

# Package ‘ReactomeGSA’

October 14, 2021

**Type** Package

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

**Version** 1.6.1

## 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 perform 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, gplots, RColorBrewer

**RoxygenNote** 7.1.1

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

**Enhances** limma, edgeR, Seurat (>= 3.0), scater

**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\_13

**git\_last\_commit** 0fa14a1

**git\_last\_commit\_date** 2021-09-10

**Date/Publication** 2021-10-14

**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	3
add_dataset,ReactomeAnalysisRequest,data.frame-method	5
add_dataset,ReactomeAnalysisRequest,DGEList-method	6
add_dataset,ReactomeAnalysisRequest,EList-method	8
add_dataset,ReactomeAnalysisRequest,ExpressionSet-method	10
add_dataset,ReactomeAnalysisRequest,matrix-method	12
analyse_sc_clusters	13
analyse_sc_clusters,Seurat-method	15
analyse_sc_clusters,SingleCellExperiment-method	16
checkRequestValidity	17
check_reactome_url	18
convert_reactome_result	18
data_frame_as_string	19
get_fc_for_dataset	19
get_is_sig_dataset	20
get_reactome_analysis_result	20
get_reactome_analysis_status	21
get_reactome_data_types	21
get_reactome_methods	22
get_result	24
get_result,ReactomeAnalysisResult-method	25
is_gsva_result	26
names,ReactomeAnalysisResult-method	26
open_reactome	27
open_reactome,ReactomeAnalysisResult-method	28
pathways	29
pathways,ReactomeAnalysisResult-method	29
perform_reactome_analysis	30
plot_correlations	32
plot_correlations,ReactomeAnalysisResult-method	33
plot_gsva_heatmap	34
plot_gsva_heatmap,ReactomeAnalysisResult-method	35
plot_gsva_pathway	36
plot_gsva_pathway,ReactomeAnalysisResult-method	37
plot_gsva_pca	38
plot_gsva_pca,ReactomeAnalysisResult-method	39
plot_volcano	39
plot_volcano,ReactomeAnalysisResult-method	40
print,ReactomeAnalysisRequest-method	41
print,ReactomeAnalysisResult-method	42
ReactomeAnalysisRequest	43

ReactomeAnalysisResult-class . . . . . 44

reactome\_links . . . . . 45

reactome\_links,ReactomeAnalysisResult-method . . . . . 46

remove\_dataset . . . . . 47

remove\_dataset,ReactomeAnalysisRequest-method . . . . . 47

result\_types . . . . . 48

result\_types,ReactomeAnalysisResult-method . . . . . 49

set\_method . . . . . 50

set\_method,ReactomeAnalysisRequest-method . . . . . 50

set\_parameters . . . . . 51

set\_parameters,ReactomeAnalysisRequest-method . . . . . 52

show,ReactomeAnalysisRequest-method . . . . . 53

show,ReactomeAnalysisResult-method . . . . . 54

start\_reactome\_analysis . . . . . 54

**Index** **56**

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.



```
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 <a href="#">get_reactome_data_types</a>
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.

<code>sample_data</code>	data.frame (optional) data.frame containing the sample metadata of the <code>expression_values</code> . Depending on the object type of <code>expression_values</code> , this information can also be extracted from there.
<code>additional_factors</code>	vector. Vector of additional sample properties that are used as blocking factors (if supported by the chosen analysis method) in the gene set analysis.
<code>overwrite</code>	boolean. If set to TRUE, datasets with the same name will be overwritten
<code>...</code>	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,ReactomeAnalysisRequest,ExpressionSet-method](#), [add\\_dataset,ReactomeAnalysisRequest,matrix](#), [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 <a href="#">get_reactome_data_types</a>
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,ReactomeAnalysisRequest,data.frame-method](#), [add\\_dataset,ReactomeAnalysisRequest,matrix-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 automatically 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 <a href="#">get_reactome_data_types</a>
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, ReactomeAnalysisRequest, data.frame-method](#), [add\\_dataset, ReactomeAnalysisRequest, matrix-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 <a href="#">get_reactome_data_types</a>
comparison_factor	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from <code>expression_values</code> or from <code>sample_data</code> .
comparison_group_1	character. Name of the first group within <code>comparison_factor</code> to use for the comparison.
comparison_group_2	character. Name of the second group within <code>comparison_factor</code> to use for the comparison.
sample_data	data.frame (optional) data.frame containing the sample metadata of the <code>expression_values</code> . Depending on the object type of <code>expression_values</code> , 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, ReactomeAnalysisRequest, data.frame-method](#), [add\\_dataset, ReactomeAnalysisRequest, matrix-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,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

<code>request</code>	ReactomeAnalysisRequest.
<code>expression_values</code>	matrix. In this case, the <code>sample_data</code> must be set.
<code>name</code>	character. Name of the dataset. This must be unique within one request.
<code>type</code>	character. The type of the dataset. Get available types using <a href="#">get_reactome_data_types</a>
<code>comparison_factor</code>	character. The name of the sample property to use for the main comparison. The sample properties are either retrieved from <code>expression_values</code> or from <code>sample_data</code> .
<code>comparison_group_1</code>	character. Name of the first group within <code>comparison_factor</code> to use for the comparison.
<code>comparison_group_2</code>	character. Name of the second group within <code>comparison_factor</code> 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, ReactomeAnalysisRequest, ExpressionSet-method](#), [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"))
```

