Package ‘gemma.R’

May 21, 2024

**Title**  A wrapper for Gemma's Restful API to access curated gene expression data and differential expression analyses

**Version**  3.0.4

**Description**  Low- and high-level wrappers for Gemma's RESTful API. They enable access to curated expression and differential expression data from over 10,000 published studies. Gemma is a web site, database and a set of tools for the meta-analysis, re-use and sharing of genomics data, currently primarily targeted at the analysis of gene expression profiles.


**License**  Apache License (>= 2)

**Encoding**  UTF-8

**Roxygen**  list(markdown = TRUE)

**RoxygenNote**  7.3.1

**BugReports**  [https://github.com/PavlidisLab/gemma.R/issues](https://github.com/PavlidisLab/gemma.R/issues)

**Imports**  magrittr, glue, memoise, jsonlite, data.table, rlang, lubridate, utils, stringr, SummarizedExperiment, Biobase, tibble, tidyr, S4Vectors, httr, rappdirs, bit64, assertthat, digest

**Suggests**  testthat (>= 2.0.0), rmarkdown, knitr, dplyr, covr, ggplot2, ggrepel, BiocStyle, microbenchmark, magick, purrr, pheatmap, viridis, poolr, kableExtra, listviewer, shiny

**Config/testthat/edition**  2

**VignetteBuilder**  knitr

**biocViews**  Software, DataImport, Microarray, SingleCell, ThirdPartyClient, DifferentialExpression, GeneExpression, Bayesian, Annotation, ExperimentalDesign, Normalization, BatchEffect, Preprocessing

**git_url**  [https://git.bioconductor.org/packages/gemma.R](https://git.bioconductor.org/packages/gemma.R)

**git_branch**  RELEASE_3_19
Contents

.git_last_commit 775417f
.git_last_commit_date 2024-05-17
Repository Bioconductor 3.19
Date/Publication 2024-05-21
Author Javier Castillo-Arnemann [aut] ([https://orcid.org/0000-0002-5626-9004]), Jordan Sicherman [aut] ([https://orcid.org/0000-0001-8160-4567]), Ogan Mancarci [cre, aut] ([https://orcid.org/0000-0002-1452-0889]), Guillaume Poirier-Morency [aut] ([https://orcid.org/0000-0002-6554-0441])
Maintainer Ogan Mancarci <ogan.mancarci@gmail.com>

Contents

.getResultSets ............................................. 4
accessField ............................................... 5
blank_processor ......................................... 5
checkBounds .............................................. 6
encode ...................................................... 6
filter_properties ......................................... 7
forget_gemma_memoised .................................. 7
gemma.R ................................................... 8
gemmaCache ............................................... 9
gemmaPath ............................................... 9
gemma_call ............................................... 9
gemma_kable .............................................. 10
gemma_memoise .......................................... 10
get_all_pages ............................................ 11
get_child_terms .......................................... 12
get_datasets ............................................. 12
get_datasets_by_ids .................................... 15
get_dataset_annotations ................................ 17
get_dataset_design ....................................... 18
get_dataset_differential_expression_analyses ........... 19
get_dataset_expression .................................. 20
get_dataset_expression_for_genes ....................... 21
get_dataset_object ...................................... 21
get_dataset_platforms ................................... 24
get_dataset_processed_expression ....................... 25
get_dataset_quantitation_types ......................... 26
get_dataset_raw_expression ................................ 28
get_dataset_samples ..................................... 29
get_differential_expression_values ...................... 30
get_genes .................................................. 31
get_gene_go_terms ....................................... 32
get_gene_locations ...................................... 33
Contents

get_gene_probes .................................................. 35
get_platforms_by_ids ............................................ 36
get_platform_annotations ....................................... 38
get_platform_datasets ........................................... 39
get_platform_element_genes ..................................... 41
get_result_sets .................................................... 42
get_taxa .......................................................... 44
get_taxa_by_ids ................................................... 45
generate_taxon .................................................... 46
isEmpty ........................................................... 48
make_design ......................................................... 49
memoise ............................................................. 49
nullCheck .......................................................... 50
processAnnotations ............................................... 50
processCharacteristicValueObject ............................... 51
processDatasetResultSets ........................................ 51
processDatasets .................................................... 52
processDEA ........................................................ 53
processDEcontrasts ............................................... 54
processDEMatrix ................................................... 54
processDesignMatrix .............................................. 55
processDifferentialExpressionAnalysisResultSetValueObject
.................................................................................. 55
processElements ..................................................... 56
processExpressionMatrix ........................................... 57
processFile .......................................................... 57
processGemmaArray ............................................... 58
processGeneLocation ............................................... 58
processGenes ......................................................... 59
processGO .......................................................... 60
processPlatforms .................................................... 60
processQuantitationTypeValueObject ............................. 61
processResultSetFactors ............................................ 62
processSamples ...................................................... 63
processSearchAnnotations ......................................... 63
processTaxon ........................................................ 64
process_search ...................................................... 65
search_annotations ................................................. 65
search_datasets ..................................................... 66
search_gemma ....................................................... 69
setGemmaPath ....................................................... 70
set_gemma_user ..................................................... 70
subset_factorValues ............................................... 71
update_result ....................................................... 71
validateBoolean ..................................................... 72
validateID ........................................................... 73
validateLimit ......................................................... 73
validateOptionalID ................................................. 74
validateOptionalQuery ............................................. 74
.getResultSets

Retrieve a single analysis result set by its identifier

Description

Retrieve a single analysis result set by its identifier

Usage

`.getResultSets(
    resultSet = NA_character_,
    raw = getOption("gemma.raw", FALSE),
    memoised = getOption("gemma.memoised", FALSE),
    file = getOption("gemma.file", NA_character_),
    overwrite = getOption("gemma.overwrite", FALSE)
)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resultSet</td>
<td>An expression analysis result set numerical identifier.</td>
</tr>
<tr>
<td>raw</td>
<td>TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.</td>
</tr>
<tr>
<td>memoised</td>
<td>Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> to clear the cache.</td>
</tr>
<tr>
<td>file</td>
<td>The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Whether or not to overwrite if a file exists at the specified filename.</td>
</tr>
</tbody>
</table>

Value

Varies
accessField

Examples

   # gemma.R:::.getResultSets(523099)

accessField

Access the field in a list

Description

This function accesses named field within the elements of a list. If an element lacks the field, it’s filled in by natype.

Usage

   accessField(d, field, natype = NA)

Arguments

   d          Input data list
   field      Field name to access in each element
   natype     What to fill in when field is unavailable

Value

   A vector of elements

blank_processor

A blank processor that returns data as is

Description

A blank processor that returns data as is

Usage

   blank_processor(data)

Arguments

   data       any data

Value

   Data as is
checkBounds  
Replace missing data with NAs

Description
Replace missing data with NAs

Usage
checkBounds(x, natype = NA)

Arguments
x  
Data
natype  
type of NA to replace the missing data with

Value
Data or NA in case of an out of bounds error

encode  
URL encode a string safely

Description
URL encode a string safely

Usage
encode(url)

Arguments
url  
The string to URL encode. Vectors are delimited by a comma.

Value
A URL encoding of url
filter_properties

Return all supported filter properties

Description

Some functions such as `get_datasets` and `get_platforms_by_ids` include a filter argument that allows creation of more complex queries. This function returns a list of supported properties to be used in those filters.

Usage

```r
filter_properties()
```

Value

A list of data.tables that contain supported properties and their data types

Examples

```r
filter_properties()
```

forget_gemma_memoised

Clear `gemma.R` cache

Description

Forget past results from memoised calls to the Gemma API (i.e. using functions with `memoised = TRUE`)

Usage

```r
forget_gemma_memoised()
```

Value

TRUE to indicate cache was cleared.

Examples

```r
forget_gemma_memoised()
```
**Description**

This package contains wrappers and convenience functions for Gemma’s RESTful API that enables access to curated expression and differential expression data from over 15,000 published studies (as of mid-2022). Gemma (https://gemma.msl.ubc.ca) is a web site, database and a set of tools for the meta-analysis, re-use and sharing of transcriptomics data, currently primarily targeted at the analysis of gene expression profiles.

**Details**

Most users will want to start with the high-level functions like `get_dataset_object`, `get_differential_expression_values` and `get_platform_annotations`. Additional lower-level methods are available that directly map to the Gemma RESTful API methods.

For more information and detailed usage instructions check the [README](https://support.bioconductor.org), the function reference and the vignette.

All software-related questions should be posted to the Bioconductor Support Site: [https://support.bioconductor.org](https://support.bioconductor.org)

**Author(s)**

Javier Castillo-Arnemann, Jordan Sicherman, Ogan Mancari, Guillaume Poirier-Morency

**References**

Lim, N. et al., Curation of over 10 000 transcriptomic studies to enable data reuse, Database, 2021. [https://doi.org/10.1093/database/baab006](https://doi.org/10.1093/database/baab006)

**See Also**

Useful links:

- [https://github.com/PavlidisLab/gemma.R](https://github.com/PavlidisLab/gemma.R)
- Report bugs at [https://github.com/PavlidisLab/gemma.R/issues](https://github.com/PavlidisLab/gemma.R/issues)
**gemmaCache**

**Gemma Cache**

**Description**

Gemma Cache

**Usage**

```r
gemmaCache()
```

**Value**

A memoise filesystem

---

**gemmaPath**

**Get gemma path**

**Description**

Get gemma path

**Usage**

```r
gemmaPath()
```

**Value**

Link to Gemma API

---

**gemma_call**

**Custom gemma call**

**Description**

A minimal function to create custom calls. Can be used to acquire unimplemented endpoints and/or raw output without any processing. Refer to the API documentation.

**Usage**

```r
gemma_call(call, ..., json = TRUE)
```
Arguments

call          Gemma API endpoint.
...          parameters included in the call
json          If TRUE will parse the content as a list

Value

A list if `json = TRUE` and an `httr` response if `FALSE`

Examples

```r
# get singular value decomposition for the dataset
gemma_call('datasets/(dataset)/svd', dataset = 1)
```

---

**gemma_kable**  
Create printable tables out of `gemma.R` outputs

**Description**

Creates a `kable` where certain columns are automatically shortened to better fit a document.

