

# Package ‘gemma.R’

September 23, 2024

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

**Version** 3.1.12

**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.

**URL** <https://pavlidislab.github.io/gemma.R/>,  
<https://github.com/PavlidisLab/gemma.R>

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

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**BugReports** <https://github.com/PavlidisLab/gemma.R/issues>

**Imports** magrittr, glue, memoise, jsonlite, data.table, rlang,  
lubridate, utils, stringr, SummarizedExperiment, Biobase,  
tibble, tidyr, S4Vectors, httr, rappdirs, bit64, assertthat,  
digest, R.utils, base64enc

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

**Config/testthat/edition** 2

**VignetteBuilder** knitr

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

**git\_url** <https://git.bioconductor.org/packages/gemma.R>

**git\_branch** devel

**git\_last\_commit** e70191e

**git\_last\_commit\_date** 2024-08-08

**Repository** Bioconductor 3.20

**Date/Publication** 2024-09-22

**Author** Javier Castillo-Arnemann [aut]

(<https://orcid.org/0000-0002-5626-9004>),

Jordan Sicherman [aut] (<https://orcid.org/0000-0001-8160-4567>),

Ogan Mancarci [cre, aut] (<https://orcid.org/0000-0002-1452-0889>),

Guillaume Poirier-Morency [aut]

(<https://orcid.org/0000-0002-6554-0441>)

**Maintainer** Ogan Mancarci <ogan.mancarci@gmail.com>

## Contents

.getResultSets . . . . .	4
accessField . . . . .	5
blank_processor . . . . .	5
checkBounds . . . . .	6
encode . . . . .	6
filter_properties . . . . .	7
forget_gemma_memoised . . . . .	7
gemma.R . . . . .	8
gemmaCache . . . . .	8
gemmaPath . . . . .	9
gemma_call . . . . .	9
gemma_kable . . . . .	10
gemma_memoise . . . . .	10
get_all_pages . . . . .	11
get_child_terms . . . . .	11
get_datasets . . . . .	12
get_datasets_by_ids . . . . .	14
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 . . . . .	22
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 . . . . .	30
get_gene_differential_expression_values . . . . .	31
get_gene_go_terms . . . . .	32
get_gene_locations . . . . .	33
get_gene_probes . . . . .	35
get_platforms_by_ids . . . . .	36
get_platform_annotations . . . . .	38
get_platform_datasets . . . . .	39
get_platform_element_genes . . . . .	40
get_result_sets . . . . .	42
get_taxa . . . . .	43

get_taxa_by_ids . . . . .	44
get_taxon_datasets . . . . .	45
isEmpty . . . . .	47
make_design . . . . .	48
memoise . . . . .	48
nullCheck . . . . .	49
processAnnotations . . . . .	49
processCharacteristicValueObject . . . . .	50
processDatasetResultSets . . . . .	50
processDatasets . . . . .	51
processDEA . . . . .	52
processDEcontrasts . . . . .	53
processDEMatrix . . . . .	53
processDesignMatrix . . . . .	54
processDifferentialExpressionAnalysisResultByGeneValueObject_tsv . . . . .	54
processDifferentialExpressionAnalysisResultSetValueObject . . . . .	55
processElements . . . . .	55
processExpressionMatrix . . . . .	56
processFile . . . . .	57
processGemmaArray . . . . .	57
processGeneLocation . . . . .	58
processGenes . . . . .	58
processGO . . . . .	59
processPlatforms . . . . .	60
processQuantitationTypeValueObject . . . . .	61
processResultSetFactors . . . . .	61
processSamples . . . . .	62
processSearchAnnotations . . . . .	63
processTaxon . . . . .	63
process_search . . . . .	64
search_annotations . . . . .	64
search_datasets . . . . .	65
search_gemma . . . . .	68
setGemmaPath . . . . .	69
set_gemma_user . . . . .	69
subset_factorValues . . . . .	70
update_result . . . . .	70
validateBoolean . . . . .	71
validateID . . . . .	71
validateLimit . . . . .	72
validateOptionalID . . . . .	72
validateOptionalQuery . . . . .	73
validateOptionalTaxon . . . . .	73
validatePositiveInteger . . . . .	74
validateQuery . . . . .	74
validateResultType . . . . .	75
validateSingleID . . . . .	75
validateSort . . . . .	76
validateTaxa . . . . .	76
validateTaxon . . . . .	77

---

<code>.getResultSets</code>	<i>Retrieve a single analysis result set by its identifier</i>
-----------------------------	--

---

## 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

<code>resultSet</code>	An expression analysis result set numerical identifier.
<code>raw</code>	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.
<code>memoised</code>	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 <a href="#">forget_gemma_memoised</a> to clear the cache.
<code>file</code>	The name of a file to save the results to, or NULL to not write results to a file. If <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
<code>overwrite</code>	Whether or not to overwrite if a file exists at the specified filename.

