a Simple Interaction Format file

Usage

```
export_to_sif(g, sif_file = "", edge_label = "relates_to")
```

Arguments

`g`	An `igraph` object
`sif_file`	Character string, the path to the file where to save the exported graph as .sif file
`edge_label`	Character string, defining the name of the interaction type. Defaults here to "relates_to"

Value

Returns the path to the exported file, invisibly

Examples

```
library("igraph")
g <- make_full_graph(5) %du% make_full_graph(5) %du% make_full_graph(5)
g <- add_edges(g, c(1, 6, 1, 11, 6, 11))
export_to_sif(g, tempfile())
```

`fgseaRes` *A sample output from fgsea*

Description

A sample output object as created from a call to the `fgsea()` function, in the `fgsea` package, as a practical framework for performing GSEA

Details

This object has been created on the data from the `macrophage` package by analyzing downstream the differentially expressed genes when comparing IFNg treated samples vs naive samples, accounting for the different cell lines included.

References

Alasoo, et al. "Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response", Nature Genetics, January 2018 doi: 10.1038/s41588-018-0046-7.

`geneinfo_2_html` *Information on a gene*

Description

Assembles information, in HTML format, regarding a gene symbol identifier

Usage

```
geneinfo_2_html(gene_id, res_de = NULL)
```

Arguments

`gene_id`	Character specifying the gene identifier for which to retrieve information
`res_de`	A `DESeqResults` object, storing the result of the differential expression analysis. If not provided, the experiment-related information is not shown, and only some generic info on the identifier is displayed. The information about the gene is retrieved by matching on the `SYMBOL` column, which should be provided in `res_de`.

Details

Creates links to the NCBI and the GeneCards databases

Value

HTML content related to a gene identifier, to be displayed in web applications (or inserted in Rmd documents)

Examples

```
geneinfo_2_html("ACTB")
geneinfo_2_html("Pf4")
```

GeneTonic *GeneTonic*

Description

GeneTonic, main function for the Shiny app

Usage

```
GeneTonic(
  dds = NULL,
  res_de = NULL,
  res_enrich = NULL,
  annotation_obj = NULL,
  gtl = NULL,
  project_id = "",
  size_gtl = 50
)
```

Arguments

dds	A DESeqDataSet object, normally obtained after running your data through the DESeq2 framework.
res_de	A DESeqResults object. As for the dds parameter, this is also commonly used in the DESeq2 framework.
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. Required columns for enjoying the full functionality of GeneTonic() include: • a gene set identifier (e.g. GeneOntology id, gs_id) and its term description (gs_description) • a numeric value for the significance of the enrichment (gs_pvalue) • a column named gs_genes containing a comma separated vector of the gene names associated to the term, one for each term • the number of genes in the geneset of interest detected as differentially expressed (gs_de_count), or in the background set of genes (gs_bg_count) See shake_topGOtableResult() or shake_enrichResult() for examples of such formatting helpers
annotation_obj	A data.frame object, containing two columns, gene_id with a set of unambiguous identifiers (e.g. ENSEMBL ids) and gene_name, containing e.g. HGNC-based gene symbols. This object can be constructed via the org.eg.XX.db packages, e.g. with convenience functions such as mosdef::get_annotation_orgdb().
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
project_id	A character string, which can be considered as an identifier for the set/session, and will be e.g. used in the title of the report created via happy_hour()

size_gtl — Numeric value, specifying the maximal size in MB for the accepted GeneTonicList object - this applies when uploading the dataset at runtime

Value

A Shiny app object is returned, for interactive data exploration

Author(s)

Federico Marini

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)


# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

# now everything is in place to launch the app
if (interactive()) {
  GeneTonic(
    dds = dds_macrophage,
    res_de = res_de,
    res_enrich = res_enrich,
```

```
    annotation_obj = anno_df,
    project_id = "myexample"
  )
}
# alternatively...
gtl_macrophage <- GeneTonicList(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)
# GeneTonic(gtl = gtl_macrophage)

# if running it "as a server", without input data specified:
if (interactive()) {
  GeneTonic(size_gtl = 300)   # for fairly large gtl objects
}
```

GeneTonic-pkg *GeneTonic*

Description

GeneTonic is a Bioconductor package that provides an interactive Shiny-based graphical user interface for streamlining the interpretation of RNA-seq data

Details

GeneTonic simplifies and optimizes the integration of all components of Differential Expression analysis, with functional enrichment analysis and the original expression quantifications. It does so in a way that makes it easier to generate insightful observations and hypothesis - combining the benefits of interactivity and reproducibility, e.g. by capturing the features and gene sets of interest highlighted during the live session, and creating an HTML report as an artifact where text, code, and output coexist.

Author(s)

Maintainer: Federico Marini <marinif@uni-mainz.de> (ORCID)

Authors:

- Annekathrin Nedwed <anneludt@uni-mainz.de> (ORCID)

Other contributors:

- Edoardo Filippi <edoardo.filippi@uni-mainz.de> [contributor]

See Also

Useful links:

- https://github.com/federicomarini/GeneTonic
- Report bugs at https://github.com/federicomarini/GeneTonic/issues

---

`GeneTonicList` *Create a GeneTonicList object*

Description

Create a list for GeneTonic from the single required components.

Usage

```
GeneTonicList(dds, res_de, res_enrich, annotation_obj)

GeneTonic_list(dds, res_de, res_enrich, annotation_obj)
```

Arguments

`dds`	A `DESeqDataSet` object, normally obtained after running your data through the `DESeq2` framework.
`res_de`	A `DESeqResults` object. As for the `dds` parameter, this is also commonly used in the `DESeq2` framework.
`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. Required columns for enjoying the full functionality of `GeneTonic()` include: • a gene set identifier (e.g. GeneOntology id, `gs_id`) and its term description (`gs_description`) • a numeric value for the significance of the enrichment (`gs_pvalue`) • a column named `gs_genes` containing a comma separated vector of the gene names associated to the term, one for each term • the number of genes in the geneset of interest detected as differentially expressed (`gs_de_count`), or in the background set of genes (`gs_bg_count`) See `shake_topGOtableResult()` or `shake_enrichResult()` for examples of such formatting helpers
`annotation_obj`	A `data.frame` object, containing two columns, `gene_id` with a set of unambiguous identifiers (e.g. ENSEMBL ids) and `gene_name`, containing e.g. HGNC-based gene symbols. This object can be constructed via the `org.eg.XX.db` packages, e.g. with convenience functions such as `mosdef::get_annotation_orgdb()`.

Details

Having this dedicated function saves the pain of remembering which names the components of the list should have. For backwards compatibility, the `GeneTonic_list` function is still provided as a synonim, and will likely be deprecated in the upcoming release cycles.

Value

A GeneTonic-list object, containing in its named slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list are specified following the requirements for using it as single input to GeneTonic()

Author(s)

Federico Marini

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)


# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gtl_macrophage <- GeneTonicList(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)

# now everything is in place to launch the app
```

```
if (interactive()) {
  GeneTonic(gtl = gtl_macrophage)
}
```

`gene_plot` *Plot expression values for a gene*

Description

Plot expression values (e.g. normalized counts) for a gene of interest, grouped by experimental group(s) of interest

Usage

```
gene_plot(
  dds,
  gene,
  intgroup = NULL,
  assay = "counts",
  annotation_obj = NULL,
  normalized = TRUE,
  transform = TRUE,
  labels_display = TRUE,
  labels_repel = TRUE,
  plot_type = "auto",
  return_data = FALSE,
  gtl = NULL
)
```

Arguments

`dds`	A `DESeqDataSet` object, normally obtained after running your data through the `DESeq2` framework.
`gene`	Character, specifies the identifier of the feature (gene) to be plotted
`intgroup`	A character vector of names in `colData(dds)` to use for grouping. Note: the vector components should be categorical variables. Defaults to NULL, which which would then select the first column of the `colData` slot.
`assay`	Character, specifies with assay of the `dds` object to use for reading out the expression values. Defaults to "counts".
`annotation_obj`	A `data.frame` object with the feature annotation information, with at least two columns, `gene_id` and `gene_name`.
`normalized`	Logical value, whether the expression values should be normalized by their size factor. Defaults to TRUE, applies when `assay` is "counts"
`transform`	Logical value, corresponding whether to have log scale y-axis or not. Defaults to TRUE.

labels_display	Logical value. Whether to display the labels of samples, defaults to TRUE.
labels_repel	Logical value. Whether to use ggrepel's functions to place labels; defaults to TRUE
plot_type	Character, one of "auto", "jitteronly", "boxplot", "violin", or "sina". Defines the type of geom_ to be used for plotting. Defaults to auto, which in turn chooses one of the layers according to the number of samples in the smallest group defined via intgroup
return_data	Logical, whether the function should just return the data.frame of expression values and covariates for custom plotting. Defaults to FALSE.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting

Details

The result of this function can be fed directly to plotly::ggplotly() for interactive visualization, instead of the static ggplot viz.

