Package 'convertid'

April 1, 2026

Type Package

Title Convert Gene IDs Between Each Other and Fetch Annotations from Biomart

Version 0.3.0

Date 2026-03-31

Author Vidal Fey [aut, cre], Henrik Edgren [aut]

Maintainer Vidal Fey <vidal.fey@gmail.com>

Description Gene Symbols or Ensembl Gene IDs are converted using the Bimap interface in 'AnnotationDbi' in convertId2() but
that function is only provided as fallback mechanism for the most common use cases in data analysis. The main function
in the package is convert.bm() which queries BioMart using the full capacity of the API provided through the
'biomaRt' package. Presets and defaults are provided for convenience but all ``marts", ``filters" and ``attributes"
can be set by the user. Function convert.alias() converts Gene Symbols to Aliases and vice versa and function likely_symbol()
attempts to determine the most likely current Gene Symbol.

Depends AnnotationDbi, R (>= 3.5.0)

Imports plyr, stringr, biomaRt, stats, xml2, utils, rappdirs, assertthat, methods, httr, BiocFileCache

Suggests BiocManager, org.Hs.eg.db, org.Mm.eg.db, testthat (>= 3.0.0), mockery

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-04-01 09:40:02 UTC

Contents

`.addToCache` *Add values to cache*

Description

Add values to cache

Usage

```
.addToCache(bfc, result, hash)
```

Arguments

`bfc`	Object of class BiocFileCache, created by a call to BiocFileCache::BiocFileCache()
`result`	character; name of the file written to chache
`hash`	unique hash representing a query.

`.checkInCache` *Check whether value in cache exists*

Description

Check whether value in cache exists

Usage

```
.checkInCache(bfc, hash, verbose = FALSE)
```

Arguments

bfc	Object of class BiocFileCache, created by a call to BiocFileCache::BiocFileCache()
hash	unique hash representing a query.
verbose	logical; should additional verbose output be printed? Not currently used.
	This function returns TRUE if a record with the requested hash already exists in the file cache, otherwise returns FALSE.

.readFromCache *Read values from cache*

Description

Read values from cache

Usage

```
.readFromCache(bfc, hash)
```

Arguments

bfc	Object of class BiocFileCache, created by a call to BiocFileCache::BiocFileCache()
hash	unique hash representing a query.

---

convert.alias *Convert Symbols to Aliases and Vice Versa.*

Description

`convert.alias()` attempts to find all possible symbol-alias combinations for a given gene symbol, i.e., it assumes the input ID to be either an Alias or a Symbol and performs multiple queries to find all possible counterparts. The input IDs are converted to title and upper case before querying and all possibilities are tested. There are species presets for Human and Mouse annotations.

Usage

```
convert.alias(id, species = c("Human", "Mouse"), db = NULL)
```

Arguments

id	(`character`). Vector of gene symbols.
species	(`character`). One of "Human" and "Mouse". Defaults to "Human".
db	(`AnnotationDb object`). Annotation package object.

Value

A `data.frame` with two columns:

'SYMBOL': The official gene symbol.
'ALIAS': All possible aliases.

See Also

`select`

Examples

```
convert.alias("TRPV4")
```

`convert.bm` *Retrieve Additional Annotations from Biomart*

Description

`convert.bm()` is a wrapper for `get.bm()` which in turn makes use of `getBM()` from the *biomaRt* package. It takes a matrix or data frame with the IDs to be converted in one column or as row names as input and returns a data frame with additional annotations after cleaning the fetched annotations and merging them with the input data frame.

Usage

```
convert.bm(
  dat,
  id = "ID",
  biom.data.set = c("human", "mouse"),
  biom.mart = c("ensembl", "mouse", "snp", "funcgen", "plants"),
  host = "https://www.ensembl.org",
  biom.filter = "ensembl_gene_id",
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description"),
  biom.cache = rappdirs::user_cache_dir("biomaRt"),
  use.cache = TRUE,
  sym.col = "hgnc_symbol",
  rm.dups = FALSE,
  verbose = FALSE
)
```

