Package ‘RITAN’

January 9, 2025

Type Package

Title Rapid Integration of Term Annotation and Network resources

Version 1.31.0

Description Tools for comprehensive gene set enrichment and extraction of multi-resource high confidence subnetworks. RITAN facilitates bioinformatic tasks for enabling network biology research.

LazyData TRUE

Depends R (>= 4.0),

Imports graphics, methods, stats, utils, grid, gridExtra, reshape2, gplots, ggplot2, plotrix, RColorBrewer, STRINGdb, MCL, linkcomm, dynamicTreeCut, gsubfn, hash, png, sqldf, igraph, BgeeDB, knitr, RITANdata, GenomicFeatures, ensembldb, AnnotationFilter, EnsDb.Hsapiens.v86

VignetteBuilder knitr

Collate 'lib_enrichment.R' 'lib_network.R' 'interconnectivity_functions.R'

RoxygenNote 7.1.1

Suggests rmarkdown, BgeeDB

License file LICENSE

biocViews QualityControl, Network, NetworkEnrichment, NetworkInference, GeneSetEnrichment, FunctionalGenomics, GraphAndNetwork

NeedsCompilation no

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

git_branch devel

git_last_commit e6e8dc4

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-01-08

Author Michael Zimmermann [aut, cre]

Maintainer Michael Zimmermann <mtzimmermann@mcw.edu>

Contents

`as.graph` *as.graph*

Description

wrapper to convert a data.frame from RITAN an igraph graph object

Usage

```
as.graph(mat, p1 = 1, p2 = 3, ...)
```

Arguments

`mat`	matrix or data frame describing a network
`p1`	[1] column of first interactor
`p2`	[3] column of second interactor
`...`	further options passed on to igraph::graph()

Value

igraph object

Examples

```
## Not run:
G <- as.graph(network_list$PID)

## End(Not run)
```

---

check_any_net_input *check_any_net_input*

Description

A Quality Control function. This function applies check_net_input() to all available resources (default).

Usage

```
check_any_net_input(set, resources = names(network_list))
```

Arguments

set	An input list of genes to check against references.
resources	The collection of network resources to check within.

Value

Logical vector indicating if the genes in "set" are within ANY of the resources.

Examples

```
#' ## Check if genes in myGeneSet are annotated by any resource in "network_list" (default).
library(RITANdata)
myGeneSet <- c('BRCA1','RAD51C','VAV1','HRAS','ABCC1','CYP1B1','CYP3A5')
yorn <- check_any_net_input( myGeneSet )
print(yorn)
```

check_net_input *check_net_input*

Description

A Quality Control function. This function will compare an input list of genes to a network reference and report if each member of the input is present in the resource.

Usage

```
check_net_input(
  set,
  ref,
  check4similar = FALSE,
  entity1name = "p1",
  entity2name = "p1"
)
```

Arguments

set	An input list of genes to check against a reference.
ref	A reference of network data. See readSIF().
check4similar	Logical flag. If TRUE, a case-insensitive grep will be used for name matching. For genes in families with many related members (e.g. ABC*, FAM*, etc.), this will not be ideal. We intend this option as a QC screening method to identify if case, punctuaiton, etc is causing fewer than expected matches.
entity1name	The column name in "ref" of the first entity. Default = "p1."
entity2name	The column name in "ref" of the second entity. Default = "p2."

Value

Character vector of "yes/no" indicating "within-ref/not"

Examples

```
## Return a "yes/no" vector indicating if each gene in myGeneSet is annotated with any term in GO
## If no match, this function can attempt to suggest closest matches (check4similar = TRUE)
library(RITANdata)
myGeneSet <- c('BRCA1','RAD51C','VAV1','HRAS','ABCC1','CYP1B1','CYP3A5')
yorn <- check_net_input( myGeneSet, network_list[["CCSB"]] )
print(yorn)

yorn <- check_net_input( myGeneSet, network_list[["PID"]] )
print(yorn)

## See check_any_net_input() for efficiently checking across all resources.
```

cov_undirected — *cov_undirected function to show the un-directed coverabe between two nodes lists from two networks*

Description

cov_undirected function to show the un-directed coverabe between two nodes lists from two networks

Usage

```
cov_undirected(this_nodes1, this_nodes2, this_net1, this_net2)
```

Arguments

`this_nodes1`	list of nodes for first network
`this_nodes2`	list of nodes for second network
`this_net1`	the first network
`this_net2`	the second network

enrichment_symbols — *enrichment_symbols*

Description

This function is called by term_enrichment() and term_enrichment_by_subset(). The user may call it directly, but we suggest using term_enrichment(). The function uses the resources currently loaded into the active_genesets vector. See load_geneset_symbols().

