Package: rigvf (via r-universe)

November 25, 2026

Title R interface to the IGVF Catalog

Version 1.6.0

Description The IGVF Catalog provides data on the impact of genomic variants on function. The `rigvf` package provides an interface to the IGVF Catalog, allowing easy integration with Bioconductor resources.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports methods, httr2, rjsoncons, dplyr, tidyr, rlang, memoise, cachem, whisker, jsonlite, GenomicRanges, IRanges, Seqinfo

URL https://IGVF.github.io/rigvf

BugReports https://github.com/IGVF/rigvf/issues

Encoding UTF-8

biocViews ThirdPartyClient, Annotation, VariantAnnotation, FunctionalGenomics, GeneRegulation, GenomicVariation, GeneTarget

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), plyranges, plotgardener, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg38.knownGene, tibble

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libssl-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:53:52 UTC

RemoteUrl https://github.com/bioc/rigvf

RemoteRef RELEASE_3_24

RemoteSha b1a5694aebfcf6c0cf2e4e0ac068d1d989843fd1

Contents

arango 2
catalog_queries 4
db_queries 6
portal_queries 7
rigvf 8

Index **9**

`arango` *Internal interface to ArangoDB REST API*

Description

Objects documented on this page are for developer use. `arango_request()` formulates 'GET' or 'POST' to the ArangoDB API.

`arango_auth()` uses username and password to authenticate against the database.

`arango_collections()` implements the `_api/collection` endpoint for available collections in the IGVF database.

`arango_cursor()` implements the `_api/cursor` endpoint to allow a user-specified query of the IGVF database.

Usage

```
arango_request(path, ..., body = NULL, jwt_token = NULL)

arango_auth(
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)

arango_collections(
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)

arango_cursor(
  query,
  ...,
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)
```

Arguments

path	character(1) path to the API end point. Note that database-specfic paths are prefixed with /_db/igvf.
...	for arango_request(), named arguments to be used a query parameters. for arango_cursor(), named arguments to be interpreted as bind variables in the query.
body	if not NULL, formulate a POST request with JSON body.
jwt_token	character(1) JWT token obtained via arango_auth().
username	character(1) ArangoDB user name. Default: "guest".
password	character(1) ArangoDB password. Default: "guestdaccigvfcatalog". A better practice is to use an environment variable to record the password, rather than encoding in a script, so password = Sys.getenv("RIGVF_ARANGODB_PASSWORD").
query	character(1) the FILE NAME (without extension .aql of the query template.

Details

arango_auth() is 'memoized', so invoked only once per hour for a particular user and password. The memoised result can be cleared with memoise::forget(arango_auth).

arango_cursor() expects queries to be written in package system files in the inst/aql directory. This allows rapid iteration during query development (the package does not need to re-loaded when the query is updated) and some opportunity for language-specific highlighting if supported by the developer's text editor.

Value

arango_request() returns the JSON response as a character(1) vector.

arango_auth() returns a JWT token to be used for authentication in subsequent calls.

arango_collections() returns a tibble with columns name, type (either 'node' or 'edge'), and count.

arango_cursor() returns the JSON character(1) 'result' of the query.

Examples

```
## available queries
templates <- system.file(package = "rigvf", "aql")
dir(templates)
```

catalog_queries *Query the IGVF Catalog via REST API*

Description

This page documents functions using the IGVF REST API, documented at https://api.catalogkg.igvf.org/#.

Note that functions will only return a limited number of responses, see `limit` and `page` arguments below for control over number of responses.

`gene_variants()` locates variants associated with a gene. Only one of `gene_id`, `hgnc`, `gene_name`, or `alias` should be specified.

`variant_genes()` locates genes associated with a variant. Only one of `spdi`, `hgvs`, `rsid`, `variant_id`, or `chr` + `position` should be specified.

`gene_elements()` locates elements associated with a gene.

`elements()` locates genomic elements based on a genomic range query.

`element_genes()` locates genomic elements and associated genes based on a genomic range query.

Usage

```
gene_variants(
  gene_id = NULL,
  hgnc = NULL,
  gene_name = NULL,
  alias = NULL,
  organism = "Homo sapiens",
  log10pvalue = NULL,
  effect_size = NULL,
  page = 0L,
  limit = 25L,
  verbose = FALSE
)

variant_genes(
  spdi = NULL,
  hgvs = NULL,
  rsid = NULL,
  variant_id = NULL,
  chr = NULL,
  position = NULL,
  organism = "Homo sapiens",
  log10pvalue = NULL,
  effect_size = NULL,
  page = 0L,
  limit = 25L,
  verbose = FALSE
```

```
)

gene_elements(gene_id = NULL, page = 0L, limit = 25L, verbose = FALSE)

elements(range = NULL, page = 0L, limit = 25L)

element_genes(range = NULL, page = 0L, limit = 25L, verbose = FALSE)
```

Arguments

gene_id	character(1) Ensembl gene identifier, e.g., "ENSG00000106633"
hgnc	character(1) HGNC identifier
gene_name	character(1) Gene symbol, e.g., "GCK"
alias	character(1) Gene alias
organism	character(1) Either 'Homo sapiens' (default) or 'Mus musculus'
log10pvalue	character(1) The following can be used to set thresholds on the negative log10pvalue: gt (>), gte (>=), lt (<), lte (<=), with a ":" following and a value, e.g., "gt:5.0"
effect_size	character(1) Optional string used for thresholding on the effect size of the variant on the gene. See 'log10pvalue'. E.g., "gt:0.5"
page	integer(1) when there are more response items than `limit`, offers pagination. starts on page 0L, next is 1L, ...
limit	integer(1) the limit parameter controls the page size and can not exceed 1000
verbose	logical(1) return additional information about variants and genes
spdi	character(1) SPDI of variant
hgvs	character(1) HGVS of variant
rsid	character(1) RSID of variant
variant_id	character(1) IGVF variant ID
chr	character(1) UCSC-style chromosome name of variant, e.g. "chr1"
position	character(1) 0-based position of variant
range	the query GRanges (expects 1-based start position)

Value

`gene_variants()` returns a tibble describing variants associated with the gene; use `verbose = TRUE` to retrieve more extensive information.

