Package 'IDConverter'

July 25, 2026

Title Convert Identifiers in Biological Databases

Version 0.4.0

Date 2026-07-25

Maintainer Shixiang Wang <w_shixiang@163.com>

Description Identifiers in biological databases connect different levels of metadata, phenotype data or genotype data. This tool is designed to easily convert identifiers within or between different biological databases (Wang, Shixiang, et al. (2021) <DOI:10.1371/journal.pgen.1009557>).

License MIT + file LICENSE

URL https://github.com/WangLabCSU/IDConverter, https://wanglabcsu.github.io/IDConverter/

BugReports https://github.com/WangLabCSU/IDConverter/issues

Depends R (>= 3.5.0)

Imports data.table, httr, tibble

Suggests biomaRt, covr, knitr, readr, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

NeedsCompilation no

Author Shixiang Wang [aut, cre] (ORCID: <https://orcid.org/0000-0001-9855-7357>)

Repository CRAN

Date/Publication 2026-07-25 17:00:11 UTC

Contents

`build_annotables` *Build Gene Annotation Tables from Ensembl BioMart*

Description

Queries Ensembl BioMart directly to build up-to-date gene annotation tables ("annotables") using recipes derived from the annotables package. This provides the latest annotations from Ensembl when internet is available, as an alternative to the fixed-version tables stored on Zenodo.

Usage

```
build_annotables(
  recipes = NULL,
  tx2gene = TRUE,
  include_synonyms = FALSE,
  mirrors = ENSEMBL_MIRRORS,
  cache_dir = getOption("IDConverter.datapath", tempdir()),
  verbose = TRUE
)
```

Arguments

`recipes`	character vector of recipe names to build, or `NULL` to build all available recipes. Use `ls_annotables()` to see which organisms are available (matching the "gene" + "tx2gene" table names). Recipe names include `"grch38"`, `"grch37"`, `"grcm38"`, `"bdgp6"`, `"galgal5"`, `"rnor6"`, `"mmul801"`, `"wbcel235"`, `"cfamiliaris"`, `"drerio"`, `"sscrofa"`.

tx2gene	if TRUE (default), also build transcript-to-gene mapping tables (appended with _tx2gene suffix).
include_synonyms	if TRUE, also fetch external_synonym (gene name aliases) from BioMart and include a synonym column in the output tables. Default FALSE for faster queries.
mirrors	character vector of Ensembl mirror URLs to try in order. The function automatically falls back to the next mirror on failure.
cache_dir	directory path to cache downloaded tables as .rda files. Set to NULL to skip caching. Default uses the package's data path (customizable via options(IDConverter.datapath = ...)).
verbose	if TRUE, print progress messages.

Details

Requirements: This function requires the Bioconductor package **biomaRt**. Install it with:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("biomaRt")
```

Mirror fallback: Ensembl mirrors can be unreliable. The function tries each mirror in mirrors for each recipe until one succeeds. If all mirrors fail for a recipe, that recipe is skipped with a warning.

Caching: When cache_dir is set, successfully downloaded tables are saved as .rda files. Subsequent calls with the same cache_dir will load from cache instead of re-querying, unless cache_dir = NULL.

Value

a named list of data.frame objects (tibbles). The list names correspond to recipe names (and _tx2gene variants if tx2gene = TRUE). Returns invisible(NULL) on total failure.

References

- https://github.com/stephenturner/annotables
- https://www.ensembl.org/info/data/biomart.html

See Also

load_data() for loading fixed-version tables from Zenodo, ls_annotables() for listing available tables.

Examples

```
# Build a single annotable table
grch38 <- build_annotables("grch38", tx2gene = FALSE)
head(grch38[[1]])

# Build all available tables (requires internet + biomaRt)
```

```
all_tables <- build_annotables()
names(all_tables)
```

---

convert_custom *Convert Identifiers with Custom Database*

Description

Convert Identifiers with Custom Database

Usage

```
convert_custom(x, from = NULL, to = NULL, dt = NULL, multiple = FALSE)
```

Arguments

x	A character vector to convert.
from	Which identifier type to be converted.
to	Identifier type convert to.
dt	A data.frame as database for conversion.
multiple	if TRUE, return a data.table instead of a string vector, so multiple identifier mappings can be kept.

Value

A character vector.

Examples

