Package 'mulea'

May 9, 2026

Type Package

Title Enrichment Analysis Using Multiple Ontologies and False Discovery Rate

Version 1.1.1

Description Background - Traditional gene set enrichment analyses are typically limited to a few ontologies and do not account for the interdependence of gene sets or terms, resulting in overcorrected p-values. To address these challenges, we introduce mulea, an R package offering comprehensive overrepresentation and functional enrichment analysis. Results - mulea employs a progressive empirical false discovery rate (eFDR) method, specifically designed for interconnected biological data, to accurately identify significant terms within diverse ontologies. mulea expands beyond traditional tools by incorporating a wide range of ontologies, encompassing Gene Ontology, pathways, regulatory elements, genomic locations, and protein domains. This flexibility enables researchers to tailor enrichment analysis to their specific questions, such as identifying enriched transcriptional regulators in gene expression data or overrepresented protein domains in protein sets. To facilitate seamless analysis, mulea provides gene sets (in standardised GMT format) for 27 model organisms, covering 22 ontology types from 16 databases and various identifiers resulting in almost 900 files. Additionally, the muleaData ExperimentData Bioconductor package simplifies access to these pre-defined ontologies. Finally, mulea's architecture allows for easy integration of user-defined ontologies, or GMT files from external sources (e.g., MSigDB or Enrichr), expanding its applicability across diverse research areas. Conclusions - mulea is distributed as a CRAN R package. It offers researchers a powerful and flexible toolkit for functional enrichment analysis, addressing limitations of traditional tools with its progressive eFDR and by supporting a variety of ontologies. Overall, mulea fosters the exploration of diverse biological questions across various model organisms.

biocViews Annotation, DifferentialExpression, GeneExpression, GeneSetEnrichment, GO, GraphAndNetwork, MultipleComparison, Pathways, Reactome, Software, Transcription, Visualization

License GPL-2

Depends R (>= 4.0.0)

Imports data.table (>= 1.13.0), dplyr, fgsea (>= 1.0.2), ggplot2, ggraph (>= 2.0.3), magrittr (>= 2.0.3), methods, parallel (>= 4.0.2), plyr (>= 1.8.4), Rcpp, readr, rlang, scales, stats, stringi, tibble, tidygraph, tidyverse

Suggests devtools, knitr, rmarkdown, testthat (>= 3.1.4)

LinkingTo Rcpp

VignetteBuilder knitr

URL https://github.com/ELTEbioinformatics/mulea

BugReports https://github.com/ELTEbioinformatics/mulea/issues

RoxygenNote 7.3.2

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

Author Cezary Turek [aut] (ORCID: <https://orcid.org/0000-0002-1445-5378>), Marton Olbei [aut] (ORCID: <https://orcid.org/0000-0002-4903-6237>), Tamas Stirling [aut, cre] (ORCID: <https://orcid.org/0000-0002-8964-6443>), Gergely Fekete [aut] (ORCID: <https://orcid.org/0000-0001-9939-4860>), Ervin Tasnadi [aut] (ORCID: <https://orcid.org/0000-0002-4713-5397>), Leila Gul [aut], Balazs Bohar [aut] (ORCID: <https://orcid.org/0000-0002-3033-5448>), Balazs Papp [aut] (ORCID: <https://orcid.org/0000-0003-3093-8852>), Wiktor Jurkowski [aut] (ORCID: <https://orcid.org/0000-0002-7820-1991>), Eszter Ari [aut, cph] (ORCID: <https://orcid.org/0000-0001-7774-1067>)

Maintainer Tamas Stirling <stirling.tamas@gmail.com>

Repository CRAN

Date/Publication 2024-11-19 10:20:11 UTC

Contents

`filter_ontology` *Filter Ontology*

Description

Filtering ontology to contain entries having number of elements (genes or proteins) between a given range. The reason for this is enrichment analysis results can sometimes be skewed by overly specific or broad entries. Filtering ontologies allows you to customize the size of ontology entries, ensuring your analysis aligns with your desired scope.

Usage

```
filter_ontology(gmt, min_nr_of_elements = NULL, max_nr_of_elements = NULL)
```

Arguments

`gmt`	A `data.frame` which contains the entries (gene or protein sets), imported from a GMT file with the `read_gmt` function.
`min_nr_of_elements`	Minimum number of elements. Ontology entries containing as many or fewer elements (genes or proteins) will be excluded.
`max_nr_of_elements`	Maximum number of elements. Ontology entries containing as many or more elements (genes or proteins) will be excluded.

