Package 'ReactomeGSA'

March 19, 2025

Type Package

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

Version 1.21.2

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

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

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

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

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

Imports Biobase, BiocSingular, dplyr, ggplot2, gplots, httr, igraph, jsonlite, methods, progress, RColorBrewer, SummarizedExperiment, tidyr

Suggests devtools, knitr, ReactomeGSA.data, rmarkdown, scater, scran, scRNAseq, scuttle, Seurat (>= 3.0), SingleCellExperiment, testthat

VignetteBuilder knitr

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

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

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

git_branch devel

git_last_commit 1612e12

git_last_commit_date 2024-11-27

Repository Bioconductor 3.21

Date/Publication 2025-03-18

Contents

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,ReactomeAnalys
add_dataset,ReactomeAnalysisRequest,ExpressionSet-method, add_dataset,ReactomeAnalysisRequest,data.f
add_dataset,ReactomeAnalysisRequest,matrix-method

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(), add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAnalysisRequest,EList-method, add_dataset,ReactomeAnalysisRequest,ExpressionSet-m add_dataset,ReactomeAnalysisRequest,matrix-method

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(), add_dataset,ReactomeAnalysisRequest,EList-method, add_dataset,ReactomeAnalysisRequest,ExpressionSet-method, add_dataset,ReactomeAnalysisRequest,data.f add_dataset,ReactomeAnalysisRequest,matrix-method

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(), add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAnalysisRequest,ExpressionSet-method, add_dataset,ReactomeAnalysisRequest,data.f add_dataset,ReactomeAnalysisRequest,matrix-method

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(), add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAnalysisRequest,EList-method, add_dataset,ReactomeAnalysisRequest,data.frame-meth
add_dataset,ReactomeAnalysisRequest,matrix-method

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,matrix-method`
add_dataset - matrix

Description

Adds a dataset to the analysis request

Usage

```
## S4 method for signature 'ReactomeAnalysisRequest,matrix'
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`	matrix. 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(), add_dataset,ReactomeAnalysisRequest,DGEList-method, add_dataset,ReactomeAnalysisRequest,EList-method, add_dataset,ReactomeAnalysisRequest,ExpressionSet-m add_dataset,ReactomeAnalysisRequest,data.frame-method

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"))
```

---

analyse_sc_clusters *analyse_sc_clusters*

Description

Analyses cell clusters of a single-cell RNA-sequencing experiment to get pathway-level expressions for every cluster of cells.

Usage

```
analyse_sc_clusters(
  object,
  use_interactors = TRUE,
  include_disease_pathways = FALSE,
  create_reactome_visualization = FALSE,
  create_reports = FALSE,
  report_email = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

object	The object containing the single-cell RNA-sequencing data.
use_interactors	If set (default), protein-protein interactors from IntAct are used to extend Reactome pathways.
include_disease_pathways	If set, disease pathways are included as well. Disease pathways in Reactome follow a different annotation approach and can therefore lead to inaccurate results.
create_reactome_visualization	If set, the interactive visualization in Reactome's PathwayBrowser is created.
create_reports	If set, PDF and Microsoft Excel reports are created. Links to these report files are send to the supplied e-mail address.
report_email	The e-mail address to which reports should be sent to.
verbose	If set, additional status messages are printed.
...	Parameters passed to the specific implementation. Detailed documentations can be found there.

Details

There are currently two specific implementations of this function, one to support `Seurat` objects and one to support Bioconductor's `SingleCellExperiment` class.

Value

A `ReactomeAnalysisResult` object.

Examples

```
# This example shows how a Seurat object can be analysed
# the approach is identical for SingleCellExperiment objects
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

analyse_sc_clusters,Seurat-method
analyse_sc_clusters - Seurat

Description

Analyses cell clusters of a single-cell RNA-sequencing experiment to get pathway-level expressions for every cluster of cells.

