Package ‘gemma.R’

January 16, 2024

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

Version 2.0.0

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.

     https://github.com/PavlidisLab/gemma.R

License  Apache License (>= 2)

Encoding  UTF-8

Roxygen  list(markdown = TRUE)

RoxygenNote  7.2.3

BugReports  https://github.com/PavlidisLab/gemma.R/issues

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

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

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

git_branch  RELEASE_3_18
git_last_commit 1f19014

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-01-16

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>

R topics documented:

- .getResultSetFactors ........................... 4
- .getResultSets ................................. 5
- accessField .................................... 5
- blank_processor ............................... 6
- checkBounds .................................... 6
- encode ........................................ 7
- filter_properties ............................. 7
- forget_gemma_memoised ...................... 8
- gemma.R ....................................... 8
- gemmaCache .................................... 9
- gemmaPath ..................................... 9
- gemma_call ................................... 10
- get_all_pages ................................ 10
- get_datasets ................................ 11
- get_datasets_by_ids ......................... 13
- get_dataset_annotations .................... 16
- get_dataset_design ......................... 17
- get_dataset_differential_expression_analyses .... 18
- get_dataset_expression ..................... 19
- get_dataset_expression_for_genes .......... 20
- get_dataset_object .......................... 21
- get_dataset_platforms ...................... 23
- get_dataset_processed_expression .......... 24
- get_dataset_quantitation_types ........... 25
- get_dataset_raw_expression ................ 26
- get_dataset_samples ........................ 27
- get_differential_expression_values ....... 28
- get_genes .................................... 29
- get_gene_go_terms ............................ 31
- get_gene_locations ........................... 32
- get_gene_probes ............................... 33
- get_platforms_by_ids ....................... 34
R topics documented:

get_platform_annotations ............................................. 36
get_platform_datasets .................................................... 37
get_platform_element_genes ............................................. 39
get_taxa ........................................................................ 40
get_taxa_by_ids ............................................................ 41
get_taxon_datasets .......................................................... 42
make_design ..................................................................... 44
nullCheck ........................................................................ 45
processAnnotations ........................................................... 45
processCharacteristicBasicValueObject .................................. 46
processDatasetResultSets .................................................... 46
processDatasets .................................................................. 47
processDEA ....................................................................... 48
processDEcontrasts ............................................................ 49
processDEMatrix ............................................................... 50
processDesignMatrix .......................................................... 50
processElements ................................................................ 51
processExpressionMatrix .................................................... 52
processFile ........................................................................ 52
processGemmaArray ........................................................... 53
processGemmaFactor .......................................................... 53
processGeneLocation ........................................................... 54
processGenes .................................................................... 54
processGO ........................................................................ 55
processPlatforms ............................................................... 56
processQuantitationTypeValueObject ..................................... 57
processResultSetFactors ....................................................... 57
processSamples .................................................................. 58
processSearchAnnotations ..................................................... 59
processTaxon ..................................................................... 59
process_search ................................................................. 60
search_annotations ............................................................. 60
search_datasets ................................................................. 61
search_gemma ................................................................... 64
set_gemma_user .................................................................. 65
validateBoolean .................................................................. 65
validateID .......................................................................... 66
validateLimit ...................................................................... 66
validateOptionalID ............................................................. 67
validateOptionalQuery ......................................................... 67
validateOptionalTaxon ........................................................ 68
validatePositiveInteger ....................................................... 68
validateQuery ..................................................................... 69
validateResultType ............................................................. 69
validateSingleID ............................................................... 70
validateSort ....................................................................... 70
validateTaxa ...................................................................... 71
validateTaxon ..................................................................... 71
.getResultSetFactors

Retrieve a single analysis result set by its identifier

Usage

.getResultSetFactors(
    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

resultSet  An expression analysis result set numerical identifier.
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

Varies
# .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

- **resultSet**: An expression analysis result set numerical identifier.
- **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

Varies

## accessField

### 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)
encode

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

---

**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**

```
filter_properties()
```

**Value**

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

**Examples**

```
filter_properties()
```
forget_gemma_memoised  
Clear gemma.R cache

Description
Forget past results from memoised calls to the Gemma API (ie. using functions with memoised = TRUE)

Usage
forget_gemma_memoised()

Value
TRUE to indicate cache was cleared.