Usage

```
enrichment_symbols(geneset, term = NULL, all_symbols = NA, ...)
```

Arguments

`geneset`	vector of gene symbols to be evaluated
`term`	a list containing specific gene set term(s) and their corresponding gene symbols contained in one of the annotation resources, default is all gene set terms in the GO, ReactomePathways, KEGG_filtered_canonical_pathways, and MSigDB_Hallmarks libraries
`all_symbols`	gene symbols to be evaluated, identified by gene symbol name. Default is all protein coding genes. This parameter should be manipulated to include only the gene symbols that pertain to the user's analysis.
`...`	additional arguments are not used

Details

Outputs a data frame containing the gene set name, a hypergeometric-test p value, the number of genes from the input gene list that occur in the gene set, the number of genes in the gene set, the gene symbols for the genes in the input gene list, and the q value.

Value

results matrix of input gene list compared to active gene sets. Q value is calculated using entire group of active gene sets.

Examples

```
require(RITANdata)
myGeneSet <- c('BRCA1','RAD51C','VAV1','HRAS','ABCC1','CYP1B1','CYP3A5')

## Not run:
## We suggest using term_enrichment() instead. E.g.:
e <- term_enrichment(myGeneSet, 'GO')

## End(Not run)

## But, you may use enrichment_symbols() directly for an individual term:
load_geneset_symbols('GO')
e <- enrichment_symbols(myGeneSet, 'DNA_repair', all_symbols = cached_coding_genes)
print(e)

## Not run:
## Gene set enrichment using intersection of gene symbols
##   provided in geneset parameter and all protein coding genes.
enrichment_symbols(geneset = vac1.day0vs31.de.genes)

## choose which terms to evaluate
t <- active_genesets[1:5]

## Test enrichment of that set of terms
enrichment_symbols(geneset = vac1.day0vs31.de.genes, term = t)

## End(Not run)
```

`geneset_overlap` *geneset_overlap*

Description

Return assymetric matrix of the fraction of genes shared between sets. E.G. The fraction of the first set that is "covered" by or "overlaps" the second set.

Usage

```
geneset_overlap(s1, s2 = s1, s.size = unlist(lapply(s1, length)))
```

Arguments

s1	The first geneset
s2	the second geneset
s.size	Denominator used in each comparison. The default is to determint the lengths of elements in "s1"

Value

results matrix of input gene list compared to active gene sets. Q value is calculated using entire group of active gene sets.

Examples

```
require(RITANdata)
r <- geneset_overlap( geneset_list$MSigDB_Hallmarks, geneset_list$NetPath_Gene_regulation )
heatmap(r, col = rev(gray(seq(0,1,length.out = 15))) )
summary(c(r))
```

---

icon_single_within *icon_single_within interconnectivity score within a network*

Description

icon_single_within interconnectivity score within a network

Usage

```
icon_single_within(nodes = NULL, net = NULL, s = 10, verbose = TRUE)
```

Arguments

nodes	the node labels to use
net	the network to use
s	[10] the number of repeated random draws to make
verbose	[TRUE] if more verbose output should be shown

icon_test *icon_test*

Description

"icon" is an abbreviation for the "interconnectivity" of a network or graph.

Usage

```
icon_test(nodes1 = NULL, nodes2 = NULL, s = 100, verbose = TRUE, ...)
```

Arguments

nodes1	[NULL] the first network. See network_overlap().
nodes2	[NULL] the second network. See network_overlap().
s	[100] the number of random permutations to make.
verbose	[TRUE] Extent of text shown in the console.
...	Additional argumetns are passed on to the specific test performed

Details

This function handles different inputs and directs them to the appropriate "icon" testing method. Depending on the values given to "nodes1" and "nodes2," a different specific test is performed.