Value

A ggplot object

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

gene_plot(dds_macrophage,
  gene = "ENSG00000125347",
  intgroup = "condition",
  annotation_obj = anno_df
)
```

get_aggrscores *Compute aggregated scores for gene sets*

Description

Computes for each gene set in the res_enrich object a Z score and an aggregated score (using the log2FoldChange values, provided in the res_de)

Usage

```
get_aggrscores(res_enrich, res_de, annotation_obj, gtl = NULL, aggrfun = mean)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
aggrfun	Specifies the function to use for aggregating the scores for each term. Common values could be mean or median.

Value

A data.frame with the same columns as provided in the input, with additional information on the z_score and the aggr_score for each gene set. This information is used by other functions such as gs_volcano() or enrichment_map()

See Also

gs_volcano() and enrichment_map() make efficient use of the computed aggregated scores

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
```

```
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)

res_enrich <- get_aggrscores(
  res_enrich,
  res_de,
  anno_df
)
```

`get_expression_values` *Get expression values*

Description

Extract expression values, with the possibility to select other assay slots

Usage

```
get_expression_values(
  dds,
  gene,
  intgroup,
  assay = "counts",
  normalized = TRUE,
  gtl = NULL
)
```

Arguments

dds	A `DESeqDataSet` object, normally obtained after running your data through the `DESeq2` framework.

gene	Character, specifies the identifier of the feature (gene) to be extracted
intgroup	A character vector of names in colData(dds) to use for grouping.
assay	Character, specifies with assay of the dds object to use for reading out the expression values. Defaults to "counts".
normalized	Logical value, whether the expression values should be normalized by their size factor. Defaults to TRUE, applies when assay is "counts"
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting

Value

A tidy data.frame with the expression values and covariates for further processing

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

df_exp <- get_expression_values(dds_macrophage,
  gene = "ENSG00000125347",
  intgroup = "condition"
)
head(df_exp)
```

---

ggs_backbone *Extract the backbone for the gene-geneset graph*

Description

Extract the backbone for the gene-geneset graph, either for the genes or for the genesets

Usage

```
ggs_backbone(
  res_enrich,
  res_de,
  annotation_obj = NULL,
  gtl = NULL,
```

```
  n_gs = 15,
  gs_ids = NULL,
  bb_on = c("genesets", "features"),
  bb_method = c("sdsm", "fdsm", "fixedrow"),
  bb_extract_alpha = 0.05,
  bb_extract_fwer = c("none", "bonferroni", "holm"),
  bb_remove_singletons = TRUE,
  color_graph = TRUE,
  color_by_geneset = "z_score",
  color_by_feature = "log2FoldChange",
  ...
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included in addition to the top ones (via n_gs)
bb_on	A character string, either "genesets" or "features", to specify which entity should be based the backbone graph on.
bb_method	A character string, referring to the function to be called (from the backbone package) for computing the backbone of the specified bipartite graph. Defaults to "sdsm", as recommended in the backbone package.
bb_extract_alpha	A numeric value, specifying the significance level to use when detecting the backbone of the network
bb_extract_fwer	A character string, defaulting to "none", specifying which method to use for the multiple testing correction for controlling the family-wise error rate
bb_remove_singletons	Logical value, defines whether to remove or leave in the returned graph the nodes that are not connected to other vertices
color_graph	Logical value, specifies whether to use information about genesets or features to colorize the nodes, e.g. for this info to be used in interactive versions of the graph

color_by_geneset
: Character string, corresponding to the column in res_enrich to be used for coloring the nodes if bb_on is set to "genesets". Defaults to the "z_score", which can be obtained via get_aggrscores()

color_by_feature
: Character string, corresponding to the column in res_de to be used for coloring the nodes if bb_on is set to "features". Defaults to the "log2FoldChange", which should be normally included in a DESeqResults object.

...
: Additional parameters to be passed internally

Value

A simple ìgraph object with the graph backbone

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

ggs_bbg <- ggs_backbone(res_enrich,
  res_de,
  anno_df,
  n_gs = 50,
```

```
  bb_on = "genesets",
  color_graph = TRUE,
  color_by_geneset = "z_score"
)
plot(ggs_bbg)

# if desired, one can also plot the interactive version
visNetwork::visIgraph(ggs_bbg)
```

---

ggs_graph *Construct a gene-geneset-graph*

Description

Construct a gene-geneset-graph from the results of a functional enrichment analysis

Usage

```
ggs_graph(
  res_enrich,
  res_de,
  annotation_obj = NULL,
  gtl = NULL,
  n_gs = 15,
  gs_ids = NULL,
  prettify = TRUE,
  geneset_graph_color = "gold",
  genes_graph_colpal = NULL
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included in addition to the top ones (via n_gs)
prettify	Logical, controlling the aspect of the returned graph object. If TRUE (default value), different shapes of the nodes are returned, based on the node type

geneset_graph_color
: Character value, specifying which color should be used for the fill of the shapes related to the gene sets.

genes_graph_colpal
: A vector of colors, also provided with their hex string, to be used as a palette for coloring the gene nodes. If unspecified, defaults to a color ramp palette interpolating from blue through yellow to red.

Value

An igraph object to be further manipulated or processed/plotted (e.g. via igraph::plot.igraph() or visNetwork::visIgraph())

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

ggs <- ggs_graph(
  res_enrich,
  res_de,
  anno_df
)
```

```
ggs

# could be viewed interactively with
# library(visNetwork)
# library(magrittr)
# ggs %>%
#   visIgraph() %>%
#   visOptions(highlightNearest = list(enabled = TRUE,
#                                      degree = 1,
#                                      hover = TRUE),
#              nodesIdSelection = TRUE)
```

`gostres_macrophage` *A sample output from g:Profiler*

Description

A sample output object as created from a call to g:Profiler, with the interface provided by `gprofiler2` - using the `gost()` function

Details

This object has been created on the data from the `macrophage` package by analyzing downstream the differentially expressed genes when comparing IFNg treated samples vs naive samples, accounting for the different cell lines included.

Details on how this object has been created are included in the `create_gt_data.R` script, included in the `scripts` folder of the `GeneTonic` package.

References

Alasoo, et al. "Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response", Nature Genetics, January 2018 doi: 10.1038/s41588-018-0046-7.

See Also

Other pathway-analysis-results: `enrichr_output_macrophage`, `topgoDE_macrophage_IFNg_vs_naive`

go_2_html *Information on a GeneOntology identifier*

Description

Assembles information, in HTML format, regarding a Gene Ontology identifier

Usage

```
go_2_html(go_id, res_enrich = NULL)
```

Arguments

go_id	Character, specifying the GeneOntology identifier for which to retrieve information
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. If not provided, the experiment-related information is not shown, and only some generic info on the identifier is displayed. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).

Details

Also creates a link to the AmiGO database

Value

HTML content related to a GeneOntology identifier, to be displayed in web applications (or inserted in Rmd documents)

Examples

```
go_2_html("GO:0002250")
go_2_html("GO:0043368")
```

gs_alluvial *Alluvial (sankey) plot for a set of genesets and the associated genes*

Description

Generate an interactive alluvial plot linking genesets to their associated genes

Usage

```
gs_alluvial(
  res_enrich,
  res_de,
  annotation_obj,
  gtl = NULL,
  n_gs = 5,
  gs_ids = NULL
)

gs_sankey(
  res_enrich,
  res_de,
  annotation_obj,
  gtl = NULL,
  n_gs = 5,
  gs_ids = NULL
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be displayed.

Value

A plotly object

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
```

```
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_alluvial(
  res_enrich = res_enrich,
  res_de = res_de,
  annotation_obj = anno_df,
  n_gs = 4
)
# or using the alias...
gs_sankey(
  res_enrich = res_enrich,
  res_de = res_de,
  annotation_obj = anno_df,
  n_gs = 4
)
```