**Usage**

`gemma_kable(table)`

**Arguments**

table          A `data.table` or `data.frame` outputted by a `gemma.R` function

---

**gemma_memoise**  
Enable and disable memoisation of `gemma.R` functions

**Description**

Enable and disable memoisation of `gemma.R` functions

**Usage**

```r
gemma_memoise(
  memoised = FALSE,
  cache = rappdirs::user_cache_dir(appname = "gemmaR")
)
```
**get_all_pages**

Get all pages of a paginated call

**Description**

Given a Gemma.R output from a function with offset and limit arguments, returns the output from all pages. All arguments other than offset, limit

**Usage**

```
get_all_pages(
  query,
  step_size = 100,
  binder = rbind,
  directory = NULL,
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

**Arguments**

- **query**: Output from a gemma.R function with offset and limit argument
- **step_size**: Size of individual calls to the server. 100 is the maximum value
- **binder**: Binding function for the calls. If raw = FALSE use rbind to combine the data.tables. If not, use c to combine lists
- **directory**: Directory to save the output from the individual calls to. If provided, each page is saved to separate files.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. This function always saves the output as an RDS file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.

**Value**

A data.table or a list containing data from all pages.
get_child_terms  
Return child terms of a term

Description

When querying for ontology terms, Gemma propagates these terms to include any datasets with their child terms in the results. This function returns these children for any number of terms, including all children and the terms itself in the output vector.

Usage

get_child_terms(terms)

Arguments

terms  An array of terms

Value

An array containing descendends of the annotation terms, including the terms themselves

Examples

get_child_terms("http://purl.obolibrary.org/obo/MONDO_0000408")

get_datasets  
Retrieve all datasets

Description

Retrieve all datasets

Usage

get_datasets(  
  query = NA_character_,  
  filter = NA_character_,  
  taxa = NA_character_,  
  uris = NA_character_,  
  offset = 0L,  
  limit = 20L,  
  sort = "+id",  
  raw = getOption("gemma.raw", FALSE),  
  memoised = getOption("gemma.memoised", FALSE),  
  file = getOption("gemma.file", NA_character_),  
  overwrite = getOption("gemma.overwrite", FALSE)  
)
Arguments

query
The search query. Queries can include plain text or ontology terms. They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").

filter
Filter results by matching expression. Use filter_properties function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as "="", "<" or "in". (e.g. "taxon.commonName = human", "taxon.commonName in (human,mouse), "id < 1000")

taxa
A vector of taxon common names (e.g. human, mouse, rat). Providing multiple species will return results for all species. These are appended to the filter and equivalent to filtering for taxon.commonName property

uris
A vector of ontology term URIs. Providing multiple terms will return results containing any of the terms and their children. These are appended to the filter and equivalent to filtering for allCharacteristics.valueUri

offset
The offset of the first retrieved result.

limit
Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.

sort
Order results by the given property and direction. The '+' sign indicate ascending order whereas the '-' indicate descending.

raw
TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised
Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

file
The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite
Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the queried dataset(s). A list if raw = TRUE. Returns an empty list if no datasets matched.

The fields of the output data.table are:

- experiment.shortName: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- experiment.name: Full title of the dataset
- experiment.ID: Internal ID of the dataset.
• **experiment.description**: Description of the dataset

• **experiment.troubled**: Did an automatic process within gemma or a curator mark the dataset as "troubled"

• **experiment.accession**: Accession ID of the dataset in the external database it was taken from

• **experiment.database**: The name of the database where the dataset was taken from

• **experiment.URI**: URI of the original database

• **experiment.sampleCount**: Number of samples in the dataset

• **experiment.batchEffectText**: A text field describing whether the dataset has batch effects

• **experiment.batchCorrected**: Whether batch correction has been performed on the dataset.

• **experiment.batchConfound**: 0 if batch info isn't available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found

• **experiment.batchEffect**: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.

• **experiment.rawData**: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches

• **geeq.qScore**: Data quality score given to the dataset by Gemma.

• **geeq.sScore**: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design

• **taxon.name**: Name of the species

• **taxon.scientific**: Scientific name for the taxon

• **taxon.ID**: Internal identifier given to the species by Gemma

• **taxon.NCBI**: NCBI ID of the taxon

• **taxon.database.name**: Underlying database used in Gemma for the taxon

• **taxon.database.ID**: ID of the underlying database used in Gemma for the taxon

### Examples

```r
get_datasets()
get_datasets(taxa = c("mouse", "human"), uris = "http://purl.obolibrary.org/obo/UBERON_0002048")
# filter below is equivalent to the call above
get_datasets(filter = "taxon.commonName in (mouse,human) and allCharacteristics.valueUri = http://purl.obolibrary"
get_datasets(query = "lung")```
get_datasets_by_ids

Retrieve datasets by their identifiers

Usage

```r
get_datasets_by_ids(
  datasets = NA_character_,
  filter = NA_character_,
  taxa = NA_character_,
  uris = NA_character_,
  offset = 0L,
  limit = 20L,
  sort = "+id",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

datasets Numerical dataset identifiers or dataset short names. If not specified, all datasets will be returned instead

filter Filter results by matching expression. Use `filter_properties` function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as "+", "+<" or "+in". (e.g.
  "taxon.commonName = human", "taxon.commonName in (human,mouse), "id < 1000")

taxa A vector of taxon common names (e.g. human, mouse, rat). Providing multiple species will return results for all species. These are appended to the filter and equivalent to filtering for `taxon.commonName` property

uris A vector of ontology term URIs. Providing multiple terms will return results containing any of the terms and their children. These are appended to the filter and equivalent to filtering for `allCharacteristics.valueUri`

offset The offset of the first retrieved result.

limit Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the `totalElements` attribute in the output to compile all data if needed.

sort Order results by the given property and direction. The '+' sign indicate ascending order whereas the '-' indicate descending.
get_datasets_by_ids

**raw**

TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

**memoised**

Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

**file**

The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

**overwrite**

Whether or not to overwrite if a file exists at the specified filename.

**Value**

A data table with information about the queried dataset(s). A list if `raw = TRUE`. Returns an empty list if no datasets matched.

The fields of the output data.table are:

- `experiment.shortName`: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- `experiment.name`: Full title of the dataset
- `experiment.ID`: Internal ID of the dataset.
- `experiment.description`: Description of the dataset
- `experiment.troubled`: Did an automatic process within gemma or a curator mark the dataset as "troubled"
- `experiment.accession`: Accession ID of the dataset in the external database it was taken from
- `experiment.database`: The name of the database where the dataset was taken from
- `experiment.URI`: URI of the original database
- `experiment.sampleCount`: Number of samples in the dataset
- `experiment.batchEffectText`: A text field describing whether the dataset has batch effects
- `experiment.batchCorrected`: Whether batch correction has been performed on the dataset.
- `experiment.batchConfound`: 0 if batch info isn’t available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
- `experiment.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.
- `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- `geeq.qScore`: Data quality score given to the dataset by Gemma.
- `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- `taxon.name`: Name of the species
get_dataset_annotations

- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlyng database used in Gemma for the taxon

Examples

```r
get_datasets_by_ids("GSE2018")
get_datasets_by_ids(c("GSE2018", "GSE2872"))
```

---

get_dataset_annotations

Retrieve the annotations of a dataset

Description

Retrieve the annotations of a dataset

Usage

```r
get_dataset_annotations(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- **dataset**: A numerical dataset identifier or a dataset short name
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.
Value

A data table with information about the annotations of the queried dataset. A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- `class.name`: Name of the annotation class (e.g. organism part)
- `class.URI`: URI for the annotation class
- `term.name`: Name of the annotation term (e.g. lung)
- `term.URI`: URI for the annotation term
- `object.class`: Class of object that the term originated from.

Examples

get_dataset_annotations("GSE2018")

---

**get_dataset_design**  
Retrieve the design of a dataset

Description

Retrieve the design of a dataset

Usage

get_dataset_design(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)

Arguments

dataset  
A numerical dataset identifier or a dataset short name

raw  
TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised  
Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

file  
The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite  
Whether or not to overwrite if a file exists at the specified filename.
Value

A data table of the design matrix for the queried dataset. A 404 error if the given identifier does not map to any object

Examples

head(get_dataset_design("GSE2018"))

---

get_dataset_differential_expression_analyses

Retrieve annotations and surface level stats for a dataset’s differential analyses

Description

Retrieve annotations and surface level stats for a dataset’s differential analyses

Usage

get_dataset_differential_expression_analyses(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)

Arguments

dataset     A numerical dataset identifier or a dataset short name
raw         TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
memoised    Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.
file        The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite   Whether or not to overwrite if a file exists at the specified filename.
get_dataset_expression

**Value**

A data table with information about the differential expression analysis of the queried dataset. Note that this function does not return differential expression values themselves. Use `get_differential_expression_values` to get differential expression values (see examples).