Usage

```
## S4 method for signature 'Seurat'
analyse_sc_clusters(
  object,
  use_interactors = TRUE,
  include_disease_pathways = FALSE,
  create_reactome_visualization = FALSE,
  create_reports = FALSE,
  report_email = NULL,
  verbose = FALSE,
  assay = "RNA",
  slot = "counts",
  ...
)
```

Arguments

object	The Seurat object containing the single cell RNA-sequencing data.
use_interactors	If set (default), protein-protein interactors from IntAct are used to extend Reactome pathways.
include_disease_pathways	If set, disease pathways are included as well. Disease pathways in Reactome follow a different annotation approach and can therefore lead to inaccurate results.
create_reactome_visualization	If set, the interactive visualization in Reactome's PathwayBrowser is created.
create_reports	If set, PDF and Microsoft Excel reports are created. Links to these report files are send to the supplied e-mail address.
report_email	The e-mail address to which reports should be sent to.
verbose	If set, additional status messages are printed.
assay	By default, the "RNA" assay is used, which contains the original read counts.
slot	The slot in the Seurat object to use. Default and recommended approach is to use the raw counts.
...	Parameters passed to the specific implementation. Detailed documentations can be found there.

Details

There are currently two specific implementations of this function, one to support `Seurat` objects and one to support Bioconductor's `SingleCellExperiment` class.

Value

A `ReactomeAnalysisResult` object.

Examples

```
# This example shows how a Seurat object can be analysed
# the approach is identical for SingleCellExperiment objects
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

`analyse_sc_clusters,SingleCellExperiment-method`
analyse_sc_clusters - SingleCellExperiment

Description

Analyses cell clusters of a single-cell RNA-sequencing experiment to get pathway-level expressions for every cluster of cells.

Usage

```
## S4 method for signature 'SingleCellExperiment'
analyse_sc_clusters(
  object,
  use_interactors = TRUE,
  include_disease_pathways = FALSE,
  create_reactome_visualization = FALSE,
  create_reports = FALSE,
  report_email = NULL,
  verbose = FALSE,
  cell_ids,
  ...
)
```

Arguments

`object`	The `SingleCellExperiment` object containing the single cell RNA-sequencing data.

use_interactors
: If set (default), protein-protein interactors from IntAct are used to extend Reactome pathways.

include_disease_pathways
: If set, disease pathways are included as well. Disease pathways in Reactome follow a different annotation approach and can therefore lead to inaccurate results.

create_reactome_visualization
: If set, the interactive visualization in Reactome's PathwayBrowser is created.

create_reports
: If set, PDF and Microsoft Excel reports are created. Links to these report files are send to the supplied e-mail address.

report_email
: The e-mail address to which reports should be sent to.

verbose
: If set, additional status messages are printed.

cell_ids
: A factor specifying the group to which each cell belongs. For example, `object$cluster`. Alternatively, a string specifying the metada field's name may be passed.

...
: Parameters passed to scater's `aggregateAcrossCells` function.

Details

There are currently two specific implementations of this function, one to support `Seurat` objects and one to support Bioconductor's `SingleCellExperiment` class.

Value

A `ReactomeAnalysisResult` object.

Examples

```
# This example shows how a Seurat object can be analysed
# the approach is identical for SingleCellExperiment objects
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

---

break_names *break_names*

Description

Introduce a line break in the middle of a long name.

Usage

```
break_names(the_names, long_name_limit = 46)
```

Arguments

the_names	A vector of names
long_name_limit	The limit to define a long name (defautl 46 chars.)

Value

The list of adapted names

---

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

fetch_public_data *fetch_public_data*

Description

Loads an already available public dataset from ReactomeGSA and returns it as a Biobase::ExpressionSet object.

Usage

```
fetch_public_data(dataset_entry, reactome_url)
```

Arguments

dataset_entry	The entry of the respective dataset as returned by the find_public_datasets function.
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 loaded data as an ExpressionSet object.

find_public_datasets *find_public_datasets*

Description

Search for a public dataset in the resources supported by ReactomeGSA as external data sources.

