Package 'ReactomeGSA'

April 15, 2020

Type Package

Title Client for the Reactome Analysis Service for comparative multi-omics gene set analysis

Version 1.0.0

Description
The ReactomeGSA packages uses Reactome's online analysis service to perform a multi-omics gene set analysis. The main advantage of this package is, that the retrieved results can be visualized using REACTOME's powerful webapplication.
Since Reactome's analysis service also uses R to perfrom the actual gene set analysis you will get similar results when using the same packages (such as limma and edgeR) locally.
Therefore, if you only require a gene set analysis, different packages are more suited.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Imports jsonlite, httr, progress, ggplot2, methods

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown, ReactomeGSA.data, Biobase, devtools

Enhances limma, edgeR

VignetteBuilder knitr

biocViews GeneSetEnrichment, Proteomics, Transcriptomics, SystemsBiology, GeneExpression, Reactome

BugReports https://github.com/reactome/ReactomeGSA/issues

URL https://github.com/reactome/ReactomeGSA

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

git_branch RELEASE_3_10

git_last_commit 16be774

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Johannes Griss [aut, cre] (<https://orcid.org/0000-0003-2206-9511>)

Maintainer Johannes Griss <johannes.griss@meduniwien.ac.at>

R topics documented:

add_dataset *add_dataset*

Description

Adds a dataset to the analysis request

Usage

```
add_dataset(request, expression_values, name, type, comparison_factor,
  comparison_group_1, comparison_group_2, sample_data = NULL,
  additional_factors = NULL, overwrite = FALSE, ...)
```

Arguments

request	The request to add the dataset to. Commonly a ReactomeAnalysisRequest object.
expression_values	Object containing the expression values of the dataset to add (multiple types supported).
name	character. Name of the dataset. This must be unique within one request.
type	character. The type of the dataset. Get available types using get_reactome_data_types
comparison_factor	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from expression_values or from sample_data.
comparison_group_1	character. Name of the first group within comparison_factor to use for the comparison.
comparison_group_2	character. Name of the second group within comparison_factor to use for the comparison.
sample_data	data.frame (optional) data.frame containing the sample metadata of the expression_values. Depending on the object type of expression_values, this information can also be extracted from there.
additional_factors	vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.
overwrite	boolean. If set to TRUE, datasets with the same name will be overwritten
...	Additional parameters passed to downstream functions. See the respective documentation of whether any additional parameters are supported.

Value

The ReactomeAnalysisRequest object with the added dataset

See Also

Other add_dataset methods: add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAn
add_dataset,ReactomeAnalysisRequest,ExpressionSet-method, add_dataset,ReactomeAnalysisRequest,dat

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)
library(methods)

my_request <- ReactomeAnalysisRequest(method = "Camera")

# since the expression_values object is a limma EList object, the sample_data is
# retrieved from there

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))
```

---

add_dataset,ReactomeAnalysisRequest,data.frame-method

add_dataset - data.frame

Description

Adds a dataset to the analysis request

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest,data.frame'
add_dataset(request,
  expression_values, name, type, comparison_factor, comparison_group_1,
  comparison_group_2, sample_data = NULL, additional_factors = NULL,
  overwrite = FALSE, ...)
```

Arguments

request	ReactomeAnalysisRequest.
expression_values	data.frame. In this case, the sample_data must be set.
name	character. Name of the dataset. This must be unique within one request.
type	character. The type of the dataset. Get available types using get_reactome_data_types
comparison_factor	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from expression_values or from sample_data.
comparison_group_1	character. Name of the first group within comparison_factor to use for the comparison.

comparison_group_2
: character. Name of the second group within comparison_factor to use for the comparison.

sample_data
: data.frame (optional) data.frame containing the sample metadata of the expression_values. Depending on the object type of expression_values, this information can also be extracted from there.

additional_factors
: vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.

overwrite
: boolean. If set to TRUE, datasets with the same name will be overwritten

...
: Additional parameters passed to downstream functions. See the respective documentation of whether any additional parameters are supported.

