Package: gwasrapidd (via r-universe)

May 28, 2026

Type Package

Title 'REST' 'API' Client for the 'NHGRI'-'EBI' 'GWAS' Catalog

Version 0.99.18

Description 'GWAS' R 'API' Data Download. This package provides easy access to the 'NHGRI'-'EBI' 'GWAS' Catalog data by accessing the 'REST' 'API' <https://www.ebi.ac.uk/gwas/rest/docs/api/>.

Depends R (>= 3.2.3)

License MIT + file LICENSE

URL https://github.com/ramiromagno/gwasrapidd, https://rmagno.eu/gwasrapidd/

BugReports https://github.com/ramiromagno/gwasrapidd/issues

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Config/Needs/website patterninstitute/chic

Imports magrittr, httr, urltools, pingr, stringr, dplyr, jsonlite, purrr, tibble, glue, tidyr (> 0.8.99), assertthat, rlang, methods, lubridate, plyr, testthat, utils, progress, writexl

Suggests httptest, spelling, knitr, rmarkdown, bookdown

Collate 'browser.R' 'cc.R' 'class-associations.R' 'class-studies.R' 'class-traits.R' 'class-variants.R' 'data.R' 'ebi_server.R' 'generics.R' 'get_associations.R' 'get_metadata.R' 'get_studies.R' 'get_traits.R' 'get_variants.R' 'gwasrapidd-package.R' 'id_mapping.R' 'list_joins.R' 'missing.R' 'parse-associations.R' 'parse-studies.R' 'parse-traits.R' 'parse-utils.R' 'parse-variants.R' 'post-studies.R' 'post-traits.R' 'post-variants.R' 'recursive_apply.R' 'request.R' 's4-utils.R' 'sure.R' 'tests.R' 'utils-pipe.R' 'utils.R' 'wrappers.R' 'write_xlsx.R'

VignetteBuilder knitr

biocViews ThirdPartyClient, BiomedicalInformatics, GenomeWideAssociation, SNP

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://mrcieu.r-universe.dev

Date/Publication 2025-05-27 10:52:42 UTC

RemoteUrl https://github.com/ramiromagno/gwasrapidd

RemoteRef HEAD

RemoteSha 86b4acfcc918b2f2d76c11ae4ecc52be6463c698

Contents

%>% . . . 3
association_to_study . . . 3
association_to_trait . . . 4
association_to_variant . . . 5
associations-class . . . 5
bind . . . 7
cytogenetic_bands . . . 8
exists_variant . . . 9
gc_examples . . . 10
get_associations . . . 11
get_child_efo . . . 12
get_metadata . . . 13
get_studies . . . 14
get_traits . . . 16
get_variants . . . 17
is_ebi_reachable . . . 19
n . . . 20
open_in_dbsnp . . . 21
open_in_gtex . . . 21
open_in_gwas_catalog . . . 22
open_in_pubmed . . . 23
setop . . . 23
studies-class . . . 25
study_to_association . . . 27
study_to_trait . . . 28
study_to_variant . . . 28
subset-associations . . . 29
subset-studies . . . 30
subset-traits . . . 31
subset-variants . . . 32
trait_to_association . . . 33
trait_to_study . . . 34
trait_to_variant . . . 34

traits-class 35
variant_to_association 35
variant_to_study 36
variant_to_trait 37
variants-class 38
write_xlsx 39

Index **40**

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Value

The same as the rhs.

Examples

```
c(1,2,3) %>% mean()
```

`association_to_study` *Map an association id to a study id*

Description

Map an association accession identifier to a study accession identifier.

Usage

```
association_to_study(association_id, verbose = FALSE, warnings = TRUE)
```

Arguments

`association_id`	A character vector of association accession identifiers.
`verbose`	Whether the function should be verbose about the different queries or not.
`warnings`	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the association identifier and the second column is the study identifier.

Examples

```
## Not run:
# Map GWAS association identifiers to study identifiers
association_to_study(c('24300097', '24299759'))

## End(Not run)
```

`association_to_trait` *Map an association id to an EFO trait id*

Description

Map an association accession identifier to an EFO trait id.

Usage

```
association_to_trait(association_id, verbose = FALSE, warnings = TRUE)
```

Arguments

`association_id`	A character vector of association accession identifiers.
`verbose`	Whether the function should be verbose about the different queries or not.
`warnings`	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the association identifier and the second column is the EFO trait identifier.

Examples

```
## Not run:
# Map GWAS association identifiers to EFO trait identifiers
association_to_trait(c('24300097', '24299759'))

## End(Not run)
```

association_to_variant *Map an association id to a variant id*

Description

Map an association accession identifier to a variant identifier.

Usage

```
association_to_variant(association_id, verbose = FALSE, warnings = TRUE)
```

Arguments

association_id	A character vector of association accession identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the association identifier and the second column is the variant identifier.

Examples

```
## Not run:
# Map GWAS association identifiers to variant identifiers
association_to_variant(c('24300097', '24299759'))

## End(Not run)
```

associations-class *An S4 class to represent a set of GWAS Catalog associations*

Description

The association object consists of six slots, each a table (tibble), that combined form a relational database of a subset of GWAS Catalog associations. Each association is an observation (row) in the associations table — main table. All tables have the column association_id as primary key.

Slots

`associations` A `tibble` listing associations. Columns:

- **association_id** GWAS Catalog association accession identifier, e.g., `"20250"`.
- **pvalue** Reported p-value for strongest variant risk or effect allele.
- **pvalue_description** Information describing context of p-value.
- **pvalue_mantissa** Mantissa of p-value.
- **pvalue_exponent** Exponent of p-value.
- **multiple_snp_haplotype** Whether the association is for a multi-SNP haplotype.
- **snp_interaction** Whether the association is for a SNP-SNP interaction.
- **snp_type** Whether the SNP has previously been reported. Either `'known'` or `'novel'`.
- **risk_frequency** Reported risk/effect allele frequency associated with strongest SNP in controls.
- **standard_error** Standard error of the effect size.
- **range** Reported 95% confidence interval associated with strongest SNP risk allele, along with unit in the case of beta coefficients. If 95% CIs have not been not reported, these are estimated using the standard error, when available.
- **or_per_copy_number** Reported odds ratio (OR) associated with strongest SNP risk allele. Note that all ORs included in the Catalog are >1.
- **beta_number** Beta coefficient associated with strongest SNP risk allele.
- **beta_unit** Beta coefficient unit.
- **beta_direction** Beta coefficient direction, either `'decrease'` or `'increase'`.
- **beta_description** Additional beta coefficient comment.
- **last_mapping_date** Last time this association was mapped to Ensembl.
- **last_update_date** Last time this association was updated.