Examples
forget_gemma_memoised()

---

gemma.R package: Access curated gene expression data and differential expression analyses

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, the function reference and the vignette.

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

Author(s)
Javier Castillo-Arnemann, Jordan Sicherman, Ogan Mancarci, 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

---

### gemmaCache

**Description**

Gemma Cache

**Usage**

```r
gemmaCache()
```

**Value**

A memoise filesystem

### gemmaPath

**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)
```

---

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

Arguments

query
Output from a `gemma.R` function with offset and query 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. 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 or a list containing data from all pages.

---

get_datasets

Retrieve all datasets

Description

Retrieve all datasets

Usage

```r
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. 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.
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

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 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. A successful response may contain 'Geeq' information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties here.

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
get_datasets_by_ids

- `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.batchEffect`: A text field describing whether the dataset has batch effects
- `geeq.batchCorrected`: Whether batch correction has been performed on the dataset.
- `geeq.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
- `geeq.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- `geeq.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

Description

Retrieve datasets by their identifiers
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 appended to the filter and equivalent to filtering for allCharacteristics.valueUri

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 "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

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 the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
get_datasets_by_ids

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. A successful response may contain 'Geeq' information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties [here](#).

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
- geeq.batchEffect: A text field describing whether the dataset has batch effects
- geeq.batchCorrected: Whether batch correction has been performed on the dataset.
- geeq.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
- geeq.batchEffect: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- geeq.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

g get_datasets_by_ids("GSE2018")
g get_datasets_by_ids(c("GSE2018", "GSE2872"))
get_dataset_annotations

Retrieve the annotations of a dataset

Description

Retrieve the annotations of a dataset

Usage

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.Type: Type of the annotation class
- 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
**get_dataset_design**

Retrieve the design of a dataset

**Usage**

```r
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**

```r
get_dataset_annotations("GSE2018")

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,
  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

dataset A numerical dataset identifier or a dataset short name
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.
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 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:
get_dataset_expression

- `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
- `baseline.category`: Category for the contrast
- `baseline.categoryURI`: URI for the baseline category
- `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
- `subsetFactor.subset`: TRUE if the result set belong to a subset, FALSE if not. Subsets are created when performing differential expression to avoid unhelpful comparisons.
- `subsetFactor.category`: Category of the subset
- `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

*Retrieves 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")
```

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))
```

get_dataset_object  Compile gene expression data and metadata

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

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 probe 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.

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 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.
get_dataset_platforms

Examples

get_dataset_object("GSE2018")

get_dataset_platforms  Retrieve the platforms of a dataset

Description

Retrieve the platforms of a dataset

Usage

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 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
get_dataset_processed_expression

Retrieve processed expression data of a dataset

Description
Retrieve processed expression data of a dataset

Usage
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)
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>A numerical dataset identifier or a dataset short name</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>

Examples
get_dataset_platforms("GSE2018")
get_dataset_quantitation_types

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

get_dataset_processed_expression("GSE2018")

Description

Retrieve quantitation types of a dataset

Usage

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)
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>A numerical dataset identifier or a dataset short name</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 <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 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 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`
- `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**

```r
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**

```r
generate 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.
get_dataset_samples

Retrieve the samples of a dataset

Description
Retrieve the samples of a dataset

Usage
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>A numerical dataset identifier or a dataset short name</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>
</tbody>
</table>
get_differential_expression_values

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

Examples

head(get_dataset_samples("GSE2018"))

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_,
  resultSet = NA_integer_,
  readableContrasts = FALSE,
  memoised = getOption("gemma.memoised", FALSE)
)
get_genes

Arguments

dataset: A dataset identifier.
resultSet: A resultSet identifier.
readableContrasts: If FALSE (default), the returned columns will use internal contrasts 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

```r
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.

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 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.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

Examples

genes("DYRK1A")
genes(c("DYRK1A", "PTEN"))
**get_gene_go_terms**

Retrieve the GO terms associated to a gene

**Description**

Retrieve the GO terms associated to a gene

**Usage**

```r
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)
)
```

**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
gene_go_terms("DYRK1A")
```
get_gene_locations  Retrieve the physical locations of a given gene

Description
Retrieve the physical locations of a given gene

Usage
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)
)

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
get_gene_probes

• 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

get_gene_locations("DYRK1A")

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 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.
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:

- `mapping.name`: Name of the mapping. Typically the probeset name
- `mapping.description`: A free text field providing optional information about the mapping
- `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.Taxon`: Species the platform was designed for
- `platform.TaxonID`: Id number of the species 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.