```
dt <- data.table::data.table(UpperCase = LETTERS[1:5], LowerCase = letters[1:5])
dt
x <- convert_custom(c("B", "C", "E", "E", "FF"), from = "UpperCase", to = "LowerCase", dt = dt)
x
```

convert_hm_genes *Convert Gene IDs between Ensembl and Hugo Symbol System*

Description

Supports human (hg38, hg19, T2T), mouse (mm10, mm9), and worm (ce11, C. elegans) genomes.

Usage

```
convert_hm_genes(
  IDs,
  type = c("ensembl", "symbol"),
  genome_build = c("hg38", "hg19", "mm10", "mm9", "ce11", "T2T"),
  multiple = FALSE
)
```

Arguments

IDs	a character vector to convert.
type	type of input IDs, could be 'ensembl' or 'symbol'.
genome_build	reference genome build.
multiple	if TRUE, return a data.table instead of a string vector, so multiple identifier mappings can be kept.

Value

a vector or a data.table.

Note

This function matches against **primary** gene symbols only. For resolving outdated or alternative gene names (aliases like "MLL" -> "KMT2A"), use build_annotables() with include_synonyms = TRUE followed by resolve_gene_aliases().

See Also

resolve_gene_aliases() for resolving gene symbol aliases.

Examples

```
convert_hm_genes("ENSG00000243485")
convert_hm_genes("ENSG00000243485", multiple = TRUE)
convert_hm_genes(c("TP53", "KRAS", "EGFR", "MYC"), type = "symbol")
```

convert_hm_orthologs *Convert Gene IDs Between Human and Mouse via Orthology*

Description

Maps gene symbols or Ensembl IDs between human and mouse using Ensembl BioMart orthology data. This resolves homologous gene relationships (e.g., human TP53 <-> mouse Trp53).

Usage

```
convert_hm_orthologs(
  IDs,
  from_species = c("human", "mouse"),
  to_species = c("mouse", "human"),
  from_type = c("symbol", "ensembl"),
  to_type = c("symbol", "ensembl", "both"),
  multiple = FALSE,
  high_confidence_only = TRUE,
  mirrors = ENSEMBL_MIRRORS,
  cache_dir = getOption("IDConverter.datapath", tempdir()),
  verbose = TRUE
)
```

Arguments

IDs	a character vector of gene symbols or Ensembl IDs.
from_species	source species: "human" (default) or "mouse".
to_species	target species: "mouse" (default) or "human". Must differ from from_species.
from_type	type of input IDs: "symbol" (default) or "ensembl".
to_type	type of output IDs: "symbol" (default), "ensembl", or "both" to return both columns.
multiple	if TRUE, return a data.frame with all ortholog matches (e.g., one-to-many relationships).
high_confidence_only	if TRUE (default), restrict to orthologs with high confidence (confidence == 1) in Ensembl.
mirrors	character vector of Ensembl mirror URLs.
cache_dir	directory to cache query results. Set NULL to skip.
verbose	if TRUE, print progress messages.

Details

Requirements: This function requires the Bioconductor package **biomaRt**. Install it with:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("biomaRt")
```

Orthology data is queried live from Ensembl and cached locally.

Value

If `multiple = FALSE`, a character vector of converted IDs (NA for unmatched). If `multiple = TRUE` or `to_type = "both"`, a `data.frame`.

Examples

```
# Human symbol -> mouse symbol
convert_hm_orthologs(c("TP53", "KRAS", "EGFR"))

# Mouse symbol -> human symbol
convert_hm_orthologs(c("Trp53", "Kras"), from_species = "mouse", to_species = "human")

# Human Ensembl -> mouse Ensembl
convert_hm_orthologs("ENSG00000141510", from_type = "ensembl", to_type = "ensembl")
```

---

convert_icgc *Convert ICGC Identifiers*

Description

Run `data("icgc")` to see detail database for conversion.

Usage

```
convert_icgc(
  x,
  from = "icgc_specimen_id",
  to = "icgc_donor_id",
  multiple = FALSE
)
```

Arguments

`x`	A character vector to convert.
`from`	Which identifier type to be converted. One of icgc_sample_id, submitted_sample_id, icgc_specimen_id, submitted_specimen_id, icgc_donor_id, submitted_donor_id.
`to`	Identifier type convert to. Same as parameter `from`.
`multiple`	if `TRUE`, return a `data.table` instead of a string vector, so multiple identifier mappings can be kept.

Value

A character vector.

Examples

```
x <- convert_icgc("SP29019")
x

## Not run:
convert_icgc("SA170678")

## End(Not run)
```

`convert_pcawg` *Convert PCAWG Identifiers*

Description

Run `data("pcawg_full")` or `data("pcawg_simple")` to see detail database for conversion. The `pcawg_simple` database only contains PCAWG white-list donors.