The fields of the output data.table are:

- `result.ID`: Result set ID of the differential expression analysis. May represent multiple factors in a single model.
- `contrast.ID`: Id of the specific contrast factor. Together with the `result.ID` they uniquely represent a given contrast.
- `experiment.ID`: Id of the source experiment
- `factor.category`: Category for the contrast
- `factor.category.URI`: URI for the contrast category
- `factor.ID`: ID of the factor
- `baseline.factors`: Characteristics of the baseline. This field is a data.table
- `experimental.factors`: Characteristics of the experimental group. This field is a data.table
- `isSubset`: TRUE if the result set belong to a subset, FALSE if not. Subsets are created when performing differential expression to avoid unhelpful comparisons.
- `subsetFactor`: Characteristics of the subset. This field is a data.table
- `probes.analyzed`: Number of probesets represented in the contrast
- `genes.analyzed`: Number of genes represented in the contrast

**Examples**

```r
result <- get_dataset_differential_expression_analyses("GSE2872")
get_differential_expression_values(resultSet = result$result.ID[1])
```

---

**get_dataset_expression**

*Retrieve processed expression data of a dataset*

**Description**

This function is deprecated in favor of `get_dataset_processed_expression`

**Usage**

```r
get_dataset_expression(
  dataset,
  filter = FALSE,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```
get_dataset_expression_for_genes

Arguments

**dataset** A numerical dataset identifier or a dataset short name

**filter** This argument is ignored due to deprecation of the function

**raw** TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

**memoised** Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

**file** The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

**overwrite** Whether or not to overwrite if a file exists at the specified filename.

Value

If raw is FALSE (default), a data table of the expression matrix for the queried dataset. If raw is TRUE, returns the binary file in raw form.

Examples

```r
get_dataset_expression("GSE2018")
```

get_dataset_expression_for_genes

*Retrieve the expression data matrix of a set of datasets and genes*

Description

Retrieve the expression data matrix of a set of datasets and genes

Usage

```r
get_dataset_expression_for_genes(
  datasets,
  genes,
  keepNonSpecific = FALSE,
  consolidate = NA_character_,
  raw =getOption("gemma.raw", FALSE),
  memoised =getOption("gemma.memoised", FALSE),
  file =getOption("gemma.file", NA_character_),
  overwrite =getOption("gemma.overwrite", FALSE)
)
```
get_dataset_object

**Arguments**

- **datasets**: A numerical dataset identifier or a dataset short name
- **genes**: An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc
- **keepNonSpecific**: logical. FALSE by default. If TRUE, results from probesets that are not specific to the gene will also be returned.
- **consolidate**: An option for gene expression level consolidation. If empty, will return every probe for the genes. "pickmax" to pick the probe with the highest expression, "pickvar" to pick the probe with the highest variance and "average" for returning the average expression
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.

**Value**

A list of data frames

**Examples**

```r
get_dataset_expression_for_genes("GSE2018", genes = c(10225, 2841))
```

**Description**

Return an annotated Bioconductor-compatible data structure or a long form tibble of the queried dataset, including expression data and the experimental design.
Usage

```r
get_dataset_object(
  datasets,
  genes = NULL,
  keepNonSpecific = FALSE,
  consolidate = NA_character_,
  resultSets = NULL,
  contrasts = NULL,
  metaType = "text",
  type = "se",
  memoised = getOption("gemma.memoised", FALSE)
)
```

Arguments

datasets A numerical dataset identifier or a dataset short name
genes An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc
keepNonSpecific logical. FALSE by default. If TRUE, results from probesets that are not specific to the gene will also be returned.
consolidate An option for gene expression level consolidation. If empty, will return every probe for the genes. "pickmax" to pick the probe with the highest expression, "pickvar" to pick the prove with the highest variance and "average" for returning the average expression
resultSets Result set IDs of the a differential expression analysis. Optional. If provided, the output will only include the samples from the subset used in the result set ID. Must be the same length as datasets.
contrasts Contrast IDs of a differential expression contrast. Optional. Need resultSets to be defined to work. If provided, the output will only include samples relevant to the specific contrasts.
metaType How should the metadata information should be included. Can be "text", "uri" or "both". "text" and "uri" options
type "se"for a SummarizedExperiment or "eset" for Expression Set. We recommend using SummarizedExperiments which are more recent. See the Summarized experiment vignette or the ExpressionSet vignette for more details. "tidy" for a long form data frame compatible with tidyverse functions. 'list' to return a list containing individual data frames containing expression values, design and the experiment.
memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
get_dataset_platforms

Value

A list of `SummarizedExperiments`, `ExpressionSets` or a tibble containing metadata and expression data for the queried datasets and genes. Metadata will be expanded to include a variable number of factors that annotates samples from a dataset but will always include single "factorValues" column that houses data.tables that include all annotations for a given sample.

Examples

```r
get_dataset_object("GSE2018")
```

---

get_dataset_platforms  Retrieve the platforms of a dataset

Description

Retrieve the platforms of a dataset

Usage

```r
get_dataset_platforms(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- `dataset` A numerical dataset identifier or a dataset short name
- `raw` TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- `memoised` Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- `file` The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- `overwrite` Whether or not to overwrite if a file exists at the specified filename.
Value
A data table with information about the platform(s). A list if \texttt{raw = TRUE}. A 404 error if the given identifier does not map to any object

The fields of the output data.table are:

- \texttt{platform.ID}: Internal identifier of the platform
- \texttt{platform.shortName}: Shortname of the platform.
- \texttt{platform.name}: Full name of the platform.
- \texttt{platform.description}: Free text description of the platform
- \texttt{platform.troubled}: Whether or not the platform was marked "troubled" by a Gemma process or a curator
- \texttt{platform.experimentCount}: Number of experiments using the platform within Gemma
- \texttt{platform.type}: Technology type for the platform.
- \texttt{taxon.name}: Name of the species platform was made for
- \texttt{taxon.scientific}: Scientific name for the taxon
- \texttt{taxon.ID}: Internal identifier given to the species by Gemma
- \texttt{taxon.NCBI}: NCBI ID of the taxon
- \texttt{taxon.database.name}: Underlying database used in Gemma for the taxon
- \texttt{taxon.database.ID}: ID of the underlying database used in Gemma for the taxon

Examples

\begin{verbatim}
get_dataset_platforms("GSE2018")
\end{verbatim}

---

\texttt{get_dataset_processed_expression}

\textit{Retrieve processed expression data of a dataset}

Description
Retrieve processed expression data of a dataset

Usage

\begin{verbatim}
get_dataset_processed_expression(
    dataset,
    raw = getOption("gemma.raw", FALSE),
    memoised = getOption("gemma.memoised", FALSE),
    file = getOption("gemma.file", NA_character_),
    overwrite = getOption("gemma.overwrite", FALSE)
)
\end{verbatim}
get_dataset_quantitation_types

Retrieve quantitation types of a dataset

Arguments

- **dataset**: A numerical dataset identifier or a dataset short name.
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.

Value

If raw is FALSE (default), a data table of the expression matrix for the queried dataset. If raw is TRUE, returns the binary file in raw form.

Examples

```r
get_dataset_processed_expression("GSE2018")
```

Description

Retrieve quantitation types of a dataset

Usage

```r
get_dataset_quantitation_types(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```
get_dataset_quantitation_types

Arguments

dataset A numerical dataset identifier or a dataset short name

raw TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

file The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite Whether or not to overwrite if a file exists at the specified filename.

Value

A data.table containing the quantitation types

The fields of the output data.table are:

- id: If of the quantitation type. Any raw quantitation type can be accessed by `get_dataset_raw_expression` function using this id.
- name: Name of the quantitation type
- description: Description of the quantitation type
- type: Type of the quantitation type. Either raw or processed. Each dataset will have one processed quantitation type which is the data returned using `get_dataset_processed_expression`
- ratio: Whether or not the quantitation type is a ratio of multiple quantitation types. Typically TRUE for processed TWOCOLOR quantitation type.
- preferred: The preferred raw quantitation type. This version is used in generation of the processed data within gemma.
- recomputed: If TRUE this quantitation type is generated by recomputing raw data files Gemma had access to.

Examples

get_dataset_quantitation_types("GSE59918")
get_dataset_raw_expression

Retrieve raw expression data of a dataset

Description

Retrieve raw expression data of a dataset

Usage

get_dataset_raw_expression(
  dataset,
  quantitationType,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)

Arguments

- **dataset**
  A numerical dataset identifier or a dataset short name

- **quantitationType**
  Quantitation type id. These can be acquired using get_dataset_quantitation_types function. This endpoint can only return non-processed quantitation types.

- **raw**
  TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

- **memoised**
  Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

- **file**
  The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

- **overwrite**
  Whether or not to overwrite if a file exists at the specified filename.

Value

If raw is FALSE (default), a data table of the expression matrix for the queried dataset. If raw is TRUE, returns the binary file in raw form.

Examples

```r
q_types <- get_dataset_quantitation_types("GSE59918")
get_dataset_raw_expression("GSE59918", q_types$id[q_types$name == "Counts"])
```
get_dataset_samples

Retrieves the samples of a dataset

Description

Retrieve the samples of a dataset

Usage

```r
get_dataset_samples(
  dataset,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- `dataset`: A numerical dataset identifier or a dataset short name
- `raw`: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- `memoised`: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- `file`: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- `overwrite`: Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the samples of the queried dataset. A list if `raw = TRUE`. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- `sample.name`: Internal name given to the sample
- `sample.ID`: Internal ID of the sample
- `sample.description`: Free text description of the sample
- `sample.outlier`: Whether or not the sample is marked as an outlier
- `sample.accession`: Accession ID of the sample in it’s original database
- `sample.database`: Database of origin for the sample
- `sample.characteristics`: Characteristics of the sample. This field is a data table
- `sample.factorValues`: Experimental factor values of the sample. This field is a data table
get_differential_expression_values

Retrieve differential expression results

Description

Retrieves the differential expression result set(s) associated with the dataset. To get more information about the contrasts in individual resultSets and annotation terms associated them, use get_dataset_differential_expression_analyses()

Usage

get_differential_expression_values(
  dataset = NA_character_,
  resultSets = NA_integer_,
  readableContrasts = FALSE,
  memoised = getOption("gemma.memoised", FALSE)
)

Arguments

dataset A dataset identifier.
resultSets resultSet identifiers. If a dataset is not provided, all result sets will be downloaded. If it is provided it will only be used to ensure all result sets belong to the dataset.
readableContrasts If FALSE (default), the returned columns will use internal constrasts IDs as names. Details about the contrasts can be accessed using get_dataset_differential_expression_analyses() If TRUE IDs will be replaced with human readable contrast information.
memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

Details

In Gemma each result set corresponds to the estimated effects associated with a single factor in the design, and each can have multiple contrasts (for each level compared to baseline). Thus a dataset with a 2x3 factorial design will have two result sets, one of which will have one contrast, and one having two contrasts.