## Value

Varies

## Examples

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

---

accessField	<i>Access the field in a list</i>
-------------	-----------------------------------

---

**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	<i>A blank processor that returns data as is</i>
-----------------	--

---

**Description**

A blank processor that returns data as is

**Usage**

```
blank_processor(data)
```

**Arguments**

data	any data
------	----------

**Value**

Data as is

---

checkBounds	<i>Replace missing data with NAs</i>
-------------	--------------------------------------

---

**Description**

Replace missing data with NAs

**Usage**

```
checkBounds(x, natype = NA)
```

**Arguments**

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

**Value**

Data or NA in case of an out of bounds error

---

encode	<i>URL encode a string safely</i>
--------	-----------------------------------

---

**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	<i>Return all supported filter properties</i>
-------------------	---

---

### 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	<i>Clear gemma.R cache</i>
-----------------------	----------------------------

---

### 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

*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](#) 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>

## See Also

Useful links:

- <https://pavlidislab.github.io/gemma.R/>
- <https://github.com/PavlidisLab/gemma.R>
- Report bugs at <https://github.com/PavlidisLab/gemma.R/issues>

---

gemmaCache

*Gemma Cache*


---

## Description

Gemma Cache

## Usage

`gemmaCache()`



**Value**

A memoise filesystem

---

gemmaPath	<i>Get gemma path</i>
-----------	-----------------------

---

**Description**

Get gemma path

**Usage**

```
gemmaPath()
```

**Value**

Link to Gemma API

---

gemma_call	<i>Custom gemma call</i>
------------	--------------------------

---

**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**

```
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**

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

---

gemma_kable	<i>Create printable tables out of gemma.R outputs</i>
-------------	---

---

### Description

Creates a [kable](#) where certain columns are automatically shortened to better fit a document.

### Usage

```
gemma_kable(table)
```

### Arguments

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

---

gemma_memoise	<i>Enable and disable memoisation of gemma.R functions</i>
---------------	--

---

### Description

Enable and disable memoisation of gemma.R functions

### Usage

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

### Arguments

memoised	boolean. If TRUE memoisation will be enabled
cache	File path or "cache_in_memory". File path will chose a location to save the cache files for memoisation. "cache_in_memory" will store the cache in the current R session

---

get_all_pages	<i>Get all pages of a paginated call</i>
---------------	--

---

### Description

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

### Usage

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

### Arguments

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

### Value

A data.table or a list containing data from all pages.

---

get_child_terms	<i>Return child terms of a term</i>
-----------------	-------------------------------------

---

### Description

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

### Usage

```
get_child_terms(terms)
```

**Arguments**

terms                      An array of terms

**Value**

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

**Examples**

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

---

get_datasets	<i>Retrieve all datasets</i>
--------------	------------------------------

---

**Description**

Retrieve all datasets

**Usage**

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

**Arguments**

query	The search query. Queries can include plain text or ontology terms They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").
filter	Filter results by matching expression. Use <a href="#">filter_properties</a> 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

<code>uris</code>	A vector of ontology term URIs. Providing multiple terms will return results containing any of the terms and their children. These are appended to the filter and equivalent to filtering for <code>allCharacteristics.valueUri</code>
<code>offset</code>	The offset of the first retrieved result.
<code>limit</code>	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with <code>offset</code> and the <code>totalElements</code> <a href="#">attribute</a> in the output to compile all data if needed.
<code>sort</code>	Order results by the given property and direction. The '+' sign indicate ascending order whereas the '-' indicate descending.
<code>raw</code>	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.
<code>memoised</code>	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 <a href="#">forget_gemma_memoised</a> to clear the cache.
<code>file</code>	The name of a file to save the results to, or NULL to not write results to a file. If <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
<code>overwrite</code>	Whether or not to overwrite if a file exists at the specified filename.

## Value

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

The fields of the output `data.table` are:

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

- `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- `geeq.qScore`: Data quality score given to the dataset by Gemma.
- `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- `taxon.name`: Name of the species
- `taxon.scientific`: Scientific name for the taxon
- `taxon.ID`: Internal identifier given to the species by Gemma
- `taxon.NCBI`: NCBI ID of the taxon
- `taxon.database.name`: Underlying database used in Gemma for the taxon
- `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon

### Examples

```
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.org/obo/UBERON_0002048")
get_datasets(query = "lung")
```

---

get_datasets_by_ids	<i>Retrieve datasets by their identifiers</i>
---------------------	---

---

### Description

Retrieve datasets by their identifiers

### Usage

```
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

<code>datasets</code>	Numerical dataset identifiers or dataset short names. If not specified, all datasets will be returned instead
-----------------------	---

filter	Filter results by matching expression. Use <a href="#">filter_properties</a> function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as "=", "<" or "in". (e.g. "taxon.commonName = human", "taxon.commonName in (human,mouse)", "id < 1000")
taxa	A vector of taxon common names (e.g. human, mouse, rat). Providing multiple species will return results for all species. These are appended to the filter and equivalent to filtering for taxon.commonName property
uris	A vector of ontology term URIs. Providing multiple terms will return results containing any of the terms and their children. These are appended to the filter and equivalent to filtering for allCharacteristics.valueUri
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> to clear the cache.
file	The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

### Value

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

The fields of the output data.table are:

- experiment.shortName: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- experiment.name: Full title of the dataset
- experiment.ID: Internal ID of the dataset.
- experiment.description: Description of the dataset
- experiment.troubled: Did an automatic process within gemma or a curator mark the dataset as "troubled"
- experiment.accession: Accession ID of the dataset in the external database it was taken from
- experiment.database: The name of the database where the dataset was taken from
- experiment.URI: URI of the original database
- experiment.sampleCount: Number of samples in the dataset

- `experiment.batchEffectText`: A text field describing whether the dataset has batch effects
- `experiment.batchCorrected`: Whether batch correction has been performed on the dataset.
- `experiment.batchConfound`: 0 if batch info isn't available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
- `experiment.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- `geeq.qScore`: Data quality score given to the dataset by Gemma.
- `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- `taxon.name`: Name of the species
- `taxon.scientific`: Scientific name for the taxon
- `taxon.ID`: Internal identifier given to the species by Gemma
- `taxon.NCBI`: NCBI ID of the taxon
- `taxon.database.name`: Underlying database used in Gemma for the taxon
- `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon

## Examples

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

---

```
get_dataset_annotations
```

*Retrieve the annotations of a dataset*

---

## Description

Retrieve the annotations of a dataset

## Usage

```
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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

**Value**

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

The fields of the output data.table are:

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

**Examples**