Value

The ReactomeAnalysisRequest object with the added dataset

See Also

Other add_dataset methods: add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAn
add_dataset,ReactomeAnalysisRequest,ExpressionSet-method, add_dataset

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)
library(methods)

my_request <- ReactomeAnalysisRequest(method = "Camera")

# since the expression_values object is a limma EList object, the sample_data is
# retrieved from there

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))
```

---

add_dataset,ReactomeAnalysisRequest,DGEList-method
add_dataset - DGEList

Description

Adds a dataset to the analysis request

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest,DGEList'
add_dataset(request,
  expression_values, name, type, comparison_factor, comparison_group_1,
  comparison_group_2, sample_data = NULL, additional_factors = NULL,
  overwrite = FALSE, ...)
```

Arguments

request	ReactomeAnalysisRequest.
expression_values	DGEList Here, the sample_data is automaticall extracted from the expression_values object unless sample_data is specified as well.
name	character. Name of the dataset. This must be unique within one request.
type	character. The type of the dataset. Get available types using get_reactome_data_types
comparison_factor	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from expression_values or from sample_data.
comparison_group_1	character. Name of the first group within comparison_factor to use for the comparison.
comparison_group_2	character. Name of the second group within comparison_factor to use for the comparison.
sample_data	data.frame (optional) data.frame containing the sample metadata of the expression_values. Depending on the object type of expression_values, this information can also be extracted from there.
additional_factors	vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.
overwrite	boolean. If set to TRUE, datasets with the same name will be overwritten
...	Additional parameters passed to downstream functions. See the respective documentation of whether any additional parameters are supported.

Value

The ReactomeAnalysisRequest object with the added dataset

See Also

Other add_dataset methods: add_dataset,ReactomeAnalysisRequest,EList-method, add_dataset,ReactomeAnal
add_dataset,ReactomeAnalysisRequest,data.frame-method, add_dataset

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)
library(methods)
```

```
my_request <- ReactomeAnalysisRequest(method = "Camera")

# since the expression_values object is a limma EList object, the sample_data is
# retrieved from there

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))
```

`add_dataset,ReactomeAnalysisRequest,EList-method`
add_dataset - EList

Description

Adds a dataset to the analysis request

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest,EList'
add_dataset(request,
  expression_values, name, type, comparison_factor, comparison_group_1,
  comparison_group_2, sample_data = NULL, additional_factors = NULL,
  overwrite = FALSE, ...)
```

Arguments

`request`	ReactomeAnalysisRequest.
`expression_values`	EList. Here, the sample_data is automaticall extracted from the `expression_values` object unless `sample_data` is specified as well.
`name`	character. Name of the dataset. This must be unique within one request.
`type`	character. The type of the dataset. Get available types using `get_reactome_data_types`
`comparison_factor`	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from `expression_values` or from `sample_data`.
`comparison_group_1`	character. Name of the first group within `comparison_factor` to use for the comparison.
`comparison_group_2`	character. Name of the second group within `comparison_factor` to use for the comparison.

sample_data	data.frame (optional) data.frame containing the sample metadata of the expression_values. Depending on the object type of expression_values, this information can also be extracted from there.
additional_factors	vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.
overwrite	boolean. If set to TRUE, datasets with the same name will be overwritten
...	Additional parameters passed to downstream functions. See the respective documentation of whether any additional parameters are supported.

Value

The ReactomeAnalysisRequest object with the added dataset

See Also

Other add_dataset methods: add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAn
add_dataset,ReactomeAnalysisRequest,data.frame-method, add_dataset

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)
library(methods)

my_request <- ReactomeAnalysisRequest(method = "Camera")

# since the expression_values object is a limma EList object, the sample_data is
# retrieved from there

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))
```

---

add_dataset,ReactomeAnalysisRequest,ExpressionSet-method
add_dataset - ExpressionSet

Description

Adds a dataset to the analysis request

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest,ExpressionSet'
add_dataset(request,
  expression_values, name, type, comparison_factor, comparison_group_1,
  comparison_group_2, sample_data = NULL, additional_factors = NULL,
  overwrite = FALSE, ...)
```

Arguments

request	ReactomeAnalysisRequest.
expression_values	ExpressionSet. Here, the sample_data is automaticall extracted from the expression_values object unless sample_data is specified as well.
name	character. Name of the dataset. This must be unique within one request.
type	character. The type of the dataset. Get available types using get_reactome_data_types
comparison_factor	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from expression_values or from sample_data.
comparison_group_1	character. Name of the first group within comparison_factor to use for the comparison.
comparison_group_2	character. Name of the second group within comparison_factor to use for the comparison.
sample_data	data.frame (optional) data.frame containing the sample metadata of the expression_values. Depending on the object type of expression_values, this information can also be extracted from there.
additional_factors	vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.
overwrite	boolean. If set to TRUE, datasets with the same name will be overwritten
...	Additional parameters passed to downstream functions. See the respective documentation of whether any additional parameters are supported.