Value

Return a `data.frame`which contains the entries (gene or protein sets) in a similar format that produced by the `read_gmt` function.

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(
    package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt,
        min_nr_of_elements = 3,
        max_nr_of_elements = 400)
```

gsea-class *Gene Set Enrichment Analysis (GSEA)*

Description

An S4 class to represent the gsea tests in mulea.

Usage

```
## S4 method for signature 'gsea'
run_test(model)
```

Arguments

model	Object of S4 class representing the mulea test.

Value

GSEA object. This object represents the result of the gsea tests.

run_test method for GSEA object. Returns results of the enrichment analysis.

Methods (by generic)

- run_test(gsea): runs test calculations.

Slots

gmt A data.frame representing the ontology GMT.

element_names A vector of elements names (gene or protein names or identifiers) to include in the analysis.

element_scores A vector of numeric values representing a score (*e.g.* *p*-value, *z*-score, log fold change) for each 'element_name', in the same number and order as element_name.

gsea_power A power of weight. Default value is 1.

element_score_type Defines the GSEA score type.

- 'pos': Only positive element_scores
- 'neg': Only negative element_scores
- 'std': standard, containing both positive and negative scores Default value is 'std'.

number_of_permutations The number of permutations used in gsea test. Default value is 1000.

test character

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(
    package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example `data.frame`
scored_gene_tab <- read.delim(file = system.file(package = "mulea", "extdata",
        "ordered_set.tsv"))

# creating the GSEA model
gsea_model <- gsea(gmt = tf_gmt_filtered,
        # the names of elements to test
        element_names = scored_gene_tab$Gene.symbol,
        # the logFC-s of elements to test
        element_scores = scored_gene_tab$logFC,
        # consider elements having positive logFC values only
        element_score_type = "pos",
        # the number of permutations
        number_of_permutations = 10000)
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
        "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
        max_nr_of_elements = 400)

# loading the example `data.frame`
scored_gene_tab <- read.delim(file = system.file(package = "mulea", "extdata",
        "ordered_set.tsv"))

# creating the GSEA model
gsea_model <- gsea(gmt = tf_gmt_filtered,
        # the names of elements to test
        element_names = scored_gene_tab$Gene.symbol,
        # the logFC-s of elements to test
        element_scores = scored_gene_tab$logFC,
        # consider elements having positive logFC values only
        element_score_type = "pos",
        # the number of permutations
        number_of_permutations = 10000)

# running the test
gsea_results <- run_test(gsea_model)
```

list_to_gmt *Convert a list to ontology (GMT)* data.frame.

Description

Converts a list of ontology elements (gene sets) to an ontology (GMT) data.frame object.

Usage

```
list_to_gmt(gmt_list)
```

Arguments

gmt_list	A list with named character vectors. The name will become the 'ontology_id', and the elements in the vector will become the 'list_of_values' in the ontology (GMT) data.frame.

Value

Returns ontology (GMT) data.frame where the 'ontology_name' contains random unique strings.

Examples

```
library(mulea)

# creating a list of gene sets
ontology_list <- list(gene_set1 = c("gene1", "gene2", "gene3"),
        gene_set2 = c("gene4", "gene5", "gene6"))

# converting the list to a ontology (GMT) object
new_ontology_object <- list_to_gmt(ontology_list)
```

MuleaHypergeometricTest-class *PRIVATE class : An S4 class to represent a Hypergeometric tests in mulea.*

Description

PRIVATE class : An S4 class to represent a Hypergeometric tests in mulea.

Usage

```
## S4 method for signature 'MuleaHypergeometricTest'
run_test(model)
```

Arguments

`model` Object of s4 class represents mulea Test.

Value

MuleaHypergeometricTest object. Used as private function.

run_test method for MuleaHypergeometricTest object. Used as private function.

Methods (by generic)

- `run_test(MuleaHypergeometricTest)`: runs test calculations.

Slots

`gmt` A data.frame representing the GMT model.

`element_names` Data to be analysed across the model.

`background_element_names` Background data used for the test.

`ora-class` *An S4 class to represent a set based tests in mulea.*

Description

An S4 class to represent a set based tests in mulea.

Value

ora object. This object represents the result of the overrepresentation test in mulea.