Arguments

dat	matrix or data.frame. Matrix or data frame with the ids to be converted in a column or as row names.
id	character. Name of the column with the ids to be converted, special name "rownames" will use the row names.
biom.data.set	character of length one. Biomart data set to use.
biom.mart	character vector. Biomart to use (uses the first element of the vector), defaults to "ensembl".
host	character of length one. Host URL.
biom.filter	character of length one. Name of biomart filter, i.e., type of query ids, defaults to "ensembl_gene_id".
biom.attributes	character vector. Biomart attributes, i.e., type of desired result(s); if biom.filter is missing from this it will be added internally as it is needed for merging query result and input data.
biom.cache	character. Path name giving the location of the cache getBM() uses if use.cache=TRUE. Defaults to the value in the *BIOMART_CACHE* environment variable.
use.cache	(logical). Should getBM() use the cache? Defaults to TRUE as in the getBM() function and is passed on to that.
sym.col	character. Name of the column in the query result with gene symbols.
rm.dups	logical. Should duplicated input IDs ('biom.filter') be removed from the result?
verbose	(logical). Should verbose output be written to the console? Defaults to FALSE.

Details

Wrapped around 'get.bm'.

Value

A data frame with the retrieved information.

Author(s)

Vidal Fey

See Also

getBM

Examples

```
## Not run:
dat <- data.frame(ID=c("ENSG00000111199", "ENSG00000134121", "ENSG00000176102", "ENSG00000171611"))
bm <- convert.bm(dat)
bm

## End(Not run)
```

convertId2 *Convert Gene Symbols to Ensembl Gene IDs or vice versa*

Description

convertId2() uses the Bimap interface in AnnotationDbi to extract information from annotation packages. The function is limited to Human and Mouse annotations and is provided only as fallback mechanism for the most common use cases in data analysis. Please use the Biomart interface function convert.bm() for more flexibility.

Usage

```
convertId2(id, species = c("Human", "Mouse"))
```

Arguments

id	(character). Vector of gene symbols.
species	(character). One of "Human" and "Mouse". Defaults to "Human".

Value

A named character vector where the input IDs are the names and the query results the values.

See Also

Bimap-envirAPI

Examples

```
convertId2("ENSG00000111199")
convertId2("TRPV4")
```

get.bm *Make a Query to Biomart.*

Description

get.bm() is a user-friendly wrapper for getBM() from the *biomaRt* package with default settings for Human and Mouse. It sets all needed variables and performs the query.

Usage

```
get.bm(
  values,
  biom.data.set = c("human", "mouse"),
  biom.mart = c("ensembl", "mouse", "snp", "funcgen", "plants"),
  host = "https://www.ensembl.org",
  biom.filter = "ensembl_gene_id",
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol", "description"),
  biom.cache = rappdirs::user_cache_dir("biomaRt"),
  use.cache = TRUE,
  verbose = FALSE
)
```

Arguments

values	character vector of ids to be converted.
biom.data.set	character of length one. Biomart data set to use. Defaults to 'human' (internally translated to "hsapiens_gene_ensembl" if biom.mart="ensembl").
biom.mart	character vector. Biomart to use (uses the first element of the vector), defaults to "ensembl".
host	character of length one. Host URL.
biom.filter	character of length one. Name of biomart filter, i.e., type of query ids, defaults to "ensembl_gene_id".
biom.attributes	character vector. Biomart attributes, i.e., type of desired result(s); make sure query id type is included!
biom.cache	character. Path name giving the location of the cache getBM() uses if use.cache=TRUE. Defaults to the value in the *BIOMART_CACHE* environment variable.
use.cache	(logical). Should getBM() use the cache? Defaults to TRUE as in the getBM() function and is passed on to that.
verbose	(logical). Should verbose output be written to the console? Defaults to FALSE.

Value

A data frame with the retrieved information.

Author(s)

Vidal Fey

See Also

getBM

Examples

```
## Not run:
val <- c("ENSG00000111199", "ENSG00000134121", "ENSG00000176102", "ENSG00000171611")
bm <- get.bm(val)
bm

## End(Not run)
```

`likely_symbol` *Retrieve Symbol Aliases and Previous symbols to determine a likely current symbol*

Description

`likely_symbol()` downloads the latest version of the HGNC gene symbol database as a text file and query it to obtain symbol aliases, previous symbols and all symbols currently in use. (Optionally) assuming the input ID to be either an Alias or a Symbol or a Previous Symbol it performs multiple queries and compares the results of all possible combinations to determine a likely current Symbol. The downloaded HGNC table is cached for the duration of the R session to avoid repeated downloads.