Usage

```
find_public_datasets(
  search_term,
  species = "Homo sapiens",
  reactome_url = NULL
)
```

Arguments

search_term	The search terms as a single string. Multiple words (seperated by a space) are combined by an "AND".
species	Limit the search to selected species. The complete list of available species can be retrieved through get_public_species. By default, entries as limited to human datasets.
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 a list of datasets found through the search.

Examples

```
# search for any public dataset relating to BRAF in melanoma
melanoma_datasets <- find_public_datasets("melanoma braf")

# it is also possible to limit this to another species than human
melanoma_mouse <- find_public_datasets("melanoma", species = "Mus musculus")

# the list of available species can be retrieved using get_public_species
all_species <- get_public_species()

# datasets can then be loaded using the load_public_dataset function
```

---

generate_metadata *generate_metadata*

Description

The pseudobulk data is generated using the generate_pseudo_bulk_data function.

Usage

```
generate_metadata(pseudo_bulk_data)
```

Arguments

pseudo_bulk_data	Pseudobulk data generated from the generate_pseudo_bulk_data function

Value

Metadata table for later use

See Also

generate_pseudo_bulk_data for generating pseudobulk data.

Examples

```
# Example pseudobulk data
pseudo_bulk_data <- data.frame(
  sample1_groupA = c(10, 20, 30),
  sample2_groupA = c(15, 25, 35),
  sample3_groupB = c(5, 10, 15)
)

# Generate metadata from pseudobulk data
metadata <- generate_metadata(pseudo_bulk_data)
```

---

generate_metadata,data.frame-method
Generate metadata

Description

Generate metadata

Usage

```
## S4 method for signature 'data.frame'
generate_metadata(pseudo_bulk_data)
```

Arguments

pseudo_bulk_data
: Pseudobulk data generated from the generate_pseudo_bulk_data function

Value

Returns metadata table for later use

generate_pseudo_bulk_data *generate_pseudo_bulk_data*

Description

generate_pseudo_bulk_data

Usage

```
generate_pseudo_bulk_data(
  object,
  group_by = NULL,
  split_by = "random",
  k_variable = 4
)
```

Arguments

object	The Seurat or SingleCellExperiment object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
split_by	variable -> split by a variable within the metadata; k must be a string random -> splits based on a random number; k must be a number Louvain, Louvain_multilevel, SLM, Leiden -> subclusters k must be a list with [resolution, cluster_1, cluster_2]
k_variable	variable dependent on the split_by

Value

returns pseudo bulk generated data

Examples

```
#using SCE object
library("scRNAseq")
SCE_OBJECT <- ZeiselBrainData()
# generating pseudo bulk data using the SCE object above,
# and clustering level level1class from the metadata

# generate pseudo bulk data based on random subsampling
SCE_RESULT_RANDOM <- generate_pseudo_bulk_data(SCE_OBJECT,
                                               group_by = "level1class",
                                               split_by = "random",
                                               k_variable = 5)

# generate pseudo bulk data based on variable within the metadata
```

```
SCE_RESULT_VARIABLE <- generate_pseudo_bulk_data(SCE_OBJECT, "level1class","variable","tissue")
```

---

generate_pseudo_bulk_data,Seurat-method
generate_pseudo_bulk_data - Seurat

Description

Generate Pseudo Bulk Data for Seurat Objects

Usage

```
## S4 method for signature 'Seurat'
generate_pseudo_bulk_data(
  object,
  group_by = NULL,
  split_by = "random",
  k_variable = 4
)
```

Arguments

object	The object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
split_by	variable -> split by a variable within the metadata; k must be a string random -> splits based on a random number; k must be a number Louvain, Louvain_multilevel, SLM, Leiden -> subclusters k must be a list with [resolution, cluster_1, cluster_2]
k_variable	variable dependent on the split_by

