Package: gesel (via r-universe)

November 25, 2026

Version 1.0.0

Date 2026-11-09

Title Search for Interesting Gene Sets

License MIT + file LICENSE

Description Query the gesel indices to identify interesting gene sets. Users can test for enrichment in their own list of genes and/or search by text in the set names or descriptions. The indices can be hosted remotely or on a local filesystem.

Imports utils, methods, rappdirs, httr2, Rcpp

Suggests BiocStyle, knitr, testthat, rmarkdown, S4Vectors, IRanges

LinkingTo assorthead, Rcpp

biocViews GeneSetEnrichment, DataImport, Pathways

URL https://github.com/gesel-inc/gesel-R

BugReports https://github.com/gesel-inc/gesel-R/issues

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libssl-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:21 UTC

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

RemoteRef RELEASE_3_24

RemoteSha 44911415bea2ccd116f4894a88733e9d840553a4

Contents

cacheDirectory 2
consolidateBlockSize 3
createGeneIdentifierMap 4

downloadDatabaseFile . . . 5
downloadDatabaseRanges . . . 6
downloadGeneFile . . . 8
effectiveNumberOfGenes . . . 9
fetchAllCollections . . . 10
fetchAllGenes . . . 11
fetchAllSets . . . 11
fetchCollectionSizes . . . 12
fetchGenesForAllSets . . . 13
fetchGenesForSomeSets . . . 14
fetchGeneTypes . . . 15
fetchGeneVersion . . . 16
fetchSetsForAllGenes . . . 17
fetchSetsForSomeGenes . . . 18
fetchSetSizes . . . 19
fetchSomeCollections . . . 20
fetchSomeSets . . . 21
flushMemoryCache . . . 22
loadAllSets . . . 22
newConfig . . . 24
prepareDatabaseFiles . . . 25
prepareGeneFiles . . . 27
querySets . . . 29
rangeConcurrency . . . 31
readDatabaseRanges . . . 32
renameGenesInSets . . . 33
searchGenes . . . 34
searchOverlappingSets . . . 35
searchSetText . . . 37
validateDatabaseFiles . . . 38
validateGeneFiles . . . 39

Index **41**

`cacheDirectory` *Gesel cache directory*

Description

Get or set the location of the Gesel cache directory, used as the default in functions like `downloadGeneFile` and `downloadDatabaseFile`.

Usage

```
cacheDirectory(cache = NULL)
```

Arguments

cache	String specifying the path to a cache directory.

Details

The cache directory defaults to a location in the user cache directory, as chosen by the **rappdirs** package. Users can modify this choice by setting the GESEL_CACHE_DIRECTORY environment variable before the first call to this function.

Value

If cache=NULL, the path to the current cache directory is returned.

If cache is provided, it is set as the path to the cache directory, and the previous location is invisibly returned.

Author(s)

Aaron Lun

Examples

```
cacheDirectory()

old <- cacheDirectory("/tmp/foo/bar")
cacheDirectory()

cacheDirectory(old)
cacheDirectory()
```

---

consolidateBlockSize *Block size for consolidation*

Description

Get or set the block size for consolidating HTTP range requests.

Usage

```
consolidateBlockSize(block.size = NULL)
```

Arguments

block.size	Integer specifying the block size in bytes. Larger sizes reduce the number of requests at the cost of increasing the size of each request.

Details

Each file is split up into blocks of size approximately equal to `consolidateBlockSize()`. When performing a range request, all ranges in the same block will be retrieved. This consolidates near-adjacent ranges into a single request, reducing the number of requests at the cost of increasing the size of each request.

All ranges associated with a block will be cached in memory, even those that were not directly requested. Subsequent function calls can then quickly retrieve ranges from this cache instead of making a new HTTP request. Downloading and caching the entire block also ensures that the same bytes will never be requested from the server twice in the same session.

Setting the block size to a value of 1 indicates that no consolidation is to be performed, i.e., each range is requested as-is.

Value

If `block.size=NULL`, the current block size is returned.

If `block.size` is provided, it is used to set the block size, and the previous value is returned invisibly.

Author(s)

Aaron Lun

Examples

```
consolidateBlockSize()
old <- consolidateBlockSize(500)
consolidateBlockSize()
consolidateBlockSize(old)
```

`createGeneIdentifierMap`

Create mapping of gene identifiers

Description

Create a mapping of gene identifiers (Ensembl, symbol, etc.) to their Gesel gene indices.

Usage

```
createGeneIdentifierMap(species, type, ignore.case = FALSE, config = NULL)
```

Arguments

species	String specifying the taxonomy ID of the species of interest.
type	String specifying the type of gene identifier. This can be any type listed in fetchGeneTypes.
ignore.case	Boolean indicating whether case should be ignored.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Value

Named list of integer vectors. Each name is an identifier of the specified type, and each vector contains the identities of genes associated with that identifier (after ignoring case, if ignore.case=TRUE). Specifically, each gene's identity is represented as a row index into the data frame returned by fetchAllGenes.

Author(s)

Aaron Lun

See Also

searchGenes, which uses the mapping when searching for genes.

Examples

```
mapping <- createGeneIdentifierMap("9606", type="symbol")

# Taking it for a spin:
found <- mapping[["SNAP25"]]
fetchAllGenes("9606")$symbol[found]
```

---

downloadDatabaseFile *Download Gesel database files*

Description

Default function to download Gesel database files.

Usage

```
downloadDatabaseFile(
  name,
  url = databaseUrl(),
  cache = cacheDirectory(),
  overwrite = FALSE
)

databaseUrl(url = NULL)
```

Arguments

name	String containing the name of a Gesel database file. This usually has the species identifier as a prefix, e.g., "9606_set2gene.tsv.gz".
url	String containing the base URL to the Gesel database files.
cache	String specifying the path to a cache directory.
overwrite	Boolean indicating whether any cached file should be overwritten.