`loci` A `tibble` listing loci. Columns:

- **association_id** GWAS Catalog association accession identifier, e.g., `"20250"`.
- **locus_id** A locus identifier referring to a single variant locus or to a multi-loci entity such as a multi-SNP haplotype.
- **haplotype_snp_count** Number of variants per locus. Most loci are single-SNP loci, i.e., there is a one to one relationship between a variant and a `locus_id` (`haplotype_snp_count == NA`). There are however cases of associations involving multiple loci at once, such as SNP-SNP interactions and multi-SNP haplotypes. This is signalled in the columns: `multiple_snp_haplotype` and `snp_interaction` with value `TRUE`.
- **description** Description of the locus identifier, e.g., `'Single variant'`, `SNP x SNP interaction`, or `3-SNP Haplotype`.

`risk_alleles` A `tibble` listing risk alleles. Columns:

- **association_id** GWAS Catalog association accession identifier, e.g., `"20250"`.
- **locus_id** A locus identifier referring to a single variant locus or to a multi-loci entity such as a multi-SNP haplotype.
- **variant_id** Variant identifier, e.g., `'rs1333048'`.
- **risk_allele** Risk allele or effect allele.
- **risk_frequency** Reported risk/effect allele frequency associated with strongest SNP in controls (if not available among all controls, among the control group with the largest sample size). If the associated locus is a haplotype the haplotype frequency will be extracted.

genome_wide Whether this variant allele has been part of a genome-wide study or not.

limited_list Undocumented.

`genes` A `tibble` listing author reported genes. Columns:

association_id GWAS Catalog association accession identifier, e.g., `"20250"`.

locus_id A locus identifier referring to a single variant locus or to a multi-loci entity such as a multi-SNP haplotype.

gene_name Gene symbol according to HUGO Gene Nomenclature (HGNC).

`ensembl_ids` A `tibble` listing Ensembl gene identifiers. Columns:

association_id GWAS Catalog association accession identifier, e.g., `"20250"`.

locus_id A locus identifier referring to a single variant locus or to a multi-loci entity such as a multi-SNP haplotype.

gene_name Gene symbol according to HUGO Gene Nomenclature (HGNC).

ensembl_id The Ensembl identifier of an Ensembl gene, see Section Gene annotation in Ensembl for more information.

`entrez_ids` A `tibble` listing Entrez gene identifiers. Columns:

association_id GWAS Catalog association accession identifier, e.g., `"20250"`.

locus_id A locus identifier referring to a single variant locus or to a multi-loci entity such as a multi-SNP haplotype.

gene_name Gene symbol according to HUGO Gene Nomenclature (HGNC).

entrez_id The Entrez identifier of a gene, see ref. doi:10.1093/nar/gkq1237 for more information.

`bind` *Bind GWAS Catalog objects*

Description

Binds together GWAS Catalog objects of the same class. Note that `bind()` preserves duplicates whereas `union` does not.

Usage

```
bind(x, ...)
```

Arguments

`x`	An object of class: studies, associations, variants, or traits.
`...`	Objects of the same class as `x`.

Value

An object of the same class as `x`.

Examples

```
# Join two studies objects.
bind(studies_ex01, studies_ex02)

# Join two associations objects.
bind(associations_ex01, associations_ex02)

# Join two variants objects.
bind(variants_ex01, variants_ex02)

# Join two traits objects.
bind(traits_ex01, traits_ex02)
```

`cytogenetic_bands` *GRCh38 human cytogenetic bands.*

Description

A dataset containing the GRCh38 human cytogenetic bands and their genomic coordinates.

Usage

```
cytogenetic_bands
```

Format

A data frame with 862 rows and 8 variables:

cytogenetic_band Cytogenetic band name. See *Cytogenetic Nomenclature* below.

chromosome Chromosome name: 1 through 22 (the autosomes), X or Y.

start Genomic start position of the cytogenetic band. Starts at 1.

end Genomic end position of the cytogenetic band. End position is included in the band interval.

length Length of the genomic interval of cytogenetic band.

assembly Assembly version, should be 'GRCh38'.

stain Giemsa stain results: Giemsa negative, `'gneg'`; Giemsa positive, of increasing intensities, `'gpos25'`, `'gpos50'`, `'gpos75'`, and `'gpos100'`; centromeric region, `'acen'`; heterochromatin, either pericentric or telomeric, `'gvar'`; and short arm of acrocentric chromosomes 13, 14, 15, 21, and 22 are coded as `'stalk'`.

last_download_date Time stamp of last time this dataset was downloaded from Ensembl.

Details

Genomic coordinates are for fully closed intervals.

Cytogenetic Nomenclature

Cytogenetic bands are numbered from the centromere outwards in both directions towards the telomeres on the shorter p arm and the longer q arm.

The first number or letter represents the chromosome. Chromosomes 1 through 22 (the autosomes) are designated by their chromosome number. The sex chromosomes are designated by X or Y. The next letter represents the arm of the chromosome: p or q.

The numbers cannot be read in the normal decimal numeric system e.g. 36, but rather 3-6 (region 3 band 6). Counting starts at the centromere as region 1 (or 1-0), to 11 (1-1) to 21 (2-1) to 22 (2-2) etc. Subbands are added in a similar way, e.g. 21.1 to 21.2, if the bands are small or only appear at a higher resolution.

Source

https://rest.ensembl.org/info/assembly/homo_sapiens?content-type=application/json&bands=1

---

exists_variant *Check if a variant exists in the Catalog.*

Description

This function attempts to get a variant by its variant identifier and checks the response code. If the response code is 200 then the response has been successful, meaning that the variant does exist in the GWAS Catalog. If the response is 404 then the variant is not found in the Catalog database. Other errors are mapped to NA.

Usage

```
exists_variant(variant_id = NULL, verbose = FALSE, page_size = 20L)
```

Arguments

variant_id	A character vector of GWAS Catalog variant identifiers.
verbose	Whether the function should be verbose about the different queries or not.
page_size	An integer scalar indicating the page value to be used in the JSON requests, can be between 1 and 1000.