Value

returns pseudo bulk generated data

generate_pseudo_bulk_data,SingleCellExperiment-method

generate_pseudo_bulk_data - SingleCellExperiment

Description

generate_pseudo_bulk_data - SingleCellExperiment

Usage

```
## S4 method for signature 'SingleCellExperiment'
generate_pseudo_bulk_data(
  object,
  group_by = NULL,
  split_by = "random",
  k_variable = 4
)
```

Arguments

object	The SingleCellExperiment object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
split_by	variable -> split by a variable within the metadata; k must be a string random -> splits based on a random number; k must be a number subclustering [resolution, cluster_1, cluster_2]
k_variable	variable dependent on the split_by

Value

returns pseudo bulk generated data

get_dataset_loading_status

Retrieves the status of the submitted dataset loading request

Description

Retrieves the status of the submitted dataset loading request

Usage

```
get_dataset_loading_status(loading_id, reactome_url = NULL)
```

Arguments

loading_id	The dataset loading process' 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_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_public_species` *get_public_species*

Description

Return the list of found species labels in the supported public data resources

Usage

```
get_public_species(reactome_url = NULL)
```

Arguments

`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 vector of species strings.

Examples

```
# get the available species
available_species <- get_public_species()

# inspect the first 1 - 3 entries
available_species[1:3]
```

`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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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)
```

`is_gsva_result` *is_gsva_result*

Description

is_gsva_result

Usage

```
is_gsva_result(object)
```

Arguments

`object`	A `ReactomeAnalysisResult` object

Value

Boolean indicating whether the object is a GSVA result.

load_public_dataset *load_public_dataset*

Description

Loads a public dataset that was found through the find_public_datasets function. The dataset is returned as a Biobase ExpressionSet object.

Usage

```
load_public_dataset(dataset_entry, verbose = FALSE, reactome_url = NULL)
```

Arguments

dataset_entry	The entry of the respective dataset as returned by the find_public_datasets function.
verbose	If set to TRUE, status messages and a status bar are displayed.
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 loaded data as an ExpressionSet object.

Examples

```
# As a first step, you need to find available datasets
available_datasets <- find_public_datasets("psoriasis tnf")

# have a quick look at the found datasets
available_datasets[, c("id", "title")]

# load the first one, use the whole row of the found datasets
# data.frame as the parameter
dataset_1 <- load_public_dataset(available_datasets[1,], verbose = TRUE)
```

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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap()`, `plot_gsva_pathway()`, `plot_heatmap()`, `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,
  compress = TRUE,
  reactome_url = NULL
)
```

Arguments

`request`	`ReactomeAnalysisRequest` to submit.
`verbose`	logical. If `FALSE` status messages are not printed to the console.
`compress`	logical. If `TRUE` (default) the request data is compressed before submitting it to the ReactomeGSA API. This is the generally recommended way and should only be disabled for debugging purposes.
`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, hide_non_sig = FALSE)
```

Arguments

x	ReactomeAnalysisResult. The result object to use as input
hide_non_sig	If set, non-significant pathways are not shown.

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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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, hide_non_sig = FALSE)
```

Arguments

x	ReactomeAnalysisResult. The result object to use as input
hide_non_sig	If set, non-significant pathways are not shown.

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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap` *plot_gsva_heatmap*

Description

Plots pathway expression values / sample as a heatmap. Ranks pathways based on their expression difference.

Usage

```
plot_gsva_heatmap(
  object,
  pathway_ids = NULL,
  max_pathways = 20,
  truncate_names = TRUE,
  ...
)
```

Arguments

`object`	The `ReactomeAnalysisResult` object.
`pathway_ids`	A vector of pathway ids. If set, only these pathways are included in the plot.
`max_pathways`	The maximum number of pathways to include. Only takes effect if `pathway_ids` is not set.
`truncate_names`	If set, long pathway names are truncated.
`...`	Additional parameters passed to specific implementations.