---

gs_dendro *Dendrogram of the gene set enrichment results*

Description

Calculate (and plot) the dendrogram of the gene set enrichment results

Usage

```
gs_dendro(
  res_enrich,
  gtl = NULL,
```

```
  n_gs = nrow(res_enrich),
  gs_ids = NULL,
  gs_dist_type = "kappa",
  clust_method = "ward.D2",
  color_leaves_by = "z_score",
  size_leaves_by = "gs_pvalue",
  color_branches_by = "clusters",
  create_plot = TRUE
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to see the formatting requirements.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included (from the top ranked ones). Defaults to the number of rows of res_enrich
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included, additionally to the ones specified via n_gs. Defaults to NULL.
gs_dist_type	Character string, specifying which type of similarity (and therefore distance measure) will be used. Defaults to kappa, which uses create_kappa_matrix()
clust_method	Character string defining the agglomeration method to be used for the hierarchical clustering. See stats::hclust() for details, defaults to ward.D2
color_leaves_by	Character string, which columns of res_enrich will define the color of the leaves. Defaults to z_score
size_leaves_by	Character string, which columns of res_enrich will define the size of the leaves. Defaults to the gs_pvalue
color_branches_by	Character string, which columns of res_enrich will define the color of the branches. Defaults to clusters, which calls dynamicTreeCut::cutreeDynamic() to define the clusters
create_plot	Logical, whether to create the plot as well.

Value

A dendrogram object is returned invisibly, and a plot can be generated as well on that object.

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
```

```
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_dendro(res_enrich,
  n_gs = 100
)
```

`gs_fuzzyclustering` *Compute fuzzy clusters of gene sets*

Description

Compute fuzzy clusters of different gene sets, aiming to identify grouped categories that can better represent the distinct biological themes in the enrichment results

Usage

```
gs_fuzzyclustering(
  res_enrich,
  gtl = NULL,
  n_gs = nrow(res_enrich),
  gs_ids = NULL,
  similarity_matrix = NULL,
  similarity_threshold = 0.35,
```

```
  fuzzy_seeding_initial_neighbors = 3,
  fuzzy_multilinkage_rule = 0.5
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be displayed.
similarity_matrix	A similarity matrix between gene sets. Can be e.g. computed with create_kappa_matrix() or create_jaccard_matrix() or a similar function, returning a symmetric matrix with numeric values (max = 1). If not provided, this will be computed on the fly with create_kappa_matrix()
similarity_threshold	A numeric value for the similarity matrix, used to determine the initial seeds as in the implementation of DAVID. Higher values will lead to more genesets being initially unclustered, leading to a functional classification result with fewer groups and fewer geneset members. Defaults to 0.35, recommended to not go below 0.3 (see DAVID help pages)
fuzzy_seeding_initial_neighbors	Integer value, corresponding to the minimum geneset number in a seeding group. Lower values will lead to the inclusion of more genesets in the functional groups, and may generate a lot of small size groups. Defaults to 3
fuzzy_multilinkage_rule	Numeric value, comprised between 0 and 1. This parameter will determine how the seeding groups merge with each other, by specifying the percentage of shared genesets required to merge the two subsets into one group. Higher values will give sharper separation between the groups of genesets. Defaults to 0.5 (50%)

Value

A data frame, shaped in a similar way as the originally provided res_enrich object, containing two extra columns: gs_fuzzycluster, to specify the identifier of the fuzzy cluster of genesets, and gs_cluster_status, which can specify whether the geneset is the "Representative" for that cluster or a simple "Member". Notably, the number of rows in the returned object can be higher than the original number of rows in res_enrich.

References

See https://david.ncifcrf.gov/helps/functional_classification.html#clustering for details on the original implementation

Examples

```
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
# taking a smaller subset
res_enrich_subset <- res_enrich[1:100, ]

fuzzy_subset <- gs_fuzzyclustering(
  res_enrich = res_enrich_subset,
  n_gs = nrow(res_enrich_subset),
  gs_ids = NULL,
  similarity_matrix = NULL,
  similarity_threshold = 0.35,
  fuzzy_seeding_initial_neighbors = 3,
  fuzzy_multilinkage_rule = 0.5
)

# show all genesets members of the first cluster
fuzzy_subset[fuzzy_subset$gs_fuzzycluster == "1", ]

# list only the representative clusters
head(fuzzy_subset[fuzzy_subset$gs_cluster_status == "Representative", ], 10)
```

---

gs_heatmap *Plot a heatmap of the gene signature on the data*

Description

Plot a heatmap for the selected gene signature on the provided data, with the possibility to compactly display also DE only genes

Usage

```
gs_heatmap(
  se,
  res_de,
  res_enrich,
  annotation_obj = NULL,
  gtl = NULL,
  geneset_id = NULL,
  genelist = NULL,
  FDR = 0.05,
  de_only = FALSE,
  cluster_rows = TRUE,
```

```
  cluster_columns = FALSE,
  center_mean = TRUE,
  scale_row = FALSE,
  winsorize_threshold = NULL,
  anno_col_info = NULL,
  plot_title = NULL,
  ...
)
```

Arguments

se	A SummarizedExperiment object, or an object derived from this class, such as a DESeqTransform object (variance stabilized transformed data, or regularized logarithm transformed), in where the transformation has been applied to make the data more homoscedastic and thus a better fit for visualization.
res_de	A DESeqResults object.
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
geneset_id	Character specifying the gene set identifier to be plotted
genelist	A vector of character strings, specifying the identifiers contained in the row names of the se input object.
FDR	Numeric value, specifying the significance level for thresholding adjusted p-values. Defaults to 0.05.
de_only	Logical, whether to include only differentially expressed genes in the plot
cluster_rows	Logical, determining if rows should be clustered, as specified by ComplexHeatmap::Heatmap()
cluster_columns	Logical, determining if columns should be clustered, as specified by ComplexHeatmap::Heatmap()
center_mean	Logical, whether to perform mean centering on the row-wise
scale_row	Logical, whether to standardize by row the expression values
winsorize_threshold	Numeric value, to be applied as value to winsorize the extreme values of the heatmap. Should be a positive number. Defaults to NULL, which corresponds to not applying any winsorization. Suggested values: enter 2 or 3 if using row-standardized values (scale_row is TRUE), or visually inspect the range of the values if using simply mean centered values.
anno_col_info	A character vector of names in colData(dds) to use for decorating the heatmap as annotation.

plot_title	Character string, to specify the title of the plot, displayed over the heatmap. If left to NULL as by default, it tries to use the information on the geneset identifier provided
...	Additional arguments passed to other methods, e.g. in the call to ComplexHeatmap::Heatmap()

Value

A plot returned by the ComplexHeatmap::Heatmap() function

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

vst_macrophage <- vst(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_heatmap(vst_macrophage,
  res_de,
  res_enrich,
  anno_df,
  geneset_id = res_enrich$gs_id[1],
  cluster_columns = TRUE,
  anno_col_info = "condition"
```

```
)
```

`gs_horizon` *Plots a summary of enrichment results*

Description

Plots a summary of enrichment results - horizon plot to compare one or more sets of results

Usage

```
gs_horizon(
  res_enrich,
  compared_res_enrich_list,
  n_gs = 20,
  p_value_column = "gs_pvalue",
  color_by = "z_score",
  ref_name = "ref_scenario",
  sort_by = c("clustered", "first_set")
)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
`compared_res_enrich_list`	A named list, where each element is a `data.frame` formatted like the standard `res_enrich` objects used by `GeneTonic`. The names of the list are the names of the scenarios.
`n_gs`	Integer value, corresponding to the maximal number of gene sets to be displayed
`p_value_column`	Character string, specifying the column of `res_enrich` where the p-value to be represented is specified. Defaults to `gs_pvalue` (it could have other values, in case more than one p-value - or an adjusted p-value - have been specified).
`color_by`	Character, specifying the column of `res_enrich` to be used for coloring the plotted gene sets. Defaults sensibly to `z_score`.
`ref_name`	Character, defining the name of the scenario to compare against (the one in `res_enrich`) - defaults to "ref_scenario".
`sort_by`	Character string, either "clustered", or "first_set". This controls the sorting order of the included terms in the final plot. "clustered" presents the terms grouped by the scenario where they assume the highest values. "first_set" sorts the terms by the significance value in the reference scenario.

Details

It makes sense to have the results in `res_enrich` sorted by increasing `gs_pvalue`, to make sure the top results are first sorted by the significance (when selecting the common gene sets across the `res_enrich` elements provided in `compared_res_enrich_list`)

The gene sets included are a subset of the ones in common to all different scenarios included in `res_enrich` and the elements of `compared_res_enrich_list`.

Value

A `ggplot` object

See Also

`gs_summary_overview()`, `gs_summary_overview_pair()`

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

res_enrich2 <- res_enrich[1:42, ]
res_enrich3 <- res_enrich[1:42, ]
```

```
res_enrich4 <- res_enrich[1:42, ]

set.seed(2 * 42)
shuffled_ones_2 <- sample(seq_len(42)) # to generate permuted p-values
res_enrich2$gs_pvalue <- res_enrich2$gs_pvalue[shuffled_ones_2]
res_enrich2$z_score <- res_enrich2$z_score[shuffled_ones_2]
res_enrich2$aggr_score <- res_enrich2$aggr_score[shuffled_ones_2]

set.seed(3 * 42)
shuffled_ones_3 <- sample(seq_len(42)) # to generate permuted p-values
res_enrich3$gs_pvalue <- res_enrich3$gs_pvalue[shuffled_ones_3]
res_enrich3$z_score <- res_enrich3$z_score[shuffled_ones_3]
res_enrich3$aggr_score <- res_enrich3$aggr_score[shuffled_ones_3]

set.seed(4 * 42)
shuffled_ones_4 <- sample(seq_len(42)) # to generate permuted p-values
res_enrich4$gs_pvalue <- res_enrich4$gs_pvalue[shuffled_ones_4]
res_enrich4$z_score <- res_enrich4$z_score[shuffled_ones_4]
res_enrich4$aggr_score <- res_enrich4$aggr_score[shuffled_ones_4]

compa_list <- list(
  scenario2 = res_enrich2,
  scenario3 = res_enrich3,
  scenario4 = res_enrich4
)

gs_horizon(res_enrich,
  compared_res_enrich_list = compa_list,
  n_gs = 50,
  sort_by = "clustered"
)
gs_horizon(res_enrich,
  compared_res_enrich_list = compa_list,
  n_gs = 20,
  sort_by = "first_set"
)
```