The methodology for differential expression is explained in Curation of over 10000 transcriptomic studies to enable data reuse. Briefly, differential expression analysis is performed on the dataset based on the annotated experimental design with up two three potentially nested factors. Gemma
attempts to automatically assign baseline conditions for each factor. In the absence of a clear control condition, a baseline is arbitrarily selected. A generalized linear model with empirical Bayes shrinkage of t-statistics is fit to the data for each platform element (probe/gene) using an implementation of the limma algorithm. For RNA-seq data, we use weighted regression, applying the voom algorithm to compute weights from the mean–variance relationship of the data. Contrasts of each condition are then computed compared to the selected baseline. In some situations, Gemma will split the data into subsets for analysis. A typical such situation is when a ‘batch’ factor is present and confounded with another factor, the subsets being determined by the levels of the confounding factor.

Value

A list of data tables with differential expression values per result set.

Examples

get_differential_expression_values("GSE2018")

get_genes

Retrieve genes matching gene identifiers

Description

Retrieve genes matching gene identifiers

Usage

get_genes(  
genes,  
raw = getOption("gemma.raw", FALSE),  
memoised = getOption("gemma.memoised", FALSE),  
file = getOption("gemma.file", NA_character_),  
overwrite = getOption("gemma.overwrite", FALSE)
)

Arguments

genes An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc

raw TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.
get_gene_go_terms

Description
Retrieve the GO terms associated to a gene

Usage
get_gene_go_terms(
  gene,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
get_gene_locations

Arguments

- **gene**: An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc.
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the GO terms assigned to the queried gene. A list if `raw = TRUE`. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- **term.name**: Name of the term
- **term.ID**: ID of the term
- **term.URI**: URI of the term

Examples

```r
get_gene_go_terms(3091)
```

Description

Retrieve the physical locations of a given gene

Usage

```r
gene_locations<- get_gene_locations(  
gene,  
raw = getOption("gemma.raw", FALSE),  
memoised = getOption("gemma.memoised", FALSE),  
file = getOption("gemma.file", NA_character_),  
overwrite = getOption("gemma.overwrite", FALSE)  
)
```
get_gene_locations

Arguments

gene An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc

raw TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

file The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the physical location of the queried gene. A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- chromosome: Name of the chromosome the gene is located
- strand: Which strand the gene is located
- nucleotide: Nucleotide number for the gene
- length: Gene length
- taxon.name: Name of the taxon
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal ID for the taxon given by Gemma
- taxon.NCBI: NCBI ID for the taxon
- taxon.database.name: Name of the database used in Gemma for the taxon

Examples

gene_locations("DYRK1A")
gen_locations(1859)
get_gene_probes

Retrieve the probes associated to a genes across all platforms

Description

Retrieve the probes associated to a genes across all platforms

Usage

get_gene_probes(
  gene,
  offset = 0L,
  limit = 20L,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)

Arguments

gene
  An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc

offset
  The offset of the first retrieved result.

limit
  Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.

raw
  TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised
  Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

file
  The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite
  Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the probes representing a gene across all platforms. A list if raw = TRUE. A 404 error if the given identifier does not map to any genes.

The fields of the output data.table are:
get_platforms_by_ids

- element.name: Name of the element. Typically the probeset name
- element.description: A free text field providing optional information about the element
- platform.shortName: Shortname of the platform given by Gemma. Typically the GPL identifier.
- platform.name: Full name of the platform
- platform.ID: Id number of the platform given by Gemma
- platform.type: Type of the platform.
- platform.description: Free text field describing the platform.
- platform.troubled: Whether the platform is marked as troubled by a Gemma curator.
- taxon.name: Name of the species platform was made for
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon

Examples

get_gene_probes(1859)

get_platforms_by_ids

Retrieve all platforms matching a set of platform identifiers

Description

Retrieve all platforms matching a set of platform identifiers

Usage

get_platforms_by_ids(
  platforms = NA_character_,
  filter = NA_character_,
  taxa = NA_character_,
  offset = 0L,
  limit = 20L,
  sort = "+id",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
get_platforms_by_ids

Arguments

platforms Platform numerical identifiers or platform short names. If not specified, all platforms will be returned instead.

filter Filter results by matching expression. Use filter_properties function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as ":", "<" or ":in". (e.g. "taxon.commonName = human", "taxon.commonName in (human,mouse), "id < 1000")

taxa A vector of taxon common names (e.g. human, mouse, rat). Providing multiple species will return results for all species. These are appended to the filter and equivalent to filtering for taxon.commonName property

offset The offset of the first retrieved result.

limit Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.

sort Order results by the given property and direction. The '+' sign indicate ascending order whereas the '-' indicate descending.

raw TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemmememoised to clear the cache.

file The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with information about the platform(s). A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- platform.ID: Internal identifier of the platform
- platform.shortName: Shortname of the platform.
- platform.name: Full name of the platform.
- platform.description: Free text description of the platform
- platform.troubled: Whether or not the platform was marked "troubled" by a Gemma process or a curator
- platform.experimentCount: Number of experiments using the platform within Gemma
- platform.type: Technology type for the platform.
get_platform_annotations

- taxon.name: Name of the species platform was made for
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon

Examples

get_platforms_by_ids("GPL1355")
get_platforms_by_ids(c("GPL1355", "GPL96"))

get_platform_annotations

Retrieve Platform Annotations by Gemma

Description

Gets Gemma's platform annotations including mappings of microarray probes to genes.

Usage

get_platform_annotations(
  platform,
  annotType = c("noParents", "allParents", "bioProcess"),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE),
  memoised = getOption("gemma.memoise", FALSE),
  unzip = FALSE
)

Arguments

- platform: A platform numerical identifiers or platform short name.
- annotType: Which GO terms should the output include
- file: Where to save the annotation file to, or empty to just load into memory
- overwrite: Whether or not to overwrite an existing file
- memoised: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.
- unzip: Whether or not to unzip the file (if @param file is not empty)
Value

A table of annotations

- **ProbeName**: Probeset names provided by the platform. Gene symbols for generic annotations
- **GeneSymbols**: Genes that were found to be aligned to the probe sequence. Note that it is possible for probes to be non-specific. Alignment to multiple genes are indicated with gene symbols separated by "|"s
- **GeneNames**: Name of the gene
- **GOTerms**: GO Terms associated with the genes. `annotType` argument can be used to choose which terms should be included.
- **GemmaIDs and NCBIIds**: respective IDs for the genes.

Examples

```r
head(get_platform_annotations("GPL96"))
head(get_platform_annotations('Generic_human_ncbiIds'))
```

get_platform_datasets

Retrieve all experiments using a given platform

Description

Retrieve all experiments using a given platform

Usage

```r
get_platform_datasets(
  platform,
  offset = 0L,
  limit = 20L,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- **platform**: A platform numerical identifier or a platform short name
- **offset**: The offset of the first retrieved result.
- **limit**: Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with `offset` and the `totalElements` attribute in the output to compile all data if needed.
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
### get_platform.datasets

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoised</td>
<td>Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> to clear the cache.</td>
</tr>
<tr>
<td>file</td>
<td>The name of a file to save the results to, or <code>NULL</code> to not write results to a file. If <code>raw == TRUE</code>, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Whether or not to overwrite if a file exists at the specified filename.</td>
</tr>
</tbody>
</table>

### Value

A data table with information about the queried dataset(s). A list if `raw = TRUE`. Returns an empty list if no datasets matched.

The fields of the output data.table are:

- `experiment.shortName`: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- `experiment.name`: Full title of the dataset
- `experiment.ID`: Internal ID of the dataset.
- `experiment.description`: Description of the dataset
- `experiment.troubled`: Did an automatic process within gemma or a curator mark the dataset as "troubled"
- `experiment.accession`: Accession ID of the dataset in the external database it was taken from
- `experiment.database`: The name of the database where the dataset was taken from
- `experiment.URI`: URI of the original database
- `experiment.sampleCount`: Number of samples in the dataset
- `experiment.batchEffectText`: A text field describing whether the dataset has batch effects
- `experiment.batchCorrected`: Whether batch correction has been performed on the dataset.
- `experiment.batchConfound`: 0 if batch info isn’t available, -1 if batch counfoud is detected, 1 if batch information is available and no batch confound found
- `experiment.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.
- `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- `geeq.qScore`: Data quality score given to the dataset by Gemma.
- `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- `taxon.name`: Name of the species
- `taxon.scientific`: Scientific name for the taxon
- `taxon.ID`: Internal identifier given to the species by Gemma
- `taxon.NCBI`: NCBI ID of the taxon
- `taxon.database.name`: Underlying database used in Gemma for the taxon
- `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon
get_platform_element_genes

Examples

head(get_platform_datasets("GPL1355"))

get_platform_element_genes

Retrieve the genes associated to a probe in a given platform

Description

Retrieve the genes associated to a probe in a given platform

Usage

get_platform_element_genes(
  platform,
  probe,
  offset = 0L,
  limit = 20L,
  raw =getOption("gemma.raw", FALSE),
  memoised =getOption("gemma.memoised", FALSE),
  file =getOption("gemma.file", NA_character_),
  overwrite =getOption("gemma.overwrite", FALSE)
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform</td>
<td>A platform numerical identifier or a platform short name</td>
</tr>
<tr>
<td>probe</td>
<td>A probe name or it's numerical identifier</td>
</tr>
<tr>
<td>offset</td>
<td>The offset of the first retrieved result</td>
</tr>
<tr>
<td>limit</td>
<td>Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.</td>
</tr>
<tr>
<td>raw</td>
<td>TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.</td>
</tr>
<tr>
<td>memoised</td>
<td>Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.</td>
</tr>
<tr>
<td>file</td>
<td>The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Whether or not to overwrite if a file exists at the specified filename.</td>
</tr>
</tbody>
</table>
Value

A data table with information about the queried gene(s) A list if \texttt{raw = TRUE}.

The fields of the output data.table are:

- \texttt{gene.symbol}: Symbol for the gene
- \texttt{gene.enssembl}: Ensembl ID for the gene
- \texttt{gene.NCBI}: NCBI id for the gene
- \texttt{gene.name}: Name of the gene
- \texttt{gene.aliases}: Gene aliases. Each row includes a vector
- \texttt{gene.MFX.rank}: Multifunctionality rank for the gene
- \texttt{taxon.name}: Name of the species
- \texttt{taxon.scientific}: Scientific name for the taxon
- \texttt{taxon.ID}: Internal identifier given to the species by Gemma
- \texttt{taxon.NCBI}: NCBI ID of the taxon
- \texttt{taxon.database.name}: Underlying database used in Gemma for the taxon
- \texttt{taxon.database.ID}: ID of the underlying database used in Gemma for the taxon

Examples

\begin{verbatim}
get_platform_element_genes("GPL1355", "AFFX_Rat_beta-actin_M.at")
\end{verbatim}

\begin{verbatim}
get_result_sets  Retrieve all result sets matching the provided criteria
\end{verbatim}

Description

Returns queried result set

Usage

\begin{verbatim}
get_result_sets(
    datasets = NA_character_,
    resultSets = NA_character_,
    filter = NA_character_,
    offset = 0,
    limit = 20,
    sort = "+id",
    raw = getOption("gemma.raw", FALSE),
    memoised = getOption("gemma.memoised", FALSE),
    file = getOption("gemma.file", NA_character_),
    overwrite = getOption("gemma.overwrite", FALSE)
)
\end{verbatim}
get_result_sets

Arguments

- **datasets**: A numerical dataset identifier or a dataset short name.
- **resultSets**: A resultSet identifier. Note that resultSet identifiers are not static and can change when Gemma re-runs analyses internally. When using these as inputs, try to make sure you access a currently existing resultSet ID by basing them on result sets returned for a particular resultSet or filter used in `get_result_sets`.
- **filter**: Filter results by matching expression. Use `filter_properties` function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as "=" or "in". (e.g. "taxon.commonName = human", "taxon.commonName in (human, mouse)", "id < 1000")
- **offset**: The offset of the first retrieved result.
- **limit**: Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the `totalElements` attribute in the output to compile all data if needed.
- **sort**: Order results by the given property and direction. The '+' sign indicate ascending order whereas the '-' indicate descending.
- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
- **file**: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.

Details

Output and usage of this function is mostly identical to `get_dataset_differential_expression_analyses`. The principal difference being the ability to restrict your result sets, being able to query across multiple datasets and being able to use the filter argument to search based on result set properties.

Value

A data table with information about the queried result sets. Note that this function does not return differential expression values themselves. Use `get_differential_expression_values` to get differential expression values.

- **result.ID**: Result set ID of the differential expression analysis. May represent multiple factors in a single model.
- **contrast.ID**: Id of the specific contrast factor. Together with the result.ID they uniquely represent a given contrast.
get_taxa

- `experiment.ID`: Id of the source experiment
- `factor.category`: Category for the contrast
- `factor.category.URI`: URI for the contrast category
- `factor.ID`: ID of the factor
- `baseline.factors`: Characteristics of the baseline. This field is a data.table
- `experimental.factors`: Characteristics of the experimental group. This field is a data.table
- `isSubset`: TRUE if the result set belong to a subset, FALSE if not. Subsets are created when performing differential expression to avoid unhelpful comparisons.
- `subsetFactor`: Characteristics of the subset. This field is a data.table

Examples