---

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**

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

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

---

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= $p \leq 0.05$ , 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**

<code>analysis_id</code>	The running analysis' id
<code>reactome_url</code>	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 <code>http://your.service:1234</code> )

**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 available 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 <a href="#">result_types</a> to retrieve all available types.
name	the name of the result. Use <a href="#">names</a> 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\\_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 <a href="#">result_types</a> to retrieve all available types.
name	the name of the result. Use <a href="#">names</a> 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\\_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.

---

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\\_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	<i>open_reactome</i>
---------------	----------------------

---

## 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\\_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 <a href="#">reactome_links</a> 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\\_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\\_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\\_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 Analysis*

---

## 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	<a href="#">ReactomeAnalysisRequest</a> 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 <code>http://your.service:1234</code> )

## 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\\_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\\_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 <a href="#">ReactomeAnalysisResult</a> 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\\_volcano\(\)](#), [reactome\\_links\(\)](#), [result\\_types\(\)](#)

## Examples

```
# load the scRNA-seq example data  
library(ReactomeGSA.data)  
data(jerby_b_cells)  
  
# perform the GSEA 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 <a href="#">ReactomeAnalysisResult</a> 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\\_volcano\(\)](#), [reactome\\_links\(\)](#), [result\\_types\(\)](#)

## Examples

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

# perform the GSEA 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 <a href="#">ReactomeAnalysisResult</a> 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\\_volcano\(\)](#), [reactome\\_links\(\)](#), [result\\_types\(\)](#)

### Examples

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

# perform the GSEA 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 <a href="#">ReactomeAnalysisResult</a> 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\\_volcano\(\)](#), [reactome\\_links\(\)](#), [result\\_types\(\)](#)

### Examples

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

# perform the GSEA 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	<i>plot_gsva_pca</i>
---------------	----------------------

---

### 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

<code>object</code>	A <a href="#">ReactomeAnalysisResult</a> object containing a ssGSEA result
<code>pathway_ids</code>	A character vector of pathway ids. If set, only these pathways will be used for the PCA analysis.
<code>...</code>	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

<code>object</code>	A <a href="#">ReactomeAnalysisResult</a> object containing a ssGSEA result
<code>pathway_ids</code>	A character vector of pathway ids. If set, only these pathways will be used for the PCA analysis.
<code>...</code>	Additional parameters are passed to <code>prcomp</code>

### Value

A `ggplot2` object representing the plot.

### Examples

```
# load the scRNA-seq example data  
library(ReactomeGSA.data)  
data(jerby_b_cells)  
  
# perform the GSEA analysis  
gsva_result <- analyse_sc_clusters(jerby_b_cells, verbose = FALSE)
```

---

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\(\)](#), [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\(\)](#), [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

<code>x</code>	<a href="#">ReactomeAnalysisRequest</a>
<code>...</code>	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\\_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\\_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	<i>remove_dataset</i>
----------------	-----------------------

---

## Description

Remove the dataset from the [ReactomeAnalysisRequest](#) object.

## Usage

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

## Arguments

x	The <a href="#">ReactomeAnalysisRequest</a> to remove the dataset from
dataset_name	character The dataset's name

## Value

The updated [ReactomeAnalysisRequest](#)

---

remove_dataset, ReactomeAnalysisRequest-method
<i>remove_dataset - ReactomeAnalysisRequest</i>

---

## 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	<i>result_types</i>
--------------	---------------------

---

**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 character 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\\_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 character 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_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	<i>set_method</i>
------------	-------------------

---

**Description**

Set the analysis method used by the [ReactomeAnalysisRequest](#)

**Usage**

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

**Arguments**

request	The <a href="#">ReactomeAnalysisRequest</a> to adjust
method	The name of the method to use. Use <a href="#">get_reactome_methods</a> 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	<i>set_method - ReactomeAnalysisRequest</i>
---	---

---

**Description**

Set the analysis method used by the [ReactomeAnalysisRequest](#)

**Usage**

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

**Arguments**

request	The <a href="#">ReactomeAnalysisRequest</a> to adjust
method	The name of the method to use. Use <a href="#">get_reactome_methods</a> 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	<i>set_parameters</i>
----------------	-----------------------

---

**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 <a href="#">ReactomeAnalysisRequest</a> to set the parameters for.
...	Any name / value pair to set a parameter (see example). For a complete list of available parameters use <a href="#">get_reactome_methods</a>

## 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 <a href="#">ReactomeAnalysisRequest</a> to set the parameters for.
...	Any name / value pair to set a parameter (see example). For a complete list of available parameters use <a href="#">get_reactome_methods</a>

## 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, 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)
```