```
get_dataset_annotations("GSE2018")
```

---

get_dataset_design	<i>Retrieve the design of a dataset</i>
--------------------	---

---

**Description**

Retrieve the design of a dataset

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

```
head(get_dataset_design("GSE2018"))
```

---

```
get_dataset_differential_expression_analyses
```

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

---

**Description**

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

**Usage**

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

**Arguments**

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

memoised	Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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:

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

## Examples

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

---

get\_dataset\_expression

*Retrieve processed expression data of a dataset*

---

## Description

This function is deprecated in favor of [get\\_dataset\\_processed\\_expression](#)

**Usage**

```
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)
)
```

**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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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**

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

---

```
get_dataset_expression_for_genes
```

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

---

**Description**

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

**Usage**

```
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)
)
```

**Arguments**

<code>datasets</code>	A vector of dataset IDs or short names
<code>genes</code>	A vector of NCBI IDs, Ensembl IDs or gene symbols.
<code>keepNonSpecific</code>	logical. FALSE by default. If TRUE, results from probesets that are not specific to the gene will also be returned.
<code>consolidate</code>	An option for gene expression level consolidation. If empty, will return every probe for the genes. "pickmax" to pick the probe with the highest expression, "pickvar" to pick the prove with the highest variance and "average" for returning the average expression
<code>raw</code>	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.
<code>memoised</code>	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 <a href="#">forget_gemma_memoised</a> to clear the cache.
<code>file</code>	The name of a file to save the results to, or NULL to not write results to a file. If <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
<code>overwrite</code>	Whether or not to overwrite if a file exists at the specified filename.

**Value**

A list of data frames

**Examples**

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

---

get_dataset_object	<i>Compile gene expression data and metadata</i>
--------------------	--

---

## 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 vector of dataset IDs or short names
genes	A vector of NCBI IDs, Ensembl IDs or gene symbols.
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 <a href="#">vignette</a> or the ExpressionSet <a href="#">vignette</a> for more details. "tidy" for a long form data frame compatible with tidyverse functions. 'list' to return a list containing individual data frames containing expression values, design and the experiment.

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

## 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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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.
- taxon.name: Name of the species platform was made for
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon

**Examples**

```
get_dataset_platforms("GSE2018")
```

---

```
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**

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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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**

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

---

```
get_dataset_quantitation_types
```

*Retrieve quantitation types of a dataset*

---

**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**

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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

### Value

A `data.table` containing the quantitation types

The fields of the output `data.table` are:

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

### Examples

```
get_dataset_quantitation_types("GSE59918")
```

---

```
get_dataset_raw_expression
```

*Retrieve raw expression data of a dataset*

---

### Description

Retrieve raw expression data of a dataset

### Usage

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

**Arguments**

dataset	A numerical dataset identifier or a dataset short name
quantitationType	Quantitation type id. These can be acquired using <a href="#">get_dataset_quantitation_types</a> function. This endpoint can only return non-processed quantitation types.
raw	TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
memoised	Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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**

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

---

get_dataset_samples	<i>Retrieve the samples of a dataset</i>
---------------------	--

---

**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**

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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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_,
  resultSets = NA_integer_,
  keepNonSpecific = FALSE,
  readableContrasts = FALSE,
  memoised = getOption("gemma.memoised", FALSE)
)
```

**Arguments**

dataset	A dataset identifier.
resultSets	resultSet identifiers. If a dataset is not provided, all result sets will be downloaded. If it is provided it will only be used to ensure all result sets belong to the dataset.
keepNonSpecific	logical. FALSE by default. If TRUE, results from probesets that are not specific to the gene will also be returned.
readableContrasts	If FALSE (default), the returned columns will use internal contrasts IDs as names. Details about the contrasts can be accessed using <a href="#">get_dataset_differential_expression</a> . 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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 to two or three potentially nested factors. Gemma attempts to automatically assign baseline conditions for each factor. In the absence of a clear control condition, a baseline is arbitrarily selected. A generalized linear model with empirical Bayes shrinkage of t-statistics is fit to the data for each platform element (probe/gene) using an implementation of the limma algorithm. For RNA-seq data, we use weighted regression, applying the voom algorithm to compute weights from the mean–variance relationship of the data. Contrasts of each condition are then computed compared to the selected baseline. In some situations, Gemma will split the data into subsets for analysis. A typical such situation is when a ‘batch’ factor is present and confounded with another factor, the subsets being determined by the levels of the confounding factor.

**Value**

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

**Examples**

```
get_differential_expression_values("GSE2018")
```

---

get_genes	<i>Retrieve genes matching gene identifiers</i>
-----------	---

---

**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	A vector of NCBI IDs, Ensembl IDs or gene symbols.
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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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.aliases`: Gene aliases. Each row includes a vector
- `gene.MFX.rank`: Multifunctionality rank for the gene
- `taxon.name`: Name of the species

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

## Examples

```
get_genes("DYRK1A")
get_genes(c("DYRK1A", "PTEN"))
```

---

```
get_gene_differential_expression_values
```

*Retrieve the differential expression results for a given gene among datasets matching the provided query and filter*

---

## Description

Retrieve the differential expression results for a given gene among datasets matching the provided query and filter

## Usage

```
get_gene_differential_expression_values(
  gene,
  query = NA_character_,
  taxa = NA_character_,
  uris = NA_character_,
  filter = NA_character_,
  threshold = 1,
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