`gs_mds` *Multi Dimensional Scaling plot for gene sets*

Description

Multi Dimensional Scaling plot for gene sets, extracted from a `res_enrich` object

Usage

```
gs_mds(
  res_enrich,
  res_de,
  annotation_obj,
```

```
  gtl = NULL,
  n_gs = nrow(res_enrich),
  gs_ids = NULL,
  similarity_measure = "kappa_matrix",
  mds_k = 2,
  mds_labels = 0,
  mds_colorby = "z_score",
  gs_labels = NULL,
  plot_title = NULL,
  return_data = FALSE
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be included (from the top ranked ones). Defaults to the number of rows of res_enrich
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included, additionally to the ones specified via n_gs. Defaults to NULL.
similarity_measure	Character, currently defaults to kappa_matrix, to specify how to compute the similarity measure between gene sets
mds_k	Integer value, number of dimensions to compute in the multi dimensional scaling procedure
mds_labels	Integer, defines the number of labels to be plotted on top of the scatter plot for the provided gene sets.
mds_colorby	Character specifying the column of res_enrich to be used for coloring the plotted gene sets. Defaults sensibly to z_score.
gs_labels	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be labeled.
plot_title	Character string, used as title for the plot. If left NULL, it defaults to a general description of the plot and of the DE contrast
return_data	Logical, whether the function should just return the data.frame of the MDS coordinates, related to the original res_enrich object. Defaults to FALSE.

Value

A ggplot object

See Also

create_kappa_matrix() is used to calculate the similarity between gene sets

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_mds(res_enrich,
  res_de,
  anno_df,
  n_gs = 200,
  mds_labels = 10
)
```

---

gs_radar *Radar (spider) plot for gene sets*

Description

Radar (spider) plot for gene sets, either for one or more results from functional enrichment analysis.

Usage

```
gs_radar(
  res_enrich,
  res_enrich2 = NULL,
  n_gs = 20,
  p_value_column = "gs_pvalue"
)

gs_spider(
  res_enrich,
  res_enrich2 = NULL,
  n_gs = 20,
  p_value_column = "gs_pvalue"
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_enrich2	Analogous to res_enrich1, another data.frame object, storing the result of the functional enrichment analysis, but for a different setting (e.g. another contrast). Defaults to NULL (in this case, a single set of enrichment results is plotted).
n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed
p_value_column	Character string, specifying the column of res_enrich where the p-value to be represented is specified. Defaults to gs_pvalue (it could have other values, in case more than one p-value - or an adjusted p-value - have been specified).

Value

A plotly object

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
```

```
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)
gs_radar(res_enrich = res_enrich)
# or using the alias...
gs_spider(res_enrich = res_enrich)

# with more than one set
res_enrich2 <- res_enrich[1:60, ]
set.seed(42)
shuffled_ones <- sample(seq_len(60)) # to generate permuted p-values
res_enrich2$gs_pvalue <- res_enrich2$gs_pvalue[shuffled_ones]
# ideally, I would also permute the z scores and aggregated scores
gs_radar(
  res_enrich = res_enrich,
  res_enrich2 = res_enrich2
)
```

`gs_scores` *Compute gene set scores*

Description

Compute gene set scores for each sample, by transforming the gene-wise change to a geneset-wise change

Usage

```
gs_scores(se, res_de, res_enrich, annotation_obj = NULL, gtl = NULL)
```

Arguments

`se`	A `SummarizedExperiment` object, or an object derived from this class, such as a `DESeqTransform` object (variance stabilized transformed data, or regularized logarithm transformed), in where the transformation has been applied to make the data more homoscedastic and thus a better fit for visualization.

res_de	A DESeqResults object.
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting

Value

A matrix with the geneset Z scores, e.g. to be plotted with gs_scoresheat()

See Also

gs_scoresheat() plots these scores

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

vst_macrophage <- vst(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
```

```
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

scores_mat <- gs_scores(
  vst_macrophage,
  res_de,
  res_enrich[1:50, ],
  anno_df
)
```

`gs_scoresheat` *Plots a matrix of geneset scores*

Description

Plots a matrix of geneset Z scores, across all samples

Usage

```
gs_scoresheat(
  mat,
  n_gs = nrow(mat),
  gs_ids = NULL,
  clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean",
  cluster_rows = TRUE,
  cluster_cols = TRUE
)
```

Arguments

`mat`	A matrix, e.g. returned by the `gs_scores()` function
`n_gs`	Integer value, corresponding to the maximal number of gene sets to be displayed.
`gs_ids`	Character vector, containing a subset of `gs_id` as they are available in `res_enrich`. Lists the gene sets to be displayed.
`clustering_distance_rows`	Character, a distance measure used in clustering rows
`clustering_distance_cols`	Character, a distance measure used in clustering columns
`cluster_rows`	Logical, determining if rows should be clustered
`cluster_cols`	Logical, determining if columns should be clustered

Value

A `ggplot` object

See Also

gs_scores() computes the scores plotted by this function

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

vst_macrophage <- vst(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

scores_mat <- gs_scores(
  vst_macrophage,
  res_de,
  res_enrich[1:30, ],
  anno_df
)
gs_scoresheat(scores_mat,
  n_gs = 30
)
```

`gs_simplify` *Simplify results from functional enrichment analysis*

Description

Simplify results from functional enrichment analysis, removing genesets that are redundant to enhance interpretation of the results

Usage

```
gs_simplify(res_enrich, gs_overlap = 0.75)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
`gs_overlap`	Numeric value, which defines the threshold for removing terms that present an overlap greater than the specified value. Changing its value can control the granularity of how redundant terms are removed from the original `res_enrich` for the next steps, e.g. plotting this via `gs_volcano()`

Value

A `data.frame` with a subset of the original gene sets

See Also

`gs_volcano()` and `ggs_graph()` can e.g. show an overview on the simplified table of gene sets

Examples

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)

dim(res_enrich)
res_enrich_simplified <- gs_simplify(res_enrich)
dim(res_enrich_simplified)
# and then use this further for all other functions expecting a res_enrich
```

gs_summary_heat *Plots a heatmap for genes and genesets*

Description

Plots a heatmap for genes and genesets, useful to spot out intersections across genesets and an overview of them

Usage

```
gs_summary_heat(res_enrich, res_de, annotation_obj, gtl = NULL, n_gs = 80)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed

Value

A ggplot object

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
```

```
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_summary_heat(
  res_enrich = res_enrich,
  res_de = res_de,
  annotation_obj = anno_df,
  n_gs = 20
)
```