Details

The database URL defaults to the GitHub releases at https://github.com/gesel-inc/feedstock. This can be altered by setting the GESEL_DATABASE_URL environment variable prior to the first call to this function.

Value

downloadDatabaseFile returns a string containing a path to the downloaded file.

For databaseUrl, if url=NULL, the function returns a string containing the URL to the Gesel database. If url is provided, it instead stores url as the URL to the database, and the previous value of url is invisibly returned.

Author(s)

Aaron Lun

Examples

```
# Download file.
downloadDatabaseFile("9606_collections.tsv.gz")

# Altering the default database URL.
databaseUrl()
old <- databaseUrl("https://foo")
databaseUrl()
databaseUrl(old)
```

---

downloadDatabaseRanges

Fetch byte ranges from a Gesel database file

Description

Download any number of byte ranges from a Gesel database file using (possibly multi-part) HTTP range requests.

Usage

```
downloadDatabaseRanges(
  name,
  start,
  end,
  url = databaseUrl(),
  multipart = FALSE,
  concurrency = rangeConcurrency()
)
```

Arguments

name	String containing the name of a Gesel database file. This usually has the species identifier as a prefix, e.g., "9606_set2gene.tsv.gz".
start	Integer vector of range start positions, see "Fetching byte ranges" in ?newConfig for details.
end	Integer vector of range end positions, see "Fetching byte ranges" in ?newConfig for details.
url	String containing the base URL to the Gesel database files.
multipart	Boolean indicating whether the server at url supports multi-part range requests.
concurrency	Integer specifying the maximum number of concurrent range requests per second. Ignored if multipart=TRUE.

Value

List of raw vectors, see "Fetching byte ranges" in ?newConfig for details.

Author(s)

Aaron Lun

See Also

readDatabaseRanges, for a local counterpart to this function that reads byte ranges from the filesystem.

Examples

```
downloadDatabaseRanges("9606_set2gene.tsv", 0L, 100L)
downloadDatabaseRanges("9606_set2gene.tsv", c(10, 100, 1000), c(20, 150, 1100))
```

`downloadGeneFile` *Fetch Gesel gene annotation files*

Description

Default function to download Gesel gene annotation files.

Usage

```
downloadGeneFile(
  name,
  url = geneUrl(),
  cache = cacheDirectory(),
  overwrite = FALSE
)

geneUrl(url = NULL)
```

Arguments

`name`	String containing the name of a Gesel gene annotation file, typically of the form `<species>_<type>.tsv.gz`, e.g., `"9606_symbol.tsv.gz"`.
`url`	String containing the base URL to the Gesel gene annotation files.
`cache`	String specifying the path to a cache directory.
`overwrite`	Boolean indicating whether any cached file should be overwritten.

Details

The gene URL defaults to the GitHub releases at https://github.com/gesel-inc/feedstock. This can be altered by setting the `GESEL_GENE_URL` environment variable prior to the first call to this function.

Value

`downloadGeneFile` returns a string containing a local path to the downloaded file.

For `geneUrl`, if `url=NULL`, the function returns a string containing the URL to the Gesel gene descriptions. If `url` is provided, it instead stores `url` as the URL to the indices, and the previous value of `url` is invisibly returned.

Author(s)

Aaron Lun

Examples

```
# Download file.
path <- downloadGeneFile("9606_gene-types.tsv")
readLines(path)

# Altering the default gene URL.
geneUrl()
old <- geneUrl("https://foo")
geneUrl()
geneUrl(old)
```

`effectiveNumberOfGenes` *Effective number of genes*

Description

Count the number of genes in the Gesel database that belong to at least one set.

Usage

```
effectiveNumberOfGenes(species, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

The return value should be used as the total number of balls when performing a hypergeometric test for gene set enrichment (see `phyper`), instead of `nrow(fetchAllGenes(species))`. This ensures that uninteresting genes like pseudo-genes or predicted genes are ignored during the calculation. Otherwise, unknown genes would inappropriately increase the number of balls and understate the enrichment p-values.

Value

Integer specifying the number of genes in Gesel that belong to at least one set.

Author(s)

Aaron Lun

Examples

```
effectiveNumberOfGenes("10090")
effectiveNumberOfGenes("9606")
effectiveNumberOfGenes("7227")
```

`fetchAllCollections` *Fetch all gene set collections*

Description

Fetch information about all gene set collections in the Gesel database.

Usage

```
fetchAllCollections(species, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

If this function is called once, the data frame will be cached in memory and re-used in subsequent calls to this function. The cached data will also be used to speed up calls to `fetchSomeCollections`.

Value

Data frame of gene set collection information. Each row represents a collection and contains:

- `title`, string containing the title of the collection.
- `description`, string containing a description of the collection.
- `maintainer`, string containing the identity of the collection's maintainer.
- `source`, string containing the source of origin of the collection.
- `start`, integer containing the set index of the first gene set in this collection. The set index refers to a row in the data frame returned by `fetchAllSets`.
- `size`, integer specifying the number of gene sets in the collection.

Author(s)

Aaron Lun

Examples

```
out <- fetchAllCollections("9606")
head(out)
```

fetchAllGenes *Fetch all genes*

Description

Fetch identifiers for all genes.

Usage

```
fetchAllGenes(species, types = NULL, config = NULL)
```

Arguments

species	String specifying the taxonomy ID of the species of interest.
types	Character vector specifying the types of gene identifiers to return. If NULL, the available types are determined from fetchGeneTypes.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Value

Data frame where each row represents a gene. Each column corresponds to one of the types and is a list of character vectors. Each vector contains identifiers of the specified type for each gene.

Author(s)

Aaron Lun

Examples

```
out <- fetchAllGenes("9606")
head(out)
head(out$symbol)
```

fetchAllSets *Fetch all gene sets*

Description

Fetch information about all gene sets in the Gesel database.