Slots

`method` The overrepresentation (ora) method. Possible values: "Hypergeometric", "SetBasedEnrichment".

`gmt` A `data.frame` representing the ontology GMT.

`element_names` A vector of elements names (gene or protein names or identifiers) representing the target set to analyse. For example differentially expressed genes.

`background_element_names` A vector of elements names (gene or protein names or identifiers) representing all the elements involved in the previous analyses For example all genes that were measured in differential expression analysis.

`p_value_adjustment_method` A character string representing the type of the *p*-value adjustment method. Possible values:

- 'eFDR': empirical false discovery rate correction method
- all `method` options from `stats::p.adjust` documentation.

`number_of_permutations` A numeric value representing the number of permutations used to calculate the eFDR values. Default value is 10000.

`nthreads` Number of processor's threads to use in calculations.

`random_seed` Optional natural number (1, 2, 3, ...) setting the seed for the random generator, to make the results reproducible.

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(
    package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
        max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
        "target_set.txt"))
background_genes <- readLines(system.file(package="mulea", "extdata",
        "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
        # the test set variable
        element_names = sign_genes,
        # the background set variable
        background_element_names = background_genes,
        # the p-value adjustment method
        p_value_adjustment_method = "eFDR",
        # the number of permutations
        number_of_permutations = 10000,
        # the number of processor threads to use
        nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)
```

---

plot_barplot *Plot Barplot*

Description

Plots barplot of p-values.

Usage

```
plot_barplot(
  reshaped_results,
  ontology_id_colname = "ontology_id",
  selected_rows_to_plot = NULL,
```

```
  p_value_type_colname = "eFDR",
  p_value_max_threshold = 0.05
)
```

Arguments

`reshaped_results`
: 'data.table' in relaxed form, obtained as the output of the `reshape_results` function. The data source for generating the barplot.

`ontology_id_colname`
: Character, specifies the column name that contains ontology IDs in the input data.

`selected_rows_to_plot`
: A numeric vector specifying which rows of the reshaped results 'data.frame' should be included in the plot. Default is 'NULL'.

`p_value_type_colname`
: Character, specifies the column name for *p*-values in the input data. Default is 'eFDR'.

`p_value_max_threshold`
: Numeric, representing the maximum *p*-value threshold for filtering data. Default is 0.05.

Details

Create a customized barplot of *p*-values, facilitating visual exploration and analysis of statistical significance within ontology categories.

Value

Returns a barplot.

See Also

`reshape_results`

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
    "target_set.txt"))
background_genes <- readLines(system.file(
    package="mulea", "extdata", "background_set.txt"))
```

```
# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

# reshaping results for visualisation
ora_reshaped_results <- reshape_results(model = ora_model,
    model_results = ora_results,
    # choosing which column to use for the indication of significance
    p_value_type_colname = "eFDR")

# Plot barplot
plot_barplot(reshaped_results = ora_reshaped_results,
    # the column containing the names we wish to plot
    ontology_id_colname = "ontology_id",
    # upper threshold for the value indicating the significance
    p_value_max_threshold = 0.05,
    # column that indicates the significance values
    p_value_type_colname = "eFDR")
```

---

plot_graph *Plot Graph (Network)*

Description

Plots graph representation of enrichment results.

Usage

```
plot_graph(
  reshaped_results,
  ontology_id_colname = "ontology_id",
  ontology_element_colname = "element_id_in_ontology",
  shared_elements_min_threshold = 0,
  p_value_type_colname = "eFDR",
  p_value_max_threshold = 0.05
)
```

Arguments

`reshaped_results`
: Character, the input `data.table` containing the reshaped results.

`ontology_id_colname`
: Character, the name of the column in the reshaped results that contains ontology identifiers or names. Default value is 'ontology_id'.

`ontology_element_colname`
: Character, the name of the column in the reshaped results that contains element identifiers within the ontology. Default value is 'element_id_in_ontology'.

`shared_elements_min_threshold`
: Numeric, threshold specifying the minimum number of shared elements required between two ontologies to consider them connected by an edge on the graph. Default value is 0.

`p_value_type_colname`
: Character, the name of the column in the reshaped results that contains the type of *p*-values associated with the ontology elements. Default value is 'eFDR'.

`p_value_max_threshold`
: Numeric, a threshold value for filtering rows in the reshaped results based on the *p*-values. Rows with *p*-values greater than this threshold will be filtered out. Default value is 0.05.