Value

A named logical vector, TRUE indicates that the variant does exist in the Catalog, FALSE otherwise. NA codes other types of errors. The names of the vector are the variant identifiers passed as variant_id.

Examples

```
exists_variant('rs12345')

exists_variant('rs11235813')
```

`gc_examples` *gwasrapidd entities' examples*

Description

These are examples of GWAS Catalog entities shipped with gwasrapidd:

Usage

```
studies_ex01

studies_ex02

associations_ex01

associations_ex02

variants_ex01

variants_ex02

traits_ex01

traits_ex02
```

Format

studies_ex01 An S4 studies object of 2 studies: `'GCST001585'` and `'GCST003985'`.

studies_ex02 An S4 studies object of 2 studies: `'GCST001585'` and `'GCST006655'`.

associations_ex01 An S4 associations object of 4 associations: `'22509'`, `'22505'`, `'19537565'` and `'19537593'`.

associations_ex02 An S4 associations object of 3 associations: `'19537593'`, `'31665940'` and `'34944736'`.

variants_ex01 An S4 variants object of 3 variants: `'rs146992477'`, `'rs56261590'` and `'rs4725504'`.

variants_ex02 An S4 variants object of 4 variants: `'rs56261590'`, `'rs4725504'`, `'rs11099757'` and `'rs16871509'`.

traits_ex01 An S4 traits object of 3 traits: `'EFO_0004884'`, `'EFO_0004343'` and `'EFO_0005299'`.

traits_ex02 An S4 traits object of 4 traits: `'EFO_0007845'`, `'EFO_0004699'`, `'EFO_0004884'` and `'EFO_0004875'`.

An object of class `studies` of length 1.

An object of class `associations` of length 1.

An object of class `associations` of length 1.

An object of class `variants` of length 1.

An object of class `variants` of length 1.

An object of class `traits` of length 1.

An object of class `traits` of length 1.

`get_associations` *Get GWAS Catalog associations*

Description

Retrieves associations via the NHGRI-EBI GWAS Catalog REST API. The REST API is queried multiple times with the criteria passed as arguments (see below). By default all associations that match the criteria supplied in the arguments are retrieved: this corresponds to the default option `set_operation` set to `'union'`. If you rather have only the associations that match simultaneously all criteria provided, then set `set_operation` to `'intersection'`.

Usage

```
get_associations(
  study_id = NULL,
  association_id = NULL,
  variant_id = NULL,
  efo_id = NULL,
  pubmed_id = NULL,
  efo_trait = NULL,
  set_operation = "union",
  interactive = TRUE,
  verbose = FALSE,
  warnings = TRUE
)
```

Arguments

`study_id`	A `character` vector of GWAS Catalog study accession identifiers.
`association_id`	A `character` vector of GWAS Catalog association identifiers.
`variant_id`	A `character` vector of GWAS Catalog variant identifiers.
`efo_id`	A character vector of EFO identifiers.
`pubmed_id`	An `integer` vector of PubMed identifiers.

efo_trait	A character vector of EFO trait descriptions, e.g., 'uric acid measurement'.
set_operation	Either 'union' or 'intersection'. This tells how associations retrieved by different criteria should be combined: 'union' binds together all results removing duplicates and 'intersection' only keeps same associations found with different criteria.
interactive	A logical. If all associations are requested, whether to ask interactively if we really want to proceed.
verbose	A logical indicating whether the function should be verbose about the different queries or not.
warnings	A logical indicating whether to print warnings, if any.

Details

Please note that all search criteria are vectorised, thus allowing for batch mode search, e.g., one can search by multiple variant identifiers at once by passing a vector of identifiers to variant_id.

Value

An associations object.

Examples

```
## Not run:
# Get an association by study identifier
get_associations(study_id = 'GCST001085', warnings = FALSE)

# Get an association by association identifier
get_associations(association_id = '25389945', warnings = FALSE)

# Get associations by variant identifier
get_associations(variant_id = 'rs3798440', warnings = FALSE)

# Get associations by EFO trait identifier
get_associations(efo_id = 'EFO_0005537', warnings = FALSE)

## End(Not run)
```

---

get_child_efo *Get all child terms of this trait in the EFO hierarchy*

Description

Get all child terms of this trait in the EFO hierarchy

Usage

```
get_child_efo(
  efo_id,
  verbose = FALSE,
  warnings = TRUE,
  page_size = 20L,
  progress_bar = TRUE
)
```

Arguments

efo_id	A character vector of EFO identifiers.
verbose	A logical indicating whether the function should be verbose about the different queries or not.
warnings	A logical indicating whether to print warnings, if any.
page_size	An integer scalar indicating the page value to be used in the JSON requests, can be between 1 and 1000.
progress_bar	Whether to show a progress bar as the paginated resources are retrieved.

Value

A named list whose values are character vectors of EFO identifiers.

Examples

```
## Not run:
get_child_efo(c('EFO_0004884', 'EFO_0004343', 'EFO_0005299'))

## End(Not run)
```

---

get_metadata *Get GWAS Catalog metadata*

Description

Provides a list of the resources the GWAS Catalog data is currently mapped against: Ensembl release number, Genome build version and dbSNP version.In addition, the date since this combination of resource versions has been in use is also returned.

Usage

```
get_metadata(verbose = FALSE, warnings = TRUE)
```

Arguments

verbose	Whether to be chatty.
warnings	Whether to trigger a warning if the request is not successful.

Value

A named `list` whose `names` are:

- `ensembl_release_number`: Ensembl release number;
- `genome_build_version`: Genome build version;
- `dbsnp_version`: dbSNP version.
- `usage_start_date`: Date since this combination of resource versions has been in use.

Examples

```
## Not run:
get_metadata(warnings = FALSE)

## End(Not run)
```

---

get_studies *Get GWAS Catalog studies*

Description

Retrieves studies via the NHGRI-EBI GWAS Catalog REST API. The REST API is queried multiple times with the criteria passed as arguments (see below). By default all studies that match the criteria supplied in the arguments are retrieved: this corresponds to the default option `set_operation` set to `'union'`. If you rather have only the studies that match simultaneously all criteria provided, then set `set_operation` to `'intersection'`.