Examples

```r
gene_probes("DYRK1A")
```

---

**get_platforms_by_ids**  
Retrieve all platforms matching a set of platform identifiers

**Description**

Retrieve all platforms matching a set of platform identifiers

**Usage**

```r
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)
)
```
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 "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  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 the raw endpoint from the API, likely a JSON file 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**

```r
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**

```r
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 identifier @seealso getPlatforms
- **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)
**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** (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.)
- **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.)

**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'))
```
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.

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.

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

A data table with information about the queried dataset(s). A list if `raw = TRUE`. Returns an empty list if no datasets matched. A successful response may contain 'Geeq' information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties [here](#).

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
- `geeq.batchEffect`: A text field describing whether the dataset has batch effects
- `geeq.batchCorrected`: Whether batch correction has been performed on the dataset.
- `geeq.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
- `geeq.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- `geeq.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
get_platform_element_genes

- **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
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**

```r
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**

- **platform**: A platform numerical identifier or a platform short name
- **probe**: A probe name or its numerical identifier
- **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.
- **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_taxa

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 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.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

Examples

get_platform_element_genes("GPL1355", "AFFX_Rat_beta-actin_M_at")

get_tauxa

Get taxa

Description

Returns taxa and their versions used in Gemma

Usage

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

get_taxa()

Description

Retrieve taxa by their identifiers.

Usage

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>NcbID</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.

Value A data table with the queried taxa’s details.

Examples

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

get_taxon_datasets Retrieve the datasets for a given taxon

Description This function is deprecated in favor of `get_datasets`

Usage

g_powdered = get_dataset(  
  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),  
  new = TRUE  
)

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:
### get_taxon_datasets

<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.

... Kept for compatibility. Ignored

### Value

A data table with information about the queried dataset(s). A list if `raw = TRUE`. Returns an empty list if no datasets matched. A successful response may contain ‘Geeq’ information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties [here](#).

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
make_design

- 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.batchEffect: A text field describing whether the dataset has batch effects
- geeq.batchCorrected: Whether batch correction has been performed on the dataset.
- geeq.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
- geeqbatchEffect: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- geeq.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

get_taxon_datasets("human")

Description

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

Usage

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"
nullCheck

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

Examples
samples <- get_dataset_samples('GSE46416')
make_design(samples)

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.Type`: Type of the annotation class
- `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

---

`processCharacteristicBasicValueObject`

*Processes JSON as a factor*

---

**Description**

Processes JSON as a factor

**Usage**

`processCharacteristicBasicValueObject(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**

`processDatasetResultSets(d)`
processDatasets  

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. A successful response may contain ‘Geeq’ information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties here.

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
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
- `baseline.category`: Category for the contrast
- `baseline.categoryURI`: URI for the baseline category
- `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
- `subsetFactor.subset`: TRUE if the result set belong to a subset, FALSE if not. Subsets are created when performing differential expression to avoid unhelpful comparisons.
- `subsetFactor.category`: Category of the subset
- `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

`processDEcontrasts(rs, rsID)`

Arguments

- `rs`  
The resultSet matrix to process

Value

A processed matrix
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Usage</th>
<th>Arguments</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>processDEMatrix</td>
<td>Processes differential expression matrix</td>
<td>processDEMatrix(m)</td>
<td>m (differential expression matrix)</td>
<td>A processed matrix</td>
</tr>
<tr>
<td>processDesignMatrix</td>
<td>Processes design matrix</td>
<td>processDesignMatrix(m)</td>
<td>m (design matrix)</td>
<td>A processed matrix</td>
</tr>
</tbody>
</table>
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:

- `mapping.name`: Name of the mapping. Typically the probeset name
- `mapping.description`: A free text field providing optional information about the mapping
- `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.Taxon`: Species the platform was designed for
- `platform.TaxonID`: Id number of the species 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.
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

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.

processGemmaFactor

Description
Processes JSON as a factor

Usage
processGemmaFactor(d)

Arguments

d The JSON to process

Value
A processed data.table
**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)
**processGO**

**Arguments**

- **d**
  
  The JSON to process

**Value**

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

The fields of the output data.table are:

- \texttt{gene.Symbol}: Symbol for the gene
- \texttt{gene.Ensembl}: Ensembl ID for the gene
- \texttt{gene.NCBI}: NCBI id for the gene
- \texttt{gene.Name}: Name of the gene
- \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

---

**processGO**

*Processes JSON as GO terms*

**Description**

Processes JSON as GO terms

**Usage**

\texttt{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 \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{term.Name}: Name of the term
- \texttt{term.ID}: ID of the term
- \texttt{term.URI}: URI of the term
processPlatforms  

Processes JSON as a vector of platforms

Description

Processes JSON as a vector of platforms

Usage

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
processQuantitationType ValueObject

Description

processQuantitationType ValueObject

Usage

processQuantitationType ValueObject(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`
- 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)
processSamples

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)

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

processTaxon

Processes JSON as a vector of taxa

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**

```r
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)
)
```
search_datasets

Arguments

query The search query
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),
    )
search_datasets

```r

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
(\texttt{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
\texttt{get\_taxa\_by\_ids} function to retrieve the necessary information. For conven-
nience, 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 \texttt{offset} and the \texttt{totalElements}
\texttt{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**
\texttt{TRUE} to receive results as-is from Gemma, or \texttt{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 \texttt{forget\_gemma\_memoised}
to clear the cache.

**file**
The name of a file to save the results to, or \texttt{NULL} to not write results to a file. If
\texttt{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 \texttt{TRUE} additional information from the call will be added into the output ob-
ject’s attributes such as offset and available elements.

... Kept for compatibility, ignored.
**Value**

A data table with information about the queried dataset(s). A list if `raw = TRUE`. Returns an empty list if no datasets matched. A successful response may contain ‘Geeq’ information, which aims to provide a unified metric to measure experiments by the quality of their data, and their suitability for use in Gemma. You can read more about the geeq properties [here](#).

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.batchEffect`: A text field describing whether the dataset has batch effects
- `geeq.batchCorrected`: Whether batch correction has been performed on the dataset.
- `geeq.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
- `geeq.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- `geeq.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")
```
Description

Search everything in Gemma

Usage

search_gemma(
  query,
  taxon = NA_character_,
  platform = NA_character_,
  limit = 20,
  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. 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 A numerical taxon identifier or an ncbi taxon identifier or a taxon identifier that matches either its scientific or common name
platform A platform numerical identifier or a platform short name
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.
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 \texttt{raw = FALSE} and \texttt{resultType} is \texttt{experiment}, \texttt{gene} or \texttt{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 \texttt{raw = TRUE}

Examples

\begin{verbatim}
search_gemma("bipolar")
\end{verbatim}

\begin{verbatim}
set_gemma_user
\end{verbatim}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
set_gemma_user(username = NULL, password = NULL)
\end{verbatim}

\textbf{Arguments}

\begin{verbatim}
username   Your username (or empty, if logging out)
password   Your password (or empty, if logging out)
\end{verbatim}

\textbf{Value}

\begin{verbatim}
TRUE if authentication is successful, FALSE if not
\end{verbatim}

\begin{verbatim}
validateBoolean
\end{verbatim}

\textbf{Description}

Validate a boolean value

\textbf{Usage}

\begin{verbatim}
validateBoolean(name, ...)
\end{verbatim}
validateLimit

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

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

Arguments

name  The variable name

...  Any identifiers

Value

The validated identifiers, or stop with an error message

validateLimit

Validate a limit value

Arguments

name  The variable name

...  Any possible integers
validateOptionalID

**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**

name  The variable name
...
...  Any identifiers

**Value**

The validated identifiers, or stop with an error message

---

validateOptionalQuery  Validate an optional query

---

**Description**

Validate an optional query

**Usage**

validateOptionalQuery(name, ...)

**Arguments**

name  The variable name
...
...  Any queries

**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
...

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
...