Usage

```
convert_pcawg(
  x,
  from = "icgc_specimen_id",
  to = "icgc_donor_id",
  db = c("full", "simple"),
  multiple = FALSE
)
```

Arguments

`x`	A character vector to convert.
`from`	Which identifier type to be converted. For db "full", one of donor_unique_id, submitter_donor_id, icgc_donor_id, aliquot_id, submitter_specimen_id, icgc_specimen_id, submitter_sample_id, icgc_sample_id. For db "simple", one of tumour_specimen_aliquot_id, normal_specimen_aliquot_id, donor_unique_id, submitted_donor_id, icgc_donor_id, icgc_sample_id, icgc_specimen_id, submitted_specimen_id, submitted_sample_id, tcga_specimen_uuid, tcga_sample_uuid, tcga_donor_uuid.
`to`	Identifier type convert to. Same as parameter `from`.
`db`	Database, one of "full" (for `data("pcawg_full")`) or "simple" (for `data("pcawg_simple")`).
`multiple`	if `TRUE`, return a `data.table` instead of a string vector, so multiple identifier mappings can be kept.

Value

A character vector.

Examples

```
x <- convert_pcawg("SP1677")
x

y <- convert_pcawg("DO804",
  from = "icgc_donor_id",
  to = "icgc_specimen_id", multiple = TRUE
)
y

## Not run:
convert_pcawg("SA5213")

## End(Not run)
```

---

convert_tcga *Convert TCGA Identifiers*

Description

Run `data("tcga")` to see detail database for conversion.

Usage

```
convert_tcga(x, from = "sample_id", to = "submitter_id", multiple = FALSE)
```

Arguments

`x`	A character vector to convert.
`from`	Which identifier type to be converted. One of case_id, aliquot_ids, submitter_aliquot_ids, submitter_id, sample_id.
`to`	Identifier type convert to. Same as parameter `from`.
`multiple`	if `TRUE`, return a `data.table` instead of a string vector, so multiple identifier mappings can be kept.

Value

A character vector.

Examples

```
x <- convert_tcga("TCGA-02-0001-10")
x

## Not run:
convert_tcga("TCGA-02-0001-10A-01W-0188-10")

## End(Not run)
```

`filter_tcga_barcodes` *Filter TCGA Replicate Sample Barcodes*

Description

Check details for filter rules.

Usage

```
filter_tcga_barcodes(
  tsb,
  analyte_target = c("DNA", "RNA"),
  decreasing = TRUE,
  analyte_position = 20,
  plate = c(22, 25),
  portion = c(18, 19),
  filter_FFPE = FALSE
)
```

Arguments

`tsb`	a vector of TCGA sample barcodes.
`analyte_target`	type of barcodes, "DNA" or "RNA".
`decreasing`	if `TRUE` (default), use decreasing order to select barcode to keep.
`analyte_position`	bit position for analyte. DON'T CHANGE IT if you don't understand.
`plate`	bit position for plate. DON'T CHANGE IT if you don't understand.
`portion`	bit position for portion. DON'T CHANGE IT if you don't understand.
`filter_FFPE`	if `TRUE` (`FALSE` is default), filter out FFPE samples.

Details

In many instances there is more than one aliquot for a given combination of individual, platform, and data type. However, only one aliquot may be ingested into Firehose. Therefore, a set of precedence rules are applied to select the most scientifically advantageous one among them. Two filters are applied to achieve this aim: an Analyte Replicate Filter and a Sort Replicate Filter.

Analyte Replicate Filter:

The following precedence rules are applied when the aliquots have differing analytes. For RNA aliquots, T analytes are dropped in preference to H and R analytes, since T is the inferior extraction protocol. If H and R are encountered, H is the chosen analyte. This is somewhat arbitrary and subject to change, since it is not clear at present whether H or R is the better protocol. If there are multiple aliquots associated with the chosen RNA analyte, the aliquot with the later plate number is chosen. For DNA aliquots, D analytes (native DNA) are preferred over G, W, or X (whole-genome amplified) analytes, unless the G, W, or X analyte sample has a higher plate number.

Sort Replicate Filter:

The following precedence rules are applied when the analyte filter still produces more than one sample. The sort filter chooses the aliquot with the highest lexicographical sort value, to ensure that the barcode with the highest portion and/or plate number is selected when all other barcode fields are identical.