Usage

```
get_studies(
  study_id = NULL,
  association_id = NULL,
  variant_id = NULL,
  efo_id = NULL,
  pubmed_id = NULL,
  user_requested = NULL,
  full_pvalue_set = NULL,
  efo_uri = NULL,
  efo_trait = NULL,
  reported_trait = NULL,
  set_operation = "union",
  interactive = TRUE,
  verbose = FALSE,
  warnings = TRUE
)
```

Arguments

study_id	A character vector of GWAS Catalog study accession identifiers.
association_id	A character vector of GWAS Catalog association identifiers.
variant_id	A character vector of GWAS Catalog variant identifiers.
efo_id	A character vector of EFO identifiers.
pubmed_id	An integer vector of PubMed identifiers.
user_requested	A logical (scalar!) indicating to retrieve either studies requested by users of the Catalog (TRUE) or otherwise (FALSE).
full_pvalue_set	A logical (scalar!) indicating to retrieve studies with full summary statistics (TRUE) or studies without it (FALSE).
efo_uri	A character vector of EFO URIs.
efo_trait	A character vector of EFO trait descriptions, e.g., 'uric acid measurement'.
reported_trait	A character vector of phenotypic traits as reported by the original authors of the study.
set_operation	Either 'union' or 'intersection'. This tells how studies retrieved by different criteria should be combined: 'union' binds together all results removing duplicates and 'intersection' only keeps same studies found with different criteria.
interactive	A logical. If all studies are requested, whether to ask interactively if we really want to proceed.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Details

Please note that all search criteria are vectorised, thus allowing for batch mode search, e.g., one can search by multiple variant identifiers at once by passing a vector of identifiers to variant_id.

Value

A studies object.

Examples

```
## Not run:
# Get a study by its accession identifier
get_studies(study_id = 'GCST001085', warnings = FALSE)

# Get a study by association identifier
get_studies(association_id = '25389945', warnings = FALSE)

# Get studies by variant identifier
get_studies(variant_id = 'rs3798440', warnings = FALSE)

# Get studies by EFO trait identifier
```

```
get_studies(efo_id = 'EFO_0005537', warnings = FALSE)

## End(Not run)
```

`get_traits` *Get GWAS Catalog EFO traits*

Description

Retrieves traits via the NHGRI-EBI GWAS Catalog REST API. The REST API is queried multiple times with the criteria passed as arguments (see below). By default all traits that match the criteria supplied in the arguments are retrieved: this corresponds to the default option `set_operation` set to `'union'`. If you rather have only the traits that match simultaneously all criteria provided, then set `set_operation` to `'intersection'`.

Usage

```
get_traits(
  study_id = NULL,
  association_id = NULL,
  efo_id = NULL,
  pubmed_id = NULL,
  efo_uri = NULL,
  efo_trait = NULL,
  set_operation = "union",
  verbose = FALSE,
  warnings = TRUE
)
```

Arguments

`study_id`	A `character` vector of GWAS Catalog study accession identifiers.
`association_id`	A `character` vector of GWAS Catalog association identifiers.
`efo_id`	A character vector of EFO identifiers.
`pubmed_id`	An `integer` vector of PubMed identifiers.
`efo_uri`	A `character` vector of EFO URIs.
`efo_trait`	A `character` vector of EFO trait descriptions, e.g., `'uric acid measurement'`.
`set_operation`	Either `'union'` or `'intersection'`. This tells how traits retrieved by different criteria should be combined: `'union'` binds together all results removing duplicates and `'intersection'` only keeps same traits found with different criteria.
`verbose`	A `logical` indicating whether the function should be verbose about the different queries or not.
`warnings`	A `logical` indicating whether to print warnings, if any.

Details

Please note that all search criteria are vectorised, thus allowing for batch mode search, e.g., one can search by multiple trait identifiers at once by passing a vector of identifiers to `efo_id`.

Value

A traits object.

Examples

```
## Not run:
# Get traits by study identifier
get_traits(study_id = 'GCST001085', warnings = FALSE)

# Get traits by association identifier
get_traits(association_id = '25389945', warnings = FALSE)

# Get a trait by its EFO identifier
get_traits(efo_id = 'EFO_0005537', warnings = FALSE)

## End(Not run)
```

`get_variants` *Get GWAS Catalog variants*

Description

Retrieves variants via the NHGRI-EBI GWAS Catalog REST API. The REST API is queried multiple times with the criteria passed as arguments (see below). By default all variants that match the criteria supplied in the arguments are retrieved: this corresponds to the default option `set_operation` set to `'union'`. If you rather have only the variants that match simultaneously all criteria provided, then set `set_operation` to `'intersection'`.

Usage

```
get_variants(
  study_id = NULL,
  association_id = NULL,
  variant_id = NULL,
  efo_id = NULL,
  pubmed_id = NULL,
  genomic_range = NULL,
  cytogenetic_band = NULL,
  gene_name = NULL,
  efo_trait = NULL,
  reported_trait = NULL,
  set_operation = "union",
```

```
  interactive = TRUE,
  std_chromosomes_only = TRUE,
  verbose = FALSE,
  warnings = TRUE
)
```

Arguments

study_id	A character vector of GWAS Catalog study accession identifiers.
association_id	A character vector of GWAS Catalog association identifiers.
variant_id	A character vector of GWAS Catalog variant identifiers.
efo_id	A character vector of EFO identifiers.
pubmed_id	An integer vector of PubMed identifiers.
genomic_range	A named list of three vectors: **chromosome** A character vector of chromosome names of the form 1–22, X or Y. **start** A numeric vector of start positions, starting at 1. **end** A numeric vector of end positions. The three vectors need to be of the same length so that chromosome names, start and end positions can be matched by position.
cytogenetic_band	A character vector of cytogenetic bands of the form '1p36.11'.
gene_name	Gene symbol according to HUGO Gene Nomenclature (HGNC).
efo_trait	A character vector of EFO trait descriptions, e.g., 'uric acid measurement'.
reported_trait	A character vector of phenotypic traits as reported by the original authors of the study.
set_operation	Either 'union' or 'intersection'. This tells how variants retrieved by different criteria should be combined: 'union' binds together all results removing duplicates and 'intersection' only keeps same variants found with different criteria.
interactive	A logical. If all variants are requested, whether to ask interactively if we really want to proceed.
std_chromosomes_only	Whether to return only variants mapped to standard chromosomes: 1 thru 22, X, Y, and MT.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Details

Please note that all search criteria are vectorised, thus allowing for batch mode search, e.g., one can search by multiple variant identifiers at once by passing a vector of identifiers to variant_id.