```r
get_result_sets(dataset = 1)
# get all contrasts comparing disease states. use filter_properties to see available options
get_result_sets(filter = "baselineGroup.characteristics.value = disease")
```

get_taxa

Get taxa

Description

Returns taxa and their versions used in Gemma

Usage

```r
get_taxa(memoised = getOption("gemma.memoised", FALSE))
```

Arguments

- `memoised`: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

Value

A data frame including the names, IDs and database information about the taxons

Examples

```r
get_taxa()
```
get_taxa_by_ids

Retrieve taxa by their identifiers

Description
Retrieve taxa by their identifiers

Usage

```r
get_taxa_by_ids(
taxa,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- **taxa**: Limits the result to entities with given identifiers. A vector of identifiers. Identifiers can be any of the following:
  - taxon ID
  - scientific name
  - common name Retrieval by ID is more efficient. Do not combine different identifiers in one query. For convenience, below is a list of officially supported taxa

<table>
<thead>
<tr>
<th>ID</th>
<th>Comm.name</th>
<th>Scient.name</th>
<th>NcbiID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>human</td>
<td>Homo sapiens</td>
<td>9606</td>
</tr>
<tr>
<td>2</td>
<td>mouse</td>
<td>Mus musculus</td>
<td>10090</td>
</tr>
<tr>
<td>3</td>
<td>rat</td>
<td>Rattus norvegicus</td>
<td>10116</td>
</tr>
<tr>
<td>11</td>
<td>yeast</td>
<td>Saccharomyces cerevisiae</td>
<td>4932</td>
</tr>
<tr>
<td>12</td>
<td>zebrafish</td>
<td>Danio rerio</td>
<td>7955</td>
</tr>
<tr>
<td>13</td>
<td>fly</td>
<td>Drosophila melanogaster</td>
<td>7227</td>
</tr>
<tr>
<td>14</td>
<td>worm</td>
<td>Caenorhabditis elegans</td>
<td>6239</td>
</tr>
</tbody>
</table>

- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.

- **file**: The name of a file to save the results to, or NULL to not write results to a file. If `raw == TRUE`, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.
get_taxon_datasets

Description

This function is deprecated in favor of get_datasets.

Usage

get_taxon_datasets(
  taxon,
  offset = 0L,
  limit = 20,
  sort = "+id",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE),
  ...
)

Arguments

taxon can either be Taxon ID, Taxon NCBI ID, or one of its string identifiers: scientific name, common name. It is recommended to use Taxon ID for efficiency. Please note, that not all taxa have all the possible identifiers available. Use the get_taxa_by_ids function to retrieve the necessary information. For convenience, below is a list of officially supported taxa:

<table>
<thead>
<tr>
<th>ID</th>
<th>Comm.name</th>
<th>Scient.name</th>
<th>NcbiID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>human</td>
<td>Homo sapiens</td>
<td>9606</td>
</tr>
<tr>
<td>2</td>
<td>mouse</td>
<td>Mus musculus</td>
<td>10090</td>
</tr>
<tr>
<td>3</td>
<td>rat</td>
<td>Rattus norvegicus</td>
<td>10116</td>
</tr>
<tr>
<td>11</td>
<td>yeast</td>
<td>Saccharomyces cerevisiae</td>
<td>4932</td>
</tr>
<tr>
<td>12</td>
<td>zebrafish</td>
<td>Danio rerio</td>
<td>7955</td>
</tr>
<tr>
<td>13</td>
<td>fly</td>
<td>Drosophila melanogaster</td>
<td>7227</td>
</tr>
<tr>
<td>14</td>
<td>worm</td>
<td>Caenorhabditis elegans</td>
<td>6239</td>
</tr>
</tbody>
</table>

offset The offset of the first retrieved result.

Value

A data table with the queried taxa’s details.

Examples

gemma.R:::get_taxa_by_ids(c("mouse", "human"))
**get_taxon_datasets**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Optional, defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.</td>
</tr>
<tr>
<td>sort</td>
<td>Order results by the given property and direction. The ‘+’ sign indicates ascending order whereas the ‘-’ indicate descending.</td>
</tr>
<tr>
<td>raw</td>
<td>TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.</td>
</tr>
<tr>
<td>memoised</td>
<td>Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.</td>
</tr>
<tr>
<td>file</td>
<td>The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be a JSON file. Otherwise, it will be a RDS file.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Whether or not to overwrite if a file exists at the specified filename.</td>
</tr>
</tbody>
</table>

**Value**

A data table with information about the queried dataset(s). A list if raw = TRUE. Returns an empty list if no datasets matched.

The fields of the output data.table are:

- **experiment.shortName**: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- **experiment.name**: Full title of the dataset
- **experiment.ID**: Internal ID of the dataset.
- **experiment.description**: Description of the dataset
- **experiment.troubled**: Did an automatic process within gemma or a curator mark the dataset as "troubled"
- **experiment.accession**: Accession ID of the dataset in the external database it was taken from
- **experiment.database**: The name of the database where the dataset was taken from
- **experiment.URI**: URI of the original database
- **experiment.sampleCount**: Number of samples in the dataset
- **experiment.batchEffectText**: A text field describing whether the dataset has batch effects
- **experiment.batchCorrected**: Whether batch correction has been performed on the dataset.
- **experiment.batchConfound**: 0 if batch info isn’t available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
- **experiment.batchEffect**: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.
• `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches

• `geeq.qScore`: Data quality score given to the dataset by Gemma.

• `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design

• `taxon.name`: Name of the species

• `taxon.scientific`: Scientific name for the taxon

• `taxon.ID`: Internal identifier given to the species by Gemma

• `taxon.NCBI`: NCBI ID of the taxon

• `taxon.database.name`: Underlying database used in Gemma for the taxon

• `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon

**Examples**

```r
get_taxon_datasets("human")
```

**isEMPTY**

*Check for empty arguments*

**Description**

Gemma functions accept typed NAs instead of simple NULLS. I believe this was done as a non-standard-in-R way of specifying data types for the users. Usually this is fine but it makes checking for emptiness a bit annoying since a user can still provide NULLs to make them empty

**Usage**

```r
isEmpty(x)
```

**Arguments**

- `x`  
  A parameter that can be NA or NULL when empty
make_design

Make simplified design frames

Description

Using on the output of `get_dataset_samples`, this function creates a simplified design table, granting one column to each experimental variable.

Usage

```r
make_design(samples, metaType = "text")
```

Arguments

- `samples`: An output from `get_dataset_samples`. The output should not be raw.
- `metaType`: Type of metadata to include in the output. "text", "uri" or "both".

Value

A data.frame including the design table for the dataset.