Details

This function generates a graph (network) visualization of the enriched ontology entries. On the plot each node represents an ontology entry below a given *p*-value threshold, and is coloured based on its significance level. A connection (edge) is drawn between two nodes if they share at least one common element (gene) belonging to the target set – in the case of ORA results – or all analysed elements – in the case of GSEA results.

Value

Returns a graph plot.

See Also

`reshape_results`

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
```

```
    "target_set.txt"))
background_genes <- readLines(system.file(
    package="mulea", "extdata", "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

# reshaping results for visualisation
ora_reshaped_results <- reshape_results(model = ora_model,
    model_results = ora_results,
    # choosing which column to use for the indication of significance
    p_value_type_colname = "eFDR")

# Plot graph
plot_graph(reshaped_results = ora_reshaped_results,
    # the column containing the names we wish to plot
    ontology_id_colname = "ontology_id",
    # upper threshold for the value indicating the significance
    p_value_max_threshold = 0.05,
    # column that indicates the significance values
    p_value_type_colname = "eFDR")
```

---

plot_heatmap *Plot Heatmap*

Description

Plots heatmap of enriched terms and obtained p-values.

Usage

```
plot_heatmap(
  reshaped_results,
  ontology_id_colname = "ontology_id",
  ontology_element_colname = "element_id_in_ontology",
  p_value_type_colname = "eFDR",
  p_value_max_threshold = 0.05
)
```

Arguments

`reshaped_results`
: data.table in relaxed form, obtained as the output of the `reshape_results` function. The data source for generating the barplot.

`ontology_id_colname`
: Character, specifies the column name that contains ontology IDs in the input data.

`ontology_element_colname`
: Character, specifying the column name that contains ontology elements or terms in the input data. Default: 'element_id_in_ontology'.

`p_value_type_colname`
: Character, specifies the column name for p-values in the input data. Default is 'eFDR'.

`p_value_max_threshold`
: Numeric, representing the maximum p-value threshold for filtering data. Default is 0.05.

Details

The `plot_heatmap` function provides a convenient way to create a ggplot2 heatmap illustrating the significance of enriched terms within ontology categories based on their associated p-values.

Value

Returns a ggplot2 heatmap.

See Also

`reshape_results`

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
    "target_set.txt"))
background_genes <- readLines(system.file(
    package="mulea", "extdata", "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
```

```
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

# reshaping results for visualisation
ora_reshaped_results <- reshape_results(
    model = ora_model,
    model_results = ora_results,
    # choosing which column to use for the indication of significance
    p_value_type_colname = "eFDR")

# Plot heatmap
plot_heatmap(reshaped_results = ora_reshaped_results,
    # the column containing the names we wish to plot
    ontology_id_colname = "ontology_id",
    # column that indicates the significance values
    p_value_type_colname = "eFDR")
```

`plot_lollipop` *Plot Lollipop*

Description

Plots lollipop plot of p-values.

Usage

```
plot_lollipop(
  reshaped_results,
  ontology_id_colname = "ontology_id",
  selected_rows_to_plot = NULL,
  p_value_type_colname = "eFDR",
  p_value_max_threshold = 0.05
)
```

Arguments

`reshaped_results`
: data.table in relaxed form, obtained as the output of the `reshape_results` function. The data source for generating the barplot.

`ontology_id_colname`
: Character, specifies the column name that contains ontology IDs in the input data.

`selected_rows_to_plot`
: A numeric vector specifying which rows of the reshaped results data frame should be included in the plot. Default is NULL. frame should be included in the plot?

`p_value_type_colname`
: Character, specifies the column name for p-values in the input data. Default is 'eFDR'.

`p_value_max_threshold`
: Numeric, representing the maximum p-value threshold for filtering data. Default is 0.05.

Details

Create a customized lollipop plot of p-values, facilitating visual exploration and analysis of statistical significance within ontology categories.