Value

A variants object.

Examples

```
# Get variants by study identifier
get_variants(study_id = 'GCST001085', warnings = FALSE)

# Get a variant by its identifier
## Not run:
get_variants(variant_id = 'rs3798440', warnings = FALSE)

## End(Not run)
```

`is_ebi_reachable` *Is the GWAS Catalog REST API server reachable?*

Description

Check if the EBI server where the GWAS Catalog REST API server is running is reachable. This function attempts to connect to https://www.ebi.ac.uk, returning `TRUE` on success, and `FALSE` otherwise. Set `chatty = TRUE` for a step by step description of the connection attempt.

Usage

```
is_ebi_reachable(url = "https://www.ebi.ac.uk", port = 443L, chatty = FALSE)
```

Arguments

`url`	NHGRI-EBI GWAS Catalog server URL. Default is `https://www.ebi.ac.uk`. You should not need to change this parameter.
`port`	Network port on which to ping the server. You should not need to change this parameter.
`chatty`	Whether to be verbose (`TRUE`) or not (`FALSE`).

Value

A logical value: `TRUE` if EBI server is reachable, `FALSE` otherwise.

Examples

```
# Check if the GWAS Catalog Server is reachable
is_ebi_reachable() # Returns TRUE or FALSE.

# Check if the GWAS Catalog Server is reachable
# and show exactly at what step is it failing (if that is the case)
is_ebi_reachable(chatty = TRUE)
```

n *Number of GWAS Catalog entities*

Description

This function returns the number of unique entities in a GWAS Catalog object.

Usage

```
n(x, unique = FALSE)

## S4 method for signature 'studies'
n(x, unique = FALSE)

## S4 method for signature 'associations'
n(x, unique = FALSE)

## S4 method for signature 'variants'
n(x, unique = FALSE)

## S4 method for signature 'traits'
n(x, unique = FALSE)
```

Arguments

x	A studies, an associations, a variants, or a traits object.
unique	Whether to count only unique entries (TRUE) or not (FALSE).

Value

An integer scalar.

Examples

```
# Determine number of studies
n(studies_ex01)

# Determine number of associations
n(associations_ex01)

# Determine number of variants
n(variants_ex01)

# Determine number of traits
n(traits_ex01)
```

open_in_dbsnp *Browse dbSNP from SNP identifiers.*

Description

This function launches the web browser at dbSNP and opens a tab for each SNP identifier.

Usage

```
open_in_dbsnp(variant_id)
```

Arguments

variant_id	A variant identifier, a character vector.

Value

Returns TRUE if successful. Note however that this function is run for its side effect.

Examples

```
open_in_dbsnp('rs56261590')
```

open_in_gtex *Browse GTEx from SNP identifiers.*

Description

This function launches the web browser at the GTEx Portal and opens a tab for each SNP identifier.

Usage

```
open_in_gtex(variant_id)
```

Arguments

variant_id	A variant identifier, a character vector.

Value

Returns TRUE if successful. Note however that this function is run for its side effect.

Examples

```
open_in_gtex('rs56261590')
```

`open_in_gwas_catalog` *Browse GWAS Catalog entities from the GWAS Web Graphical User Interface*

Description

This function launches the web browser and opens a tab for each identifier on the GWAS web graphical user interface: https://www.ebi.ac.uk/gwas.

Usage

```
open_in_gwas_catalog(
  identifier,
 gwas_catalog_entity = c("study", "variant", "trait", "gene", "region", "publication")
)
```

Arguments

`identifier`	A vector of identifiers. The identifiers can be: study accession identifiers, variant identifiers, EFO trait identifiers, gene symbol names, cytogenetic regions, or PubMed identifiers.
`gwas_catalog_entity`	Either `'study'` (default), `'variant'`, `'trait'`, `'gene'`, `'region'` or `'publication'`, a scalar character. This argument indicates the type of the identifiers passed in `identifier`.

Value

Returns `TRUE` if successful, or `FALSE` otherwise. But note that this function is run for its side effect.

Examples

```
# Open studies in GWAS Web Graphical User Interface
open_in_gwas_catalog(c('GCST000016', 'GCST001115'))

# Open variants
open_in_gwas_catalog(c('rs146992477', 'rs56261590'),
  gwas_catalog_entity = 'variant')

# Open EFO traits
open_in_gwas_catalog(c('EFO_0004884', 'EFO_0004343'),
  gwas_catalog_entity = 'trait')

# Open genes
open_in_gwas_catalog(c('DPP6', 'MCCC2'),
  gwas_catalog_entity = 'gene')

# Open cytogenetic regions
open_in_gwas_catalog(c('2q37.1', '1p36.11'),
```

```
  gwas_catalog_entity = 'region')

# Open publications
open_in_gwas_catalog(c('25533513', '24376627'),
  gwas_catalog_entity = 'publication')
```

`open_in_pubmed` *Browse PubMed from PubMed identifiers.*

Description

This function launches the web browser and opens a tab for each PubMed citation.

Usage

```
open_in_pubmed(pubmed_id)
```

Arguments

`pubmed_id`	A PubMed identifier, either a character or an integer vector.

Value

Returns `TRUE` if successful. Note however that this function is run for its side effect.

Examples

```
open_in_pubmed(c('26301688', '30595370'))
```

`setop` *Set operations on GWAS Catalog objects.*

Description

Performs set union, intersection, and (asymmetric!) difference on two objects of either class studies, associations, variants, or traits. Note that `union()` removes duplicated entities, whereas `bind()` does not.

Usage

```
union(x, y, ...)

intersect(x, y, ...)

setdiff(x, y, ...)

setequal(x, y, ...)
```

Arguments

x, y	Objects of either class studies, associations, variants, or traits.
...	other arguments passed on to methods.

Value

An object of the same class as x and y, i.e., studies, associations, variants, or traits.