Usage

```
fetchAllSets(species, config = NULL)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Details

If this function is called once, the data frame will be cached in memory and re-used in subsequent calls to this function. The cached data will also be used to speed up calls to fetchSomeSets.

Value

Data frame of gene set information. Each row represents a gene set and contains:

- name, string containing the name of the gene set.
- description, string containing a description of the gene set.
- size, integer specifying the number of genes in this gene set.
- collection, integer containing the collection index of the collection that contains this gene set. The collection index refers to a row of the data frame returned by fetchAllCollections.
- number, integer containing the position of the gene set inside the specified collection. The set index of the current gene set is defined by adding number - 1 to the collection's start.

Author(s)

Aaron Lun

Examples

```
out <- fetchAllSets("9606")
head(out)
```

---

fetchCollectionSizes *Size of collections*

Description

Quickly get the sizes of all gene set collections in the Gesel database. This is more efficient than fetchAllCollections when only the sizes are of interest.

Usage

```
fetchCollectionSizes(species, config = NULL)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Value

Integer vector containing the size of each collection (i.e., the number of gene sets).

Author(s)

Aaron Lun

Examples

```
head(fetchCollectionSizes("9606"))
```

---

fetchGenesForAllSets *Fetch genes for all sets*

Description

Fetch the gene membership of all sets in the Gesel database.

Usage

```
fetchGenesForAllSets(species, config = NULL)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Details

If this function is called once, the returned list will be cached in memory and re-used in subsequent calls to this function. The cached data will also be used to speed up calls to fetchGenesForSomeSets.

Value

List of integer vectors. Each vector represents a gene set, corresponding to the rows of the data frame returned by fetchAllSets. Each vector contains the identities of the genes in that set, where each integer is a gene index that refers to a row of the data frame returned by fetchAllGenes.

Author(s)

Aaron Lun

See Also

renameGenesInSets, to easily convert the gene indices to the usual identifiers (symbols, Ensembl, etc.).

Examples

```
all.sets <- fetchGenesForAllSets("9606")
length(all.sets)

# Genes in the first set:
fetchAllGenes("9606")$symbol[all.sets[[1]]]

# Details about the first set:
fetchAllSets("9606")[1,]
```

---

fetchGenesForSomeSets *Fetch genes for some sets*

Description

Fetch the gene membership of some sets in the Gesel database. This can be more efficient than fetchGenesForAllSets if only a few sets are of interest.

Usage

```
fetchGenesForSomeSets(species, sets, config = NULL)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
sets	Integer vector containing set indices. Each set index refers to a row in the data frame returned by fetchAllSets.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Details

Every time this function is called, information from the requested sets will be added to an in-memory cache. Subsequent calls to this function will re-use as many of the cached sets as possible before making new requests to the Gesel database.

If fetchGenesForAllSets was previously called, its cached data will be directly used by fetchGenesForSomeSets to avoid performing extra requests to the database. If sets is large, it may be more efficient to call fetchGenesForAllSets to prepare the cache before calling this function.

Value

List of integer vectors. Each vector corresponds to a set in `sets` and contains the identities of its member genes. Each gene is defined by its gene index, which refers to a row of the data frame returned by `fetchAllGenes`.

Author(s)

Aaron Lun

See Also

`renameGenesInSets`, to easily convert the gene indices to the usual identifiers (symbols, Ensembl, etc.).

Examples

```
first.set <- fetchGenesForSomeSets("9606", 1:5)
str(first.set)

# Genes in the first set:
gene.symbols <- fetchAllGenes("9606")$symbol
head(gene.symbols[first.set[[1]]])

# Identities of the requested sets.
set.info <- fetchAllSets("9606")[1:5,]
set.info
```

`fetchGeneTypes` *Fetch available gene identfier types*

Description

Fetch the list of available gene identifier types for this species.

Usage

```
fetchGeneTypes(species, config = NULL)
```

Arguments

`species`	String specifying the taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

In older versions (0.1.0) of the Gesel gene annotation file specification, an explicit listing of types was not generated, so this function will just assume that `"ensembl"`, `"entrez"` and `"symbol"` are available.

Value

Character vector of the available types.

Author(s)

Aaron Lun

Examples

```
fetchGeneTypes("9606")
fetchGeneTypes("10090")
```

`fetchGeneVersion` *Fetch version of the gene annotation files*

Description

Get the version of the Gesel gene annotation file specification used by the species of interest.

Usage

```
fetchGeneVersion(species, config = NULL)
```

Arguments

`species`	String specifying the taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Value

String containing the version of the Gesel gene annotation file specification.

Author(s)

Aaron Lun

Examples

```
fetchGeneVersion("9606")
```

`fetchSetsForAllGenes` *Fetch sets for all genes*

Description

Fetch the identities of the sets that contain each gene in the Gesel database.

Usage

```
fetchSetsForAllGenes(species, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

If this function is called once, the returned list will be cached in memory and re-used in subsequent calls to this function. The cached data will also be used to speed up calls to `fetchSetsForSomeGenes`.

Value

List of integer vectors. Each vector corresponds to a gene, corresponding to a row of the data frame returned by `fetchAllGenes`. Each vector contains the identities of the sets that contain that gene, where each integer is a set index that refers to a row of the data frame returned by `fetchAllSets`.

Author(s)

Aaron Lun

Examples

```
all.genes <- fetchSetsForAllGenes("9606")
length(all.genes)

# Sets containing the first gene:
head(fetchAllSets("9606")[all.genes[[1]],])

# Details about the first gene:
fetchAllGenes("9606")$symbol[1]
```

`fetchSetsForSomeGenes` *Fetch sets for some genes*

Description

Fetch the identities of sets that contain some genes in the Gesel database. This can be more efficient than `fetchSetsForAllGenes` if only a few genes are of interest.

Usage

```
fetchSetsForSomeGenes(species, genes, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`genes`	Integer vector containing gene indices. Each gene index refers to a row of the data frame returned by `fetchAllGenes`).
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

Every time this function is called, information from the requested `genes` will be added to an in-memory cache. Subsequent calls to this function will re-use as many of the cached genes as possible before making new requests to the Gesel database.