Value

The ReactomeAnalysisRequest object with the added dataset

See Also

Other add_dataset methods: add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAn
add_dataset,ReactomeAnalysisRequest,data.frame-method, add_dataset

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)
library(methods)
```

```
my_request <- ReactomeAnalysisRequest(method = "Camera")

# since the expression_values object is a limma EList object, the sample_data is
# retrieved from there

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))
```

`checkRequestValidity` *Check's if a ReactomeAnalysisRequest object is valid*

Description

Check's if a ReactomeAnalysisRequest object is valid

Usage

```
checkRequestValidity(object)
```

Arguments

`object`	The request object to check.

Value

TRUE if the object is valid or a string with the reason why it is not

---

`check_reactome_url` *check_reactome_url*

Description

Makes sure the passed URL is valid. If not URL is passed, the one stored in the options is retrieved

Usage

```
check_reactome_url(reactome_url)
```

Arguments

`reactome_url`	character The URL to test. If `NULL` the URL is retrieved from the options.

Value

character The potentially cleaned / retrieved URL with a trailing "/"

convert_reactome_result *Convert the Reactome JSON result to a ReactomeAnalysisResult object*

Description

Convert the Reactome JSON result to a ReactomeAnalysisResult object

Usage

```
convert_reactome_result(reactome_result)
```

Arguments

reactome_result	The JSON result already converted to R objects (name list)

Value

A ReactomeAnalysisResult object

data_frame_as_string *Converts a data.frame to a string representation*

Description

A data.frame is converted into a single string using '\t' (the characters, not tab) as field delimiter and '\n' (the characters, not newline) as line delimiter

Usage

```
data_frame_as_string(data)
```

Arguments

data	The data.frame to convert

Value

A string representing the passed data.frame

get_fc_for_dataset *get_fc_for_dataset*

Description

Retrieve the fold-changes for all pathways of the defined dataset

Usage

```
get_fc_for_dataset(dataset, pathway_result)
```

Arguments

dataset	Name of the dataset to retrieve the fold changes for.
pathway_result	The data.frame created by the pathways function.

Value

A vector of fold-changes

get_is_sig_dataset *get_is_sig_dataset*

Description

Determines how significant a pathway is across the datasets. Returns the lowest significance.

Usage

```
get_is_sig_dataset(dataset, pathway_result)
```

Arguments

dataset	Name of the dataset
pathway_result	data.frame created by the pathways function

Value

A vector with 3=non-significant, 2=p<=0.05, 1=p<0.01

get_reactome_analysis_result

Retrieves the result of the submitted analysis using perform_reactome_analysis

Description

The result is only available if get_reactome_analysis_status indicates that the analysis is complete.

Usage

```
get_reactome_analysis_result(analysis_id, reactome_url = NULL)
```

Arguments

analysis_id	The running analysis' id
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Value

The result object

get_reactome_analysis_status

Retrieves the status of the submitted analysis using start_reactome_analysis

Description

Retrieves the status of the submitted analysis using start_reactome_analysis

Usage

```
get_reactome_analysis_status(analysis_id, reactome_url = NULL)
```

Arguments

analysis_id	The running analysis' id
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Value

A list containing the id, status (can be "running", "complete", "failed"), description, and completed (numeric between 0 - 1)

get_reactome_data_types *ReactomeGSA supported data types*

Description

ReactomeGSA supported data types

Usage

```
get_reactome_data_types(print_types = TRUE, return_result = FALSE,
  reactome_url = NULL)
```

Arguments

print_types	If set to TRUE (default) a (relatively) nice formatted version of the result is printed.
return_result	If set to TRUE, the result is returned as a data.frame (see below)
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Value

A data.frame containing one row per data type with its id and description.

Author(s)

Johannes Griss

See Also

Other Reactome Service functions: get_reactome_methods

Examples

```
# retrieve the avialable data types
available_types <- get_reactome_data_types(print_types = FALSE,  return_result = TRUE)

# print all data type ids
available_types$id

# simply print the available methods
get_reactome_data_types()
```

get_reactome_methods *get_reactome_methods*

Description

Returns all available analysis methods from the Reactome analysis service.

Usage