Value

Returns a lollipop plot

See Also

`reshape_results`

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
    "target_set.txt"))
background_genes <- readLines(system.file(
    package="mulea", "extdata", "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
```

```
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

# reshaping results for visualisation
ora_reshaped_results <- reshape_results(
    model = ora_model,
    model_results = ora_results,
    # choosing which column to use for the indication of significance
    p_value_type_colname = "eFDR")

# Plot lollipop
plot_lollipop(reshaped_results = ora_reshaped_results,
    # the column containing the names we wish to plot
    ontology_id_colname = "ontology_id",
    # upper threshold for the value indicating the significance
    p_value_max_threshold = 0.05,
    # column that indicates the significance values
    p_value_type_colname = "eFDR")
```

`read_gmt` *Read GMT File*

Description

Reads gene set or ontology data from a Gene Matrix Transposed (GMT) file and parse into a `data.frame`.

Usage

```
read_gmt(file)
```

Arguments

`file`	Character, a path to a file.

Value

Returns a `data.frame` with three columns:

- 'ontology_id': Ontology identifier that uniquely identifies the element within the referenced ontology.
- 'ontology_name': Ontology name or description that provides a user-friendly label or textual description for the 'ontology_id'.
- 'list_of_values': Associated genes or proteins that is a vector of identifiers of genes or proteins belonging to the 'ontology_id'.

Examples

```
# import example gene set
library(mulea)
tf_gmt <- read_gmt(file = system.file(
    package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
```

`reshape_results` *Reshape Results*

Description

This function takes a model and model_results data, reshapes them into a suitable format for plotting, and returns the resulting data frame, which can be used for further analysis or visualization.

Usage

```
reshape_results(
  model = NULL,
  model_results = NULL,
  model_ontology_col_name = "ontology_id",
  ontology_id_colname = "ontology_id",
  p_value_type_colname = "eFDR",
  p_value_max_threshold = TRUE
)
```

Arguments

`model`	a mulea model, created by the `ora` or the `gsea` functions.
`model_results`	Result `data.frame` returned by the `run_test` function.
`model_ontology_col_name`	Character, specifies the column name in the model that contains ontology IDs. It defines which column in the model should be used for matching ontology IDs. Possible values are 'ontology_id' and 'ontology_name'. The default value is 'ontology_id'.
`ontology_id_colname`	Character, specifies the column name for ontology IDs in the model results. It indicates which column in the model results contains ontology IDs for merging. Possible values are 'ontology_id' and 'ontology_name'. The default value is 'ontology_id'.
`p_value_type_colname`	Character, specifies the column name for the type or raw or adjusted *p*-value in the result `data.frame` returned by the `run_test` function. The default value is 'eFDR'.
`p_value_max_threshold`	Logical, indicating whether to apply a *p*-value threshold when filtering the resulting data. If TRUE, the function filters the data based on a *p*-value threshold.

Value

Return detailed and relaxed `data.table` where model and results are merged for plotting purposes.

See Also

plot_graph, plot_barplot, plot_heatmap

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(
    package = "mulea", "extdata", "target_set.txt"))
background_genes <- readLines(
    system.file(package="mulea", "extdata", "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

# reshaping results for visualisation
ora_reshaped_results <- reshape_results(model = ora_model,
    model_results = ora_results,
    # choosing which column to use for the indication of significance
    p_value_type_colname = "eFDR")
```

`run_test` *Run enrichment analysis procedure*

Description

This is a generic function that chooses an enrichment analysis procedure based on the model class and runs the analysis.

Usage

```
run_test(model)

## S4 method for signature 'ora'
run_test(model)
```

Arguments

model	Object of S4 class representing the mulea test.

Details

The function requires the definition of a model. Models currently implemented in mulea include Gene Set Enrichment Analysis (GSEA) and Over-Representation Analysis (ORA). These models must be defined through their specific functions which are provided in this package.

Value

Results in form of data.frame. Structure of data.frame depends on object processed by this generic method. In the case of run_test was used with the model generated by the ora function the returned data.frame contains the following columns:

1. 'ontology_id': Identifiers of the ontology elements.
2. 'ontology_name': Names of the ontology elements.
3. 'nr_common_with_tested_elements': Number of common elements between the ontology element and the vector defined by the element_names parameter of the ora function.
4. 'nr_common_with_background_elements': Number of common elements between the ontology element and the vector defined by the background_element_names parameter of the ora function.
5. 'p_value': The raw *p*-value of the overrepresentation analysis.
6. The adjusted *p*-value. The column named based on the p_value_adjustment_method parameter of the ora function, *e.g.* 'eFDR'

In the case of run_test was used with the model generated by the gsea function the returned data.frame contains the following columns:

1. 'ontology_id': Identifiers of the ontology elements.
2. 'ontology_name': Names of the ontology elements.
3. 'nr_common_with_tested_elements': Number of common elements between the ontology element and the vector defined by the element_names parameter of the gsea function.
4. 'p_value': The raw *p*-value of the gene set enrichment analysis.
5. 'adjusted_p_value': The adjusted *p*-value.

run_test method for ora object. Returns the results of the overrepresentation analysis.