# Index

- \* **Reactome Service functions**
  - get\_reactome\_data\_types, 21
  - get\_reactome\_methods, 22
- \* **ReactomeAnalysisResult functions**
  - get\_result, 24
  - names, ReactomeAnalysisResult-method, 26
  - open\_reactome, 27
  - pathways, 29
  - plot\_correlations, 32
  - plot\_gsva\_heatmap, 34
  - plot\_gsva\_pathway, 36
  - plot\_volcano, 39
  - reactome\_links, 45
  - result\_types, 48
- \* **add\_dataset methods**
  - add\_dataset, 3
  - add\_dataset, ReactomeAnalysisRequest, data.frame-method, 24, 26–30, 32–37, 40, 41, 44–46, 48, 49
  - add\_dataset, ReactomeAnalysisRequest, DGEList-method, ReactomeAnalysisResult-method, 25
  - add\_dataset, ReactomeAnalysisRequest, EList-method, 8
  - add\_dataset, ReactomeAnalysisRequest, ExpressionSet-method, 10
  - add\_dataset, ReactomeAnalysisRequest, matrix-method, 12
- add\_dataset, 3, 6, 8, 9, 11, 13, 23
- add\_dataset, ReactomeAnalysisRequest, data.frame-method, 41, 44–46, 48, 49
- add\_dataset, ReactomeAnalysisRequest, DGEList-method, 28
- add\_dataset, ReactomeAnalysisRequest, EList-method, 8
- add\_dataset, ReactomeAnalysisRequest, ExpressionSet-method, 10
- add\_dataset, ReactomeAnalysisRequest, matrix-method, 12
- analyse\_sc\_clusters, 13
- analyse\_sc\_clusters, Seurat-method, 15
- analyse\_sc\_clusters, SingleCellExperiment-method, 16
- check\_reactome\_url, 18
- checkRequestValidity, 17
- convert\_reactome\_result, 18
- data\_frame\_as\_string, 19
- get\_fc\_for\_dataset, 19
- get\_is\_sig\_dataset, 20
- get\_reactome\_analysis\_result, 20, 44
- get\_reactome\_analysis\_status, 20, 21
- get\_reactome\_data\_types, 4, 5, 7, 9, 11, 12, 21, 23
- get\_reactome\_methods, 22, 22, 50–52
- get\_result, ReactomeAnalysisResult-method, 25
- is\_gsva\_result, 26
- names, 24, 25, 44
- names, ReactomeAnalysisResult-method, 26
- open\_reactome, 24–26, 27, 29, 30, 32–37, 40, 41, 44–46, 48, 49
- open\_reactome, ReactomeAnalysisResult-method, 28
- pathways, 24–28, 29, 32–37, 40, 41, 44–46, 48, 49
- pathways, ReactomeAnalysisResult-method, 29
- perform\_reactome\_analysis, 20, 30, 44, 55
- plot\_correlations, 24–30, 32, 34–37, 40, 41, 45, 46, 48, 49



plot\_correlations, ReactomeAnalysisResult-method, 33  
 plot\_correlations, ReactomeAnalysisRequest-method, 53  
 plot\_gsva\_heatmap, 24–30, 32, 33, 34, 36, 37, 40, 41, 45, 46, 48, 49  
 plot\_gsva\_heatmap, ReactomeAnalysisResult-method, 35  
 plot\_gsva\_pathway, 24–30, 32–35, 36, 40, 41, 45, 46, 48, 49  
 plot\_gsva\_pathway, ReactomeAnalysisResult-method, 37  
 plot\_gsva\_pca, 38  
 plot\_gsva\_pca, ReactomeAnalysisResult-method, 39  
 plot\_volcano, 24–30, 32–37, 39, 45, 46, 48, 49  
 plot\_volcano, ReactomeAnalysisResult-method, 40  
 print, ReactomeAnalysisRequest-method, 41  
 print, ReactomeAnalysisResult-method, 42  
  
 reactome\_links, 24–30, 32–37, 40, 41, 44, 45, 48, 49  
 reactome\_links, ReactomeAnalysisResult-method, 46  
 ReactomeAnalysisRequest, 3, 4, 6, 7, 9, 11, 13, 30, 31, 41, 43, 47, 48, 50–54  
 ReactomeAnalysisResult, 14, 16–18, 24–26, 34–39, 42, 48, 49, 54  
 ReactomeAnalysisResult  
     (ReactomeAnalysisResult-class), 44  
 ReactomeAnalysisResult-class, 44  
 remove\_dataset, 47  
 remove\_dataset, ReactomeAnalysisRequest-method, 47  
 result\_types, 24–30, 32–37, 40, 41, 44–46, 48  
 result\_types, ReactomeAnalysisResult-method, 49  
  
 set\_method, 50  
 set\_method, ReactomeAnalysisRequest-method, 50  
 set\_parameters, 23, 51  
 set\_parameters, ReactomeAnalysisRequest-method, 52