```
get_reactome_methods(print_methods = TRUE, print_details = FALSE,
  return_result = FALSE, method = NULL, reactome_url = NULL)
```

Arguments

print_methods	If set to TRUE (default) a (relatively) nice formatted version of the result is printed.
print_details	If set to TRUE detailed information about every method, including available parameters and description are displayed. This does not affect the data returned if return_result is TRUE.
return_result	If set to TRUE, the result is returned as a data.frame (see below)
method	If set to a method's id, only information for this method will be shown. This is especially useful if detailed information about a single method should be retrieved. This does not affect the data returned if return_result is TRUE.
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Details

Every method has a type, a scope, and sometimes a list of allowed values. The type (string, int = integer, float) define the expected data type. The **scope** defines at what level the parameter can be set. *dataset* level parameters can be set at the dataset level (using the add_dataset function) or at the analysis request level (using set_parameters). If these parameters are set at the analysis request level, this overwrites the default value for all datasets. *analysis* and *global* level parameters must only be set at the analysis request level using set_parameters. The difference between these two types of parameters is that while *analysis* parameters influence the results, *global* parameters only influence the behaviour of the analysis system (for example whether a Reactome visualization is created).

Value

If return_result is set to TRUE, a data.frame with one row per method. Each method has a name, description, and (optional) a list of parameters. Parameters again have a name, type, and description.

Author(s)

Johannes Griss

See Also

Other Reactome Service functions: get_reactome_data_types

Examples

```
# retrieve the available methods only in an object
available_methods <- get_reactome_methods(print_methods = FALSE, return_result = TRUE)

# print all method names
available_methods$name

# list all parameters for the first method
first_method_parameters <- available_methods[1, "parameters"]
first_method_parameters

# simply print the available methods
get_reactome_methods()

# get the details for PADOG
get_reactome_methods(print_details = TRUE, method = "PADOG")
```

`get_result` *get_result*

Description

Retrieves a result from a `ReactomeAnalysisResult` object.

Usage

```
get_result(x, type, name)
```

Arguments

`x`	ReactomeAnalysisResult.
`type`	the type of result. Use `result_types` to retrieve all available types.
`name`	the name of the result. Use `names` to retrieve all available results.

Value

A `data.frame` containing the respective result.

See Also

Other ReactomeAnalysisResult functions: `names`,`ReactomeAnalysisResult-method`, `open_reactome`, `pathways`, `plot_correlations`, `plot_volcano`, `reactome_links`, `result_types`

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the available result types
result_types(griss_melanoma_result)

# get the dataset names
```

```
names(griss_melanoma_result)

# get the fold_changes for the first dataset
prot_fc <- get_result(griss_melanoma_result, type = "fold_changes", name = "proteomics")

head(prot_fc)
```

`get_result,ReactomeAnalysisResult-method`
ReactomeAnalysisResult - get_result

Description

Retrieves a result from a `ReactomeAnalysisResult` object.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
get_result(x, type, name)
```

Arguments

x	ReactomeAnalysisResult.
type	the type of result. Use `result_types` to retrieve all available types.
name	the name of the result. Use `names` to retrieve all available results.

Value

A `data.frame` containing the respective result.

See Also

Other ReactomeAnalysisResult functions: `names,ReactomeAnalysisResult-method`, `open_reactome`, `pathways`, `plot_correlations`, `plot_volcano`, `reactome_links`, `result_types`

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the available result types
result_types(griss_melanoma_result)

# get the dataset names
names(griss_melanoma_result)

# get the fold_changes for the first dataset
prot_fc <- get_result(griss_melanoma_result, type = "fold_changes", name = "proteomics")

head(prot_fc)
```

names,ReactomeAnalysisResult-method *ReactomeAnalysisResult - names*

Description

Retrieves the names of the contained datasets within an ReactomeAnalysisResult object.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
names(x)
```

Arguments

x	ReactomeAnalysisResult.

Value

character vector with the names of the contained datasets

See Also

Other ReactomeAnalysisResult functions: get_result, open_reactome, pathways, plot_correlations, plot_volcano, reactome_links, result_types

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the names of the available datasets
names(griss_melanoma_result)
```

open_reactome *open_reactome*

Description

Opens the specified Reactome visualization in the system's default browser.

Usage

```
open_reactome(x, ...)
```

Arguments

x	ReactomeAnalysisResult.
...	Additional parameters passed to downstream functions.

Value

The opened link

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, pathways, plot_correlations, plot_volcano, reactome_links, result_types

Examples

```
# Note: This function only works with a newly created result
# since the visualization links only stay active for 7 days

# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the reactome link - this does only work
# with new results
# open_reactome(griss_melanoma_result)
```

---

open_reactome,ReactomeAnalysisResult-method

open_reactome - ReactomeAnalysisResult

Description

Opens the specified Reactome visualization in the system's default browser.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
open_reactome(x, n_visualization = 1,
  ...)
```

Arguments

x	ReactomeAnalysisResult.
n_visualization	numeric The index of the visualization to display (default 1). Use reactome_links to retrieve all available visualizations and their index. By default, the first visualization is opened.
...	Additional parameters passed to downstream functions.

Value

The opened link

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, pathways, plot_correlations, plot_volcano, reactome_links, result_types

Examples