`gs_summary_overview` *Plots a summary of enrichment results*

Description

Plots a summary of enrichment results for one set

Usage

```
gs_summary_overview(
  res_enrich,
  gtl = NULL,
  n_gs = 20,
  p_value_column = "gs_pvalue",
  color_by = "z_score",
  return_barchart = FALSE
)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
`gtl`	A `GeneTonic`-list object, containing in its slots the arguments specified above: `dds`, `res_de`, `res_enrich`, and `annotation_obj` - the names of the list *must* be specified following the content they are expecting

n_gs	Integer value, corresponding to the maximal number of gene sets to be displayed
p_value_column	Character string, specifying the column of res_enrich where the p-value to be represented is specified. Defaults to gs_pvalue (it could have other values, in case more than one p-value - or an adjusted p-value - have been specified).
color_by	Character, specifying the column of res_enrich to be used for coloring the plotted gene sets. Defaults sensibly to z_score.
return_barchart	Logical, whether to return a barchart (instead of the default dot-segment plot); defaults to FALSE.

Value

A ggplot object

See Also

gs_summary_overview_pair(), gs_horizon()

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive


# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
```

```
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_summary_overview(res_enrich)

# if desired, it can also be shown as a barplot
gs_summary_overview(res_enrich, n_gs = 30, return_barchart = TRUE)
```

`gs_summary_overview_pair` *Plots a summary of enrichment results*

Description

Plots a summary of enrichment results - for two sets of results

Usage

```
gs_summary_overview_pair(
  res_enrich,
  res_enrich2,
  n_gs = 20,
  p_value_column = "gs_pvalue",
  color_by = "z_score",
  alpha_set2 = 1
)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present).
`res_enrich2`	As `res_enrich`, the result of functional enrichment analysis, in a scenario/contrast different than the first set.
`n_gs`	Integer value, corresponding to the maximal number of gene sets to be displayed
`p_value_column`	Character string, specifying the column of `res_enrich` where the p-value to be represented is specified. Defaults to `gs_pvalue` (it could have other values, in case more than one p-value - or an adjusted p-value - have been specified).
`color_by`	Character, specifying the column of `res_enrich` to be used for coloring the plotted gene sets. Defaults sensibly to `z_score`.
`alpha_set2`	Numeric value, between 0 and 1, which specified the alpha transparency used for plotting the points for gene set 2.

Value

A `ggplot` object

See Also

gs_summary_overview(), gs_horizon()

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

res_enrich2 <- res_enrich[1:42, ]
set.seed(42)
shuffled_ones <- sample(seq_len(42)) # to generate permuted p-values
res_enrich2$gs_pvalue <- res_enrich2$gs_pvalue[shuffled_ones]
res_enrich2$z_score <- res_enrich2$z_score[shuffled_ones]
res_enrich2$aggr_score <- res_enrich2$aggr_score[shuffled_ones]
# ideally, I would also permute the z scores and aggregated scores
gs_summary_overview_pair(
  res_enrich = res_enrich,
  res_enrich2 = res_enrich2
)
```

gs_upset *Upset plot for genesets*

Description

Create an upset plot for genesets

Usage

```
gs_upset(
  res_enrich,
  res_de = NULL,
  annotation_obj = NULL,
  n_gs = 10,
  gtl = NULL,
  gs_ids = NULL,
  add_de_direction = FALSE,
  add_de_gsgenes = FALSE,
  col_upDE = "#E41A1C",
  col_downDE = "#377EB8",
  upset_geom = geom_point(size = 2),
  return_upsetgsg = FALSE
)
```

Arguments

res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
res_de	A DESeqResults object.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
n_gs	Integer value, corresponding to the maximal number of gene sets to be included
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be included in addition to the top ones (via n_gs)
add_de_direction	Logical, whether to add an annotation with info on the DE direction of single genes
add_de_gsgenes	Logical, if set to TRUE adds an annotation with detail on the single components of each defined subset
col_upDE	Character, specifying the color value to be used to mark upregulated genes

col_downDE Character, specifying the color value to be used to mark downregulated genes

upset_geom A geom specification to be used in the upset chart. Defaults sensibly to geom_point(size = 2)

return_upsetgsg
Logical, controlling the returned value. If set to TRUE, this function will not generate the plot but only create the corresponding data.frame, in case the user wants to proceed with a custom call to create an upset plot.

Value

A ggplot object (if plotting), or alternatively a data.frame

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)
gs_upset(res_enrich,
  n_gs = 10
)

gs_upset(res_enrich,
  res_de = res_de, annotation_obj = anno_df,
  n_gs = 8,
```

```
  add_de_direction = TRUE, add_de_gsgenes = TRUE
)

# or using the practical gtl (GeneTonicList)
gtl_macrophage <- GeneTonic_list(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)

gs_upset(
  gtl = gtl_macrophage,
  n_gs = 15,
  add_de_direction = TRUE, add_de_gsgenes = TRUE
)
```

`gs_volcano` *Volcano plot for gene sets*

Description

Volcano plot for gene sets, to summarize visually the functional enrichment results

Usage

```
gs_volcano(
  res_enrich,
  gtl = NULL,
  p_threshold = 0.05,
  color_by = "aggr_score",
  volcano_labels = 10,
  scale_circles = 1,
  gs_ids = NULL,
  plot_title = NULL
)
```

Arguments

`res_enrich`	A `data.frame` object, storing the result of the functional enrichment analysis. See more in the main function, `GeneTonic()`, to check the formatting requirements (a minimal set of columns should be present). This object needs to be processed first by a function such as `get_aggrscores()` to compute the term-wise `z_score` or `aggr_score`, which will be used for plotting
`gtl`	A `GeneTonic`-list object, containing in its slots the arguments specified above: `dds`, `res_de`, `res_enrich`, and `annotation_obj` - the names of the list *must* be specified following the content they are expecting

p_threshold	Numeric, defines the threshold to be used for filtering the gene sets to display. Defaults to 0.05
color_by	Character specifying the column of res_enrich to be used for coloring the plotted gene sets. Defaults to aggr_score.
volcano_labels	Integer, maximum number of labels for the gene sets to be plotted as labels on the volcano scatter plot.
scale_circles	A numeric value, to define the scaling factor for the circle sizes. Defaults to 1.
gs_ids	Character vector, containing a subset of gs_id as they are available in res_enrich. Lists the gene sets to be labeled.
plot_title	Character string, used as title for the plot. If left NULL, it defaults to a general description of the plot and of the DE contrast

Details

It is also possible to reduce the redundancy of the input res_enrich object, if it is passed in advance to the gs_simplify() function.

Value

A ggplot object

See Also

gs_simplify() can be applied in advance to res_enrich to reduce the redundancy of the displayed gene sets

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)
```

```
# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

gs_volcano(res_enrich)
```

---

happy_hour *Happy hour!*

Description

Start the happy hour, creating a report containing a document full of goodies derived from the provided objects.

Usage

```
happy_hour(
  dds,
  res_de,
  res_enrich,
  annotation_obj,
  gtl = NULL,
  project_id,
  mygenesets,
  mygenes,
  mygroup = NULL,
  usage_mode = "batch_mode",
  input_rmd = NULL,
  output_file = "my_first_GeneTonic_happyhour.html",
  output_dir = tempdir(),
  output_format = NULL,
  force_overwrite = FALSE,
  knitr_show_progress = FALSE,
  ignore_pandoc = FALSE,
  open_after_creating = TRUE,
  ...
)
```

Arguments

dds	A `DESeqDataSet` object, normally obtained after running your data through the `DESeq2` framework.

res_de	A DESeqResults object. As for the dds parameter, this is also commonly used in the DESeq2 framework.
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See GeneTonic() for the formatting requirements.
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name. See GeneTonic() for the formatting requirements.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting
project_id	A character string, which can be considered as an identifier for the set/session, and will be e.g. used in the title of the report created via happy_hour()
mygenesets	A vector of character strings, containing the genesets to focus on in the report - for each geneset, e.g. a signature heatmap can be created.
mygenes	A vector of character strings, containing the genes to focus on in the report - for each gene, the plot of the expression values is included.
mygroup	A character string, or a vector thereof. Contains the experimental variables to be used to split into groups the expression data, and color accordingly.
usage_mode	A character string, which controls the behavior of the Rmd document, based on whether the rendering is triggered while using the app ("shiny_mode"), or offline, in batch mode. Defaults to "batch_mode".
input_rmd	Character string with the path to the RMarkdown (.Rmd) file that will be used as the template for generating the report. Defaults to NULL, which will then use the one provided with the GeneTonic package.
output_file	Character string, specifying the file name of the output report. The file name extension must be either .html or .pdf, and consistent with the value of output_format.
output_dir	Character, defining the path to the output directory where the report will be generated. Defaults to the temp directory (tempdir()).
output_format	The format of the output report. Either html_document or pdf_document. The file name extension of output_file must be consistent with this choice. Can also be left empty and determined accordingly.
force_overwrite	Logical, whether to force overwrite an existing report with the same name in the output directory. Defaults to FALSE.
knitr_show_progress	Logical, whether to display the progress of knitr while generating the report. Defaults to FALSE.
ignore_pandoc	Logical, controlling how the report generation function will behave if pandoc or pandoc-citeproc are missing.
open_after_creating	Logical, whether to open the report in the default browser after being generated. Defaults to TRUE.
...	Other arguments that will be passed to rmarkdown::render().