Note that the specific functions called make use of the "param" attribute of each input. These parameters are populated by network_overlap() so that the permutation reflects the exact procedure that was done to generate "nodes1" and/or "nodes2."

Value

metrics and significance of the network overlap

Examples

```
## Not run:
icon_test( nodes1=n, s=10)

## End(Not run)
```

load_all_protein_coding_symbols
load_all_protein_coding_symbols

Description

The character array returned is, by default, all human protein coding gene symbols. This variable defines the "universe of possible genes" for use in enrichment. Users should load a different "universe" or filter this one down to the most appropriate setting for their current study. For example, if running RNA-Seq, genes are in the universie if they are detected in any sample.

Usage

```
load_all_protein_coding_symbols()
```

Value

A unique list of gene symbols for protein coding genes according to EnsDb.Hsapiens.v86

load_geneset_symbols *load_geneset_symbols*

Description

For most applications, this function is used internally by term_enrichment(). Users may call this function directly in some cases to force FDR adjustment to be across multiple resources. See Vignette for more details.

Usage

```
load_geneset_symbols(gmt = NA, gmt_dir = "", verbose = TRUE)
```

Arguments

`gmt`	Either 1) name of pre-loaded resource (i.e. names(geneset_list)) or 2) gmt file containing annotation resources for enrichment annotation
`gmt_dir`	location of gmt file named in gmt parameter
`verbose`	print results to screen

Details

load_geneset_symbols allows the user to specify an annotation resource (e.g. Gene Ontology terms) to use in enrichment analysis. The expectation is that the annotation resource contains of at least one set of genes in the form of a list. The RITAN package comes with 15 pre-loaded annotation resources. The default active annotation resources are GO, ReactomePathways, KEGG_filtered_canonical_pathways, and MSigDB_Hallmarks.

The result of calling this function is to set the variable "active_genesets" which will be used by further functions.

Value

R list object named active_genesets

Examples

```
## Load generic GO-slim terms
require(RITANdata)
load_geneset_symbols("GO_slim_generic")
print(length(active_genesets))
print(head(active_genesets[[1]]))

## Not run:
## load the default set of resources into "active_genesets"
load_geneset_symbols()

## Use only the Reactome Pathways annotation resource.
load_geneset_symbols(gmt="ReactomePathways")

## Suppresses output message describing the annotation resource and size.
load_geneset_symbols(gmt="ReactomePathways", verbose=FALSE)

## To list the available resources within RITAN:
print(names(geneset_list))

## You can also load your own data
load_geneset_symbols(gmt="myFile.gmt")

## End(Not run)
```

`network_overlap` *network_overlap*

Description

network_overlap

Usage

```
network_overlap(
  gene_list = NA,
  resources = c("PID", "TFe", "dPPI", "CCSB", "STRING"),
  minStringScore = 700,
  minHumanNetScore = 0.4,
  minScore = 0,
  verbose = TRUE,
  dedup = TRUE,
  directed_net = FALSE,
  include_neighbors = FALSE,
  STRING_cache_directory = NA,
  STRING_species = 9606,
  STRING_version = "10"
)
```

Arguments

`gene_list`	A list of genes to use. The function will identify edges across resources for or among these genes; identify the induced subnetwork around the gene_list.
`resources`	Name of network resource(s) to use.
`minStringScore`	If STRING is among the resources, only edges of at least the indicated score will be included.
`minHumanNetScore`	If HumanNet is among the resources, only edges of at least the indicated score will be included.
`minScore`	Same as above, but used for any other networks where "score" is provided
`verbose`	If TRUE (default), the function will update the user on what it is doing and how many edges are identified for each resource.
`dedup`	If TRUE (Default = TRUE), remove edges reported by multiple resources. The edge type will be a semi-colon delimited list of the resources that had reported the interaction.
`directed_net`	Logical indicating if the network resources should be interpreted as directed.
`include_neighbors`	Logical to include 1st neighbors of "gene_list" (genes not in gene_list, but directly connected to them) in the induced subnetwork.
`STRING_cache_directory`	A direcotry where STRING data files are cached to speed up subsequent queries; no need to re-download. If NA (the default), caches STRING data in your Rpackages directory. If "", uses a temporary directory that is cleared when the R-session closes.
`STRING_species`	Sepcies taxon ID (number) to use in searching STRING data. (Default = 9606)
`STRING_version`	Version of the STRING database (Default = "10")

Value

Data table describing the induced subnetwork for "gene_list" across the requested resources.