```
# Note: This function only works with a newly created result
# since the visualization links only stay active for 7 days

# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the reactome link - this does only work
# with new results
# open_reactome(griss_melanoma_result)
```

`pathways` *pathways*

Description

Combines and returns the pathways of all analysed datasets.

Usage

```
pathways(x, ...)
```

Arguments

`x`	ReactomeAnalysisResult.
`...`	Additional parameters for specific implementations.

Value

A `data.frame` containing all merged pathways.

See Also

Other ReactomeAnalysisResult functions: `get_result`, `names,ReactomeAnalysisResult-method`, `open_reactome`, `plot_correlations`, `plot_volcano`, `reactome_links`, `result_types`

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the combined pathway result
pathway_result <- pathways(griss_melanoma_result)

head(pathway_result)
```

pathways,ReactomeAnalysisResult-method
ReactomeAnalysisResult - pathways

Description

Combines and returns the pathways of all analysed datasets.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
pathways(x, p = 0.01,
  order_by = NULL, ...)
```

Arguments

x	ReactomeAnalysisResult.
p	Minimum p-value to accept a pathway as significantly regulated. Default is 0.01.
order_by	Name of the dataset to sort the result list by. By default, the results are sorted based on the first dataset.
...	Additional parameters for specific implementations.

Value

A data.frame containing all merged pathways.

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, plot_correlations, plot_volcano, reactome_links, result_types

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the combined pathway result
pathway_result <- pathways(griss_melanoma_result)

head(pathway_result)
```

perform_reactome_analysis *Perform a Reactome Analaysis*

Description

This function wraps all steps required to perform an Analysis using the Reactome Analysis Service. It submits the passed ReactomeAnalysisRequest object to the Reactome Analysis Service API, checks the submitted analysis' status and returns the result once the analysis is complete.

Usage

```
perform_reactome_analysis(request, verbose = TRUE, reactome_url = NULL)
```

Arguments

request	ReactomeAnalysisRequest to submit.
verbose	logical. If FALSE status messages are not printed to the console.
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Value

The analysis' result

Examples

```
# create a request using Camera as an analysis
library(ReactomeGSA.data)
data(griss_melanoma_proteomics)

my_request <- ReactomeAnalysisRequest(method = "Camera")

# set maximum missing values to 0.5 and do not create any reactome visualizations
my_request <- set_parameters(request = my_request,
                             max_missing_values = 0.5,
                             create_reactome_visualization = FALSE)

# add the dataset
my_request <- add_dataset(request = my_request,
                          expression_values = griss_melanoma_proteomics,
                          name = "Proteomics",
                          type = "proteomics_int",
                          comparison_factor = "condition",
                          comparison_group_1 = "MOCK",
                          comparison_group_2 = "MCM",
                          additional_factors = c("cell.type", "patient.id"))

# perform the analysis
my_result <- perform_reactome_analysis(request = my_request, verbose = FALSE)
```

plot_correlations *plot_correlations*

Description

Plots correlations of the average fold-changes of all pathways between the different datasets. This function is only available to GSA based results (not GSVA ones).

Usage

```
plot_correlations(x)
```

Arguments

x	ReactomeAnalysisResult. The result object to use as input

Value

A list of ggplot2 plot objects representing one plot per combination

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, pathways, plot_volcano, reactome_links, result_types

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# create the correlation plots
plot_objs <- plot_correlations(griss_melanoma_result)

# only one plot created for this result as it contains two datasets
length(plot_objs)

# show the plot using `print(plot_objs[[1]])`
```

plot_correlations,ReactomeAnalysisResult-method
plot_correlations - ReactomeAnalysisResult

Description

Plots correlations of the average fold-changes of all pathways between the different datasets. This function is only available to GSA based results (not GSVA ones).

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_correlations(x)
```

Arguments

x	ReactomeAnalysisResult. The result object to use as input

Value

A list of ggplot2 plot objects representing one plot per combination

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, pathways, plot_volcano, reactome_links, result_types

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# create the correlation plots
plot_objs <- plot_correlations(griss_melanoma_result)

# only one plot created for this result as it contains two datasets
length(plot_objs)

# show the plot using `print(plot_objs[[1]])`
```

---

plot_volcano *plot_volcano*

Description

Creates a volcano plot for the pathway analysis result. Every point represents one pathway, the x-axis the log fold-change and the y-axis the adjusted p-value (-log10).

Usage