NOTE: Basically, user provides tsb and analyte_target is fine.

Value

a barcode list.

References

Rules:

- https://confluence.broadinstitute.org/display/GDAC/FAQ#FAQ-sampleTypesQWhatTCGAsampletypesareFi

FFPE cases:

- http://gdac.broadinstitute.org/runs/sampleReports/latest/FPPP_FFPE_Cases.html

Examples

```
filter_tcga_barcodes(c("TCGA-44-2656-01B-06D-A271-08", "TCGA-44-2656-01B-06D-A273-01"))
filter_tcga_barcodes(c("TCGA-44-2656-01B-06D-A271-08", "TCGA-44-2656-01B-06D-A273-01"),
  filter_FFPE = TRUE
)
```

---

icgc *ICGC Sample Identifiers*

Description

ICGC Sample Identifiers

Format

A data frame with 155874 rows and 6 variables.

Source

https://dcc.icgc.org/repositories

Examples

```
load_data("icgc")
```

---

load_data *Load Data from Local or Remote Zenodo Repository*

Description

Data are stored in remote Zenodo repo. This function will help download required data and load it into R. For datasets bundled with the package (`tcga`, `icgc`, `pcawg_full`, `pcawg_simple`), local data is used directly without network access.

Usage

```
load_data(x)
```

Arguments

x	a dataset name.

Value

typically a `data.frame`, depends on `x`.

Note

The Zenodo repository contains fixed-version data. For the latest Ensembl gene annotations, use `build_annotables()` to query BioMart directly.

See Also

`build_annotables()` for live Ensembl BioMart queries.

Examples

```
load_data("pcawg_full")
load_data("pcawg_simple")
load_data("tcga")
load_data("icgc")
```

`ls_annotables` *List Available Annotation Tables*

Description

Lists all annotation tables that can be loaded via `load_data()` from the Zenodo repository (fixed Ensembl version). For the latest Ensembl annotations, use `build_annotables()` to query BioMart directly.

Usage

```
ls_annotables()
```

Details

These tables have basic annotation information from **Ensembl Genes** for:

- Human build 38 (`grch38`)
- Human build 37 (`grch37`)
- Mouse (`grcm38`)
- Rat (`rnor6`)
- Chicken (`galgal5`)
- Worm (`wbcel235`)
- Fly (`bdgp6`)
- Macaque (`mmul801`) Where each table contains:
- `ensgene`: Ensembl gene ID
- `entrez`: Entrez gene ID
- `symbol`: Gene symbol
- `chr`: Chromosome
- `start`: Start
- `end`: End
- `strand`: Strand
- `biotype`: Protein coding, pseudogene, mitochondrial tRNA, etc.
- `description`: Full gene name/description Additionally, there are `tx2gene` tables that link Ensembl gene IDs to Ensembl transcript IDs.

Use `build_annotables()` to fetch the latest annotations from Ensembl BioMart, which also supports additional organisms (dog, zebrafish, pig).

Value

a character vector of available table names.

References

https://github.com/stephenturner/annotables

See Also

build_annotables() for live Ensembl BioMart queries.

Examples

```
ls_annotables()
load_data(ls_annotables()[1])
```

---

pair_gdc_samples *Pair Tumor-Normal Samples from GDC Manifest*

Description

Parses a GDC manifest file (or result from parse_gdc_file_uuid()) and creates paired tumor-normal sample information. This is useful for generating matched pair lists for downstream genomic analyses.

Usage

```
pair_gdc_samples(x, prefer_blood_normal = TRUE)
```

Arguments

x	a path to a GDC manifest file, a vector of GDC file UUIDs, or a data.frame returned by parse_gdc_file_uuid().
prefer_blood_normal	if TRUE (default), prefer blood-derived normal samples over solid tissue normals when both are available for the same case.

Details

The function identifies tumor vs normal samples based on the TCGA barcode: samples with position 14-15 less than "10" are classified as tumor, others as normal. When both blood-derived and solid tissue normals are available for a case, blood normal is preferred by default.