Usage

```
likely_symbol(
  syms,
  alias_sym = TRUE,
  prev_sym = TRUE,
  orgnsm = "human",
  hgnc = NULL,
  hgnc_url = NULL,
  output = c("likely", "symbols", "all"),
  index_threshold = 10L,
  refresh = FALSE,
  verbose = TRUE
)
```

Arguments

`syms`	(`character`). Vector of Gene Symbols to be tested.
`alias_sym`	(`logical`). Should the input be assumed to be an Alias? Defaults to `TRUE`.
`prev_sym`	(`logical`). Should the input be assumed to be a Previous Symbol? Defaults to `TRUE`.
`orgnsm`	(`character`). The organism for which the Symbols are tested.
`hgnc`	(`data.frame`). An optional data frame with the needed HGNC annotations. (Needs to match the format available at `hgnc_url`!) When supplied, bypasses both the cache and any download.

`hgnc_url` (`character`). URL where to download the HGNC annotation dataset. Defaults to `"https://storage.googleapis.com/public-download-files/hgnc/tsv/tsv/hgnc_complete_s`

`output` (`character`). One of `"likely"`, `"symbols"` and `"all"`. Determines the scope of the output data frame. Defaults to `"likely"` which will return the input Symbol and the determined likely Symbol.

`index_threshold`
(`integer`). Minimum number of unique input symbols above which inverted indices are pre-built for alias and previous symbol lookups, giving a substantial speedup for large inputs. Below this threshold the original row-scan is used, which is faster for very small inputs (e.g. a single symbol lookup) where the index-building overhead would dominate. Defaults to `10L`.

`refresh` (`logical`). Should the cached HGNC table be discarded and re-downloaded? Defaults to `FALSE`. Use `TRUE` to force a fresh download within the same R session, e.g. after a known HGNC update.

`verbose` (`logical`). Should messages be written to the console? Defaults to `TRUE`.

Details

The HGNC table is downloaded once per R session and cached in a package-level environment. Subsequent calls reuse the cached table without any network access. If the cached table is more than 3 days old a warning message is emitted recommending a refresh, since the HGNC database is updated monthly. To force a fresh download within the same session use `refresh = TRUE` or start a new R session.

When the number of unique input symbols is at or above `index_threshold`, inverted indices (hash tables) are pre-built from the HGNC table so that each per-symbol lookup is O(1) rather than O(nrow(hgnc)), giving roughly a 50-100x speedup for batch inputs. For small inputs the original row-scan is retained to avoid the index-building overhead.

Value

A `data.frame` with the following columns depending on the `output` setting. `output="likely"`:

'likely_symbol'
'input_symbol'

`output="symbols"`:

'current_symbols'
'likely_symbol'
'input_symbol'
'all_symbols'

`output="all"`:

'orig_input'
'organism'
'current_symbols'
'likely_symbol'
'input_symbol'
'all_symbols'

Note

Only fully implemented for Human for now.

Examples

```
## Not run:
# Single symbol lookup (uses row-scan, no index overhead)
likely_symbol("CCBL1")

# Second call reuses cached HGNC table — no download
likely_symbol("KAAT1")

# Force a fresh download within the same session
likely_symbol("CCBL1", refresh = TRUE)

# Batch lookup (builds index for speed)
likely_symbol(c("ABCC4", "ACPP", "KIAA1524"))

# Supply a pre-loaded table to bypass cache and download entirely
likely_symbol(c("ABCC4", "ACPP"), hgnc = my_hgnc_table)

## End(Not run)
```

`todisp2` *Convenience Function to Convert Ensembl Gene IDs to Gene Symbols*

Description

`todisp2()` uses Biomart by employing `get.bm()` to retrieve Gene Symbols for a set of Ensembl Gene IDs. It is mainly meant as a fast way to convert IDs in standard gene expression analysis output to Symbols, e.g., for visualisation, which is why the input ID type is hard-coded to ENSG IDs. If Biomart is not available the function can fall back to use `convertId2()` or a user-provided data frame with corresponding ENSG IDs and Symbols.