```
plot_volcano(x, ...)
```

Arguments

x	ReactomeAnalysisResult. The analysis result to plot the volcano plot for.
...	Additional parameters for specific implementations.

Details

This function is only available for GSA-based analysis results.

Value

A ggplot2 plot object representing the volcano plot.

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, pathways, plot_correlations, reactome_links, result_types

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# create the volcano plot for the first dataset
plot_obj <- plot_volcano(griss_melanoma_result)

# display the plot using `print(plot_obj)`
```

---

plot_volcano,ReactomeAnalysisResult-method

ReactomeAnalysisResult - plot_volcano

Description

Creates a volcano plot for the pathway analysis result. Every point represents one pathway, the x-axis the log fold-change and the y-axis the adjusted p-value (-log10).

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_volcano(x, dataset = 1, ...)
```

Arguments

x	ReactomeAnalysisResult. The analysis result to plot the volcano plot for.
dataset	The name or index of the dataset to plot (first one by default).
...	Additional parameters for specific implementations.

Details

This function is only available for GSA-based analysis results.

Value

A ggplot2 plot object representing the volcano plot.

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, pathways, plot_correlations, reactome_links, result_types

Examples

```
# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# create the volcano plot for the first dataset
plot_obj <- plot_volcano(griss_melanoma_result)

# display the plot using `print(plot_obj)`
```

---

print,ReactomeAnalysisRequest-method
print - ReactomeAnalysisRequest

Description

Shows a ReactomeAnalysisRequest object summary.

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest'
print(x, ...)
```

Arguments

x	ReactomeAnalysisRequest
...	Not used

Value

The classname of the object

Examples

```
library(methods)

request <- ReactomeAnalysisRequest(method = "Camera")
print(request)

# add additional parameters
request <- set_parameters(request, "max_missing_values" = 0.5)
show(request)
```

print,ReactomeAnalysisResult-method
print - ReactomeAnalysisResult

Description

Displays basic information about the ReactomeAnalysisResult object.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
print(x, ...)
```

Arguments

x	ReactomeAnalysisResult.
...	Not used

Value

character classname of the object

Examples

```
library(ReactomeGSA.data)
data(griss_melanoma_result)

print(griss_melanoma_result)
```

ReactomeAnalysisRequest
ReactomeAnalysisRequest class

Description

This class is used to collect all information required to submit an analysis request to the Reactome Analysis System.

Usage

```
ReactomeAnalysisRequest(method)
```

Arguments

method	character. Name of the method to use.

Value

A ReactomeAnalysisRequest object.

Slots

`method` character. Name of the method to use

`request_object` list. This slot should not be set manually. It stores the internal request representation and should be modified using the classes' functions. To add parameters, use `set_parameters,ReactomeAnalysisRequest-method`

Examples

```
library(ReactomeGSA.data)
library(methods)

# create the request method and specify its method
request <- ReactomeAnalysisRequest(method = "Camera")

# add a dataset to the request
data(griss_melanoma_proteomics)

request <- add_dataset(request = request,
             expression_values = griss_melanoma_proteomics,
             name = "Proteomics",
             type = "proteomics_int",
             comparison_factor = "condition",
             comparison_group_1 = "MOCK",
             comparison_group_2 = "MCM",
             additional_factors = c("cell.type", "patient.id"))

# to launch the actual analysis use the perform_reactome_analysis function
```

`ReactomeAnalysisResult-class` *ReactomeAnalysisResult class*

Description

A ReactomeAnalysisResult object contains the pathway analysis results of all submitted datasets at once.

Details

This class represents a result retrieved from the Reactome Analysis Service. It is returned by `get_reactome_analysis_result` and its wrapper `perform_reactome_analysis`. Generally, object of this class should not be created manually.

Value

A ReactomeAnalysisResult object.

Slots

`reactome_release` The Reactome version used to create this result.

`mappings` Stores the mapping results that were generated for this analysis.

`results` A named list containing the actual analysis results for every dataset and possibly combined results as well.

`reactome_links` Links pointing to reactome results as a list.

Methods

`names`: Retrieves the names of all datasets in the result object

`result_types`: Retrieves the available result types

`pathways`: Merges the pathway results of all analysed datasets.

`get_result`: Retrieve a specific result as data.frame

`reactome_links`: Displays / retrieves the URLs to the available visualizations in Reactome's pathway browser.

`open_reactome`: Opens the specified Reactome visualization in the system's default browser.

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# retrieve the names of all datasets in the result
names(griss_melanoma_result)

# get the combined pathway result
pathway_result <- pathways(griss_melanoma_result)

# check which result types are available
result_types(griss_melanoma_result)

# get the fold changes for the first dataset
first_dataset_name <- names(griss_melanoma_result)[1]

first_fc <- get_result(griss_melanoma_result, "fold_changes", first_dataset_name)
```

`reactome_links` *reactome_links*

Description

Displays detailed information about the result visualizations in Reactome.