Examples

```
## Get interactions among a list of genes from the PID: Pathway Interaction Database
require(RITANdata)
myGeneSet <- c('BRCA1','RAD51C','VAV1','HRAS','ABCC1','CYP1B1','CYP3A5')
sif <- network_overlap( myGeneSet, resources = 'PID')
print(sif)

## Not run:
## Get the PPI network induced by genes within myGeneSet
## Use 4 seperate resources, but trim STRING to only include more confident interactions
sif <- network_overlap( myGeneSet, c('dPPI','PID','CCSB','STRING'), minStringScore = 500 )

## End(Not run)
```

---

plot.term_enrichment *plot.term_enrichment*

Description

plot.term_enrichment

Usage

```
## S3 method for class 'term_enrichment'
plot(x = NA, min_q = 0.05, max_terms = 25, extend_mar = c(0, 10, 0, 0), ...)
```

Arguments

x	data frame returned by term_enrichment
min_q	Only q-values more significant than this threshold will be plotted. Default = 0.05.
max_terms	Up to max_terms will be plotted. Default = 25.
extend_mar	Term names can be long. We attempt to keep them readable by extending the left-hand-side margins automatically. Default = c(0,10,0,0) added to par()$mar.
...	Additional arguments are passed on to plot()

Value

silent return from plot

Examples

```
require(RITANdata)
e <- term_enrichment(vac1.day0vs31.de.genes, resources = 'GO_slim_generic')
plot(e, min_q = .1)
```

plot.term_enrichment_by_subset
plot.term_enrichment_by_subset

Description

plot.term_enrichment_by_subset

Usage

```
## S3 method for class 'term_enrichment_by_subset'
plot(
  x,
  show_values = TRUE,
  annotation_matrix = NA,
  low = "white",
  high = "#2166AC",
  return_ggplot_object = FALSE,
  label_size_x = 16,
  label_angle_x = -30,
  label_size_y = 9,
  wrap_y_labels = 20,
  grid_line_color = "white",
  mid = 0,
  cap = NA,
 annotation_palates = c("Reds", "Greens", "Purples", "Greys", "BuPu", "RdPu", "BrBG",
    "PiYG", "Spectral"),
  annotation_legend_x = -0.3,
  trim_resource_names = TRUE,
  ...
)
```

Arguments

`x` data frame returned by term_enrichment_by_subset

`show_values` True or False, plot values on the heatmap

`annotation_matrix`
a matrix() of group-levle characteristics - same number of columns as "m"

`low` color for low end of range

`high` color for high end of range

`return_ggplot_object`
logical flag (default FALSE) that if TRUE, the ggplot object for the plot is returned

`label_size_x` size of text for x label. Default lable_size_x=16

`label_angle_x` angle for text for x label. Default is -30 degrees label_size_y size of text for y label. Default label_size_y=9

wrap_y_labels Number of characters to wrap row labels

grid_line_color
color o grid lines between cells. Default is white.

mid sets lower threshold for color scale

cap Clip numeric values to this maximum threshold

annotation_palates
Color palates (RColorBrewer) used for each row of the annotation matrix

annotation_legend_x
offset for placing the legend

trim_resource_names
[TRUE] remove any text in rownames preceeding a period characte. This convension is usually used in RITAN to prepend the resource name to the term name, which may not be needed in plotting.

... further areguments are not used at this time. If the user wants to modify the plot, use return_ggplot_object = TRUE.

Value

silent return, unless return_ggplot_object==TRUE. Then, the ggplot object for the plot is returned.

Examples

```
## Create list of gene sets to evaluate.
##  This example is from a vaccine study where we pre-generated differentially expressed genes.
##   This object will be passed to the groups parameter.
require(RITANdata)
vac1.de.genes <- list(vac1.day0vs31.de.genes, vac1.day0vs56.de.genes)
names(vac1.de.genes) <- c("Day0vs31", "Day0vs56")
print(str(vac1.de.genes))

## Not run:
## Run term_enrichment_by_subset on the two results.
##   This function usually takes a few seconds to a minute to run.
m <- term_enrichment_by_subset(groups = vac1.de.genes, q_value_threshold = .9)
summary(m)
plot( m, label_size_y = 4, show_values = FALSE )

## End(Not run)
```

---

readGMT *readGMT*

Description

Created for simplification of reading .gmt files into RITAN.

