Package 'IDConverter'

March 14, 2023

Title Convert Identifiers in Biological Databases

Version 0.3.4

Date 2023-03-14

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/ShixiangWang/IDConverter

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

Depends R (>= 3.5.0)

Imports data.table, httr, tibble

Suggests covr, readr, testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

NeedsCompilation no

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

Repository CRAN

Date/Publication 2023-03-14 06:40:06 UTC

R topics documented:

`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 Human/Mouse Gene IDs between Ensembl and Hugo Symbol System*

Description

Convert Human/Mouse Gene IDs between Ensembl and Hugo Symbol System

Usage

```
convert_hm_genes(
  IDs,
  type = c("ensembl", "symbol"),
  genome_build = c("hg38", "hg19", "mm10", "mm9"),
  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.

Examples

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

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 .
`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 . For db "simple", one of .
`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 .
`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.

Usage

```
load_data(x)
```

Arguments

x	a dataset name.

Value

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

Examples

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

ls_annotables *List Annotation Tables from* annotables *package*

Description

The tables are obtained from annotables package and stored in Zenodo for better management. They can be downloaded and loaded with `load_data()`. See details for more info.

Usage

```
ls_annotables()
```

Details

Many bioinformatics tasks require converting gene identifiers from one convention to another, or annotating gene identifiers with gene symbol, description, position, etc. Sure, biomaRt does this for you, but users may get tired of remembering biomaRt syntax and hammering Ensembl's servers every time. 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.

NOTE, the description above is copied from README of `annotables` package. If you are unclear to the data tables, please refer to annotables.

Value

a `data.frame`

References

https://github.com/stephenturner/annotables

Examples

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

`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`

Examples

```
parse_gdc_file_uuid("fe522fc8-e690-49b9-b3b6-fa3658705057")
parse_gdc_file_uuid(
  c(
    "fe522fc8-e690-49b9-b3b6-fa3658705057",
    "2c16506f-1110-4d60-81e3-a85233c79909"
  )
)
```

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")
```

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