Methods (by class)

- run_test(ora): ora test.

See Also

gsea, ora

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file( package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
    max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
    "target_set.txt"))
background_genes <- readLines(system.file(package="mulea", "extdata", "
    background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
    # the test set variable
    element_names = sign_genes,
    # the background set variable
    background_element_names = background_genes,
    # the p-value adjustment method
    p_value_adjustment_method = "eFDR",
    # the number of permutations
    number_of_permutations = 10000,
    # the number of processor threads to use
    nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)

library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(
        package="mulea", "extdata",
        "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))
tf_gmt_filtered <- filter_ontology(gmt = tf_gmt, min_nr_of_elements = 3,
        max_nr_of_elements = 400)

# loading the example data
sign_genes <- readLines(system.file(package = "mulea", "extdata",
        "target_set.txt"))
background_genes <- readLines(system.file(package="mulea", "extdata",
        "background_set.txt"))

# creating the ORA model
ora_model <- ora(gmt = tf_gmt_filtered,
        # the test set variable
```

```
        element_names = sign_genes,
        # the background set variable
        background_element_names = background_genes,
        # the p-value adjustment method
        p_value_adjustment_method = "eFDR",
        # the number of permutations
        number_of_permutations = 10000,
        # the number of processor threads to use
        nthreads = 2)
# running the ORA
ora_results <- run_test(ora_model)
```

`SetBasedEnrichmentTest-class` *PRIVATE class : An S4 class to represent a Hypergeometric tests in mulea.*

Description

PRIVATE class : An S4 class to represent a Hypergeometric tests in mulea.

Usage

```
## S4 method for signature 'SetBasedEnrichmentTest'
run_test(model)
```

Arguments

`model`	Object of s4 class represents mulea Test.

Value

SetBasedEnrichmentTest object. Used as private function.

run_test method for SetBasedEnrichmentTest object. Used as private function.

Methods (by generic)

- `run_test(SetBasedEnrichmentTest)`: runs test calculations.

Slots

`gmt` A data.frame representing GMT's representation of model.

`element_names` A data from experiment to analyse across model.

`pool` A background data to count test.

`nthreads` Number of processor's threads used in calculations.

`random_seed` Setup seed for random generator.

SubramanianTest-class *PRIVATE class : An S4 class to represent a ranked based tests in mulea.*

Description

PRIVATE class : An S4 class to represent a ranked based tests in mulea.

Usage

```
## S4 method for signature 'SubramanianTest'
run_test(model)
```

Arguments

model	Object of s4 class represents mulea Test.

Value

data.frame with presented columns 'ontology_id', 'ontology_name', 'nr_common_with_tested_elements', 'p_value', 'adjusted_p_value'

run_test method for SubramanianTest object. Used as private function.

Methods (by generic)

- run_test(SubramanianTest): runs test calculations.

Slots

gmt A data.frame representing the ontology GMT.

element_names A vector of elements names (gene or protein names or identifiers) to include in the analysis.

element_scores A vector of numeric values representing a score (*e.g.* *p*-value, *z*-score, log fold change) for each 'element_name', in the same number and order as element_name.

p A power of weight.

element_score_type Defines the GSEA score type.

- "pos": Only positive element_scores
- "neg": only negative element_scores - "neg" and mixed
- "std": standard – containing both positive and negative scores Default value is "std".

write_gmt *Write GMT file*

Description

Writes gene set or ontology data.frame with specific formatting (columns representing ontology identifiers, descriptions, and associated lists of values) and writes it to a file in a standardized Gene Matrix Transposed (GMT) file format.

Usage

```
write_gmt(gmt, file)
```

Arguments

gmt	A data.frame containing the data to be written, imported from a GMT file with the read_gmt function.
file	Character, a path to a file.

Value

Returns the input as a GMT file at a specific location.

Examples

```
library(mulea)

# loading and filtering the example ontology from a GMT file
tf_gmt <- read_gmt(file = system.file(
    package="mulea", "extdata",
    "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt"))

# writing the filtered ontology to a GMT file

write_gmt(
    gmt = tf_gmt,
    file = "Transcription_factor_RegulonDB_Escherichia_coli_GeneSymbol.gmt")
```

Index