Usage

```
todisp2(
  ensg,
  lab = NULL,
  biomart = TRUE,
```

```
  biom.data.set = "hsapiens_gene_ensembl",
  biom.mart = "ensembl",
  host = "https://www.ensembl.org",
  biom.filter = "ensembl_gene_id",
  biom.attributes = c("ensembl_gene_id", "hgnc_symbol"),
  biom.cache = rappdirs::user_cache_dir("biomaRt"),
  use.cache = TRUE,
  keep.original = TRUE,
  verbose = FALSE
)
```

Arguments

ensg	(character). Vector of Ensemble Gene IDs. Other ID types are not yet supported.
lab	(data.frame). A data frame with Ensembl Gene IDs as row names and Gene Symbols in the only column.
biomart	(logical). Should Biomart be used? Defaults to TRUE.
biom.data.set	character of length one. Biomart data set to use. Defaults to 'hsapiens_gene_ensembl'
biom.mart	character vector. Biomart to use (uses the first element of the vector), defaults to "ensembl".
host	character of length one. Host URL.
biom.filter	character of length one. Name of biomart filter, i.e., type of query ids, defaults to "ensembl_gene_id".
biom.attributes	character vector. Biomart attributes, i.e., type of desired result(s); make sure query id type is included!
biom.cache	character. Path name giving the location of the cache getBM() uses if use.cache=TRUE. Defaults to the value in the *BIOMART_CACHE* environment variable.
use.cache	(logical). Should getBM() use the cache? Defaults to TRUE as in the getBM() function and is passed on to that.
keep.original	(logical). Should the order and length of the input vector be preserved, i.e., should also IDs missing after conversion be kept? Defaults to TRUE.
verbose	(logical). Should verbose output be written to the console? Defaults to FALSE.

Value

A character vector of Gene Symbols.

See Also

get.bm

Examples

```
## Not run:
val <- c("ENSG00000111199", "ENSG00000134121", "ENSG00000176102", "ENSG00000171611")
sym <- todisp2(val)
sym

## End(Not run)
```

`unify_gene_ids` *Unify gene IDs from BioMart and AnnotationDbi lookups*

Description

Takes a data frame with Ensembl gene IDs (and optionally gene symbols) and returns a deduplicated data frame with unified HGNC symbols, using a priority-based reconciliation of BioMart and AnnotationDbi results.

Usage

```
unify_gene_ids(
  genes,
  ensg_col = "ensembl_gene_id",
  symbol_col = NULL,
  host = "https://www.ensembl.org",
  biomart_fallback = c("https://uswest.ensembl.org", "https://asia.ensembl.org",
    "https://useast.ensembl.org"),
  keep_intermediates = FALSE,
  verbose = FALSE
)
```

Arguments

`genes`	A data frame with at minimum an Ensembl gene ID column or a character vector of Ensembl gene IDs.
`ensg_col`	Name of the column containing Ensembl gene IDs. Default: `"ensembl_gene_id"`.
`symbol_col`	Name of the column containing gene symbols, or `NULL` if absent (ENSG-only mode). Default: `NULL`.
`host`	BioMart host URL. Default: `"https://www.ensembl.org"`.
`biomart_fallback`	Character vector of fallback BioMart host URLs to try if the primary host fails. Set to `NULL` to disable fallback.
`keep_intermediates`	Logical; if `TRUE`, the intermediate lookup columns `hgnc_symbol_2` and `ensg_2` are retained in the output. Useful for debugging. Default: `FALSE`.
`verbose`	Logical; if `TRUE`, print progress and summary messages. Default: `FALSE`.

Details

Requires the Bioconductor packages **org.Hs.eg.db** and **AnnotationDbi**. These are not hard dependencies but will be checked at runtime with an informative error if missing.

Deduplication passes

The function performs two sequential deduplication passes via the internal `dedup_gene_ids()` function:

1. Deduplicate by `gene_name` (if available) or `ensembl_gene_id`, resolving multiple ENSG IDs mapping to the same gene name.
2. Deduplicate by `hgnc_symbol`, resolving cases where multiple gene names resolve to the same symbol.