If `fetchSetsForAllGenes` is called, its cached data will be directly used by `fetchSomeSets` to avoid extra requests to the database. If `genes` is large, it may be more efficient to call `fetchSetsForAllGenes` to prepare the cache before calling this function.

Value

List of integer vectors. Each vector corresponds to a gene in `genes` and contains the identities of the sets containing that gene. Each set is defined by its set index, which refers to a row of the data frame returned by `fetchAllSets`.

Author(s)

Aaron Lun

Examples

```
first.gene <- fetchSetsForSomeGenes("9606", 1:5)
str(first.gene)

# Sets containing the first gene.
all.set.info <- fetchAllSets("9606")
head(all.set.info[first.gene[[1]],])
```

```
# Identities of the requested genes.
fetchAllGenes("9606")[1:5,]
```

`fetchSetSizes` *Size of gene sets*

Description

Quickly get the size of the sets in the Gesel database. This is more efficient than `fetchAllSets` when only the sizes are of interest.

Usage

```
fetchSetSizes(species, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Value

Integer vector containing the size of each set (i.e., the number of gene sets).

Author(s)

Aaron Lun

Examples

```
head(fetchSetSizes("9606"))
```

`fetchSomeCollections` *Fetch some collections*

Description

Fetch the details of some gene set collections from the Gesel database. This can be more efficient than `fetchAllCollections` when only a few collections are of interest.

Usage

```
fetchSomeCollections(species, collections, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`collections`	Integer vector containing collection indices. Each entry refers to a row of the data frame returned by `fetchAllCollections`).
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

Every time this function is called, information from the requested `collections` will be added to an in-memory cache. Subsequent calls to this function will re-use as many of the cached collections as possible before making new requests to the Gesel database.

If `fetchAllCollections` was previously called, its cached data will be used by `fetchSomeCollections` to avoid extra requests to the database. If `collections` is large, it may be more efficient to call `fetchAllCollections` to prepare the cache before calling this function.

Value

Data frame with the same columns as the return value of `fetchAllCollections`, where each row corresponds to an entry of `collections`.

Author(s)

Aaron Lun

Examples

```
fetchSomeCollections("9606", 1:5)
```

`fetchSomeSets` *Fetch some sets*

Description

Fetch the details of some gene sets from the Gesel database. This can be more efficient than calling `fetchAllSets` when only a few sets are of interest.

Usage

```
fetchSomeSets(species, sets, config = NULL)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`sets`	Integer vector of set indices, where each set index refers to a row in the data frame returned by `fetchAllSets`.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

Every time this function is called, information from the requested `sets` will be added to an in-memory cache. Subsequent calls to this function will re-use as many of the cached sets as possible before making new requests to the Gesel database.

If `fetchAllSets` was previously called, its cached data will be directly used by `fetchSomeSets` to avoid performing extra requests to the database. If `sets` is large, it may be more efficient to call `fetchAllSets` to prepare the cache before calling this function.

Value

Data frame with the same columns as the return value of `fetchAllSets`, where each row corresponds to an entry of `sets`.

Author(s)

Aaron Lun

Examples

```
fetchSomeSets("9606", 1:5)
```

flushMemoryCache *Flush the in-memory cache*

Description

Flush the in-memory cache for **gesel** data structures in the current R session.

Usage

```
flushMemoryCache(config = NULL)
```

Arguments

config	A configuration list. If NULL, the default configuration is used.

Details

By default, the **gesel** package caches the data structures in the current R session to avoid unnecessary requests to the filesystem and remote server. On rare occasion, these cached data structures may be out of date when the Gesel database files change. In such cases, the cache can be flushed to ensure that the various **gesel** functions operate on the latest version of the database.

Value

The in-memory cache in config is cleared. NULL is invisibly returned.

Author(s)

Aaron Lun

Examples

```
flushMemoryCache()
```

loadAllSets *Load all gene sets for a species*

Description

Load information about all gene sets for a species. This includes its gene membership as well as details like its name and description.

Usage

```
loadAllSets(species, type, config = NULL, as.compressed = FALSE)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
type	String specifying the type of gene identifier to report. This can be any type listed in fetchGeneTypes.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.
as.compressed	Boolean indicating whether to return a CompressedCharacterList.

Value

If as.compressed = FALSE, a list is returned containing:

- sets, a name list of length equal to the total number of sets for species. Each list element is a character vector that corresponds to a gene set and contains the genes in that set. Each gene is represented by zero, one or more identifiers of the specified type (see renameGenesInSets for details). The positional index of each set in sets can be used as the Gesel set index in other **gesel** functions like fetchGenesForSomeSets. Each list element is also named after its set (i.e., the name column in details).
- details, a data frame where each row corresponds to a gene set in sets. The data frame contains the following columns:
  - name, string containing the name of the gene set.
  - description, string containing a description of the gene set.
  - size, integer specifying the number of genes in this gene set. Note that this may not equal lengths(sets) if any gene does not have exactly one identifier of the specified type.
  - collection, the name of the collection that contains this gene set.

  The row index of each set in details can be used as the Gesel set index in other **gesel** functions like fetchGenesForSomeSets.

If as.compressed = TRUE, a CompressedCharacterList of length equal to the total number of sets for species is returned. Each element is a character vector that contains the identifiers of genes in that set, as described above for sets. The mcols contains more details about each set as described above for details.

Author(s)

Aaron Lun

See Also

fetchGenesForAllSets, to obtain the internal Gesel gene indices for all sets.

renameGenesInSets, to convert Gesel gene indices to identifiers.

fetchAllSets, to obtain information about all sets.