Usage

```
reactome_links(x, ...)
```

Arguments

`x`	ReactomeAnalysisResult.
`...`	Additional parameters for specific implementations.

Value

If `return_result` is set to `TRUE`, a vector of the available visualizations.

See Also

Other ReactomeAnalysisResult functions: `get_result`, `names,ReactomeAnalysisResult-method`, `open_reactome`, `pathways`, `plot_correlations`, `plot_volcano`, `result_types`

Examples

```
# Note: This function only works with a newly created result
# since the visualization links only stay active for 7 days

# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the reactome link - this does only work
# with new results
reactome_links(griss_melanoma_result)
```

`reactome_links,ReactomeAnalysisResult-method`
ReactomeAnalysisResult - reactome_links

Description

Displays detailed information about the result visualizations in Reactome.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
reactome_links(x, print_result = TRUE,
  return_result = FALSE)
```

Arguments

`x`	ReactomeAnalysisResult.
`print_result`	If set to `FALSE` the links are not printed to the console.
`return_result`	If `TRUE` the available visualizations are returned as a list containing named vectors for every visualization. These vectors' have a `url`, `name`, and optionally a `description` slot.

Value

If `return_result` is set to `TRUE`, a vector of the available visualizations.

See Also

Other ReactomeAnalysisResult functions: `get_result`, `names,ReactomeAnalysisResult-method`, `open_reactome`, `pathways`, `plot_correlations`, `plot_volcano`, `result_types`

Examples

```
# Note: This function only works with a newly created result
# since the visualization links only stay active for 7 days

# load an example result
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the reactome link - this does only work
# with new results
reactome_links(griss_melanoma_result)
```

`remove_dataset` *remove_dataset*

Description

Remove the dataset from the `ReactomeAnalysisRequest` object.

Usage

```
remove_dataset(x, dataset_name)
```

Arguments

`x`	The `ReactomeAnalysisRequest` to remove the dataset from
`dataset_name`	character The dataset's name

Value

The updated `ReactomeAnalysisRequest`

`remove_dataset,ReactomeAnalysisRequest-method` *remove_dataset - ReactomeAnalysisRequest*

Description

Remove the dataset from the `ReactomeAnalysisRequest` object.

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest'
remove_dataset(x, dataset_name)
```

Arguments

`x`	The `ReactomeAnalysisRequest` to remove the dataset from
`dataset_name`	character The dataset's name

Value

The updated `ReactomeAnalysisRequest`

`result_types` *result_types*

Description

Retrieves the available result types for the `ReactomeAnalysisResult` object. Currently, the Reactome Analysis System supports `pathways` and gene level `fold_changes` as result types. Not all analysis methods return both data types though. Use the `names` function to find out which datasets are available in the result object.

Usage

```
result_types(x)
```

Arguments

`x`	ReactomeAnalysisResult.

Value

A cacharacter vector of result types.

See Also

Other ReactomeAnalysisResult functions: `get_result`, `names,ReactomeAnalysisResult-method`, `open_reactome`, `pathways`, `plot_correlations`, `plot_volcano`, `reactome_links`

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the available result types
result_types(griss_melanoma_result)
```

`result_types,ReactomeAnalysisResult-method`
ReactomeAnalysisResult - result_types

Description

Retrieves the available result types for the `ReactomeAnalysisResult` object. Currently, the Reactome Analysis System supports `pathways` and gene level `fold_changes` as result types. Not all analysis methods return both data types though. Use the `names` function to find out which datasets are available in the result object.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
result_types(x)
```

Arguments

x	ReactomeAnalysisResult.

Value

A cacharacter vector of result types.