Examples

```r
samples <- get_dataset_samples('GSE46416')
make_design(samples)
```

memoise

Memoise doc

Description

Memoise doc

Arguments

- `memoised`: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
nullCheck  
*Avoid NULLS as data.table columns*

**Description**

Avoid NULLS as data.table columns

**Usage**

`nullCheck(x, natype = NA)`

**Arguments**

- **x**: A value that might be null
- **natype**: What to fill in when data is unavailable

**Value**

- x as is or natype

---

processAnnotations  
*Processes JSON as annotations*

**Description**

Processes JSON as annotations

**Usage**

`processAnnotations(d)`

**Arguments**

- **d**: The JSON to process

**Value**

A data table with information about the annotations of the queried dataset. A list if `raw = TRUE`. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- `class.name`: Name of the annotation class (e.g. organism part)
- `class.URI`: URI for the annotation class
- `term.name`: Name of the annotation term (e.g. lung)
- `term.URI`: URI for the annotation term
- `object.class`: Class of object that the term originated from.
processCharacteristicValueObject

*Processes JSON as a factor*

**Description**
Processes JSON as a factor

**Usage**
```r
processCharacteristicValueObject(d)
```

**Arguments**
- `d` The JSON to process

**Value**
A processed data.table

---

processDatasetResultSets

*Processes JSON as a datasets result set*

**Description**
Processes JSON as a datasets result set

**Usage**
```r
processDatasetResultSets(d)
```

**Arguments**
- `d` The JSON to process

**Value**
A data table with the queried datasets’ resultSet ID(s). A list if `raw = TRUE`. Use `get_differential_expression_values` to get differential expression values (see examples). Use `get_dataset_differential_expression_analyses` to get more detailed information about a result set.

The fields of the output data.table are:

- `resultSet.id`: Internal ID given to the result set. Can be used to access the results using `get_differential_expression_values`
• factor.category: What is the category splitting the experimental groups in the result set (e.g. disease)
• factor.levels: What are the conditions that are compared in the result set (e.g control, bipolar disorder)

processDatasets

Processes JSON as a vector of datasets

Description

Processes JSON as a vector of datasets

Usage

processDatasets(d)

Arguments

d The JSON to process

Value

A data table with information about the queried dataset(s). A list if raw = TRUE. Returns an empty list if no datasets matched.

The fields of the output data.table are:

• experiment.shortName: Shortname given to the dataset within Gemma. Often corresponds to accession ID
• experiment.name: Full title of the dataset
• experiment.ID: Internal ID of the dataset.
• experiment.description: Description of the dataset
• experiment.troubled: Did an automatic process within gemma or a curator mark the dataset as "troubled"
• experiment.accession: Accession ID of the dataset in the external database it was taken from
• experiment.database: The name of the database where the dataset was taken from
• experiment.URI: URI of the original database
• experiment.sampleCount: Number of samples in the dataset
• experiment.batchEffectText: A text field describing whether the dataset has batch effects
• experiment.batchCorrected: Whether batch correction has been performed on the dataset.
• experiment.batchConfound: 0 if batch info isn’t available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
processDEA

**Processes JSON as a differential expression analysis**

### Description
Processes JSON as a differential expression analysis

### Usage
processDEA(d)

### Arguments
- **d**
  - The JSON to process

### Value
A data table with information about the differential expression analysis of the queried dataset. Note that this function does not return differential expression values themselves. Use `get_differential_expression_values` to get differential expression values (see examples).

The fields of the output data.table are:

- **result.ID**: Result set ID of the differential expression analysis. May represent multiple factors in a single model.
- **contrast.ID**: Id of the specific contrast factor. Together with the result.ID they uniquely represent a given contrast.
- **experiment.ID**: Id of the source experiment
- **factor.category**: Category for the contrast

- **experiment.batchEffect**: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.
- **experiment.rawData**: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- **geeq.qScore**: Data quality score given to the dataset by Gemma.
- **geeq.sScore**: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- **taxon.name**: Name of the species
- **taxon.scientific**: Scientific name for the taxon
- **taxon.ID**: Internal identifier given to the species by Gemma
- **taxon.NCBI**: NCBI ID of the taxon
- **taxon.database.name**: Underlying database used in Gemma for the taxon
- **taxon.database.ID**: ID of the underlying database used in Gemma for the taxon
- `factor.category.URI`: URI for the contrast category
- `factor.ID`: ID of the factor
- `baseline.factors`: Characteristics of the baseline. This field is a data.table
- `experimental.factors`: Characteristics of the experimental group. This field is a data.table
- `isSubset`: TRUE if the result set belong to a subset, FALSE if not. Subsets are created when performing differential expression to avoid unhelpful comparisons.
- `subsetFactor`: Characteristics of the subset. This field is a data.table
- `probes.analyzed`: Number of probesets represented in the contrast
- `genes.analyzed`: Number of genes represented in the contrast

### processDEcontrasts

*Replaces factor ids by the factors strings in DE table columns*

**Description**

Replaces factor ids by the factors strings in DE table columns

**Usage**

```r
processDEcontrasts(rs, rsID)
```

**Arguments**

- `rs` : The resultSet matrix to process

**Value**

A processed matrix

### processDEMatrix

*Processes differential expression matrix*

**Description**

Processes differential expression matrix

**Usage**

```r
processDEMatrix(m)
```

**Arguments**

- `m` : The differential expression matrix to process

**Value**

A processed matrix
processDesignMatrix  Processes design matrix

Description
Processes design matrix

Usage
processDesignMatrix(m)

Arguments

m  The design matrix to process

Value
A processed matrix

processDifferentialExpressionAnalysisResultSetValueObject  Process JSON of a result set

Description
Process JSON of a result set

Usage
processDifferentialExpressionAnalysisResultSetValueObject(d)

Value
A data table with information about the queried result sets. Note that this function does not return
differential expression values themselves. Use get_differential_expression_values to get
differential expression values

- result.ID: Result set ID of the differential expression analysis. May represent multiple fac-
tors in a single model.
- contrast.ID: Id of the specific contrast factor. Together with the result.ID they uniquely
  represent a given contrast.
- experiment.ID: Id of the source experiment
- factor.category: Category for the contrast
- factor.category.URI: URI for the contrast category
processElements

Processes JSON as a vector of elements

Description

Processes JSON as a vector of elements

Usage

processElements(d)

Arguments

d The JSON to process

Value

A data table with information about the probes representing a gene across all platforms. A list if raw = TRUE. A 404 error if the given identifier does not map to any genes.

The fields of the output data.table are:

- element.name: Name of the element. Typically the probeset name
- element.description: A free text field providing optional information about the element
- platform.shortName: Shortname of the platform given by Gemma. Typically the GPL identifier.
- platform.name: Full name of the platform
- platform.ID: Id number of the platform given by Gemma
- platform.type: Type of the platform.
- platform.description: Free text field describing the platform.
- platform.troubled: Whether the platform is marked as troubled by a Gemma curator.
- taxon.name: Name of the species platform was made for
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon
processExpressionMatrix

Processes expression matrix

Description
Processes expression matrix

Usage
processExpressionMatrix(m)

Arguments
m The expression matrix to process

Value
A processed matrix

processFile Processes a response as a gzip file

Description
Processes a response as a gzip file

Usage
processFile(content)

Arguments
content The content from an http_get request

Value
A processed data.table
processGemmaArray  

**Processes JSON as an array**

**Description**
Processes JSON as an array

**Usage**
processGemmaArray(d)

**Arguments**
- **d**  
The JSON to process

**Value**
A data table with information about the probes representing the gene across different platforms.

---

processGeneLocation  

**Processes JSON as a vector of gene locations**

**Description**
Processes JSON as a vector of gene locations

**Usage**
processGeneLocation(d)

**Arguments**
- **d**  
The JSON to process

**Value**
A data table with information about the physical location of the queried gene. A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- **chromosome**: Name of the chromosome the gene is located
- **strand**: Which strand the gene is located
- **nucleotide**: Nucleotide number for the gene
- **length**: Gene length
- **taxon.name**: Name of the taxon
• taxon.scientific: Scientific name for the taxon
• taxon.ID: Internal ID for the taxon given by Gemma
• taxon.NCBI: NCBI ID for the taxon
• taxon.database.name: Name of the database used in Gemma for the taxon

processGenes  Processes JSON as a vector of genes

Description
Processes JSON as a vector of genes

Usage
processGenes(d)

Arguments
d  The JSON to process

Value
A data table with information about the queried gene(s) A list if raw = TRUE.
The fields of the output data.table are:
• gene.symbol: Symbol for the gene
• gene.ensembl: Ensembl ID for the gene
• gene.NCBI: NCBI id for the gene
• gene.name: Name of the gene
• gene.aliases: Gene aliases. Each row includes a vector
• gene.MFX.rank: Multifunctionality rank for the gene
• taxon.name: Name of the species
• taxon.scientific: Scientific name for the taxon
• taxon.ID: Internal identifier given to the species by Gemma
• taxon.NCBI: NCBI ID of the taxon
• taxon.database.name: Underlying database used in Gemma for the taxon
• taxon.database.ID: ID of the underlying database used in Gemma for the taxon
processGO

*Processes JSON as GO terms*

**Description**

Processes JSON as GO terms

**Usage**

```r
processGO(d)
```

**Arguments**

- `d`
  
  The JSON to process

**Value**

A data table with information about the GO terms assigned to the queried gene. A list if `raw = TRUE`. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- `term.name`: Name of the term
- `term.ID`: ID of the term
- `term.URI`: URI of the term

processPlatforms

*Processes JSON as a vector of platforms*

**Description**

Processes JSON as a vector of platforms

**Usage**

```r
processPlatforms(d)
```

**Arguments**

- `d`
  
  The JSON to process
Value

A data table with information about the platform(s). A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- platform.ID: Internal identifier of the platform
- platform.shortName: Shortname of the platform.
- platform.name: Full name of the platform.
- platform.description: Free text description of the platform.
- platform.troubled: Whether or not the platform was marked "troubled" by a Gemma process or a curator.
- platform.experimentCount: Number of experiments using the platform within Gemma.
- platform.type: Technology type for the platform.
- taxon.name: Name of the species platform was made for.
- taxon.scientific: Scientific name for the taxon.
- taxon.ID: Internal identifier given to the species by Gemma.
- taxon.NCBI: NCBI ID of the taxon.
- taxon.database.name: Underlying database used in Gemma for the taxon.
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon.