Examples

```
everything <- loadAllSets("7227", "symbol")
head(everything$sets)
head(everything$details)

everything2 <- loadAllSets("7227", "symbol", as.compressed = TRUE)
everything2
S4Vectors::mcols(everything2)
```

`newConfig` *Create a new configuration object*

Description

Create a new configuration object to specify how the Gesel database should be queried. This can be used by applications to point to a different Gesel database from the default.

Usage

```
newConfig(
  fetch.gene = NULL,
  fetch.gene.args = list(),
  gene.version = NULL,
  fetch.file = NULL,
  fetch.file.args = list(),
  fetch.ranges = NULL,
  fetch.ranges.args = list(),
  consolidate.block.size = NULL
)
```

Arguments

`fetch.gene`	Function that accepts the name of a Gesel gene annotation file and returns an absolute path to the file. If `NULL`, it defaults to `downloadGeneFile`.
`fetch.gene.args`	Named list of arguments to pass to `fetch.gene`.
`gene.version`	String containing the version of the Gesel gene annotation file specification used by all gene annotation instances. If `NULL`, this is automatically determined by `fetchGeneVersion`. This can also be set explicitly to skip a query and save some time.
`fetch.file`	Function that accepts the name of a Gesel database file and returns an absolute path to the file. If `NULL`, it defaults to `downloadDatabaseFile`.
`fetch.file.args`	Named list of arguments to pass to `fetch.file`.

`fetch.ranges` Function that downloads byte ranges, see "Fetching byte ranges" below. If `NULL`, it defaults to `downloadDatabaseRanges`.

`fetch.ranges.args`
Named list of arguments to pass to `fetch.ranges`.

`consolidate.block.size`
Integer specifying the block size for consolidated requests. If `NULL`, it defaults to `consolidateBlockSize`.

Details

The configuration list returned by `newConfig` can be passed to each **gesel** function to alter its behavior in a consistent manner. For example, we can override `fetch.file` to retrieve database files from a shared filesystem instead of performing a HTTP request.

The configuration list also contains a cache of data structures that can be populated by **gesel** functions. This avoids unnecessary fetch requests upon repeated calls to the same function. If the cache becomes stale or too large, it can be cleared by calling `flushMemoryCache`.

If no configuration list is supplied to **gesel** functions, the default configuration is used. The default is created by calling `newConfig` without any arguments.

Value

A list containing Gesel configuration settings.

Author(s)

Aaron Lun

Examples

```
config <- newConfig()
```

---

prepareDatabaseFiles *Prepare the Gesel database*

Description

Prepare Gesel database files from various pieces of gene set information.

Usage

```
prepareDatabaseFiles(
  species,
  collections,
  set.info,
  set.membership,
  num.genes,
```

```
  path = ".",
  validate = TRUE
)
```

Arguments

species	String specifying the species in the form of its NCBI taxonomy ID.
collections	Data frame of information about each gene set collection, where each row corresponds to a collection. This data frame should contain the title, description, source and maintainer columns as described in ?fetchAllCollections.
set.info	List of data frames of length equal to nrow(collections). Each data frame corresponds to a collection where each row corresponds to a gene set. Each data frame should have the name and description columns as described in ?fetchAllSets.
set.membership	List of lists of integer vectors. Each inner list corresponds to a collection and each of its vectors corresponds to a gene set in that collection, e.g., set.membership[[i]][[j]] contains the j-th set in the i-th collection. Each vector contains the identities of the constituent genes in that set, where each gene is identified as a row index into the data frame returned by fetchAllGenes. All gene indices should be positive and no greater than num.genes. Vectors may be unsorted and contain duplicate entries.
num.genes	Integer specifying the total number of genes available for this species.
path	String containing the path to a directory in which to create the database files.
validate	Boolean indicating whether to run validateDatabaseFiles on the newly created files.

Value

Several files are produced at path with the <species>_ prefix. NULL is invisibly returned.

Author(s)

Aaron Lun

See Also

prepareGeneFiles, to create Gesel gene annotation files containing the gene identifiers.

Examples

```
# Mocking up some information.
collections <- data.frame(
    title=c("FOO", "BAR"),
    description=c("I am a foo", "I am a bar"),
    maintainer=c("Aaron", "Aaron"),
    source=c("https://foo", "https://bar")
)
```

```
set.info <- list(
    data.frame(
        name=sprintf("FOO_%i", seq_len(20)),
        description=sprintf("this is FOO %i", seq_len(20))
    ),
    data.frame(
        name=sprintf("BAR_%i", seq_len(50)),
        description=sprintf("this is BAR %i", seq_len(50))
    )
)

# Mocking up the gene sets.
num.genes <- 10000
set.membership <- list(
    lapply(seq_len(nrow(set.info[[1]])), function(i) {
        sample(num.genes, sample(500, 1))
    }),
    lapply(seq_len(nrow(set.info[[2]])), function(i) {
        sample(num.genes, sample(200, 1))
    })
)

# Now making the database files.
output <- tempfile()
dir.create(output)
prepareDatabaseFiles(
    "1234",
    collections,
    set.info,
    set.membership,
    num.genes,
    output
)

# We can then read directly from them:
config <- newConfig(fetch.file=function(x) file.path(output, x))
head(fetchAllSets("1234", config))
```

`prepareGeneFiles` *Prepare the Gesel gene annotation files*

Description

Prepare Gesel gene annotation files containing the gene identifiers.

Usage

```
prepareGeneFiles(
  species,
```

```
  genes,
  path = ".",
  validate = TRUE,
  version = c("0.2.0", "0.1.0")
)
```

Arguments

species	String specifying the species in the form of its NCBI taxonomy ID.
genes	Named list of lists of character vectors. Each inner list corresponds to an identifier type and is named accordingly, e.g., `genes[["ensembl"]]` for Ensembl. Each inner list should be of length equal to the total number of genes. Each entry of the inner list corresponds to a gene and is a character vector containing identifiers of the specified type for that gene, e.g., `genes[["ensembl"]][[10]]` is a vector of Ensembl IDs for the 10th gene. A character vector may be empty if no identifiers are available for a gene. Alternatively, a data frame where each row corresponds to a gene and each column is a list of character vectors as described above, see the output of `fetchAllGenes` for details.
path	String containing the path to a directory in which to create the gene annotation files.
validate	Boolean indicating whether to run `validateGeneFiles` on the newly created files.
version	String specifying the version of the Gesel gene annotation file specification to use for saving genes.