## Arguments

<code>gene</code>	An ensembl gene identifier which typically starts with <code>ensg</code> or an ncbi gene identifier or an official gene symbol approved by hgnc
<code>query</code>	The search query. Queries can include plain text or ontology terms They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").
<code>taxa</code>	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 <code>taxon.commonName</code> property
<code>uris</code>	A vector of ontology term URIs. Providing multiple terms will return results containing any of the terms and their children. These are appended to the filter and equivalent to filtering for <code>allCharacteristics.valueUri</code>

filter	Filter results by matching expression. Use <a href="#">filter_properties</a> 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")
threshold	number
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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

### Value

A `data.table` containing differential expression results. This table is stripped down some relevant information for speed of execution. Details about the contrasts can be accessed via [get\\_result\\_sets](#) function

The fields of the output `data.table` are:

- `result.ID`: Result set ID of the differential expression analysis. May represent multiple factors in a single model.
- `contrast.ID`: Id of the specific contrast factor. Together with the `result.ID` they uniquely represent a given contrast.
- `experiment.ID`: Id of the source experiment
- `factor.coefficient`: Model coefficient calculated for the specific contrast factor
- `factor.logfc`: Log 2 fold change calculated for the specific contrast factor
- `factor.pvalue`: p values calculated for the specific contrast factor

### Examples

```
# get all differential expression results for ENO2
# from datasets marked with the ontology term for brain
head(get_gene_differential_expression_values(2026, uris = "http://purl.obolibrary.org/obo/UBERON_0000955"))
```

---

get_gene_go_terms	<i>Retrieve the GO terms associated to a gene</i>
-------------------	---

---

### Description

Retrieve the GO terms associated to a gene



**Usage**

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

**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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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**

```
get_gene_go_terms(3091)
```

---

get_gene_locations	<i>Retrieve the physical locations of a given gene</i>
--------------------	--

---

**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 <code>ensg</code> 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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <code>forget_gemma_memoised</code> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

**Value**

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

The fields of the output `data.table` are:

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

**Examples**

```

get_gene_locations("DYRK1A")
get_gene_locations(1859)

```

---

get_gene_probes	<i>Retrieve the probes associated to a genes across all platforms</i>
-----------------	---

---

## Description

Retrieve the probes associated to a genes across all platforms

## Usage

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

## Arguments

gene	An ensembl gene identifier which typically starts with ensg or an ncbi gene identifier or an official gene symbol approved by hgnc
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> 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 platrofms. A list if raw = TRUE. A 404 error if the given identifier does not map to any genes.

The fields of the output data.table are:

- element.name: Name of the element. Typically the probeset name
- element.description: A free text field providing optional information about the element
- platform.shortName: Shortname of the platform given by Gemma. Typically the GPL identifier.

- platform.name: Full name of the platform
- platform.ID: Id number of the platform given by Gemma
- platform.type: Type of the platform.
- platform.description: Free text field describing the platform.
- platform.troubled: Whether the platform is marked as troubled by a Gemma curator.
- taxon.name: Name of the species platform was made for
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon

### Examples

```
get_gene_probes(1859)
```

---

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

---

### Description

Retrieve all platforms matching a set of platform identifiers

### Usage

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

### 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 <a href="#">filter_properties</a> 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 <code>taxon.commonName</code> property
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with <code>offset</code> and the <code>totalElements</code> <a href="#">attribute</a> 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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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.
- `taxon.name`: Name of the species platform was made for
- `taxon.scientific`: Scientific name for the taxon
- `taxon.ID`: Internal identifier given to the species by Gemma
- `taxon.NCBI`: NCBI ID of the taxon
- `taxon.database.name`: Underlying database used in Gemma for the taxon
- `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon

### Examples

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

---

get\_platform\_annotations

*Retrieve Platform Annotations by Gemma*


---

## Description

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

## Usage

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

## Arguments

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

## Value

A table of annotations

- **ElementName:** Probeset names provided by the platform. Gene symbols for generic annotations typically used for RNA-seq experiments.
- **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

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

---

get\_platform\_datasets *Retrieve all experiments using a given platform*

---

## Description

Retrieve all experiments using a given platform

## Usage

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

## Arguments

platform	A platform numerical identifier or a platform short name
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> to clear the cache.
file	The name of a file to save the results to, or NULL to not write results to a file. If raw == TRUE, the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

## Value

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

The fields of the output data.table are:

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

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

## Examples

```
head(get_platform_datasets("GPL1355"))
```

---

```
get_platform_element_genes
```

*Retrieve the genes associated to a probe in a given platform*

---

## Description

Retrieve the genes associated to a probe in a given platform



**Usage**

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

**Arguments**

platform	A platform numerical identifier or a platform short name
probe	A probe name or it's numerical identifier
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> 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.alias: Gene aliases. Each row includes a vector
- gene.MFX.rank: Multifunctionality rank for the gene
- taxon.name: Name of the species
- taxon.scientific: Scientific name for the taxon
- taxon.ID: Internal identifier given to the species by Gemma
- taxon.NCBI: NCBI ID of the taxon
- taxon.database.name: Underlying database used in Gemma for the taxon
- taxon.database.ID: ID of the underlying database used in Gemma for the taxon

## Examples

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

---

get_result_sets	<i>Retrieve all result sets matching the provided criteria</i>
-----------------	--

---

## Description

Returns queried result set

## Usage

```
get_result_sets(
  datasets = NA_character_,
  resultSets = NA_character_,
  filter = NA_character_,
  offset = 0,
  limit = 20,
  sort = "+id",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
  file = getOption("gemma.file", NA_character_),
  overwrite = getOption("gemma.overwrite", FALSE)
)
```

## Arguments

datasets	A vector of dataset IDs or short names
resultSets	A resultSet identifier. Note that result set identifiers are not static and can change when Gemma re-runs analyses internally. When using these as inputs, try to make sure you access a currently existing result set ID by basing them on result sets returned for a particular dataset or filter used in <a href="#">get_result_sets</a>
filter	Filter results by matching expression. Use <a href="#">filter_properties</a> function to get a list of all available parameters. These properties can be combined using "and" "or" clauses and may contain common operators such as "=", "<" or "in". (e.g. "taxon.commonName = human", "taxon.commonName in (human,mouse)", "id < 1000")
offset	The offset of the first retrieved result.
limit	Defaults to 20. Limits the result to specified amount of objects. Has a maximum value of 100. Use together with offset and the totalElements <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

### Details

Output and usage of this function is mostly identical to [get\\_dataset\\_differential\\_expression\\_analyses](#). The principal difference being the ability to restrict your result sets, being able to query across multiple datasets and being able to use the filter argument to search based on result set properties.