Details

When happy_hour is called, a RMarkdown template file will be copied into the output directory, and rmarkdown::render() will be called to generate the final report.

As a default template, happy_hour uses the one delivered together with the GeneTonic package, which provides a comprehensive overview of what the user can extract. Experienced users can take that as a starting point to further edit and customize.

If there is already a .Rmd file with the same name in the output directory, the function will raise an error and stop, to avoid overwriting the existing file. The reason for this behaviour is that the copied template in the output directory will be deleted once the report is generated.

Credits to the original implementation proposed by Charlotte Soneson, upon which this function is **heavily** inspired.

Value

Generates a fully fledged report in the output_dir directory, called output_file and returns (invisibly) the name of the generated report.

See Also

GeneTonic(), shake_topGOtableResult(), shake_enrichResult()

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive
```

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)
## Not run:
happy_hour(
  dds = dds_macrophage,
  res_de = res_de,
  res_enrich = res_enrich,
  annotation_obj = anno_df,
  project_id = "examplerun",
  mygroup = "condition",
  # mygroup = "line",   # alternatively
  mygenesets = res_enrich$gs_id[c(1:5, 11, 31)],
  mygenes = c(
    "ENSG00000125347",
    "ENSG00000172399",
    "ENSG00000137496"
  )
)

## End(Not run)
```

`map2color` *Maps numeric values to color values*

Description

Maps numeric continuous values to values in a color palette

Usage

```
map2color(x, pal, symmetric = TRUE, limits = NULL)
```

Arguments

`x`	A character vector of numeric values (e.g. log2FoldChange values) to be converted to a vector of colors
`pal`	A vector of characters specifying the definition of colors for the palette, e.g. obtained via `RColorBrewer::brewer.pal()`
`symmetric`	Logical value, whether to return a palette which is symmetrical with respect to the minimum and maximum values - "respecting" the zero. Defaults to `TRUE`.
`limits`	A vector containing the limits of the values to be mapped. If not specified, defaults to the range of values in the `x` vector.

Value

A vector of colors, each corresponding to an element in the original vector

Examples

```
a <- 1:9
pal <- RColorBrewer::brewer.pal(9, "Set1")
map2color(a, pal)
plot(a, col = map2color(a, pal), pch = 20, cex = 4)

b <- 1:50
pal2 <- grDevices::colorRampPalette(
  RColorBrewer::brewer.pal(name = "RdYlBu", 11)
)(50)
plot(b, col = map2color(b, pal2), pch = 20, cex = 3)
```

---

overlap_coefficient *Calculate overlap coefficient*

Description

Calculate similarity coefficient between two sets, based on the overlap

Usage

```
overlap_coefficient(x, y)
```

Arguments

x	Character vector, corresponding to set 1
y	Character vector, set 2

Value

A numeric value between 0 and 1

See Also

https://en.wikipedia.org/wiki/Overlap_coefficient

Examples

```
a <- seq(1, 21, 2)
b <- seq(1, 11, 2)
overlap_coefficient(a, b)
```

overlap_jaccard_index *Calculate Jaccard Index between two sets*

Description

Calculate similarity coefficient with the Jaccard Index

Usage

```
overlap_jaccard_index(x, y)
```

Arguments

x	Character vector, corresponding to set 1
y	Character vector, corresponding to set 2

Value

A numeric value between 0 and 1

Examples

```
a <- seq(1, 21, 2)
b <- seq(1, 11, 2)
overlap_jaccard_index(a, b)
```

res_macrophage_IFNg_vs_naive *A sample* DESeqResults *object*

Description

A sample DESeqResults object, generated in the DESeq2 framework

Details

This DESeqResults object on the data from the macrophage package has been created comparing IFNg treated samples vs naive samples, accounting for the different cell lines included.

Details on how this object has been created are included in the create_gt_data.R script, included in the scripts folder of the GeneTonic package.

References

Alasoo, et al. "Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response", Nature Genetics, January 2018 doi: 10.1038/s41588-018-0046-7.

shake_davidResult *Convert the output of DAVID*

Description

Convert the output of DAVID for straightforward use in `GeneTonic()`

Usage

```
shake_davidResult(david_output_file)
```

Arguments

david_output_file	The location of the text file output, as exported from DAVID

Value

A `data.frame` compatible for use in `GeneTonic()` as `res_enrich`

See Also

Other shakers: `shake_enrichResult()`, `shake_enrichrResult()`, `shake_fgseaResult()`, `shake_gprofilerResult()`, `shake_gsenrichResult()`, `shake_topGOtableResult()`

Examples

```
david_output_file <- system.file("extdata",
  "david_output_chart_BPonly_ifng_vs_naive.txt",
  package = "GeneTonic"
)
res_enrich <- shake_davidResult(david_output_file)
```

shake_enrichResult *Convert an enrichResult object*

Description

Convert an enrichResult object for straightforward use in `GeneTonic()`

Usage

```
shake_enrichResult(obj)
```

Arguments

obj	An `enrichResult` object, obtained via `clusterProfiler` (or also via `reactomePA`)

Details

This function is able to handle the output of `clusterProfiler` and `reactomePA`, as they both return an object of class `enrichResult` - and this in turn contains the information required to create correctly a `res_enrich` object.

Value

A `data.frame` compatible for use in `GeneTonic()` as `res_enrich`

See Also

Other shakers: `shake_davidResult()`, `shake_enrichrResult()`, `shake_fgseaResult()`, `shake_gprofilerResult()`, `shake_gsenrichResult()`, `shake_topGOtableResult()`

Examples

```
# dds
library("macrophage")
library("DESeq2")
data(gse)
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive
de_symbols_IFNg_vs_naive <- res_macrophage_IFNg_vs_naive[
  (!(is.na(res_macrophage_IFNg_vs_naive$padj))) &
    (res_macrophage_IFNg_vs_naive$padj <= 0.05), "SYMBOL"
]
bg_ids <- rowData(dds_macrophage)$SYMBOL[rowSums(counts(dds_macrophage)) > 0]
## Not run:
library("clusterProfiler")
library("org.Hs.eg.db")
ego_IFNg_vs_naive <- enrichGO(
  gene = de_symbols_IFNg_vs_naive,
  universe = bg_ids,
  keyType = "SYMBOL",
  OrgDb = org.Hs.eg.db,
  ont = "BP",
  pAdjustMethod = "BH",
  pvalueCutoff = 0.01,
  qvalueCutoff = 0.05,
  readable = FALSE
)

res_enrich <- shake_enrichResult(ego_IFNg_vs_naive)
head(res_enrich)

## End(Not run)
```

shake_enrichrResult *Convert the output of Enrichr*

Description

Convert the output of Enrichr for straightforward use in GeneTonic()

Usage

```
shake_enrichrResult(enrichr_output_file, enrichr_output = NULL)
```

Arguments

enrichr_output_file	The location of the text file output, as exported from Enrichr
enrichr_output	A data.frame with the output of enrichr, related to a specific set of genesets. Usually it is one of the members of the list returned by the initial call to enrichr.

Value

A data.frame compatible for use in GeneTonic() as res_enrich

See Also

Other shakers: shake_davidResult(), shake_enrichResult(), shake_fgseaResult(), shake_gprofilerResult(), shake_gsenrichResult(), shake_topGOtableResult()

Examples

```
# library("enrichR")
# dbs <- c("GO_Molecular_Function_2018",
#          "GO_Cellular_Component_2018",
#          "GO_Biological_Process_2018",
#          "KEGG_2019_Human",
#          "Reactome_2016",
#          "WikiPathways_2019_Human")
# degenes <- (mosdef::deresult_to_df(res_macrophage_IFNg_vs_naive, FDR = 0.01)$SYMBOL)
# if called directly within R...
# enrichr_output_macrophage <- enrichr(degenes, dbs)
# or alternatively, if downloaded from the website in tabular format
enrichr_output_file <- system.file("extdata",
  "enrichr_tblexport_IFNg_vs_naive.txt",
  package = "GeneTonic"
)
res_from_enrichr <- shake_enrichrResult(enrichr_output_file = enrichr_output_file)
# res_from_enrichr2 <- shake_enrichrResult(
#   enrichr_output = enrichr_output_macrophage[["GO_Biological_Process_2018"]])
```

shake_fgseaResult *Convert the output of fgsea*

Description

Convert the output of fgsea for straightforward use in GeneTonic()

Usage

```
shake_fgseaResult(fgsea_output)
```

Arguments

fgsea_output	A data.frame with the output of fgsea() in fgsea.