`variant_genes()` returns a tibble describing genes associated with a variant; use `verbose = TRUE` to retrieve more extensive information.

`gene_elements()` returns a tibble describing elements associated with the gene; use `verbose = TRUE` to retrieve more extensive information.

`elements()` returns a GRanges object describing elements.

`element_genes()` returns a tibble describing genomic element and gene pairs.

Examples

```
rigvf::gene_variants(gene_name = "GCK")

rigvf::gene_variants(gene_name = "GCK", effect_size="gt:0.5")

rigvf::gene_variants(gene_name = "GCK", verbose = TRUE)

rigvf::variant_genes(spdi = "NC_000001.11:920568:G:A")

rigvf::gene_elements(gene_id = "ENSG00000187961")

rng <- GenomicRanges::GRanges("chr1", IRanges::IRanges(1157520,1158189))

rigvf::elements(range = rng)

rigvf::element_genes(range = rng)
```

---

db_queries *Query the IGVF Catalog via ArangoDB*

Description

`edges()` and `nodes()` identify edges or nodes in the data base.

`db_gene_variants()` locates variants associated with a (Ensembl) gene identifier.

`db_gene_elements()` locates genomic elements associated with a (Ensembl) gene identifier.

Usage

```
db_edges(
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)

db_nodes(
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)

db_gene_variants(
  gene_id,
  threshold,
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)

db_gene_elements(
```

```
  gene_id,
  threshold,
  username = rigvf_config$get("username"),
  password = rigvf_config$get("password")
)
```

Arguments

username	character(1) ArangoDB user name. Default: "guest".
password	character(1) ArangoDB password. Default: "guestdaccigvfcatalog". A better practice is to use an environment variable to record the password, rather than encoding in a script, so `password = Sys.getenv("RIGVF_ARANGODB_PASSWORD")`.
gene_id	character(1) Ensembl gene identifier
threshold	numeric(1) minimum association statistic, minus log10 p-value for variants, and score for elements

Value

`edges()` and `nodes()` return a tibble with the edge or node name and count of occurrences in the database.

`db_gene_variants()` returns a tibble summarizing variants associated with the gene.

`db_gene_elements()` returns a tibble summarizing genomic elements associated with the gene.

Examples

```
db_edges()

db_nodes()

db_gene_variants("ENSG00000106633", threshold = 4.0)

db_gene_elements("ENSG00000106633", threshold = 0.5)
```

---

portal_queries *Query the IGVF Data Portal*

Description

`portal_files()` searches for files on the IGVF Data Portal by content type and optional additional filters. The API is documented at https://data.igvf.org/help/igvf-api-spec/.

Usage

```
portal_files(content_type, frame = "object", limit = 25L, ...)
```

Arguments

content_type	character(1) the content type of files to retrieve, e.g. "reporter genomic variant effects".
frame	character(1) "object" (default) returns full properties with embedded objects as URI paths; "embedded" fully expands nested objects.
limit	integer(1) or "all" to return all matching results. Defaults to 25L. Note the portal rate limit is 10 requests/sec.
...	additional filter parameters passed as query parameters to /search/, e.g. status = "released". Use the syntax field != "value" filters via named arguments where needed.

Details

Some columns in the returned tibble are list-columns where each element is a length-1 character vector. Use purrr::map_chr(column, 1) to flatten these before further manipulation.

Value

portal_files() returns a tibble of matching files, with columns derived from the @graph array of the JSON-LD response.

Examples

```
portal_files("reporter genomic variant effects")

portal_files("reporter genomic variant effects", limit = 5L)
```

---

rigvf *Package configuration global variable management*

Description

Objects documented on this page are for developer use.

rigvf_config provides a simple interface to manage 'package global' variables via rigvf_config$get(), rigvf_config$set(), etc. The default username and password provide guest access.

Usage

```
rigvf_config
```

Format

rigvf_config is a list of functions for listing (ls())and manipulating (get(), set(), unset()) package-global variables.

Index

arango, 2
arango_auth *(arango)*, 2
arango_collections *(arango)*, 2
arango_cursor *(arango)*, 2
arango_request *(arango)*, 2

catalog_queries, 4

db_edges *(db_queries)*, 6
db_gene_elements *(db_queries)*, 6
db_gene_variants *(db_queries)*, 6
db_nodes *(db_queries)*, 6
db_queries, 6

element_genes *(catalog_queries)*, 4
elements *(catalog_queries)*, 4

gene_elements *(catalog_queries)*, 4
gene_variants *(catalog_queries)*, 4

portal_files *(portal_queries)*, 7
portal_queries, 7

rigvf, 8
rigvf_config *(rigvf)*, 8

variant_genes *(catalog_queries)*, 4