### Value

A data table with information about the queried result sets. Note that this function does not return differential expression values themselves. Use [get\\_differential\\_expression\\_values](#) to get differential expression values

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

### Examples

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

---

get\_taxa

*Get taxa*


---

### Description

Returns taxa and their versions used in Gemma

### Usage

```
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 taxa

Examples

`get_taxa()`

---

<code>get_taxa_by_ids</code>	<i>Retrieve taxa by their identifiers</i>
------------------------------	---

---

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 the 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

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

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 <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
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	<i>Retrieve the datasets for a given taxon</i>
--------------------	--

---

Description

This function is deprecated in favor of [get\\_datasets](#)

Usage

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

Arguments

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

ID	Comm.name	Scient.name	NcbiID
----	-----------	-------------	--------

1	human	Homo sapiens	9606
2	mouse	Mus musculus	10090
3	rat	Rattus norvegicus	10116
11	yeast	Saccharomyces cerevisiae	4932
12	zebrafish	Danio rerio	7955
13	fly	Drosophila melanogaster	7227
14	worm	Caenorhabditis elegans	6239

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 <a href="#">attribute</a> 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 <a href="#">forget_gemma_memoised</a> 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.

The fields of the output data.table are:

- experiment.shortName: Shortname given to the dataset within Gemma. Often corresponds to accession ID
- experiment.name: Full title of the dataset
- experiment.ID: Internal ID of the dataset.
- experiment.description: Description of the dataset
- experiment.troubled: Did an automatic process within gemma or a curator mark the dataset as "troubled"
- experiment.accession: Accession ID of the dataset in the external database it was taken from
- experiment.database: The name of the database where the dataset was taken from
- experiment.URI: URI of the original database
- experiment.sampleCount: Number of samples in the dataset
- experiment.batchEffectText: A text field describing whether the dataset has batch effects
- experiment.batchCorrected: Whether batch correction has been performed on the dataset.

- `experiment.batchConfound`: 0 if batch info isn't available, -1 if batch confound is detected, 1 if batch information is available and no batch confound found
- `experiment.batchEffect`: -1 if batch p value < 0.0001, 1 if batch p value > 0.1, 0 if otherwise and when there is no batch information is available or when the data is confounded with batches.
- `experiment.rawData`: -1 if no raw data available, 1 if raw data was available. When available, Gemma reprocesses raw data to get expression values and batches
- `geeq.qScore`: Data quality score given to the dataset by Gemma.
- `geeq.sScore`: Suitability score given to the dataset by Gemma. Refers to factors like batches, platforms and other aspects of experimental design
- `taxon.name`: Name of the species
- `taxon.scientific`: Scientific name for the taxon
- `taxon.ID`: Internal identifier given to the species by Gemma
- `taxon.NCBI`: NCBI ID of the taxon
- `taxon.database.name`: Underlying database used in Gemma for the taxon
- `taxon.database.ID`: ID of the underlying database used in Gemma for the taxon

## Examples

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

---

isEmpty

*Check for empty arguments*

---

## Description

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

## Usage

```
isEmpty(x)
```

## Arguments

x                      A parameter that can be NA or NULL when empty

---

make_design	<i>Make simplified design frames</i>
-------------	--------------------------------------

---

**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 <code>get_dataset_samples</code> . The output should not be raw
metaType	Type of metadata to include in the output. "text", "uri" or "both"

**Value**

A data.frame including the design table for the dataset

**Examples**

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

---

memoise	<i>Memoise doc</i>
---------	--------------------

---

**Description**

Memoise doc

**Arguments**

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



---

nullCheck	<i>Avoid NULLS as data.table columns</i>
-----------	--

---

**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 natypee

---

processAnnotations	<i>Processes JSON as annotations</i>
--------------------	--------------------------------------

---

**Description**

Processes JSON as annotations

**Usage**

```
processAnnotations(d)
```

**Arguments**

d	The JSON to process
---	---------------------

**Value**

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

The fields of the output data.table are:

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

---

```
processCharacteristicValueObject
```

*Processes JSON as a factor*

---

**Description**

Processes JSON as a factor

**Usage**

```
processCharacteristicValueObject(d)
```

**Arguments**

d                      The JSON to process

**Value**

A processed data.table

---

```
processDatasetResultSets
```

*Processes JSON as a datasets result set*

---

**Description**

Processes JSON as a datasets result set

**Usage**

```
processDatasetResultSets(d)
```

**Arguments**

d                      The JSON to process

**Value**

A data table with the queried datasets' resultSet ID(s). A list if raw = TRUE. Use [get\\_differential\\_expression\\_value](#) 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	<i>Processes JSON as a vector of datasets</i>
-----------------	---

---

## Description

Processes JSON as a vector of datasets

## Usage