Value

A data.frame compatible for use in GeneTonic() as res_enrich

See Also

Other shakers: shake_davidResult(), shake_enrichResult(), shake_enrichrResult(), shake_gprofilerResult(), shake_gsenrichResult(), shake_topGOtableResult()

Examples

```
data(fgseaRes, package = "GeneTonic")
res_from_fgsea <- shake_fgseaResult(fgseaRes)
```

shake_gprofilerResult *Convert the output of g:Profiler*

Description

Convert the output of g:Profiler for straightforward use in GeneTonic()

Usage

```
shake_gprofilerResult(gprofiler_output_file, gprofiler_output = NULL)
```

Arguments

gprofiler_output_file	The location of the text file output, as exported from g:Profiler
gprofiler_output	A data.frame with the output of gost() in gprofiler2. Usually it is one of the members of the list returned by the initial call to gost.

Value

A data.frame compatible for use in GeneTonic() as res_enrich

See Also

Other shakers: shake_davidResult(), shake_enrichResult(), shake_enrichrResult(), shake_fgseaResult(), shake_gsenrichResult(), shake_topGOtableResult()

Examples

```
# degenes <- (mosdef::deresult_to_df(res_macrophage_IFNg_vs_naive, FDR = 0.01)$SYMBOL)
# if called directly within R...
# enrichr_output_macrophage <- enrichr(degenes, dbs)
# or alternatively, if downloaded from the website in tabular format
gprofiler_output_file <- system.file(
  "extdata",
  "gProfiler_hsapiens_5-25-2020_tblexport_IFNg_vs_naive.csv",
  package = "GeneTonic"
)
res_from_gprofiler <- shake_gprofilerResult(gprofiler_output_file = gprofiler_output_file)

data(gostres_macrophage, package = "GeneTonic")
res_from_gprofiler_2 <- shake_gprofilerResult(
  gprofiler_output = gostres_macrophage$result
)
```

---

shake_gsenrichResult *Convert a gseaResult object*

Description

Convert a gseaResult object for straightforward use in GeneTonic()

Usage

```
shake_gsenrichResult(obj)
```

Arguments

obj	A gseaResult object, obtained via clusterProfiler

Details

This function is able to handle the output of clusterProfiler's gseGO and GSEA, as they both return an object of class gseaResult - and this in turn contains the information required to create correctly a res_enrich object.

Value

A data.frame compatible for use in GeneTonic() as res_enrich

See Also

Other shakers: shake_davidResult(), shake_enrichResult(), shake_enrichrResult(), shake_fgseaResult(), shake_gprofilerResult(), shake_topGOtableResult()

Examples

```
# dds
library("macrophage")
library("DESeq2")
data(gse)
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)

# res object
data(res_de_macrophage, package = "GeneTonic")
sorted_genes <- sort(
  setNames(res_macrophage_IFNg_vs_naive$log2FoldChange,
           res_macrophage_IFNg_vs_naive$SYMBOL),
  decreasing = TRUE
)
## Not run:
library("clusterProfiler")
library("org.Hs.eg.db")
gsego_IFNg_vs_naive <- gseGO(
  geneList = sorted_genes,
  ont = "BP",
  OrgDb = org.Hs.eg.db,
  keyType = "SYMBOL",
  minGSSize = 10,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  verbose = TRUE
)

res_enrich <- shake_gsenrichResult(gsego_IFNg_vs_naive)
head(res_enrich)
gtl_macrophage <- GeneTonicList(
  dds = dds_macrophage,
  res_de = res_macrophage_IFNg_vs_naive,
  res_enrich = res_enrich,
  annotation_obj = anno_df
)

## End(Not run)
```

shake_topGOtableResult *Convert a topGOtableResult object*

Description

Convert a topGOtableResult object for straightforward use in `GeneTonic()`

Usage

```
shake_topGOtableResult(obj, p_value_column = "p.value_elim")
```

Arguments

`obj`	A `topGOtableResult` object
`p_value_column`	Character, specifying which column the p value for enrichment has to be used. Example values are "p.value_elim" or "p.value_classic"

Value

A `data.frame` compatible for use in `GeneTonic()` as `res_enrich`

See Also

Other shakers: `shake_davidResult()`, `shake_enrichResult()`, `shake_enrichrResult()`, `shake_fgseaResult()`, `shake_gprofilerResult()`, `shake_gsenrichResult()`

Examples

```
# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")

res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
```

signature_volcano *Plot a volcano plot of a geneset*

Description

Plot a volcano plot for the geneset of the provided data, with the remaining genes as shaded dots in the background of the plot.

Usage

```
signature_volcano(
  res_de,
  res_enrich,
  annotation_obj = NULL,
  gtl = NULL,
  geneset_id = NULL,
  genelist = NULL,
  FDR = 0.05,
  color = "#1a81c2",
  volcano_labels = 25,
  plot_title = NULL
)
```

Arguments

res_de	A DESeqResults object.
res_enrich	A data.frame object, storing the result of the functional enrichment analysis. See more in the main function, GeneTonic(), to check the formatting requirements (a minimal set of columns should be present).
annotation_obj	A data.frame object with the feature annotation information, with at least two columns, gene_id and gene_name.
gtl	A GeneTonic-list object, containing in its slots the arguments specified above: dds, res_de, res_enrich, and annotation_obj - the names of the list *must* be specified following the content they are expecting.
geneset_id	Character specifying the gene set identifier to be plotted.
genelist	A vector of character strings, specifying the identifiers contained in the rownames of the res_de input object.
FDR	Numeric value, specifying the significance level for thresholding adjusted p-values. Defaults to 0.05.
color	Character string to specify color of filtered points in the plot. Defaults to #1a81c2 (shade of blue).
volcano_labels	Integer, maximum number of labels for the gene sets to be plotted as labels on the volcano scatter plot. Defaults to 25.
plot_title	Character string, to specify the title of the plot, displayed over the volcano plot. If left to NULL as by default, it tries to use the information on the geneset identifier provided.

Value

A ggplot object

Examples

```
library("macrophage")
library("DESeq2")
```

```
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)


# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

signature_volcano(res_de,
  res_enrich,
  anno_df,
  geneset_id = res_enrich$gs_id[1]
)

# alternatively

chemokine_list <- c(
  "ENSG00000108702",
  "ENSG00000172156",
  "ENSG00000181374",
  "ENSG00000276409"
)

signature_volcano(res_de,
  res_enrich,
  anno_df,
  genelist = chemokine_list
)
```

styleColorBar_divergent *Style DT color bars*

Description

Style DT color bars for values that diverge from 0.

Usage

```
styleColorBar_divergent(data, color_pos, color_neg)
```

Arguments

data	The numeric vector whose range will be used for scaling the table data from 0-100 before being represented as color bars. A vector of length 2 is acceptable here for specifying a range possibly wider or narrower than the range of the table data itself.
color_pos	The color of the bars for the positive values
color_neg	The color of the bars for the negative values

Details

This function draws background color bars behind table cells in a column, width the width of bars being proportional to the column values *and* the color dependent on the sign of the value.

A typical usage is for values such as log2FoldChange for tables resulting from differential expression analysis. Still, the functionality of this can be quickly generalized to other cases - see in the examples.

The code of this function is heavily inspired from styleColorBar, and borrows at full hands from an excellent post on StackOverflow - https://stackoverflow.com/questions/33521828/stylecolorbar-center-and-shift-left-right-dependent-on-sign/33524422#33524422

Value

This function generates JavaScript and CSS code from the values specified in R, to be used in DT tables formatting.