Usage

processQuantitationTypeValueObject(d)

Arguments

d The JSON to process.
Value

A data.table containing the quantitation types

The fields of the output data.table are:

- **id**: If of the quantitation type. Any raw quantitation type can be accessed by `get_dataset_raw_expression` function using this id.
- **name**: Name of the quantitation type
- **description**: Description of the quantitation type
- **type**: Type of the quantitation type. Either raw or processed. Each dataset will have one processed quantitation type which is the data returned using `get_dataset_processed_expression`
- **ratio**: Whether or not the quantitation type is a ratio of multiple quantitation types. Typically TRUE for processed TWOCOLOR quantitation type.
- **preferred**: The preferred raw quantitation type. This version is used in generation of the processed data within gemma.
- **recomputed**: If TRUE this quantitation type is generated by recomputing raw data files Gemma had access to.

---

**processResultSetFactors**

*Processes JSON as a result set*

---

Description

Processes JSON as a result set

Usage

`processResultSetFactors(d)`

Arguments

- **d**: The JSON to process

Value

A processed data.table
processSamples

Processes JSON as a vector of samples

**Description**

Processes JSON as a vector of samples

**Usage**

processSamples(d)

**Arguments**

d  The JSON to process

**Value**

A data table with information about the samples of the queried dataset. A list if raw = TRUE. A 404 error if the given identifier does not map to any object.

The fields of the output data.table are:

- **sample.name**: Internal name given to the sample.
- **sample.ID**: Internal ID of the sample
- **sample.description**: Free text description of the sample
- **sample.outlier**: Whether or not the sample is marked as an outlier
- **sample.accession**: Accession ID of the sample in it’s original database
- **sample.database**: Database of origin for the sample
- **sample.characteristics**: Characteristics of the sample. This field is a data table
- **sample.factorValues**: Experimental factor values of the sample. This field is a data table

processSearchAnnotations

Processes JSON as an annotation

**Description**

Processes JSON as an annotation

**Usage**

processSearchAnnotations(d)
processTaxon

Arguments

   d       The JSON to process

Value

A data table with annotations (annotation search result value objects) matching the given identifiers. A list if raw = TRUE. A 400 error if required parameters are missing.

The fields of the output data.table are:

   • category.name: Category that the annotation belongs to
   • category.URI: URI for the category.name
   • value.name: Annotation term
   • value.URI: URI for the value.name

Description

Processes JSON as a vector of taxa

Usage

   processTaxon(d)

Arguments

   d       The JSON to process

Value

A processed data.table

   • taxon.name: Name of the species
   • taxon.scientific: Scientific name for the taxon
   • taxon.ID: Internal identifier given to the species by Gemma
   • taxon.NCBI: NCBI ID of the taxon
   • taxon.database.name: Underlying database used in Gemma for the taxon
   • taxon.database.ID: ID of the underlying database used in Gemma for the taxon
**process_search**

*Returns the ids of the found results*

**Description**

Returns the ids of the found results

**Usage**

`process_search(d)`

**Value**

A data.table or a list of resultObjects

---

**search_annotations**

*Search for annotation tags*

**Description**

Search for annotation tags

**Usage**

```r
search_annotations(
  query,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

**Arguments**

- **query**
  
  The search query. Queries can include plain text or ontology terms. They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").

- **raw**
  
  TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

- **memoised**
  
  Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing `options(gemma.memoised = TRUE)` will ensure that the cache is always used. Use `forget_gemma_memoised` to clear the cache.
file

The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

overwrite

Whether or not to overwrite if a file exists at the specified filename.

Value

A data table with annotations (annotation search result value objects) matching the given identifiers. A list if raw = TRUE. A 400 error if required parameters are missing.

The fields of the output data.table are:

- category.name: Category that the annotation belongs to
- category.URI: URI for the category.name
- value.name: Annotation term
- value.URI: URI for the value.name

Examples

search_annotations("traumatic")

search_datasets

Retrieve datasets associated to an annotation tags search

Description

This function is deprecated in favor of get_datasets

Usage

search_datasets(
    query,
    taxon = NA_character_,
    offset = 0L,
    limit = 20L,
    sort = "+id",
    raw = getOption("gemma.raw", FALSE),
    memoised = getOption("gemma.memoised", FALSE),
    file = getOption("gemma.file", NA_character_),
    overwrite = getOption("gemma.overwrite", FALSE),
    attributes = getOption("gemma.attributes", TRUE),
    ...
)
Arguments

query  The search query. Either plain text (‘traumatic’), or an ontology term URI ('http://purl.obolibrary.org/obo/UBERON_0002048’). Datasets that contain the given string in their short or full name will also be matched. Can be multiple identifiers separated by commas.

taxon  Can either be Taxon ID, Taxon NCBI ID, or one of its string identifiers: scientific name, common name. It is recommended to use Taxon ID for efficiency. Please note, that not all taxa have all the possible identifiers available. Use the get_taxa_by_ids function to retrieve the necessary information. For convenience, below is a list of officially supported taxa:

<table>
<thead>
<tr>
<th>ID</th>
<th>Comm.name</th>
<th>Scient.name</th>
<th>NcbiID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>human</td>
<td>Homo sapiens</td>
<td>9606</td>
</tr>
<tr>
<td>2</td>
<td>mouse</td>
<td>Mus musculus</td>
<td>10090</td>
</tr>
<tr>
<td>3</td>
<td>rat</td>
<td>Rattus norvegicus</td>
<td>10116</td>
</tr>
<tr>
<td>11</td>
<td>yeast</td>
<td>Saccharomyces cerevisiae</td>
<td>4932</td>
</tr>
<tr>
<td>12</td>
<td>zebrafish</td>
<td>Danio rerio</td>
<td>7955</td>
</tr>
<tr>
<td>13</td>
<td>fly</td>
<td>Drosophila melanogaster</td>
<td>7227</td>
</tr>
<tr>
<td>14</td>
<td>worm</td>
<td>Caenorhabditis elegans</td>
<td>6239</td>
</tr>
</tbody>
</table>

offset  The offset of the first retrieved result.

limit  Optional, defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements attribute in the output to compile all data if needed.

sort  Order results by the given property and direction. The ‘+’ sign indicate ascending order whereas the ‘-’ indicate descending.

raw  TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

memoised  Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

file  The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be a JSON file. Otherwise, it will be a RDS file.

overwrite  Whether or not to overwrite if a file exists at the specified filename.

attributes  If TRUE additional information from the call will be added into the output object’s attributes such as offset and available elements.

Value

A data table with information about the queried dataset(s). A list if raw = TRUE. Returns an empty list if no datasets matched.

The fields of the output data.table are:
• experiment.shortName: Shortname given to the dataset within Gemma. Often corresponds to accession ID
• experiment.name: Full title of the dataset
• experiment.ID: Internal ID of the dataset.
• experiment.description: Description of the dataset
• experiment.troubled: Did an automatic process within gemma or a curator mark the dataset as "troubled"
• experiment.accession: Accession ID of the dataset in the external database it was taken from
• experiment.database: The name of the database where the dataset was taken from
• experiment.URI: URI of the original database
• experiment.sampleCount: Number of samples in the dataset
• experiment.batchEffectText: A text field describing whether the dataset has batch effects
• experiment.batchCorrected: Whether batch correction has been performed on the dataset.
• experiment.batchConfound: 0 if batch info isn’t available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
• experiment.batchEffect: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 otherwise and when there is no batch information is available or when the data is confounded with batches.
• experiment.rawData: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
• geeq.qScore: Data quality score given to the dataset by Gemma.
• geeq.sScore: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
• taxon.name: Name of the species
• taxon.scientific: Scientific name for the taxon
• taxon.ID: Internal identifier given to the species by Gemma
• taxon.NCBI: NCBI ID of the taxon
• taxon.database.name: Underlying database used in Gemma for the taxon
• taxon.database.ID: ID of the underlying database used in Gemma for the taxon

Examples

search_datasets("bipolar", taxon = "human")
search_gemma

Search everything in Gemma

Description
Search everything in Gemma

Usage

```r
search_gemma(
  query,
  taxon = NA_character_,
  platform = NA_character_,
  limit = 100,
  resultType = "experiment",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

Arguments

- **query**: The search query. Queries can include plain text or ontology terms. They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~")

- **taxon**: A numerical taxon identifier or an ncbi taxon identifier or a taxon identifier that matches either its scientific or common name

- **platform**: A platform taxon identifier or a platform short name

- **limit**: Defaults to 100 with a maximum value of 2000. Limits the number of returned results. Note that this function does not support pagination.

- **resultType**: The kind of results that should be included in the output. Can be experiment, gene, platform or a long object type name, documented in the API documentation.

- **raw**: TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.

- **memoised**: Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing options(gemma.memoised = TRUE) will ensure that the cache is always used. Use forget_gemma_memoised to clear the cache.

- **file**: The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.

- **overwrite**: Whether or not to overwrite if a file exists at the specified filename.
Value

If `raw = FALSE` and `resultType` is `experiment`, `gene` or `platform`, a `data.table` containing the search results. If it is any other type, a list of results. A list with additional details about the search if `raw = TRUE`.

Examples

```r
search_gemma("bipolar")
```

---

**setGemmaPath** *Set gemma path*

**Description**

Set gemma path

**Usage**

```r
setGemmaPath(path)
```

**Arguments**

- `path` "dev", "prod" or a link to use to access gemma API

**Value**

Link to Gemma API

---

**set_gemma_user** *Authentication by user name*

**Description**

Allows the user to access information that requires logging in to Gemma. To log out, run `set_gemma_user` without specifying the username or password.