Value

None

See Also

Other ReactomeAnalysisResult functions: get_result(), names,ReactomeAnalysisResult-method, open_reactome(), pathways(), plot_correlations(), plot_gsva_pathway(), plot_heatmap(), plot_volcano(), reactome_links(), result_types()

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)

# plot the heatmap
relevant_pathways <- c("R-HSA-983170", "R-HSA-388841", "R-HSA-2132295",
                       "R-HSA-983705", "R-HSA-5690714")
plot_gsva_heatmap(gsva_result,
                  pathway_ids = relevant_pathways, # limit to these pathways
                  margins = c(6,30), # adapt the figure margins in heatmap.2
                  dendrogram = "col", # only plot column dendrogram
                  scale = "row", # scale for each pathway
                  key = FALSE, # don't display the color key
                  lwid=c(0.1,4)) # remove the white space on the left
```

---

plot_gsva_heatmap,ReactomeAnalysisResult-method

plot_gsva_heatmap - ReactomeAnalysisResult function

Description

Plots pathway expression values / sample as a heatmap. Ranks pathways based on their expression difference.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_gsva_heatmap(
  object,
  pathway_ids = NULL,
  max_pathways = 20,
  truncate_names = TRUE,
  ...
)
```

Arguments

object	The ReactomeAnalysisResult object.
pathway_ids	A vector of pathway ids. If set, only these pathways are included in the plot.
max_pathways	The maximum number of pathways to include. Only takes effect if pathway_ids is not set.
truncate_names	If set, long pathway names are truncated.
...	Additional parameters passed to the heatmap.2 function.

Value

None

See Also

Other ReactomeAnalysisResult functions: get_result(), names,ReactomeAnalysisResult-method, open_reactome(), pathways(), plot_correlations(), plot_gsva_pathway(), plot_heatmap(), plot_volcano(), reactome_links(), result_types()

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)

# plot the heatmap
relevant_pathways <- c("R-HSA-983170", "R-HSA-388841", "R-HSA-2132295",
                       "R-HSA-983705", "R-HSA-5690714")
plot_gsva_heatmap(gsva_result,
                  pathway_ids = relevant_pathways, # limit to these pathways
                  margins = c(6,30), # adapt the figure margins in heatmap.2
                  dendrogram = "col", # only plot column dendrogram
                  scale = "row", # scale for each pathway
                  key = FALSE, # don't display the color key
                  lwid=c(0.1,4)) # remove the white space on the left
```

---

plot_gsva_pathway *plot_gsva_pathway*

Description

Plots the expression of a specific pathway from a ssGSEA result.

Usage

```
plot_gsva_pathway(object, pathway_id, ...)
```

Arguments

object	The ReactomeAnalysisResult object.
pathway_id	The pathway's id
...	Additional parameters for specific implementations.

Value

A ggplot2 plot object

See Also

Other ReactomeAnalysisResult functions: get_result(), names,ReactomeAnalysisResult-method, open_reactome(), pathways(), plot_correlations(), plot_gsva_heatmap(), plot_heatmap(), plot_volcano(), reactome_links(), result_types()

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)

# create the plot
plot_obj <- plot_gsva_pathway(gsva_result, "R-HSA-389542")
```

---

plot_gsva_pathway,ReactomeAnalysisResult-method

ReactomeAnalysisResult - plot_gsva_pathway

Description

Plots the expression of a specific pathway from a ssGSEA result.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_gsva_pathway(object, pathway_id, ...)
```

Arguments

object	The ReactomeAnalysisResult object.
pathway_id	The pathway's id
...	Additional parameters for specific implementations.

Value

A ggplot2 plot object

See Also

Other ReactomeAnalysisResult functions: `get_result()`, `names,ReactomeAnalysisResult-method`, `open_reactome()`, `pathways()`, `plot_correlations()`, `plot_gsva_heatmap()`, `plot_heatmap()`, `plot_volcano()`, `reactome_links()`, `result_types()`

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)

# create the plot
plot_obj <- plot_gsva_pathway(gsva_result, "R-HSA-389542")
```

---

plot_gsva_pca *plot_gsva_pca*

Description

Runs a Principal Component analysis (using `prcomp`) on the samples based on the pathway analysis results.