Examples

```
#
# union()
#
# Combine studies and remove duplicates
union(studies_ex01, studies_ex02)

# Combine associations and remove duplicates
union(associations_ex01, associations_ex02)

# Combine variants and remove duplicates
union(variants_ex01, variants_ex02)

# Combine traits and remove duplicates
union(traits_ex01, traits_ex02)

#
# intersect()
#
# Intersect common studies
intersect(studies_ex01, studies_ex02)

# Intersect common associations
intersect(associations_ex01, associations_ex02)

# Intersect common variants
intersect(variants_ex01, variants_ex02)

# Intersect common traits
intersect(traits_ex01, traits_ex02)

#
# setdiff()
#
# Remove studies from ex01 that are also present in ex02
setdiff(studies_ex01, studies_ex02)

# Remove associations from ex01 that are also present in ex02
setdiff(associations_ex01, associations_ex02)

# Remove variants from ex01 that are also present in ex02
setdiff(variants_ex01, variants_ex02)
```

```
# Remove traits from ex01 that are also present in ex02
setdiff(traits_ex01, traits_ex02)

#
# setequal()
#
# Compare two studies objects
setequal(studies_ex01, studies_ex01)
setequal(studies_ex01, studies_ex02)

# Compare two associations objects
setequal(associations_ex01, associations_ex01)
setequal(associations_ex01, associations_ex02)

# Compare two variants objects
setequal(variants_ex01, variants_ex01)
setequal(variants_ex01, variants_ex02)

# Compare two traits objects
setequal(traits_ex01, traits_ex01)
setequal(traits_ex01, traits_ex02)
```

`studies-class` *An S4 class to represent a set of GWAS Catalog studies*

Description

The studies object consists of eight slots, each a table (`tibble`), that combined form a relational database of a subset of GWAS Catalog studies. Each study is an observation (row) in the `studies` table — main table. All tables have the column `study_id` as primary key.

Slots

`studies` **study_id** GWAS Catalog study accession identifier, e.g., `"GCST002735"`.

reported_trait Phenotypic trait as reported by the authors of the study, e.g. `"Breast cancer"`.

initial_sample_size Free text description of the initial cohort sample size.

replication_sample_size Free text description of the replication cohort sample size.

gxe Whether the study investigates a gene-environment interaction.

gxg Whether the study investigates a gene-gene interaction.

snp_count Number of variants passing quality control.

qualifier Qualifier of number of variants passing quality control.

imputed Whether variants were imputed.

pooled Whether samples were pooled.

study_design_comment Any other relevant study design information.

full_pvalue_set Whether full summary statistics are available for this study.

user_requested Whether the addition of this study to the GWAS Catalog was requested by a user.

`genotyping_techs` A `tibble` listing genotyping technologies employed in each study. Columns:

study_id GWAS Catalog study accession identifier.

genotyping_technology Genotyping technology employed, e.g. `"Exome genotyping array"`, `"Exome-wide sequencing"`, `"Genome-wide genotyping array"`, `"Genome-wide sequencing"`, or `"Targeted genotyping array"`.

`platforms` A `tibble` listing platforms used per study.

study_id GWAS Catalog study accession identifier.

manufacturer Platform manufacturer, e.g., `"Affymetrix"`, `"Illumina"`, or `"Perlegen"`.

`ancestries` A `tibble` listing ancestry of samples used in each study.

study_id GWAS Catalog study accession identifier.

ancestry_id Ancestry identifier.

type Stage of the ancestry sample: either `'initial'` or `'replication'`.

number_of_individuals Number of individuals comprising this ancestry sample.

`ancestral_groups` A `tibble` listing ancestral groups used in each ancestry.

study_id GWAS Catalog study accession identifier.

ancestry_id Ancestry identifier.

ancestral_group Genetic ancestry groups present in the sample.

`countries_of_origin` A `tibble` listing countries of origin of samples.

study_id GWAS Catalog study accession identifier.

ancestry_id Ancestry identifier.

country_name Country name, according to The United Nations M49 Standard of Geographic Regions.

major_area Region name, according to The United Nations M49 Standard of Geographic Regions.

region Sub-region name, according to The United Nations M49 Standard of Geographic Regions.

`countries_of_recruitment` A `tibble` listing countries of recruitment of samples.

study_id GWAS Catalog study accession identifier.

ancestry_id Ancestry identifier.

country_name Country name, according to The United Nations M49 Standard of Geographic Regions.

major_area Region name, according to The United Nations M49 Standard of Geographic Regions.

region Sub-region name, according to The United Nations M49 Standard of Geographic Regions.

`publications` A `tibble` listing publications associated with each study.

study_id GWAS Catalog study accession identifier.

pubmed_id PubMed identifier.

publication_date Publication date (online date if available) formatted as `ymd`.

publication Abbreviated journal name.

title Publication title.

author_fullname Last name and initials of first author.

author_orcid Author's ORCID iD (Open Researcher and Contributor ID).

`study_to_association` *Map a study id to an association id*

Description

Map a study accession identifier to an association accession identifier.

Usage

```
study_to_association(study_id, verbose = FALSE, warnings = TRUE)
```

Arguments

`study_id`	A character vector of study accession identifiers.
`verbose`	Whether the function should be verbose about the different queries or not.
`warnings`	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the study identifier and the second column is the association identifier.

Examples

```
## Not run:
# Map GWAS study identifiers to association identifiers
study_to_association(c('GCST001084', 'GCST001085'))

## End(Not run)
```

study_to_trait *Map a study id to a EFO trait id*

Description

Map a study accession identifier to a EFO trait identifier.

Usage

```
study_to_trait(study_id, verbose = FALSE, warnings = TRUE)
```

Arguments

study_id	A character vector of study accession identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the study identifier and the second column is the EFO identifier.

Examples

```
## Not run:
# Map GWAS study identifiers to EFO trait identifiers
study_to_trait(c('GCST001084', 'GCST001085'))

## End(Not run)
```

study_to_variant *Map a study id to a variant id*

Description

Map a study accession identifier to a variant accession identifier.

Usage

```
study_to_variant(study_id, verbose = FALSE, warnings = TRUE)
```

Arguments

study_id	A character vector of study accession identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the study identifier and the second column is the variant identifier.

Examples

```
## Not run:
# Map GWAS study identifiers to variant identifiers
study_to_variant(c('GCST001084', 'GCST001085'))

## End(Not run)
```

`subset-associations` *Subset an associations object*

Description

You can subset associations by identifier or by position using the `[` operator.