See Also

Other ReactomeAnalysisResult functions: get_result, names,ReactomeAnalysisResult-method, open_reactome, pathways, plot_correlations, plot_volcano, reactome_links

Examples

```
# load an example result object
library(ReactomeGSA.data)
data(griss_melanoma_result)

# get the available result types
result_types(griss_melanoma_result)
```

---

set_method *set_method*

Description

Set the analysis method used by the ReactomeAnalysisRequest

Usage

```
set_method(request, method, ...)
```

Arguments

request	The ReactomeAnalysisRequest to adjust
method	The name of the method to use. Use get_reactome_methods to retrieve all available methods
...	Additional parameters passed to specific implementations

Value

The ReactomeAnalysisRequest with the adapted method

Examples

```
# create a request using Camera as an analysis
data(griss_melanoma_proteomics)
library(methods)

my_request <- ReactomeAnalysisRequest(method = "Camera")

print(my_request)
```

```
# change the method to ssGSEA
my_request <- set_method(my_request, "ssGSEA")

print(my_request)
```

`set_method,ReactomeAnalysisRequest-method`
set_method - ReactomeAnalysisRequest

Description

Set the analysis method used by the `ReactomeAnalysisRequest`

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest'
set_method(request, method, ...)
```

Arguments

`request`	The `ReactomeAnalysisRequest` to adjust
`method`	The name of the method to use. Use `get_reactome_methods` to retrieve all available methods
`...`	Additional parameters passed to specific implementations

Value

The `ReactomeAnalysisRequest` with the adapted method

Examples

```
# create a request using Camera as an analysis
data(griss_melanoma_proteomics)
library(methods)

my_request <- ReactomeAnalysisRequest(method = "Camera")

print(my_request)

# change the method to ssGSEA
my_request <- set_method(my_request, "ssGSEA")

print(my_request)
```

set_parameters *set_parameters*

Description

Sets the analysis parameters for the given ReactomeAnalysisRequest. If the parameter is already set, it is overwritten. Use get_reactome_methods to get a list of all available parameters for each available method.

Usage

```
set_parameters(request, ...)
```

Arguments

request	The ReactomeAnalysisRequest to set the parameters for.
...	Any name / value pair to set a parameter (see example). For a complete list of available parameters use get_reactome_methods

Details

Both, parameters with the scope "dataset" as well as "analysis" can be set on the analysis level. In this case, these parameters overwrite the system's default values. If a parameter with the scope "dataset" is defined again at the dataset level, this value will overwrite the analysis' scope value for the given dataset.

Value

The modified ReactomeAnalysisRequest object

Examples

```
library(methods)

# create a request object
request <- ReactomeAnalysisRequest(method = "Camera")

# add a parameter
request <- set_parameters(request, max_missing_values = 0.5, discrete_norm_function = "TMM")
```

---

set_parameters,ReactomeAnalysisRequest-method
ReactomeAnalysisRequest - set_parameters

Description

Sets the analysis parameters for the given ReactomeAnalysisRequest. If the parameter is already set, it is overwritten. Use get_reactome_methods to get a list of all available parameters for each available method.

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest'
set_parameters(request, ...)
```

Arguments

request	The ReactomeAnalysisRequest to set the parameters for.
...	Any name / value pair to set a parameter (see example). For a complete list of available parameters use get_reactome_methods

Details

Both, parameters with the scope "dataset" as well as "analysis" can be set on the analysis level. In this case, these parameters overwrite the system's default values. If a parameter with the scope "dataset" is defined again at the dataset level, this value will overwrite the analysis' scope value for the given dataset.

Value

The modified ReactomeAnalysisRequest object

Examples

```
library(methods)

# create a request object
request <- ReactomeAnalysisRequest(method = "Camera")

# add a parameter
request <- set_parameters(request, max_missing_values = 0.5, discrete_norm_function = "TMM")
```

---

show,ReactomeAnalysisRequest-method

print - ReactomeAnalysisRequest

Description

Shows a ReactomeAnalysisRequest object summary.

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest'
show(object)
```

Arguments

object	ReactomeAnalysisRequest

Value

The classname of the object

Examples

```
library(methods)

request <- ReactomeAnalysisRequest(method = "Camera")
print(request)

# add additional parameters
request <- set_parameters(request, "max_missing_values" = 0.5)
show(request)
```

---

show,ReactomeAnalysisResult-method

show - ReactomeAnalysisResult

Description

Displays basic information about the ReactomeAnalysisResult object.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
show(object)
```

Arguments

object	ReactomeAnalysisResult.

Value

character classname of the object

Examples

```
library(ReactomeGSA.data)
data(griss_melanoma_result)

show(griss_melanoma_result)
```

---

start_reactome_analysis

Start Reactome Analysis

Description

Submits a ReactomeAnalysisRequest to the Reactome Analysis Service API and returns the analysis id of the submitted job.

Usage

```
start_reactome_analysis(request, reactome_url = NULL)
```

Arguments

request	ReactomeAnalysisRequest object to submit.
reactome_url	URL of the Reactome API Server. Overwrites the URL set in the 'reactome_gsa.url' option. Specific ports can be set using the standard URL specification (for example http://your.service:1234)

Value

character The analysis job's id.

Index