Usage

```
plot_gsva_pca(object, pathway_ids = NULL, ...)
```

Arguments

`object`	A `ReactomeAnalysisResult` object containing a ssGSEA result
`pathway_ids`	A character vector of pathway ids. If set, only these pathways will be used for the PCA analysis.
`...`	Additional paramters passed to specific implementations.

Value

A ggplot2 object representing the plot.

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

`plot_gsva_pca,ReactomeAnalysisResult-method`

plot_gsva_pca - ReactomeAnalysisResult

Description

Runs a Principal Component analysis (using `prcomp`) on the samples based on the pathway analysis results.

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_gsva_pca(object, pathway_ids = NULL, ...)
```

Arguments

`object`	A `ReactomeAnalysisResult` object containing a ssGSEA result
`pathway_ids`	A character vector of pathway ids. If set, only these pathways will be used for the PCA analysis.
`...`	Additional parameters are passed to `prcomp`

Value

A ggplot2 object representing the plot.

Examples

```
# load the scRNA-seq example data
library(ReactomeGSA.data)
data(jerby_b_cells)

# perform the GSVA analysis
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

plot_heatmap *plot_heatmap*

Description

Creates a heatmap to show which pathways are up- and down-regulated in different datasets

Usage

```
plot_heatmap(
  x,
  fdr = 0.01,
  max_pathways = 30,
  break_long_names = TRUE,
  return_data = FALSE
)
```

Arguments

x	ReactomeAnalysisResult. The result object to use as input
fdr	numeric. The minimum FDR to consider a pathways as significantly regulated. (Default 0.01)
max_pathways	numeric. The maximum number of pathways to plot. Pathways are sorted based on in how many datasets they are significantly regulated. This has no effect if `return_data` is set to `TRUE`.
break_long_names	logical. If set, long pathway names are broken into two lines.
return_data	logical. If set, only the plotting data, but not the plot object itself is returned. This can be used to create customized plots that use the same data structure.

Value

A ggplot2 plot object representing the heatmap of pathways

See Also

Other ReactomeAnalysisResult functions: `get_result()`, `names,ReactomeAnalysisResult-method`, `open_reactome()`, `pathways()`, `plot_correlations()`, `plot_gsva_heatmap()`, `plot_gsva_pathway()`, `plot_volcano()`, `reactome_links()`, `result_types()`

Examples

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

# create the heatmap plot
```

```
plot_obj <- plot_heatmap(griss_melanoma_result)

# show the plot
print(plot_obj)
```

`plot_heatmap,ReactomeAnalysisResult-method`
plot_heatmap - ReactomeAnalysisResult

Description

Creates a heatmap to show which pathways are up- and down-regulated in different datasets

Usage

```
## S4 method for signature 'ReactomeAnalysisResult'
plot_heatmap(
  x,
  fdr = 0.01,
  max_pathways = 30,
  break_long_names = TRUE,
  return_data = FALSE
)
```

Arguments

`x`	ReactomeAnalysisResult. The result object to use as input
`fdr`	numeric. The minimum FDR to consider a pathways as significantly regulated. (Default 0.01)
`max_pathways`	numeric. The maximum number of pathways to plot. Pathways are sorted based on in how many datasets they are significantly regulated. This has no effect if `return_data` is set to `TRUE`.
`break_long_names`	logical. If set, long pathway names are broken into two lines.
`return_data`	logical. If set, only the plotting data, but not the plot object itself is returned. This can be used to create customized plots that use the same data structure.

Value

A ggplot2 plot object representing the heatmap of pathways

See Also

Other ReactomeAnalysisResult functions: `get_result()`, `names,ReactomeAnalysisResult-method`, `open_reactome()`, `pathways()`, `plot_correlations()`, `plot_gsva_heatmap()`, `plot_gsva_pathway()`, `plot_volcano()`, `reactome_links()`, `result_types()`

Examples

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

# create the heatmap plot
plot_obj <- plot_heatmap(griss_melanoma_result)

# show the plot
print(plot_obj)
```