Value

Several files are produced inside `path` with the `<species>_` prefix. `NULL` is invisibly returned.

Author(s)

Aaron Lun

See Also

`prepareDatabaseFiles`, to create Gesel database files containing the gene set information.

Examples

```
genes <- list(
    ensembl = list("ENSG1", c("ENSG2", "ENSG3"), character(0), "ENSG4"),
    entrez = list("1", character(0), c("2", "3", "4"), c("5", "6")),
    foobar = list("malat1", "neat1", "Gm1234", "LINC0000001")
)

tmp <- tempfile()
dir.create(tmp)
prepareGeneFiles("1234", genes, tmp)
```

```
# We can then read directly from them:
config <- newConfig(fetch.gene=function(x) file.path(tmp, x))
head(fetchAllGenes("1234", config=config))
```

---

querySets *Query gene sets*

Description

Query gene sets based on overlaps with genes of interest or matches to keyboards in their names/descriptions.

Usage

```
querySets(
  species,
  genes = NULL,
  text = NULL,
  types = NULL,
  counts.only = TRUE,
  config = NULL
)
```

Arguments

species	String specifying the taxonomy ID of the species of interest.
genes	Character vector of gene identifiers of any supported type. These are typically Ensembl/Entrez identifiers or gene symbols. If not NULL, this function will search for gene sets that overlap any of the supplied genes.
text	String containing one or more keywords to search on, see the query= argument in searchSetText. If not NULL, this function will search for gene sets that contain the (tokenized) keywords in their names or descriptions.
types	Character vector specifying the types of gene identifiers to consider for genes. This can contain one or more of any type listed in fetchGeneTypes. If NULL, it defaults to all available types in fetchGeneTypes.
counts.only	Boolean indicating whether to return a list of the overlapping genes in each set. Only used if genes is provided.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Details

This is a user-friendly wrapper for quick and convenient searching of the Gesel database. Developers may prefer to use the lower-level **gesel** functions for more customization and flexibility.

Value

A data frame containing one row per set that matches the query conditions. This contains the following columns:

- `name`, string containing the name of the gene set.
- `description`, string containing a description of the gene set.
- `size`, integer specifying the number of genes in this gene set.
- `collection`, the name of the collection that contains this gene set.
- `set`, integer specifying the Gesel set index that can be used in other **gesel** functions, e.g., `fetchGenesForSomeSets`.

If `genes` is provided, the data frame will additionally contain:

- `count`, an integer column containing the number of overlaps between the genes in the set and those in `genes`.
- `which`, a nested list where each entry is a character vector containing the genes in `genes` that are present in each set. Only reported if `counts.only = FALSE`.
- `pvalue`, a numeric column containing the hypergeometric p-value for overrepresentation of `genes` in each set. Rows will be sorted by this column if it is present.

Author(s)

Aaron Lun

See Also

`searchGenes`, to convert gene identifiers into internal Gesel indices.

`searchOverlappingSets`, to find the sets that overlap the genes of interest.

`searchSetText`, to find sets based on keywords in their names/descriptions.

`fetchSomeSets`, to get the details for each set.

`fetchSomeCollections`, to get the details for each collection.

Examples

```
out <- querySets(
    species = "9606",
    genes = c("tead1", "tead2", "tead3", "tead4"),
    text = "transcription"
)
head(out)

out2 <- querySets(
    species = "9606",
    genes = c("SNAP25", "neurod4", "neurod6"),
    text = "neuro*",
    counts.only = FALSE
)
head(out2)
```

rangeConcurrency *Concurrency of range requests*

Description

Get or set the maximum number of concurrent HTTP range requests that can be performed per second in downloadDatabaseRanges. Setting this to a smaller number avoids excessive load on the server.

Usage

```
rangeConcurrency(concurrency = NULL)
```

Arguments

concurrency	Integer containing the maximum number of concurrent requests per second.

Value

If concurrency=NULL, the maximum number of concurrent requests is returned.

If concurrency is provided, it is set to the maximum number of concurrent requests, and the previous maximum is returned.

Author(s)

Aaron Lun

See Also

req_throttle, for the logic behind the requests-per-second limit.

Examples

```
rangeConcurrency()
old <- rangeConcurrency(5)
rangeConcurrency()
rangeConcurrency(old)
rangeConcurrency()
```

`readDatabaseRanges` *Read byte ranges from a Gesel database file*

Description

Read any number of byte ranges from a Gesel database file on the local filesystem.

Usage

```
readDatabaseRanges(dir, name, start, end)
```

Arguments

`dir`	String containing the path to a directory containing all Gesel database files.
`name`	String containing the name of a Gesel database file. This usually has the species identifier as a prefix, e.g., `"9606_set2gene.tsv.gz"`.
`start`	Integer vector of range start positions, see "Fetching byte ranges" in ?`newConfig` for details.
`end`	Integer vector of range end positions, see "Fetching byte ranges" in ?`newConfig` for details.

Value

List of length equal to `length(start)`. Each entry is a raw vector representing the contents of the corresponding byte range in `start`. The `retrieved.start` and `retrieved.end` attributes store `start` and `end`, respectively, as stated in the "Fetching byte ranges" section of ?`newConfig`.

See Also

`downloadDatabaseRanges`, for a remote counterpart to this function that requests the byte ranges from a server.