Examples

```
data(res_de_macrophage, package = "GeneTonic")
res_df <- mosdef::deresult_to_df(res_macrophage_IFNg_vs_naive)
library("magrittr")
library("DT")
DT::datatable(res_df[1:50, ],
  options = list(
    pageLength = 25,
    columnDefs = list(
```

```
      list(className = "dt-center", targets = "_all")
    )
  )
) %>%
  formatRound(columns = c("log2FoldChange"), digits = 3) %>%
  formatStyle(
    "log2FoldChange",
    background = styleColorBar_divergent(
      res_df$log2FoldChange,
      scales::alpha("navyblue", 0.4),
      scales::alpha("darkred", 0.4)
    ),
    backgroundSize = "100% 90%",
    backgroundRepeat = "no-repeat",
    backgroundPosition = "center"
  )


simplest_df <- data.frame(
  a = c(rep("a", 9)),
  value = c(-4, -3, -2, -1, 0, 1, 2, 3, 4)
)

# or with a very simple data frame
DT::datatable(simplest_df) %>%
  formatStyle(
    "value",
    background = styleColorBar_divergent(
      simplest_df$value,
      scales::alpha("forestgreen", 0.4),
      scales::alpha("gold", 0.4)
    ),
    backgroundSize = "100% 90%",
    backgroundRepeat = "no-repeat",
    backgroundPosition = "center"
  )
```

`summarize_ggs_hubgenes` *Summarize information on the hub genes*

Description

Summarize information on the hub genes in the Gene-Geneset graph

Usage

```
summarize_ggs_hubgenes(g)
```

Arguments

g	An igraph object, as generated by the ggs_graph() function

Value

A data.frame object, formatted for use in DT::datatable()

Examples

```
library("macrophage")
library("DESeq2")
library("org.Hs.eg.db")
library("AnnotationDbi")

# dds object
data("gse", package = "macrophage")
dds_macrophage <- DESeqDataSet(gse, design = ~ line + condition)
rownames(dds_macrophage) <- substr(rownames(dds_macrophage), 1, 15)
dds_macrophage <- estimateSizeFactors(dds_macrophage)

# annotation object
anno_df <- data.frame(
  gene_id = rownames(dds_macrophage),
  gene_name = mapIds(org.Hs.eg.db,
    keys = rownames(dds_macrophage),
    column = "SYMBOL",
    keytype = "ENSEMBL"
  ),
  stringsAsFactors = FALSE,
  row.names = rownames(dds_macrophage)
)

# res object
data(res_de_macrophage, package = "GeneTonic")
res_de <- res_macrophage_IFNg_vs_naive

# res_enrich object
data(res_enrich_macrophage, package = "GeneTonic")
res_enrich <- shake_topGOtableResult(topgoDE_macrophage_IFNg_vs_naive)
res_enrich <- get_aggrscores(res_enrich, res_de, anno_df)

ggs <- ggs_graph(
  res_enrich,
  res_de,
  anno_df
)
dt_df <- summarize_ggs_hubgenes(ggs)
DT::datatable(dt_df, escape = FALSE)
```

topgoDE_macrophage_IFNg_vs_naive
A sample res_enrich *object*

Description

A sample res_enrich object, generated with the topGOtable function (from the pcaExplorer package).

Details

This res_enrich object on the data from the macrophage package has been created by analyzing downstream the differentially expressed genes when comparing IFNg treated samples vs naive samples, accounting for the different cell lines included.

Details on how this object has been created are included in the create_gt_data.R script, included in the scripts folder of the GeneTonic package.

References

Alasoo, et al. "Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response", Nature Genetics, January 2018 doi: 10.1038/s41588-018-0046-7.

See Also

Other pathway-analysis-results: enrichr_output_macrophage, gostres_macrophage

Index

∗ **Details on how this object has been created are included in the**
fgseaRes, 23
∗ **folder of the**
fgseaRes, 23
∗ **internal**
GeneTonic-pkg, 27
∗ **list(GeneTonic)**
fgseaRes, 23
∗ **list(create_gt_data.R)**
fgseaRes, 23
∗ **list(scripts)**
fgseaRes, 23
∗ **package.**
fgseaRes, 23
∗ **pathway-analysis-results**
enrichr_output_macrophage, 21
fgseaRes, 23
gostres_macrophage, 39
topgoDE_macrophage_IFNg_vs_naive, 87
∗ **script, included in the**
fgseaRes, 23
∗ **shakers**
shake_davidResult, 75
shake_enrichResult, 75
shake_enrichrResult, 77
shake_fgseaResult, 78
shake_gprofilerResult, 78
shake_gsenrichResult, 79
shake_topGOtableResult, 81
∗
fgseaRes, 23
.check_pandoc, 3

check_colors, 7
checkup_GeneTonic, 4
checkup_gtl, 5
cluster_markov, 8
ComplexHeatmap::Heatmap(), *47, 48*
create_jaccard_matrix, 9
create_jaccard_matrix(), *45*
create_kappa_matrix, 10
create_kappa_matrix(), *43*, *45*, *53*
create_upsetdata, 11

deprecated, 12
describe_gtl, 13
DESeq2::DESeqDataSet(), *22*
DESeq2::DESeqResults(), *22*
deseqresult2df, 14
deseqresult2df(), *12*
distill_enrichment, 14
dynamicTreeCut::cutreeDynamic(), *43*

editor_to_vector_sanitized, 16
enhance_table, 17
enrichment_map, 19
enrichment_map(), *10*, *32*
enrichr_output_macrophage, 21, *39*, *87*
export_for_iSEE, 22
export_to_sif, 23

fgseaRes, 23

gene_plot, 30
gene_plot(), *12*
geneinfo_2_html, 24
geneinfo_2_html(), *12*
GeneTonic, 25
GeneTonic(), *4*, *10, 11*, *17*, *19, 20*, *22*, *25*, *28*, *32*, *35*, *37*, *40, 41*, *43*, *45*, *47*, *49*, *52*, *54*, *56*, *59–61*, *63*, *65*, *67*, *70, 71*, *75–82*
GeneTonic-package *(*GeneTonic-pkg*)*, 27
GeneTonic-pkg, 27
GeneTonic_list *(*GeneTonicList*)*, 28
GeneTonicList, 28
get_aggrscores, 32
get_aggrscores(), *67*

get_expression_values, 33
get_expression_values(), *12*
ggs_backbone, 34
ggs_graph, 37
ggs_graph(), *59*
go_2_html, 40
go_2_html(), *12*
gostres_macrophage, *21*, 39, *87*
grDevices::col2rgb(), *7*
gs_alluvial, 40
gs_dendro, 42
gs_fuzzyclustering, 44
gs_heatmap, 46
gs_horizon, 49
gs_horizon(), *62*, *64*
gs_mds, 51
gs_mds(), *10, 11*
gs_radar, 53
gs_sankey *(*gs_alluvial*)*, 40
gs_scores, 55
gs_scores(), *57, 58*
gs_scoresheat, 57
gs_scoresheat(), *56*
gs_simplify, 59
gs_simplify(), *68*
gs_spider *(*gs_radar*)*, 53
gs_summary_heat, 60
gs_summary_overview, 61
gs_summary_overview(), *50*, *64*
gs_summary_overview_pair, 63
gs_summary_overview_pair(), *50*, *62*
gs_upset, 65
gs_volcano, 67
gs_volcano(), *32*, *59*

happy_hour, 69
happy_hour(), *25*, *70*

igraph::communities(), *9*
igraph::plot.igraph(), *19*, *38*

map2color, 72
map2color(), *13*
mosdef::buttonifier(), *13*
mosdef::create_link_GeneCards(), *13*
mosdef::create_link_GO(), *13*
mosdef::create_link_NCBI(), *13*
mosdef::deresult_to_df(), *12*
mosdef::gene_plot(), *12*
mosdef::geneinfo_to_html(), *12*
mosdef::get_annotation_orgdb(), *25*, *28*
mosdef::get_expr_values(), *12*
mosdef::go_to_html(), *12*
mosdef::map_to_color(), *13*
mosdef::styleColorBar_divergent(), *13*

overlap_coefficient, 73
overlap_jaccard_index, 74

plotly::ggplotly(), *31*

RColorBrewer::brewer.pal(), *72*
res_macrophage_IFNg_vs_naive, 74
rmarkdown::render(), *70, 71*

shake_davidResult, 75, *76–81*
shake_enrichResult, *75*, 75, *77–81*
shake_enrichResult(), *25*, *28*, *71*
shake_enrichrResult, *75, 76*, 77, *78–81*
shake_fgseaResult, *75–77*, 78, *79–81*
shake_gprofilerResult, *75–78*, 78, *80, 81*
shake_gsenrichResult, *75–79*, 79, *81*
shake_topGOtableResult, *75–80*, 81
shake_topGOtableResult(), *25*, *28*, *71*
signature_volcano, 81
stats::hclust(), *43*
styleColorBar_divergent, 84
styleColorBar_divergent(), *13*
summarize_ggs_hubgenes, 85

topgoDE_macrophage_IFNg_vs_naive, *21*, *39*, 87

visNetwork::visIgraph(), *19*, *38*