Package 'dbparser'

November 8, 2026

Title Drugs Databases Parser

Version 2.3.0

Description Parses drug and gene databases from their downloaded files: the 'DrugBank' XML database <https://go.drugbank.com/>, the 'OnSIDES' and 'TWOSIDES' adverse drug event databases, Human Phenotype Ontology ('HPO') gene-phenotype associations, the 'HGNC' gene nomenclature files, Health Canada's Drug Product Database ('DPD') data extract, and the 'openFDA' 'Drugs@FDA' and 'NDC' directory downloads. The parsed tables are returned in an 'R' object called 'dvobject', which records the source release, licence and file hashes.

License MIT + file LICENSE

Encoding UTF-8

Imports data.table, dplyr, jsonlite, progress, purrr, tibble, tools, utils, XML

Suggests canvasXpress, knitr, rmarkdown, testthat, tidyr

VignetteBuilder knitr

URL https://docs.ropensci.org/dbparser/, https://github.com/ropensci/dbparser, https://interstellar-egypt.com

BugReports https://github.com/ropensci/dbparser/issues

Depends R (>= 3.5)

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Mohammed Ali [aut, cre], Ali Ezzat [aut], Hao Zhu [rev], Emma Mendelsohn [rev], Interstellar [cph, fnd] (Interstellar for Consultation Services, URL: https://interstellar-egypt.com)

Maintainer Mohammed Ali <mohammed.ali@interstellar-egypt.com>

Repository CRAN

Date/Publication 2026-11-08 21:30:08 UTC

Contents

add_database_info . . . 2
cett_nodes_options . . . 3
drug_node_options . . . 3
get_genes_for_phenotype . . . 4
get_source_metadata . . . 5
merge_drugbank_hpo . . . 5
merge_drugbank_onsides . . . 7
merge_drugbank_twosides . . . 8
parseDrugBank . . . 9
parseHealthCanadaDPD . . . 10
parseHGNC . . . 11
parseHPO . . . 13
parseOnSIDES . . . 14
parseOpenFDA . . . 15
parseTWOSIDES . . . 16
references_node_options . . . 18
search_phenotypes . . . 18
show_dvobject_metadata . . . 19
subset_drugbank_dvobject . . . 20
subset_onsides_dvobject . . . 21

Index **22**

`add_database_info` *add_database_info Assign passed databases db metadata to passed dvobject*

Description

add_database_info Assign passed databases db metadata to passed dvobject

Usage

```
add_database_info(
  dvobject,
  db_type = "DrugBank",
  db_version = NULL,
  db_exported_date = NULL
)
```

Arguments

`dvobject`	dvobject to assign metadata to it
`db_type`	database type (defualt="DrugBank")
`db_version`	database version as string
`db_exported_date`	database official export date

Value

dvobject

See Also

Other utility: get_genes_for_phenotype(), get_source_metadata(), search_phenotypes(), show_dvobject_metadata(), subset_drugbank_dvobject(), subset_onsides_dvobject()

---

cett_nodes_options *returns carriers, enzymes,targets and transporters node valid options.*

Description

returns carriers, enzymes,targets and transporters node valid options.

Usage

```
cett_nodes_options()
```

Value

list of CETT valid options

See Also

Other parsers: drug_node_options(), parseDrugBank(), parseHGNC(), parseHPO(), parseHealthCanadaDPD(), parseOnSIDES(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

---

drug_node_options *returns drug node valid options.*

Description

returns drug node valid options.

Usage

```
drug_node_options()
```

Value

list of drug valid options

See Also

Other parsers: cett_nodes_options(), parseDrugBank(), parseHGNC(), parseHPO(), parseHealthCanadaDPD(), parseOnSIDES(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

get_genes_for_phenotype *Get Genes Associated with a Phenotype*

Description

Returns the genes associated with a phenotype, given either its HPO identifier (e.g. `"HP:0001250"`) or its name.