```
processDatasets(d)
```

## Arguments

d                      The JSON to process

## Value

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

The fields of the output data.table are:

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

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

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\\_value](#) to get differential expression values (see examples).

The fields of the output `data.table` are:

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

---

processDEcontrasts	<i>Replaces factor ids by the factors strings in DE table columns</i>
--------------------	---

---

**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

---

processDEMatrix	<i>Processes differential expression matrix</i>
-----------------	---

---

**Description**

Processes differential expression matrix

**Usage**

```
processDEMatrix(m)
```

**Arguments**

m	The differential expression matrix to process
---	---

**Value**

A processed matrix

---

processDesignMatrix	<i>Processes design matrix</i>
---------------------	--------------------------------

---

**Description**

Processes design matrix

**Usage**

```
processDesignMatrix(m)
```

**Arguments**

m	The design matrix to process
---	------------------------------

**Value**

A processed matrix

---

processDifferentialExpressionAnalysisResultByGeneValueObject_tsv
<i>processDifferentialExpressionAnalysisResultByGeneValueObject_tsv</i>

---

**Description**

processDifferentialExpressionAnalysisResultByGeneValueObject\_tsv

**Usage**

```
processDifferentialExpressionAnalysisResultByGeneValueObject_tsv(content)
```

**Value**

A data.table containing differential expression results. This table is stripped down some relevant information for speed of execution. Details about the contrasts can be accessed via [get\\_result\\_sets](#) function

The fields of the output data.table are:

- result.ID: Result set ID of the differential expression analysis. May represent multiple factors in a single model.
- contrast.ID: Id of the specific contrast factor. Together with the result.ID they uniquely represent a given contrast.
- experiment.ID: Id of the source experiment
- factor.coefficient: Model coefficient calculated for the specific contrast factor
- factor.logfc: Log 2 fold change calculated for the specific contrast factor
- factor.pvalue: p values calculated for the specific contrast factor

---

processDifferentialExpressionAnalysisResultSetValueObject

*Process JSON of a result set*


---

### Description

Process JSON of a result set

### Usage

```
processDifferentialExpressionAnalysisResultSetValueObject(d)
```

### Value

A data table with information about the queried result sets. Note that this function does not return differential expression values themselves. Use [get\\_differential\\_expression\\_values](#) to get differential expression values

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

---

processElements

*Processes JSON as a vector of elements*


---

### Description

Processes JSON as a vector of elements

### Usage

```
processElements(d)
```

### Arguments

`d`                      The JSON to process

**Value**

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

The fields of the output data.table are:

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

---

processExpressionMatrix

*Processes expression matrix*

---

**Description**

Processes expression matrix

**Usage**

```
processExpressionMatrix(m)
```

**Arguments**

`m`                      The expression matrix to process

**Value**

A processed matrix



---

processFile	<i>Processes a response as a gzip file</i>
-------------	--

---

**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	<i>Processes JSON as an array</i>
-------------------	-----------------------------------

---

**Description**

Processes JSON as an array

**Usage**

```
processGemmaArray(d)
```

**Arguments**

d	The JSON to process
---	---------------------

**Value**

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

---

processGeneLocation	<i>Processes JSON as a vector of gene locations</i>
---------------------	---

---

**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	<i>Processes JSON as a vector of genes</i>
--------------	--

---

**Description**

Processes JSON as a vector of genes

**Usage**

```
processGenes(d)
```

**Arguments**

d	The JSON to process
---	---------------------

**Value**

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

The fields of the output data.table are:

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

---

processGO	<i>Processes JSON as GO terms</i>
-----------	-----------------------------------

---

**Description**

Processes JSON as GO terms

**Usage**

```
processGO(d)
```

**Arguments**

d                      The JSON to process

**Value**

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

The fields of the output data.table are:

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

---

processPlatforms	<i>Processes JSON as a vector of platforms</i>
------------------	--

---

## 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

---

processQuantitationTypeValueObject  
*processQuantitationTypeValueObject*

---

**Description**

processQuantitationTypeValueObject

**Usage**

processQuantitationTypeValueObject(d)

**Arguments**

d                      The JSON to process

**Value**

A data.table containing the quantitation types

The fields of the output data.table are:

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

---

processResultSetFactors  
*Processes JSON as a result set*

---

**Description**

Processes JSON as a result set

**Usage**

processResultSetFactors(d)

**Arguments**

d                      The JSON to process

**Value**

A processed data.table

---

processSamples	<i>Processes JSON as a vector of samples</i>
----------------	--

---

**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

---

<code>process_search</code>	<i>Returns the ids of the found results</i>
-----------------------------	---

---

**Description**

Returns the ids of the found results

**Usage**

```
process_search(d)
```

**Value**

A data.table or a list of resultObjects

---

<code>search_annotations</code>	<i>Search for annotation tags</i>
---------------------------------	-----------------------------------

---

**Description**

Search for annotation tags

**Usage**

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



**Arguments**

query	The search query. Queries can include plain text or ontology terms. They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").
raw	TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
memoised	Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

**Value**

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

The fields of the output `data.table` are:

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

**Examples**

```
search_annotations("traumatic")
```

---

search\_datasets

---

*Retrieve datasets associated to an annotation tags search*


---

**Description**

This function is deprecated in favor of [get\\_datasets](#)

**Usage**

```
search_datasets(
  query,
  taxon = NA_character_,
  offset = 0L,
  limit = 20L,
  sort = "+id",
  raw = getOption("gemma.raw", FALSE),
  memoised = getOption("gemma.memoised", FALSE),
```