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

Validates 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  

Validates 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

Validate taxa using the acceptable taxa entries

Description
Validate 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

Validate a taxon using the acceptable taxa entries

Description
Validate 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, 16
  get_dataset_design, 17
  get_dataset_differential_expression_analyses, 18
  get_dataset_expression_for_genes, 20
  get_dataset_object, 21
  get_dataset_platforms, 23
  get_dataset_processed_expression, 24
  get_dataset_quantitation_types, 25
  get_dataset_raw_expression, 26
  get_dataset_samples, 27
  get_datasets, 11
  get_datasets_by_ids, 13
  get_differential_expression_values, 28

* gene
  get_gene_go_terms, 31
  get_gene_locations, 32
  get_gene_probes, 33
  getgenes, 29

* internal
  .getResultSetFactors, 4
  .getResultSets, 5
  accessField, 5
  blank_processor, 6
  checkBounds, 6
  encode, 7
  gemmaCache, 9
  gemmaPath, 9
  get_dataset_expression, 19
  get_taxa_by_ids, 41
  get_taxon_datasets, 42
  nullCheck, 45
  process_search, 60
  processAnnotations, 45
  processCharacteristicBasicValueObject,
  processDatasetResultSets, 46
  processDatasets, 47
  processDEA, 48
  processDEcontrasts, 49
  processDEMatrix, 50
  processDesignMatrix, 50
  processElements, 51
  processExpressionMatrix, 52
  processFile, 52
  processGemmaArray, 53
  processGemmaFactor, 53
  processGeneLocation, 54
  processGenes, 54
  processGO, 55
  processPlatforms, 56
  processQuanititationTypeValueObject,
  processResultSetFactors, 57
  processSamples, 58
  processSearchAnnotations, 59
  processTaxon, 59
  search_datasets, 61
  validateBoolean, 65
  validateID, 66
  validateLimit, 66
  validateOptionalID, 67
  validateOptionalQuery, 67
  validateOptionalTaxon, 68
  validatePositiveInteger, 68
  validateQuery, 69
  validateResultType, 68
  validateSingleID, 70
  validateSort, 70
  validateTaxa, 71
  validateTaxon, 71

* misc
  filter_properties, 7
  forget_gemma_memoised, 8
INDEX

get_all_pages, 10
get_taxa, 40
make_design, 44
search_annotations, 60
search_gemma, 64
set_gemma_user, 65

platform
get_platform_annotations, 36
get_platform_datasets, 37
get_platform_element_genes, 39
get_platforms_by_ids, 34

attribute, 12, 14, 18, 33, 35, 37, 39, 43, 62, 64

blank_processor, 6
checkBounds, 6

encode, 7
ExpressionSet, 22

filter_properties, 7, 12, 14, 35
forget_gemma_memoised, 4, 5, 8, 12, 14,
16–18, 20–25, 27–33, 35, 38–40,
42, 43, 61, 62, 64

gamma.R, 8
gamma_call, 10
gammaCache, 9
gammaPath, 9
get_all_pages, 10
get_dataset_annotations, 16
get_dataset_design, 17
get_dataset_differential_expression_analyses
18, 29, 47
get_dataset_differential_expression_analyses
18, 28, 47, 49
get_dataset_samples, 27, 44
get_datasets, 7, 11, 42, 61
get_datasets_by_ids, 13
get_differential_expression_values, 8,
18, 28, 47, 49
get_gene_go_terms, 31
get_gene_locations, 32
get_gene_probes, 33
get_genes, 29
get_platform_annotations, 8, 36
get_platform_datasets, 37
get_platform_element_genes, 39
get_platforms_by_ids, 7, 34
get_taxa, 40
gene_by_ids, 41, 42, 62
gene_sets, 42

make_design, 44

nullCheck, 45

process_search, 60
processAnnotations, 45
processCharacteristicBasicValueObject, 46
processDatasetResultSets, 46
processDatasets, 47
processDEA, 48
processDEcontrasts, 49
processDEMatrix, 50
processDesignMatrix, 50
processElements, 51
processExpressionMatrix, 52
processFile, 52
processGemmaArray, 53
processGemmaFactor, 53
processGeneLocation, 54
processGenes, 54
processGO, 55
processPlatforms, 56
processQuantitationTypeValueObject, 57
processResultSetFactors, 57
processSamples, 58
processSearchAnnotations, 59
processTaxon, 59

search_annotations, 60
search_datasets, 61
search_gemma, 64
set_gemma_user, 65
SummarizedExperiment, 22

validateBoolean, 65
validateID, 66
validateLimit, 66
validateOptionalID, 67
validateOptionalQuery, 67
validateOptionalTaxon, 68
validatePositiveInteger, 68
validateQuery, 69
validateResultType, 69
validateSingleID, 70
validateSort, 70
validateTaxa, 71
validateTaxon, 71