Examples

```
path <- downloadDatabaseFile("9606_set2gene.tsv")
readDatabaseRanges(dirname(path), basename(path), 0L, 100L)
readDatabaseRanges(dirname(path), basename(path), c(10, 100, 1000), c(20, 150, 1100))
```

`renameGenesInSets` *Rename genes in their sets*

Description

Convert genes in sets from row indices to their identifiers.

Usage

```
renameGenesInSets(species, sets, type, config = NULL)
```

Arguments

`species`	String specifying the taxonomy ID of the species of interest.
`sets`	List of integer vectors, typically generated by `fetchGenesForAllSets` or `fetchGenesForSomeSets`. Each vector represents a gene set and contains gene indices, i.e., row indices into the data frame returned by `fetchAllGenes`.
`type`	String specifying the type of gene identifier to convert to. This can be any type listed in `fetchGeneTypes`.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

If a gene has no identifiers of the specified `type`, it is omitted from the affected character vectors in the output list. If a gene has multiple identifiers of the specified `type`, all identifiers are reported in the affected character vectors in the output list.

Value

A list of the same length as `sets`. Each entry is now a character vector containing the specified identifiers for the genes in the corresponding set.

Author(s)

Aaron Lun

Examples

```
example.sets <- fetchGenesForSomeSets("9606", 1:50)
head(example.sets)
renamed.sets <- renameGenesInSets("9606", example.sets, "symbol")
head(renamed.sets)
```

searchGenes *Search for genes*

Description

Search for genes by converting gene identifiers to gene indices.

Usage

```
searchGenes(species, genes, types = NULL, ignore.case = TRUE, config = NULL)
```

Arguments

species	String specifying the taxonomy ID of the species of interest.
genes	Character vector of gene identifiers of any type specified in types.
types	Character vector specifying the types of gene identifiers in genes. If NULL, the available types are determined from fetchGeneTypes.
ignore.case	Boolean indicating whether case should be ignored.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Value

List of length equal to genes. Each entry is an integer vector of gene indices that refer to rows of the data frame returned by fetchAllGenes; these rows represent the genes with identifiers that match to the corresponding entry of genes.

Author(s)

Aaron Lun

See Also

createGeneIdentifierMap, which creates the mapping used here to search for genes.

Examples

```
mapping <- searchGenes("9606", c("SNAP25", "NEUROD6", "ENSG00000139618"))
str(mapping)

# Checking that our genes were correctly identified:
ref <- fetchAllGenes("9606")
ref[mapping[[1]],]
ref[mapping[[2]],]
ref[mapping[[3]],]
```

`searchOverlappingSets` *Search for overlapping gene sets*

Description

Search for gene sets that overlap with genes in a user-supplied list.

Usage

```
searchOverlappingSets(
  species,
  genes,
  counts.only = TRUE,
  test.enrichment = TRUE,
  config = NULL
)
```

Arguments

`species`	String containing the NCBI taxonomy ID of the species of interest.
`genes`	Integer vector containing gene indices. Each gene index refers to a row of the data frame returned by `fetchAllGenes`. Duplicate entries are ignored.
`counts.only`	Boolean indicating whether to only report the number of overlapping genes for each set.
`test.enrichment`	Boolean indicating whether to compute a hypergeometric p-value for enrichment of `genes` in each set.
`config`	Configuration list, typically created by `newConfig`. If `NULL`, the default configuration is used.

Details

When performing a hypergeometric test for gene set enrichment, `present` should be used as the number of draws instead of `length(genes)`. This ensures that we do not consider genes in `genes` that are not present in any gene sets in Gesel, e.g., due to changes in annotation across genome versions or because they are pseudogenes or predicted genes. Otherwise, unknown genes would inappropriately increase the number of draws and inflate the enrichment p-value.

Value

A data frame where each row represents a set that overlaps with `genes`. This contains the following columns:

- `set`, an integer column containing the set index. This refers to a row of the data frame returned by `fetchAllSets`.
- `count`, an integer column containing the number of overlaps between the genes in the set and those in `genes`.

- which, a nested list that contains the entries of genes that overlap with those in the set. Only reported if counts.only = FALSE.
- size column, an integer column containing the size of each set. Only reported if test.enrichment = TRUE, as it is a by-product of the p-value calculation.
- pvalue column, a numeric column containing the hypergeometric p-value for overrepresentation of genes in the set. Only reported if test.enrichment = TRUE.

The row order is arbitrary.

If test.enrichment = TRUE, the data frame will also contain a present attribute. This is the number of genes in genes that are present in at least one set in the Gesel database for species.

Author(s)

Aaron Lun

Examples

```
# Pretend like we're interested in the first 10 genes.
overlaps <- searchOverlappingSets("9606", 1:10)
head(overlaps)

# More details on the overlapping sets.
all.sets <- fetchAllSets("9606")
all.sets[head(overlaps$set),]

# Computing the enrichment p-value manually. We take the upper tail after
# subtracting 1 to ensure that the probability mass of the observed
# number of overlapping genes is included in the p-value.
set.size <- all.sets$size[overlaps$set]
universe <- effectiveNumberOfGenes("9606")
p <- phyper(
    q = overlaps$count - 1,
    m = set.size,
    n = universe - set.size,
    k = attr(overlaps, "present"),
    lower.tail=FALSE
)
stopifnot(identical(p, overlaps$pvalue))

# For multiple testing correction, it is necessary to consider all sets
# in the database, as these were implicitly considered during the search
# though only a subset of them are reported by searchOverlappingSets.
fdr <- p.adjust(p, method="BH", n=nrow(all.sets))
summary(fdr <= 0.05)
```

searchSetText *Search set text*

Description

Search for sets based on their names and descriptions.

Usage

```
searchSetText(
  species,
  query,
  use.name = TRUE,
  use.description = TRUE,
  config = NULL
)
```

Arguments

species	String containing the NCBI taxonomy ID of the species of interest.
query	String containing one or more words to search on. A set is only matched if it matches to all of the tokens in the query. The * and ? wildcards can be used to match to any or one character, respectively.
use.name	Boolean indicating whether to search on the name of the set.
use.description	Boolean indicating whether to search on the description of the set.
config	Configuration list, typically created by newConfig. If NULL, the default configuration is used.