```

file = getOption("gemma.file", NA_character_),
overwrite = getOption("gemma.overwrite", FALSE),
attributes = getOption("gemma.attributes", TRUE),
...
)

```

## Arguments

**query** The search query. Either plain text ('traumatic'), or an ontology term URI ('http://purl.obolibrary.org/obo/UBERON\_0002048'). Datasets that contain the given string in their short or full name will also be matched. Can be multiple identifiers separated by commas.

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

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

**offset** The offset of the first retrieved result.

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

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

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

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

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

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

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

**...** 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.

The fields of the output `data.table` are:

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

**Examples**

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

search\_gemma

*Search everything in Gemma***Description**

Search everything in Gemma

**Usage**

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

**Arguments**

query	The search query. Queries can include plain text or ontology terms. They also support conjunctions ("alpha AND beta"), disjunctions ("alpha OR beta") grouping ("(alpha OR beta) AND gamma"), prefixing ("alpha*"), wildcard characters ("BRCA?") and fuzzy matches ("alpha~").
taxon	A numerical taxon identifier or an ncbi taxon identifier or a taxon identifier that matches either its scientific or common name
platform	A platform numerical identifier or a platform short name
limit	Defaults to 100 with a maximum value of 2000. Limits the number of returned results. Note that this function does not support pagination.
resultType	The kind of results that should be included in the output. Can be experiment, gene, platform or a long object type name, documented in the API documentation.
raw	TRUE to receive results as-is from Gemma, or FALSE to enable parsing. Raw results usually contain additional fields and flags that are omitted in the parsed results.
memoised	Whether or not to save to cache for future calls with the same inputs and use the result saved in cache if a result is already saved. Doing <code>options(gemma.memoised = TRUE)</code> will ensure that the cache is always used. Use <a href="#">forget_gemma_memoised</a> 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 <code>raw == TRUE</code> , the output will be the raw endpoint from the API, likely a JSON or a gzip file. Otherwise, it will be a RDS file.
overwrite	Whether or not to overwrite if a file exists at the specified filename.

**Value**

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

**Examples**

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

---

setGemmaPath	<i>Set gemma path</i>
--------------	-----------------------

---

**Description**

Set gemma path

**Usage**

```
setGemmaPath(path)
```

**Arguments**

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

**Value**

Link to Gemma API

---

set_gemma_user	<i>Authentication by user name</i>
----------------	------------------------------------

---

**Description**

Allows the user to access information that requires logging in to Gemma. To log out, run set\_gemma\_user without specifying the username or password.

**Usage**

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

**Arguments**

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

**Value**

TRUE if authentication is successful, FALSE if not

---

subset_factorValues	<i>Get a subset of an array of factorValues</i>
---------------------	---

---

### Description

Get a subset of an array of factorValues

### Usage

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

### Arguments

factorValue      unimplemented  
differential\_expressions

### Value

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

---

update_result	<i>Update result</i>
---------------	----------------------

---

### Description

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

### Usage

```
update_result(query)
```

### Arguments

query              Output from a gemma.R function

### Details

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

**Examples**

```

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

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

```

---

validateBoolean	<i>Validate a boolean value</i>
-----------------	---------------------------------

---

**Description**

Validate a boolean value

**Usage**

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

**Arguments**

name	The variable name
...	Any boolean types

**Value**

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

---

validateID	<i>Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)</i>
------------	--

---

**Description**

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

**Usage**

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

**Arguments**

name	The variable name
...	Any identifiers

**Value**

The validated identifiers, or stop with an error message

---

validateLimit	<i>Validate a limit value</i>
---------------	-------------------------------

---

**Description**

Validate a limit value

**Usage**

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

**Arguments**

name	The variable name
...	Any possible integers

**Value**

The validated integers, or stop with an error message

---

validateOptionalID	<i>Validate identifiers (ie. gene ID, platform ID, etc.) that are homogeneous (either all numerics or all not)</i>
--------------------	--

---

**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**

<code>name</code>	The variable name
<code>...</code>	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**

<code>name</code>	The variable name
<code>...</code>	Any queries

**Value**

The validated queries, or stop with an error message

---

validateResultType	<i>Validate result types</i>
--------------------	------------------------------

---

**Description**

Validate result types

**Usage**