Value

a data.frame with columns:

pair_id	unique pair identifier (generated from tumor sample ID)
case_id	TCGA case (patient) ID (first 12 characters of barcode)
tumor_sample	TCGA tumor sample barcode (first 15 characters), NA if no tumor for this case

normal_sample	TCGA normal sample barcode (first 15 characters), NA if no normal for this case
file_id_tumor	GDC file UUID for the tumor sample
file_id_normal	GDC file UUID for the normal sample
tissue_type	tissue type string from manifest (e.g. "Blood Derived Normal")

Examples

```
# Mock data example (works offline)
mock <- data.frame(
  submitter_id = c("TCGA-02-0001-01B-02D-A271-08",
                   "TCGA-02-0001-10B-01D-A273-01"),
  sample_type  = c("Primary Tumor", "Blood Derived Normal"),
  file_id      = c("fe522fc8-e690-49b9-b3b6-fa3658705057",
                   "2c16506f-1110-4d60-81e3-a85233c79909"),
  stringsAsFactors = FALSE
)
pair_gdc_samples(mock)

## Not run:
# From a real GDC manifest file
info <- pair_gdc_samples("gdc_manifest.txt")
head(info)

## End(Not run)
```

---

parse_gdc_file_uuid *Parse Sample ID from GDC Portal File UUID*

Description

Parse Sample ID from GDC Portal File UUID

Usage

```
parse_gdc_file_uuid(
  x,
  legacy = FALSE,
  fields = "cases.samples.submitter_id,cases.samples.sample_type,file_id",
  token = NULL,
  max_try = 5L
)
```

Arguments

x	a GDC manifest file or a vector of file UUIDs.
legacy	if use GDC legacy data.

fields	a list of fields to query. If it is a string, then fields should be separated by comma. It could also be a vector. See https://docs.gdc.cancer.gov/API/Users_Guide/Appendix_A_Available_Fields/#file-fields for list.
token	the token used for querying.
max_try	maximum try time.

Value

a data.frame

See Also

pair_gdc_samples() for pairing tumor-normal samples from the parsed manifest result.

Examples

```
## Not run:
parse_gdc_file_uuid("fe522fc8-e690-49b9-b3b6-fa3658705057")
parse_gdc_file_uuid(
  c(
    "fe522fc8-e690-49b9-b3b6-fa3658705057",
    "2c16506f-1110-4d60-81e3-a85233c79909"
  )
)

## End(Not run)
```

---

pcawg_full *PCAWG Full Sample Identifiers*

Description

PCAWG Full Sample Identifiers

Format

A data frame with 7255 rows and 8 variables.

Source

https://dcc.icgc.org/releases/PCAWG

Examples

```
load_data("pcawg_full")
```

pcawg_simple *PCAWG Mutation Related Simplified Sample Identifiers*

Description

This dataset contains less records than `data("pcawg_full")` but with more ID columns. Of note, only white-list donors included.

Format

A data frame with 2583 rows and 12 variables.

Source

https://www.nature.com/articles/s41586-020-1969-6

Examples

```
load_data("pcawg_simple")
```

resolve_gene_aliases *Resolve Gene Symbol Aliases*

Description

Maps gene symbols through known aliases (synonyms) to their current official symbols and Ensembl gene IDs. This helps when working with outdated or alternative gene names (e.g., "MLL" -> "KMT2A").

Usage

```
resolve_gene_aliases(symbols, annotable, multiple = FALSE)
```

Arguments

`symbols`	a character vector of gene symbols to resolve.
`annotable`	a `data.frame` from `build_annotables()` with `include_synonyms = TRUE`. Must contain `symbol` and `synonym` columns.
`multiple`	if `TRUE`, return a `data.frame` with all matches (including cases where one alias maps to multiple genes).

Value

a data.frame (tibble) with columns:

query	the input gene symbols
symbol	resolved official gene symbol (NA if unmatched)
ensgene	resolved Ensembl gene ID (NA if unmatched)

When multiple = TRUE, each row is a single query->match pair, so one query may appear in multiple rows.

Examples

```
# Build annotables with synonym support
ann <- build_annotables("grch38", include_synonyms = TRUE, tx2gene = FALSE)

# Resolve aliases
resolve_gene_aliases(c("TP53", "MLL", "NOTAGENE"), ann[[1]])

# Multiple match mode
resolve_gene_aliases(c("TP53", "MLL"), ann[[1]], multiple = TRUE)
```

---

tcga *TCGA Case Identifiers*

Description

How to get the dataset can be viewed in code under data-raw. Cases in case_id column can be directly mapped to a GDC portal page, e.g. https://portal.gdc.cancer.gov/cases/30a1fe5e-5b12-472c-aa86-c2db81

Format

A data frame with 150849 rows and 5 variables.

Source

https://portal.gdc.cancer.gov/

Examples

```
load_data("tcga")
```

Index