Usage

```
## S4 method for signature 'associations,missing,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'associations,numeric,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'associations,character,missing,missing'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A associations object.
`i`	Position of the identifier or the name of the identifier itself.
`j`	Not used.
`...`	Additional arguments not used here.
`drop`	Not used.

Value

A associations object.

Examples

```
# Subset an associations object by identifier
associations_ex01['22505']

# Or by its position in table associations
associations_ex01[2]

# Keep all associations except the second
associations_ex01[-2]
```

`subset-studies` *Subset a studies object*

Description

You can subset studies by identifier or by position using the `` `[` `` operator.

Usage

```
## S4 method for signature 'studies,missing,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'studies,numeric,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'studies,character,missing,missing'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A studies object.
`i`	Position of the identifier or the name of the identifier itself.
`j`	Not used.
`...`	Additional arguments not used here.
`drop`	Not used.

Value

A studies object.

Examples

```
# Subset a studies object by identifier
studies_ex01['GCST001585']

# Or by its position in table studies
studies_ex01[1]

# Keep all studies except the first
studies_ex01[-1]
```

`subset-traits` *Subset a traits object*

Description

You can subset traits by identifier or by position using the `[` operator.

Usage

```
## S4 method for signature 'traits,missing,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'traits,numeric,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'traits,character,missing,missing'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A traits object.
`i`	Position of the identifier or the name of the identifier itself.
`j`	Not used.
`...`	Additional arguments not used here.
`drop`	Not used.

Value

A traits object.

Examples

```
# Subset a traits object by identifier
traits_ex01['EFO_0004884']

# Or by its position in table traits
traits_ex01[1]

# Keep all traits except the second
traits_ex01[-2]
```

`subset-variants` *Subset a variants object*

Description

You can subset variants by identifier or by position using the `[` operator.

Usage

```
## S4 method for signature 'variants,missing,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'variants,numeric,missing,missing'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'variants,character,missing,missing'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A variants object.
`i`	Position of the identifier or the name of the identifier itself.
`j`	Not used.
`...`	Additional arguments not used here.
`drop`	Not used.

Value

A variants object.

Examples

```
# Subset a variants object by identifier
variants_ex01['rs4725504']

# Or by its position in table variants
variants_ex01[3]

# Keep all variants except the third
variants_ex01[-3]
```

`trait_to_association` *Map an EFO trait id to an association id*

Description

Map an EFO trait id to an association identifier.

Usage

```
trait_to_association(efo_id, verbose = FALSE, warnings = TRUE)
```

Arguments

`efo_id`	A character vector of EFO trait identifiers.
`verbose`	Whether the function should be verbose about the different queries or not.
`warnings`	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the EFO trait identifier and the second column is the association identifier.

Examples

```
## Not run:
# Map EFO trait identifiers to association identifiers
trait_to_association(c('EFO_0005108', 'EFO_0005109'))

## End(Not run)
```

trait_to_study *Map an EFO trait id to a study id*

Description

Map an EFO trait id to a study accession identifier.

Usage

```
trait_to_study(efo_id, verbose = FALSE, warnings = TRUE)
```

Arguments

efo_id	A character vector of EFO trait identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the EFO trait identifier and the second column is the study identifier.

Examples

```
## Not run:
# Map EFO trait identifiers to study identifiers
trait_to_study(c('EFO_0005108', 'EFO_0005109'))

## End(Not run)
```

trait_to_variant *Map an EFO trait id to a variant id*

Description

Map an EFO trait id to a variant identifier.

Usage

```
trait_to_variant(efo_id, verbose = FALSE, warnings = TRUE)
```

Arguments

efo_id	A character vector of EFO trait identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the EFO trait identifier and the second column is the variant identifier.

Examples

```
## Not run:
# Map EFO trait identifiers to variant identifiers
trait_to_variant('EFO_0005229')

## End(Not run)
```

---

traits-class *An S4 class to represent a set of GWAS Catalog EFO traits.*

Description

The traits object consists of one slot only, a table (tibble) of GWAS Catalog EFO traits. Each EFO trait is an observation (row) in the traits table — main table.

Slots

traits A tibble listing EFO traits. Columns:

- **efo_id** EFO identifier.
- **trait** EFO trait description.
- **uri** The full URI of the EFO term.

---

variant_to_association *Map a variant id to an association id*

Description

Map a variant identifier to an association identifier.

Usage

```
variant_to_association(variant_id, verbose = FALSE, warnings = TRUE)
```

Arguments

variant_id	A character vector of variant identifiers.
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the variant identifier and the second column is the association identifier.

Examples

```
## Not run:
# Map GWAS variant identifiers to association identifiers
variant_to_association(c('rs7904579', 'rs138331350'))

## End(Not run)
```

`variant_to_study` *Map a variant id to a study id*

Description

Map a variant identifier to a study accession identifier.

Usage

```
variant_to_study(variant_id, verbose = FALSE, warnings = TRUE)
```

Arguments

`variant_id`	A character vector of variant identifiers.
`verbose`	Whether the function should be verbose about the different queries or not.
`warnings`	Whether to print warnings.

Value

A dataframe of two identifiers. First column is the variant identifier and the second column is the study identifier.

Examples

```
## Not run:
# Map GWAS variant identifiers to study identifiers
variant_to_study(c('rs7904579', 'rs138331350'))

## End(Not run)
```

variant_to_trait *Map a variant id to an EFO trait*

Description

Map a variant identifier to an EFO trait identifier. Variants are first mapped to association identifiers, and then to EFO traits. Set the option keep_association_id to TRUE to keep the intermediate mapping, i.e., the association identifiers.

Usage

```
variant_to_trait(
  variant_id,
  keep_association_id = FALSE,
  verbose = FALSE,
  warnings = TRUE
)
```

Arguments

variant_id	A character vector of variant identifiers.
keep_association_id	Whether to keep the association identifier in the final output (default is FALSE).
verbose	Whether the function should be verbose about the different queries or not.
warnings	Whether to print warnings.

Value

A dataframe of two or three identifiers. If keep_association_id is set to FALSE, the first column is the variant identifier and the second column is the EFO trait identifier, otherwise the variable association_id is also included as the second column.