```
validateResultType(name, ...)
```

**Arguments**

name	The variable name
...	result types

**Value**

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

---

validateSingleID	<i>Validate a single identifier(ie. gene ID, platform ID, etc.)</i>
------------------	---

---

**Description**

Validate a single identifier(ie. 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	<i>Validate a sort argument</i>
--------------	---------------------------------

---

**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	<i>Validate taxa using the acceptable taxa entries</i>
--------------	--

---

**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	<i>Validate a taxon using the acceptable taxa entries</i>
---------------	---

---

**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, [22](#)
- 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, [12](#)
- get\_datasets\_by\_ids, [14](#)
- get\_differential\_expression\_values, [28](#)

## \* gene

- get\_gene\_differential\_expression\_values, [31](#)
- get\_gene\_go\_terms, [32](#)
- get\_gene\_locations, [33](#)
- get\_gene\_probes, [35](#)
- get\_genes, [30](#)

## \* internal

- .getResultSets, [4](#)
- accessField, [5](#)
- blank\_processor, [5](#)
- checkBounds, [6](#)
- encode, [6](#)
- gemmaCache, [8](#)
- gemmaPath, [9](#)
- get\_dataset\_expression, [19](#)
- get\_taxa\_by\_ids, [44](#)
- get\_taxon\_datasets, [45](#)
- isEmpty, [47](#)
- memoise, [48](#)
- nullCheck, [49](#)
- process\_search, [64](#)
- processAnnotations, [49](#)
- processCharacteristicValueObject, [50](#)

- processDatasetResultSets, [50](#)

- processDatasets, [51](#)

- processDEA, [52](#)

- processDEcontrasts, [53](#)

- processDEMatrix, [53](#)

- processDesignMatrix, [54](#)

- processDifferentialExpressionAnalysisResultByGeneValue, [54](#)

- processDifferentialExpressionAnalysisResultSetValue, [55](#)

- processElements, [55](#)

- processExpressionMatrix, [56](#)

- processFile, [57](#)

- processGemmaArray, [57](#)

- processGeneLocation, [58](#)

- processGenes, [58](#)

- processGO, [59](#)

- processPlatforms, [60](#)

- processQuantitationTypeValueObject, [61](#)

- processResultSetFactors, [61](#)

- processSamples, [62](#)

- processSearchAnnotations, [63](#)

- processTaxon, [63](#)

- search\_datasets, [65](#)

- setGemmaPath, [69](#)

- subset\_factorValues, [70](#)

- validateBoolean, [71](#)

- validateID, [71](#)

- validateLimit, [72](#)

- validateOptionalID, [72](#)

- validateOptionalQuery, [73](#)

- validateOptionalTaxon, [73](#)

- validatePositiveInteger, [74](#)

- validateQuery, [74](#)

- validateResultType, [75](#)

- validateSingleID, [75](#)

- validateSort, [76](#)

- validateTaxa, [76](#)

- validateTaxon, [77](#)

## \* misc

- filter\_properties, [7](#)

- forget\_gemma\_memoised, [7](#)

- gemma\_call, 9
- gemma\_kable, 10
- gemma\_memoise, 10
- get\_all\_pages, 11
- get\_child\_terms, 11
- get\_result\_sets, 42
- get\_taxa, 43
- make\_design, 48
- search\_annotations, 64
- search\_gemma, 68
- set\_gemma\_user, 69
- update\_result, 70
- \* platform**
  - get\_platform\_annotations, 38
  - get\_platform\_datasets, 39
  - get\_platform\_element\_genes, 40
  - get\_platforms\_by\_ids, 36
- .getResultSets, 4
- accessField, 5
- attribute, 13, 15, 35, 37, 39, 41, 42, 46, 66
- blank\_processor, 5
- checkBounds, 6
- encode, 6
- ExpressionSet, 23
- filter\_properties, 7, 12, 15, 32, 36, 42
- forget\_gemma\_memoised, 4, 7, 13, 15, 17–21, 23, 25–30, 32–35, 37–39, 41, 42, 44–46, 48, 65, 66, 68
- gemma.R, 8
- gemma.R-package (gemma.R), 8
- gemma\_call, 9
- gemma\_kable, 10
- gemma\_memoise, 10
- gemmaCache, 8
- gemmaPath, 9
- get\_all\_pages, 11
- get\_child\_terms, 11
- get\_dataset\_annotations, 16
- get\_dataset\_design, 17
- get\_dataset\_differential\_expression\_analyses, 18, 29, 43, 50
- get\_dataset\_differential\_expression\_analyses(), 28
- get\_dataset\_expression, 19
- get\_dataset\_expression\_for\_genes, 20
- get\_dataset\_object, 8, 22
- get\_dataset\_platforms, 23
- get\_dataset\_processed\_expression, 19, 24, 26, 61
- get\_dataset\_quantitation\_types, 25, 27
- get\_dataset\_raw\_expression, 26, 26, 61
- get\_dataset\_samples, 27, 48
- get\_datasets, 7, 12, 45, 65
- get\_datasets\_by\_ids, 14
- get\_differential\_expression\_values, 8, 19, 28, 43, 50, 52, 55
- get\_gene\_differential\_expression\_values, 31
- get\_gene\_go\_terms, 32
- get\_gene\_locations, 33
- get\_gene\_probes, 35
- get\_genes, 30
- get\_platform\_annotations, 8, 38
- get\_platform\_datasets, 39
- get\_platform\_element\_genes, 40
- get\_platforms\_by\_ids, 7, 36
- get\_result\_sets, 32, 42, 42, 54
- get\_taxa, 43
- get\_taxa\_by\_ids, 44, 45, 66
- get\_taxon\_datasets, 45
- isEmpty, 47
- kable, 10
- make\_design, 48
- memoise, 48
- nullCheck, 49
- process\_search, 64
- processAnnotations, 49
- processCharacteristicValueObject, 50
- processDatasetResultSets, 50
- processDatasets, 51
- processDEA, 52
- processDEcontrasts, 53
- processDEMatrix, 53
- processDesignMatrix, 54
- processDifferentialExpressionAnalysisResultByGeneValue, 54
- processDifferentialExpressionAnalysisResultSetValueObject, 55
- processElements, 55
- processExpressionMatrix, 56
- processFile, 57
- processGemmaArray, 57
- processGeneLocation, 58
- processGenes, 58
- processG0, 59

- processPlatforms, [60](#)
- processQuantitationTypeValueObject, [61](#)
- processResultSetFactors, [61](#)
- processSamples, [62](#)
- processSearchAnnotations, [63](#)
- processTaxon, [63](#)
  
- search\_annotations, [64](#)
- search\_datasets, [65](#)
- search\_gemma, [68](#)
- set\_gemma\_user, [69](#)
- setGemmaPath, [69](#)
- subset\_factorValues, [70](#)
- SummarizedExperiment, [23](#)
  
- update\_result, [70](#)
  
- validateBoolean, [71](#)
- validateID, [71](#)
- validateLimit, [72](#)
- validateOptionalID, [72](#)
- validateOptionalQuery, [73](#)
- validateOptionalTaxon, [73](#)
- validatePositiveInteger, [74](#)
- validateQuery, [74](#)
- validateResultType, [75](#)
- validateSingleID, [75](#)
- validateSort, [76](#)
- validateTaxa, [76](#)
- validateTaxon, [77](#)