Symbol assignment priority

The guiding principle is that **AnnotationDbi confirmation outranks BioMart ordering**. AnnotationDbi (**org.Hs.eg.db**) reflects a stable, versioned annotation database, while BioMart returns the current Ensembl release which may be ahead of annotations used to build real-world count matrices. Preferring AnnotationDbi-confirmed IDs therefore maximises compatibility with count matrices from sequencing providers whose pipelines are not frequently updated.

Within each group of rows sharing a `gene_name`, the following priority order is applied until a single row is selected:

1. **Pre-filter:** If any row has `hgnc_symbol_2 == gene_name` (AnnotationDbi confirms the symbol), rows with `hgnc_symbol_2 == NA` are discarded first. This ensures that an AnnotationDbi-confirmed row is never passed over in favour of an unconfirmed one merely because the latter happens to have `hgnc_symbol == gene_name` from BioMart.
2. **BioMart symbol match:** Rows where `hgnc_symbol == gene_name` (and is not a raw ENSG placeholder).
3. **AnnotationDbi symbol match:** Rows where `hgnc_symbol_2 == gene_name` (and is not a raw ENSG placeholder).
4. **Both sources agree:** Rows where `hgnc_symbol == hgnc_symbol_2`, indicating cross-source confirmation.
5. **BioMart ENSG confirmation:** Rows whose `ensembl_gene_id` matches the first entry in the `ensg_2` `///`-separated list returned by AnnotationDbi. Note that `ensg_2` list ordering is not considered a reliable preference signal on its own; this filter is intentionally placed after source-agreement filters.
6. **Drop ENSG placeholders:** Rows where `hgnc_symbol` is still a raw ENSG ID are deprioritised.
7. **Last resort:** When all disambiguation fields (`hgnc_symbol_2`, `ensg_2`) are `NA` across the entire group, the first row is taken. When rows are otherwise identical in all metadata, the newer ENSG ID (as returned by BioMart) is preferred as the more current annotation.

The second pass (by `hgnc_symbol`) applies the same principle but additionally prefers rows whose `hgnc_symbol` matches `gene_name`, and uses AnnotationDbi ENSG confirmation as a tiebreaker before falling back to `x[1, ]`.

ENSG placeholder resolution

After the filter chain, any remaining rows where hgnc_symbol is a raw ENSG placeholder are fixed: if hgnc_symbol_2 is available it is used; otherwise gene_name is used (or ensembl_gene_id in ENSG-only mode). This allows rows with ENSG placeholders from BioMart to be correctly resolved in the second pass via their hgnc_symbol_2 value.

BioMart fallback

BioMart queries are attempted with graceful fallback through mirror hosts. If all hosts fail the function proceeds with AnnotationDbi results only. If both BioMart and AnnotationDbi fail entirely, the input is returned with ENSG IDs used as hgnc_symbol values.

Value

A deduplicated data frame with unified HGNC symbols in the hgnc_symbol column, plus hgnc_symbol_2 and ensg_2 columns from the AnnotationDbi lookups.

Examples

```
## Not run:
# Example input: two-column data frame with Ensembl IDs and gene symbols,
# as typically produced by a sequencing provider's count matrix annotation
my_genes <- data.frame(
  gene_id   = c("ENSG00000000003", "ENSG00000000419", "ENSG00000000460",
                "ENSG00000012048", "ENSG00000075624", "ENSG00000111640",
                "ENSG00000141510", "ENSG00000146648"),
  gene_name = c("TSPAN6", "DPM1", "FIRRM",
                "BRCA1",  "ACTB",  "GAPDH",
                "TP53",   "EGFR"),
  stringsAsFactors = FALSE
)

# With gene symbols (full mode)
result <- unify_gene_ids(my_genes,
                         ensg_col   = "gene_id",
                         symbol_col = "gene_name",
                         verbose    = TRUE)

# ENSG-only (e.g. from count matrix row names, no symbol column available)
ensg_only <- data.frame(
  ensembl_gene_id  = my_genes$gene_id,
  stringsAsFactors = FALSE
)
result_ensg <- unify_gene_ids(ensg_only, verbose = TRUE)

## End(Not run)
```

Index