Usage

```
readGMT(f = NA)
```

Arguments

f	GMT file name. Please provide a full path if the file is not in the current working directory.

Value

A list() where the name of each entry is the term (first column of GMT file) and the value is a chr array of genes associated with the term.

Examples

```
# Make an example list() to show the GMT format
set <- list( term1=c('gene_name1','gene_name2'),
             term2=c('gene_name3','gene_name4','gene_name5') )
## Not run:
# Write a GMT file for "set"
writeGMT( set, 'my_file.gmt' )

# Reading GMT files
geneset <- readGMT( 'my_file.gmt' )

# Additional GMT files are available from multiple sources.
#   We recommend:
#   http://software.broadinstitute.org/gsea/msigdb/

## End(Not run)
```

---

readSIF *readSIF*

Description

This function reads a data table into R; the data table describes network interactions. It is named for the Simple Interaction Format (SIF), but can read any data table if the users identifies which columns contain the pertinent data (see below).

Usage

```
readSIF(
  file = NA,
  header = FALSE,
  sep = "\t",
  as.is = TRUE,
```

```
  p1 = 1,
  p2 = 2,
  et = 3,
  score = NA,
  ...
)
```

Arguments

`file`	location of file
`header`	indicator of presense of header on file
`sep`	file delimiter - used by read.table()
`as.is`	logical (default TRUE)
`p1`	Column number for the 1st entity. Default = 1.
`p2`	Column number for the 2nd entity. Default = 2.
`et`	Column number for the edge type. Default = 3. Optionally, it may be a string label to be used as the edge type for all interactions from the input file.
`score`	Column number for edge scores or weights. Default = NA (no score read).
`...`	Other options to read.table().

Details

The SIF file format is a 3-column format, with an optional 4th column: <entity-1><tab><edge-type><tab><entity-2><tab><score>

Entities may be genes, proteins, metabolites, etc. The edge type typically conveys the type of relationship that exists between the two entities, such as physical interaciton, phosphorylation, or activation.

Value

Returns a data.frame with 3 (or 4) columns of data.

Examples

```
# Make a simple example to show the SIF file format
s <- matrix(c('gene1','gene2','PPI',
              'gene1','gene3','Chip-Seq',
              'gene4','gene5','PPI'), ncol=3, byrow=TRUE)
## Not run:
# Read a SIF file
write.table( s, "myFile.sif", sep='\t', col.names=FALSE, row.names=FALSE )
sif <- readSIF("myFile.sif")

## End(Not run)
```

resource_reduce

resource_reduce Merge terms across resources to reduce the number of redundant and semi-redundant terms

Description

resource_reduce Merge terms across resources to reduce the number of redundant and semi-redundant terms

Usage

```
resource_reduce(genesets = NULL, min_overlap = 0.8, verbose = TRUE)
```

Arguments

genesets	the input genesets to consider. May be from one or multiple resources.
min_overlap	terms that share at least this fraction of genes will be merged
verbose	if TRUE, print status and summary output

Value

the list of terms, after merging to reduce redundnat and semi-redundant terms

Examples

```
require(RITANdata)
r <- resource_reduce( geneset_list$DisGeNet )
```

show_active_genesets_hist

show_active_genesets_hist

Description

function to plot distribution of size of active_genesets object

Usage

```
show_active_genesets_hist(nbins = 50, ...)
```

Arguments

nbins	Number of bins to include in histogram
...	further argumants are passed on to plot()

Value

NULL. The plot is shown.