Value

Integer vector of set indices for the matching gene sets. Each set index refers to a row in the data frame returned by fetchAllSets.

Author(s)

Aaron Lun

Examples

```
out <- searchSetText("9606", "cancer")
fetchSomeSets("9606", head(out))

out <- searchSetText("9606", "innate immun*")
fetchSomeSets("9606", head(out))
```

`validateDatabaseFiles` *Validate Gesel database files*

Description

Validate Gesel database files against the specification at https://github.com/gesel-inc/gesel-spec.

Usage

```
validateDatabaseFiles(path, species, num.genes)
```

Arguments

`path`	String containing the path to a directory containing the database files.
`species`	String specifying the species in the form of its NCBI taxonomy ID.
`num.genes`	Integer specifying the total number of genes available for this species. This can be obtained as the return value of `validateGeneFiles`.

Value

`NULL`, invisibly. Invalid formatting of the files will cause an error to be raised.

Author(s)

Aaron Lun

Examples

```
# Mocking up some information.
collections <- data.frame(
    title=c("FOO", "BAR"),
    description=c("I am a foo", "I am a bar"),
    maintainer=c("Aaron", "Aaron"),
    source=c("https://foo", "https://bar")
)

set.info <- list(
    data.frame(
        name=sprintf("FOO_%i", seq_len(20)),
        description=sprintf("this is FOO %i", seq_len(20))
    ),
    data.frame(
        name=sprintf("BAR_%i", seq_len(50)),
        description=sprintf("this is BAR %i", seq_len(50))
    )
)

# Mocking up the gene sets.
num.genes <- 10000
```

```
set.membership <- list(
    lapply(seq_len(nrow(set.info[[1]])), function(i) {
        sample(num.genes, sample(500, 1))
    }),
    lapply(seq_len(nrow(set.info[[2]])), function(i) {
        sample(num.genes, sample(200, 1))
    })
)

# Now making the database files.
output <- tempfile()
dir.create(output)
prepareDatabaseFiles(
    "1234",
    collections,
    set.info,
    set.membership,
    num.genes,
    output
)

# Finally, validating them.
validateDatabaseFiles(output, "1234", num.genes)
```

`validateGeneFiles` *Validate Gesel gene annotation files*

Description

Validate Gesel gene annotation files against the specification at https://github.com/gesel-inc/gesel-spec.

Usage

```
validateGeneFiles(path, species, types = NULL)
```

Arguments

`path`	String containing the path to a directory containing the gene annotation files.
`species`	String specifying the species in the form of its NCBI taxonomy ID.
`types`	Character vector specifying the types of gene identifiers to validate, e.g., `"symbol"`, `"entrez"`, or `"ensembl"`, If `NULL`, all detected files for `species` in `path` are checked.

Value

The number of genes, to be used as `num.genes` in `validateDatabaseFiles`. Invalid formatting of the files will cause an error to be raised.

Author(s)

Aaron Lun

Examples

```
genes <- list(
    ensembl = list("ENSG1", c("ENSG2", "ENSG3"), character(0), "ENSG4"),
    entrez = list("1", character(0), c("2", "3", "4"), c("5", "6")),
    foobar = list("malat1", "neat1", "Gm1234", "LINC0000001")
)

tmp <- tempfile()
dir.create(tmp)
prepareGeneFiles("1234", genes, tmp)

validateGeneFiles(tmp, "1234")
```

Index

cacheDirectory, 2
CompressedCharacterList, *23*
consolidateBlockSize, 3, *25*
createGeneIdentifierMap, 4, *34*

databaseUrl *(*downloadDatabaseFile*)*, 5
downloadDatabaseFile, *2*, 5, *24*
downloadDatabaseRanges, 6, *25*, *31, 32*
downloadGeneFile, *2*, 8, *24*
downloadMultipartRanges
        *(*downloadDatabaseRanges*)*, 6

effectiveNumberOfGenes, 9

fetchAllCollections, 10, *12*, *20*, *26*
fetchAllGenes, *5*, 11, *13*, *15*, *17, 18*, *26*, *28*, *33–35*
fetchAllSets, *10*, 11, *13, 14*, *17–19*, *21*, *23*, *26*, *35*, *37*
fetchCollectionSizes, 12
fetchGenesForAllSets, 13, *14*, *23*, *33*
fetchGenesForSomeSets, *13*, 14, *23*, *30*, *33*
fetchGeneTypes, *5*, *11*, 15, *23*, *29*, *33, 34*
fetchGeneVersion, 16, *24*
fetchSetsForAllGenes, 17, *18*
fetchSetsForSomeGenes, *17*, 18
fetchSetSizes, 19
fetchSomeCollections, *10*, 20, *30*
fetchSomeSets, *12*, 21, *30*
findOverlappingSets
        *(*searchOverlappingSets*)*, 35
flushMemoryCache, 22, *25*

geneUrl *(*downloadGeneFile*)*, 8

loadAllSets, 22

mapGenesByName
        *(*createGeneIdentifierMap*)*, 4
mcols, *23*

newConfig, *5*, *7*, *9–21*, *23*, 24, *29*, *32–35*, *37*

phyper, *9*
prepareDatabaseFiles, 25, *28*
prepareGeneFiles, *26*, 27

querySets, 29

rangeConcurrency, 31
readDatabaseRanges, *7*, 32
renameGenesInSets, *14, 15*, *23*, 33
req_throttle, *31*

searchGenes, *5*, *30*, 34
searchOverlappingSets, *30*, 35
searchSetText, *29, 30*, 37

validateDatabaseFiles, *26*, 38, *39*
validateGeneFiles, *28*, *38*, 39