Examples

```
## Not run:
# Map GWAS variant identifiers to EFO trait identifiers
variant_to_trait(c('rs7904579', 'rs138331350'))

# Map GWAS variant identifiers to EFO trait identifiers
# but keep the intermediate association identifier
variant_to_trait(c('rs7904579', 'rs138331350'), keep_association_id = TRUE)

## End(Not run)
```

variants-class *An S4 class to represent a set of GWAS Catalog variants*

Description

The variants object consists of four slots, each a table (tibble), that combined form a relational database of a subset of GWAS Catalog variants. Each variant is an observation (row) in the variants table — main table. All tables have the column variant_id as primary key.

Slots

variants A tibble listing variants. Columns:

- **variant_id** Variant identifier, e.g., 'rs1333048'.
- **merged** Whether this SNP has been merged with another SNP in a newer genome build.
- **functional_class** Class according to Ensembl's predicted consequences that each variant allele may have on transcripts. See Ensembl Variation - Calculated variant consequences.
- **chromosome_name** Chromosome name.
- **chromosome_position** Chromosome position.
- **chromosome_region** Cytogenetic location.
- **last_update_date** Last time this variant was updated.

genomic_contexts A tibble listing genomic contexts associated with each variant. Columns:

- **variant_id** Variant identifier.
- **gene_name** Gene symbol according to HUGO Gene Nomenclature (HGNC).
- **chromosome_name** Chromosome name.
- **chromosome_position** Chromosome position.
- **distance** Genomic distance between the variant and the gene (in base pairs).
- **is_mapped_gene** Whether this is a mapped gene to this variant. A mapped gene is either an overlapping gene with the variant or the two closest genes upstream and downstream of the variant. Moreover, only genes whose mapping source is 'Ensembl' are considered.
- **is_closest_gene** Whether this is the closest gene to this variant.
- **is_intergenic** Whether this variant is intergenic, i.e, if there is no gene up or downstream within 100kb.
- **is_upstream** Whether this variant is upstream of this gene.
- **is_downstream** Whether this variant is downstream of this gene.
- **source** Gene mapping source, either Ensembl or NCBI.
- **mapping_method** Gene mapping method.

ensembl_ids A tibble listing gene Ensembl identifiers associated with each genomic context. Columns:

- **variant_id** Variant identifier.
- **gene_name** Gene symbol according to HUGO Gene Nomenclature (HGNC).
- **ensembl_id** The Ensembl identifier of an Ensembl gene, see Section Gene annotation in Ensembl for more information.

`entrez_ids` A `tibble` listing gene Entrez identifiers associated with each genomic context. Columns:

variant_id Variant identifier.

gene_name Gene symbol according to HUGO Gene Nomenclature (HGNC).

entrez_id The Entrez identifier of a gene, see ref. doi:10.1093/nar/gkq1237 for more information.

`write_xlsx` *Export a GWAS Catalog object to xlsx*

Description

This function exports a GWAS Catalog object to Microsoft Excel xlsx file. Each table (slot) is saved in its own sheet.

Usage

```
write_xlsx(x, file = stop("`file` must be specified"))
```

Arguments

`x`	A studies, associations, variants or traits object.
`file`	A file name to write to.

Value

Although this function is run for its side effect of writing an xlsx file, the path to the exported file is returned.

Examples

```
# Initial setup
.old_wd <- setwd(tempdir())

# Save an `associations` object, e.g. `associations_ex01`, to xlsx.
write_xlsx(associations_ex01, "associations.xlsx")

# Cleanup
unlink("associations.xlsx")
setwd(.old_wd)
```

Index

∗ **datasets**
    cytogenetic_bands, 8
    gc_examples, 10
[,associations,character,missing,missing-method *(subset-associations)*, 29
[,associations,missing,missing,missing-method *(subset-associations)*, 29
[,associations,numeric,missing,missing-method *(subset-associations)*, 29
[,studies,character,missing,missing-method *(subset-studies)*, 30
[,studies,missing,missing,missing-method *(subset-studies)*, 30
[,studies,numeric,missing,missing-method *(subset-studies)*, 30
[,traits,character,missing,missing-method *(subset-traits)*, 31
[,traits,missing,missing,missing-method *(subset-traits)*, 31
[,traits,numeric,missing,missing-method *(subset-traits)*, 31
[,variants,character,missing,missing-method *(subset-variants)*, 32
[,variants,missing,missing,missing-method *(subset-variants)*, 32
[,variants,numeric,missing,missing-method *(subset-variants)*, 32
%>%, 3, *3*

association_to_study, 3
association_to_trait, 4
association_to_variant, 5
associations, *7*, *10*, *12*, *20*, *23*, *24*, *29*, *39*
associations-class, 5
associations_ex01 *(gc_examples)*, 10
associations_ex02 *(gc_examples)*, 10

bind, 7, *23*

cytogenetic_bands, 8

exists_variant, 9

gc_examples, 10
get_associations, 11
get_child_efo, 12
get_metadata, 13
get_studies, 14
get_traits, 16
get_variants, 17

intersect *(setop)*, 23
is_ebi_reachable, 19

list, *14*

n, 20
n,associations-method *(n)*, 20
n,studies-method *(n)*, 20
n,traits-method *(n)*, 20
n,variants-method *(n)*, 20
names, *14*

open_in_dbsnp, 21
open_in_gtex, 21
open_in_gwas_catalog, 22
open_in_pubmed, 23

setdiff *(setop)*, 23
setequal *(setop)*, 23
setop, 23
studies, *7*, *10*, *15*, *20*, *23*, *24*, *30*, *39*
studies-class, 25
studies_ex01 *(gc_examples)*, 10
studies_ex02 *(gc_examples)*, 10
study_to_association, 27
study_to_trait, 28
study_to_variant, 28
subset-associations, 29
subset-studies, 30
subset-traits, 31
subset-variants, 32

tibble, *5–7*, *25, 26*, *35*, *38, 39*
trait_to_association, 33
trait_to_study, 34
trait_to_variant, 34
traits, *7*, *10, 11*, *17*, *20*, *23, 24*, *31*, *39*
traits-class, 35
traits_ex01 *(*gc_examples*)*, 10
traits_ex02 *(*gc_examples*)*, 10

union, *7*
union *(*setop*)*, 23

variant_to_association, 35
variant_to_study, 36
variant_to_trait, 37
variants, *7*, *10*, *19, 20*, *23, 24*, *32*, *39*
variants-class, 38
variants_ex01 *(*gc_examples*)*, 10
variants_ex02 *(*gc_examples*)*, 10

write_xlsx, 39

ymd, *26*