Examples

```
require(RITANdata)
load_geneset_symbols('GO_slim_generic')
show_active_genesets_hist()

## Not run:
## Show the distribution of geneset sizes for the default set of geneset resources
load_geneset_symbols()
show_active_genesets_hist()

## Show the distribution of geneset sizes for a specific resource
load_geneset_symbols(gmt="ReactomePathways")
show_active_genesets_hist()


## End(Not run)
```

`summary.term_enrichment`

summary.term_enrichment

Description

summary.term_enrichment

Usage

```
## S3 method for class 'term_enrichment'
summary(object, ...)
```

Arguments

`object`	data frame returned by term_enrichment()
`...`	Further arguments are passed on to head()

Value

the data.frame of top enrichment results

Examples

```
require(RITANdata)
e <- term_enrichment( vac1.day0vs31.de.genes, "MSigDB_Hallmarks" )
summary(e, n=3)
```

summary.term_enrichment_by_subset
summary.term_enrichment_by_subset

Description

summary.term_enrichment_by_subset

Usage

```
## S3 method for class 'term_enrichment_by_subset'
summary(object, verbose = TRUE, ...)
```

Arguments

`object`	data frame returned by term_enrichment_by_subset()
`verbose`	if TRUE (default), print a header describing the data type
`...`	Further arguments are passed on to head()

Value

the data.frame of top enrichment results

Examples

```
require(RITANdata)
vac1.de.genes <- list(vac1.day0vs31.de.genes, vac1.day0vs56.de.genes)
names(vac1.de.genes) <- c("Day0vs31", "Day0vs56")
e <- term_enrichment_by_subset(vac1.de.genes, "MSigDB_Hallmarks", q_value_threshold = 0.1 )
summary(e)
```

term_enrichment *term_enrichment*

Description

term_enrichment evaluates the input gene list for enrichment within each of the annotation resources. This differs from the enrichment_symbols function which evaluates the gene list for enrichment against all of the annotation resources grouped together.

Usage

```
term_enrichment(
  geneset,
  resources = resources.default,
  report_resources_separately = FALSE,
  verbose = TRUE,
  all_symbols = NA,
  filter_to_intersection = FALSE,
  ...
)
```

Arguments

`geneset`	vector of gene symbols to be evaluated
`resources`	list containing the reference gene sets to test for enrichment
`report_resources_separately`	logical (default FALSE) flag to report enrichments seperately for each requested resource, or to combine them and produce FDR adjustment across the combined set
`verbose`	print the top results for each annotation resource
`all_symbols`	the background/global set of gene symbols (study dependent; we provide all protien coding genes as a default)
`filter_to_intersection`	[FALSE] should the background and foreground genesets be subsetted to one another?
`...`	further arguments are passed on to enrichment_symbols()

Value

results matrix of input gene list compared to active gene sets. Q value is calculated within each of the active gene sets.

Examples

```
## Check if there is enrichment for any "Hallmark" functions within a input set of genes
require(RITANdata)
myGeneSet <- c('BRCA1','RAD51C','VAV1','HRAS','ABCC1','CYP1B1','CYP3A5')
e <- term_enrichment(myGeneSet, "MSigDB_Hallmarks")
print( e[1:2, -6] )

## Not run:
term_enrichment(geneset = vac1.day0vs31.de.genes)
term_enrichment(geneset = vac1.day0vs31.de.genes, resources = "MSigDB_Hallmarks")
vac1.day0v31.enrichment <- term_enrichment(geneset = vac1.day0vs31.de.genes, verbose = FALSE)

## End(Not run)
```

`term_enrichment_by_subset`
term_enrichment_by_subset

Description

Run enrichment simultaneously across a group of prioritized gene lists. For example, in a time course dataset, one may have a different list of genes that are differentially expressed at each time point. This function facilitates rapid evaluation of term enrichment across time point comparisons. Alternatively, one may have a different list of differentially expressed genes by drug treatment, environmental condition, ect.

Usage

```
term_enrichment_by_subset(
  groups = NA,
  resources = resources.default,
  q_value_threshold = 0.01,
  verbose = TRUE,
  display_type = "q",
  phred = TRUE,
  ...
)
```

Arguments

`groups`	A list() of genes for enrichment. Each entry in the list() is an input set of genes. Enrichment is performed for each of these entries.
`resources`	character vector for which resources to use in enrichment
`q_value_threshold`	minimum q-value (FDR adjusted p-value) in any group for the term to be included in results
`verbose`	print additional status updates on what the function is doing
`display_type`	Flag for which data type will be returned. One of "q" (default) for q-values, "p" for unadjusted p-values, or "n" for the number of genes overlapping the term.
`phred`	Logical flag (default TRUE) to return the -log10 of p/q values
`...`	Further arguments are passed on to enrichment_symbols()