**Usage**

```r
set_gemma_user(username = NULL, password = NULL)
```

**Arguments**

- `username` Your username (or empty, if logging out)
- `password` Your password (or empty, if logging out)

**Value**

TRUE if authentication is successful, FALSE if not
subset_factorValues  

Get a subset of an array of factorValues

Description

Get a subset of an array of factorValues

Usage

subset_factorValues(
  factorValues,
  factorValue = NULL,
  differential_expressions = NULL,
  resultSet = NULL,
  contrast = NULL
)

Arguments

  factorValue       unimplemented
  differential_expressions

Value

  a boolean vector, samples representing the resultSet and/or the contrast are set to TRUE

update_result  

Update result

Description

Re-runs the function used to create a gemma.R output to update the data at hand. Useful if you have a reason to believe parts of the data has changed since your last accession and you wish to update while decoupling the update process from your original code used to generate the data.

Usage

update_result(query)

Arguments

  query        Output from a gemma.R function
validateBoolean

Details

Note that if you have used the file and overwrite arguments with the original call, this will also repeat to regenerate the file based on your initial preference.

Examples

```r
annots <- get_dataset_annotations(1)
# wait for a couple of years..
# wonder if the results are the same
updated_annots <- update_result(annots)

# also works with outputs of get_all_pages
platforms <- get_all_pages(get_platforms_by_ids())
updated_platforms <- update_result(platforms)
```

validateBoolean Validate a boolean value

Description

Validate a boolean value

Usage

```r
validateBoolean(name, ...)
```

Arguments

- `name` The variable name
- `...` Any boolean types

Value

The validated boolean as a character string (true or false), or stop with an error message.
validateID

**validateID**

Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)

**Description**

Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)

**Usage**

```
validateID(name, ...)
```

**Arguments**

- `name` The variable name
- `...` Any identifiers

**Value**

The validated identifiers, or stop with an error message

validateLimit

**validateLimit**

Validate a limit value

**Description**

Validate a limit value

**Usage**

```
validateLimit(name, ...)
```

**Arguments**

- `name` The variable name
- `...` Any possible integers

**Value**

The validated integers, or stop with an error message
validateOptionalID | Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)

**Description**

Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)

**Usage**

validateOptionalID(name, ...)

**Arguments**

<table>
<thead>
<tr>
<th>name</th>
<th>The variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Any identifiers</td>
</tr>
</tbody>
</table>

**Value**

The validated identifiers, or stop with an error message

validateOptionalQuery | Validate an optional query

**Description**

Validate an optional query

**Usage**

validateOptionalQuery(name, ...)

**Arguments**

<table>
<thead>
<tr>
<th>name</th>
<th>The variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Any queries</td>
</tr>
</tbody>
</table>

**Value**

The validated queries
validateOptionalTaxon

Validate a taxon using the acceptable taxa entries

Description

Validate a taxon using the acceptable taxa entries

Usage

validateOptionalTaxon(name, ...)

Arguments

name The variable name
...
Any taxa to validate

Value

The validated taxon, or stop with an error message

validatePositiveInteger

Validate a non-negative integer value

Description

Validate a non-negative integer value

Usage

validatePositiveInteger(name, ...)

Arguments

name The variable name
...
Any possible integers

Value

The validated integers, or stop with an error message
validateQuery 

**Validate a query**

**Description**

Validate a query

**Usage**

validateQuery(name, ...)

**Arguments**

name 

The variable name

... 

Any queries

**Value**

The validated queries, or stop with an error message

---

validateResultType 

**Validate result types**

**Description**

Validate result types

**Usage**

validateResultType(name, ...)

**Arguments**

name 

The variable name

... 

result types

**Value**

Validated result types. Either returned as they are or they will be replaced from human readable variants
validateSingleID

Validate a single identifier (i.e. gene ID, platform ID, etc.)

Description

Validate a single identifier (i.e. gene ID, platform ID, etc.)

Usage

validateSingleID(name, ...)

Arguments

- name: The variable name
- ...: An identifier

Value

The validated identifier, or stop with an error message

validateSort

Validate a sort argument

Description

Validate a sort argument

Usage

validateSort(name, ...)

Arguments

- name: The variable name
- ...: Any sort arguments

Value

The validated sort arguments, or stop with an error message
validateTaxa

Validates taxa using the acceptable taxa entries

Usage

validateTaxa(name, ...)

Arguments

name
The variable name

... Any taxa to validate

Value

The validated taxa, or stop with an error message

validateTaxon

Validates a taxon using the acceptable taxa entries

Usage

validateTaxon(name, ...)

Arguments

name
The variable name

... Any taxa to validate

Value

The validated taxon, or stop with an error message
Index

* **dataset**
  - get_dataset_annotations, 17
  - get_dataset_design, 18
  - get_dataset_differential_expression_analyses, 19
  - get_dataset_expression_for_genes, 21
  - get_dataset_object, 22
  - get_dataset_platforms, 24
  - get_dataset_processed_expression, 25
  - get_dataset_quantitation_types, 26
  - get_dataset_raw_expression, 28
  - get_dataset_samples, 29
  - get_datasets, 12
  - get_datasets_by_ids, 15
  - get_differential_expression_values, 30

* **gene**
  - get_gene_go_terms, 32
  - get_gene_locations, 33
  - get_gene_probes, 35
  - get_genes, 31

* **internal**
  - .getResultSets, 4
  - accessField, 5
  - blank_processor, 5
  - checkBounds, 6
  - encode, 6
  - gemmaCache, 9
  - gemmaPath, 9
  - get_dataset_expression, 20
  - get_taxa_by_ids, 45
  - get_taxon_datasets, 46
  - isEmpty, 48
  - memoise, 49
  - nullCheck, 50
  - process_search, 65
  - processAnnotations, 50
  - processCharacteristicValueObject, 51
  - processDatasetResultSets, 51
  - processDatasets, 52
  - processDEA, 53
  - processDEcontrasts, 54
  - processDEMatrix, 54
  - processDesignMatrix, 55
  - processDifferentialExpressionAnalysisResultSetValueObject, 55
  - processElements, 56
  - processExpressionMatrix, 57
  - processFile, 57
  - processGemmaArray, 58
  - processGeneLocation, 58
  - processGenes, 59
  - processGO, 60
  - processPlatforms, 60
  - processQuantitationTypeValueObject, 61
  - processResultSetFactors, 62
  - processSamples, 63
  - processSearchAnnotations, 63
  - processTaxon, 64
  - search_datasets, 66
  - setGemmaPath, 70
  - subset_factorValues, 71
  - validateBoolean, 72
  - validateID, 73
  - validateLimit, 73
  - validateOptionalID, 74
  - validateOptionalQuery, 74
  - validateOptionalTaxon, 75
  - validatePositiveInteger, 75
  - validateQuery, 76
  - validateResultType, 76
  - validateSingleID, 77
  - validateSort, 77
  - validateTaxa, 78
validateTaxon, 78
* misc
  filter_properties, 7
  forget_gemma_memoised, 7
  gemma_call, 9
  gemma_kable, 10
  gemma_memoise, 10
  get_all_pages, 11
  get_child_terms, 12
  get_result_sets, 42
  get_taxa, 44
  make_design, 49
  search_annotations, 65
  search_gemma, 69
  set_gemma_user, 70
  update_result, 71
* platform
  get_platform_annotations, 38
  get_platform_datasets, 39
  get_platform_element_genes, 41
  get_platforms_by_ids, 36
  .getResultSets, 4
accessField, 5
attribute, 13, 15, 35, 37, 39, 41, 43, 47, 67
blank_processor, 5
checkBounds, 6
encode, 6
ExpressionSet, 24
filter_properties, 7, 13, 15, 37, 43
gemma.R, 8
gemma.R-package (gemma.R), 8
gemma_call, 9
gemma_kable, 10
gemma_memoise, 10
gemmaCache, 9
gemmaPath, 9
get_all_pages, 11
get_child_terms, 12
get_dataset_annotations, 17
get_dataset_design, 18
get_dataset_differential_expression_analyses, 19, 30, 43, 51
get_dataset_differential_expression_analyses(), 30
get_dataset_expression, 20
get_dataset_expression_for_genes, 21
get_dataset_object, 8, 22
get_dataset_platforms, 24
get_dataset_processed_expression, 20, 25, 27, 62
get_dataset_quantitation_types, 26, 28
get_dataset_raw_expression, 27, 28, 62
get_dataset_samples, 29, 49
get_datasets, 7, 12, 46, 66
get_datasets_by_ids, 15
get_differential_expression_values, 8, 20, 30, 43, 51, 53, 55
gene_go_terms, 32
gene_locations, 33
gene_probes, 35
genes, 31
get_platform_annotations, 8, 38
get_platform_datasets, 39
get_platform_element_genes, 41
get_platforms_by_ids, 7, 36
get_result_sets, 42, 43
get_taxa, 44
get_taxa_by_ids, 45, 46, 67
get_taxon_datasets, 46
isEmpty, 48
kable, 10
make_design, 49
memoise, 49
nullCheck, 50
process_search, 65
processAnnotations, 50
processCharacteristicValueObject, 51
processDatasetResultSets, 51
processDatasets, 52
processDEA, 53
processDEcontrasts, 54
processDEMatrix, 54
processDesignMatrix, 55
processDifferentialExpressionAnalysisResultSetValueObject,
INDEX

processElements, 56
processExpressionMatrix, 57
processFile, 57
processGemmaArray, 58
processGeneLocation, 58
processGenes, 59
processGO, 60
processPlatforms, 60
processQuantitationTypeValueObject, 61
processResultSetFactors, 62
processSamples, 63
processSearchAnnotations, 63
processTaxon, 64

search_annotations, 65
search_datasets, 66
search_gemma, 69
set_gemma_user, 70
setGemmaPath, 70
subset_factorValues, 71
SummarizedExperiment, 24

update_result, 71

validateBoolean, 72
validateID, 73
validateLimit, 73
validateOptionalID, 74
validateOptionalQuery, 74
validateOptionalTaxon, 75
validatePositiveInteger, 75
validateQuery, 76
validateResultType, 76
validateSingleID, 77
validateSort, 77
validateTaxa, 78
validateTaxon, 78