---

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(), plot_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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()`, `plot_gsva_heatmap()`, `plot_gsva_pathway()`, `plot_heatmap()`, `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)

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_gsva_heatmap()`, `plot_gsva_pathway()`, `plot_heatmap()`, `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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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_gsva_heatmap(), plot_gsva_pathway(), plot_heatmap(), 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)
```

---

split_clustering *method implementation subclustering*

Description

method implementation subclustering

Usage

```
split_clustering(seurat_object, group_by, res, alg, cluster1, cluster2)
```

Arguments

seurat_object	The Seurat object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
res	The clustering resolution to use.
alg	Seurat subclustering algorithm id
cluster1	cluster to subcluster
cluster2	cluster to subcluster

Value

returns pseudo bulk generated data

`split_random_sce` *split SCE Object with random pooling*

Description

split SCE Object with random pooling

Usage

```
split_random_sce(sce_object, group_by, k_variable)
```

Arguments

`sce_object`	The SingleCellExperiment object to analyse.
`group_by`	entry in metadata table, based on these cluster annotation pseudo bulk is performed
`k_variable`	number of pools that should be created

Value

returns pseudo bulk generated data

`split_subclustering_sce` *split SCE Object with random pooling*

Description

split SCE Object with random pooling

Usage

```
split_subclustering_sce(
  sce_object,
  group_by,
  resolution,
  subcluster_ref,
  subcluster_comp
)
```

Arguments

sce_object	The SingleCellExperiment object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
resolution	resolution
subcluster_ref	cluster to subcluster as areference
subcluster_comp	cluster to subcluster for comparison

Value

returns pseudo bulk generated data

---

split_variable *split Seurat object by variable*

Description

split Seurat object by variable

Usage

```
split_variable(seurat_object, group_by, k_variable)
```

Arguments

seurat_object	The Seurat object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
k_variable	variable dependent on the split_by -> meta data entry

Value

returns pseudo bulk generated data

split_variable_random *split Seurat object by random pooling*

Description

split Seurat object by random pooling

Usage

```
split_variable_random(seurat_object, group_by, k_variable)
```

Arguments

seurat_object	The Seurat object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
k_variable	number of random pools

Value

returns pseudo bulk generated data

split_variable_sce *split SCE Object by variable*

Description

split SCE Object by variable

Usage

```
split_variable_sce(sce_object, group_by, k_variable)
```

Arguments

sce_object	The SingleCellExperiment object to analyse.
group_by	entry in metadata table, based on these cluster annotation pseudo bulk is performed
k_variable	variable for sub setting must be in the metadata

Value

returns pseudo bulk generated data

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, compress = TRUE, reactome_url = NULL)
```

Arguments

request	ReactomeAnalysisRequest object to submit.
compress	If set (default) the JSON request data is compressed using gzip.
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

This function should only be used for very large requests that likely take a long time to complete. By default, users should use the perform_reactome_analysis function to run an analysis.

Value

character The analysis job's id.

#' @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")) # start the analysis analysis_id <- start_reactome_analysis(my_request)

wait_for_loading_dataset *wait_for_loading_dataset*

Description

This function loops until the dataset is available. If verbose is set to TRUE, the progress is displayed in a status bar.

Usage

```
wait_for_loading_dataset(request, verbose, reactome_url)
```

Arguments

request	The httr request object of the dataset loading request.
verbose	If set to TRUE, the progress is displayed as a status bar.
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)

Index