Value

Returns a term-by-study matrix of enrichment values (value determined by "display_type")

Examples

```
## Create list of gene sets to evaluate.
##  This example is from a vaccine study where we pre-generated differentially expressed genes.
##   This object will be passed to the groups parameter.
require(RITANdata)
vac1.de.genes <- list(vac1.day0vs31.de.genes, vac1.day0vs56.de.genes)
names(vac1.de.genes) <- c("Day0vs31", "Day0vs56")
print(str(vac1.de.genes))

## Not run:
## Run term_enrichment_by_subset on the two results.
##   This function usually takes a few seconds to a minute to run.
m <- term_enrichment_by_subset(groups = vac1.de.genes, q_value_threshold = .9)
summary(m)
plot( m, label_size_y = 4, show_values = FALSE )

## End(Not run)
```

---

vac1.day0vs31.de.genes *This dataset is included as an example in the package:*

Description

This dataset is included as an example in the package:

Usage

```
vac1.day0vs31.de.genes
```

Format

An object of class `character` of length 669.

Value

differentially expressed genes at 31 days post-vaccination with vaccine1

References

https://www.ncbi.nlm.nih.gov/pubmed/26755593

Examples

```
## Not run:
 #data("vac1.day0vs31.de.genes")
 te <- term_enrichment(geneset = vac1.day0vs31.de.genes)

## End(Not run)
```

---

vac1.day0vs56.de.genes
This dataset is included as an example in the package:

Description

This dataset is included as an example in the package:

Usage

```
vac1.day0vs56.de.genes
```

Format

An object of class `character` of length 471.

Value

differentially expressed genes at 56 days post-vaccination with vaccine1

References

https://www.ncbi.nlm.nih.gov/pubmed/26755593

Examples

```
## Not run:
 #data("vac1.day0vs56.de.genes")
 te <- term_enrichment(geneset = vac1.day0vs56.de.genes)

## End(Not run)
```

vac2.day0vs31.de.genes *This dataset is included as an example in the package:*

Description

This dataset is included as an example in the package:

Usage

```
vac2.day0vs31.de.genes
```

Format

An object of class `character` of length 522.

Value

differentially expressed genes at 31 days post-vaccination with vaccine2

References

https://www.ncbi.nlm.nih.gov/pubmed/26755593

Examples

```
## Not run:
 #data("vac2.day0vs31.de.genes")
 te <- term_enrichment(geneset = vac2.day0vs31.de.genes)

## End(Not run)
```

vac2.day0vs56.de.genes *This dataset is included as an example in the package:*

Description

This dataset is included as an example in the package:

Usage

```
vac2.day0vs56.de.genes
```

Format

An object of class `character` of length 660.

Value

differentially expressed genes at 56 days post-vaccination with vaccine2

References

https://www.ncbi.nlm.nih.gov/pubmed/26755593

Examples

```
## Not run:
 #data("vac2.day0vs56.de.genes")
 te <- term_enrichment(geneset = vac2.day0vs56.de.genes)

## End(Not run)
```

`writeGMT` *writeGMT*

Description

Created for future use and simplification of writing .gmt files from the package.

Usage

```
writeGMT(s, file = NA, link = rep("", length(s)))
```

Arguments

`s`	list of gene sets in current R session. Each entry will become a row in the GMT file.
`file`	file name to write to
`link`	default is "". This is the second column of a GMT file and is usually a hyperlink or note about the origin of the term

Value

Nothing is returned. A file is written.

Examples

```
# Make an example list() to show the GMT format
set <- list( term1=c('gene_name1','gene_name2'),
             term2=c('gene_name3','gene_name4','gene_name5') )
## Not run:
# Write a GMT file for "set"
writeGMT( set, 'my_file.gmt')

## End(Not run)
```

---

write_simple_table *write_simple_table*

Description

This is a simple wrapper around "write.table" that writes a tab-delimited table with column names, no quoting, and no row names.

Usage

```
write_simple_table(d = NULL, f = NULL, ...)
```

Arguments

d	R data object
f	file path
...	further options passed on to write.table

Value

invisible (nothing is returned)

Examples

```
## Not run:
simple wrapper around write.table for writing a tab-delimieted, no row names, tab-seperated file

## End(Not run)
```

Index