Usage

```
get_genes_for_phenotype(hpo_db, phenotype, exact = FALSE)
```

Arguments

hpo_db	An HPO dvobject returned by parseHPO.
phenotype	Character. An HPO identifier or a phenotype name.
exact	Logical. Used when phenotype is a name. If TRUE, the name must match exactly (ignoring case). If FALSE (default), any phenotype whose name contains phenotype matches.

Value

A tibble with columns gene_symbol, entrez_id, hpo_id, phenotype_name and association_weight, ordered by gene symbol.

See Also

Other utility: add_database_info(), get_source_metadata(), search_phenotypes(), show_dvobject_metadata(), subset_drugbank_dvobject(), subset_onsides_dvobject()

Examples

```
hpo_db <- parseHPO(system.file("extdata", "hpo_gene_attribute_edges.txt",
                               package = "dbparser"))
get_genes_for_phenotype(hpo_db, "HP:0001250")
get_genes_for_phenotype(hpo_db, "thrombosis")
```

get_source_metadata *Get the source metadata of a dvobject*

Description

Returns where each source in a dvobject came from: release, dates, URLs, licence, attribution, the parsed files with their SHA-256 hashes, and the dbparser version that parsed them. A merged dvobject has one row per source. Unknown values are NA.

Usage

```
get_source_metadata(dvobject)
```

Arguments

dvobject	a dvobject returned by a parser or a merge function

Value

a tibble with one row per source and columns metadata_version, source, source_name, release, release_date, retrieved_at, source_url, licence, licence_url, attribution, files (a list column of data.frames with name and sha256) and parser. It has no rows when the dvobject carries no source metadata (e.g. one saved by an older dbparser version).

See Also

Other utility: add_database_info(), get_genes_for_phenotype(), search_phenotypes(), show_dvobject_metadata(
subset_drugbank_dvobject(), subset_onsides_dvobject()

Examples

```
drugbank <- parseDrugBank(
  system.file("extdata", "drugbank_record_biotech.xml",
              package = "dbparser"))
get_source_metadata(drugbank)
```

merge_drugbank_hpo *Merge DrugBank and HPO Gene-Phenotype Database Objects*

Description

Creates an integrated dvobject by linking DrugBank target gene data with HPO gene-phenotype associations using gene symbols as the bridge.

Usage

```
merge_drugbank_hpo(db_object, hpo_db)
```

Arguments

db_object	A dvobject from parseDrugBank() OR an existing merged dvobject (containing $drugbank).
hpo_db	A dvobject produced by dbparser::parseHPO().

Details

This function performs the following key steps: 1. Extracts gene names from DrugBank polypeptide data ($cett$targets$polypeptides$general_information). 2. Links gene names to DrugBank IDs via the target_id bridge ($cett$targets$general_information). 3. Optionally resolves DrugBank IDs to drug names. 4. Enriches the HPO gene-phenotype association table by adding drugbank_id and drug_name columns. 5. Assembles a merged object containing all original data plus the enriched table and the gene mapping bridge.

Supports piping and chaining with other merge functions.

Value

A new dvobject containing the integrated data.

See Also

Other mergers: merge_drugbank_onsides(), merge_drugbank_twosides()

Examples

```
## Not run:
drugbank <- parseDrugBank("path/to/drugbank.xml")
hpo      <- parseHPO("path/to/gene_attribute_edges.tsv")

# Standalone merge
merged_db <- merge_drugbank_hpo(drugbank, hpo)

# Chained with other merges (any order)
full_db <- drugbank %>%
  merge_drugbank_hpo(hpo) %>%
  merge_drugbank_onsides(onsides) %>%
  merge_drugbank_twosides(twosides)

# --- Example: Find drugs targeting genes associated with seizures ---
seizure_drugs <- full_db$integrated_data$gene_phenotype_enriched %>%
  filter(grepl("seizure", phenotype_name, ignore.case = TRUE)) %>%
  filter(!is.na(drugbank_id)) %>%
  select(drug_name, drugbank_id, gene_symbol, phenotype_name) %>%
  distinct()

## End(Not run)
```

merge_drugbank_onsides *Merge DrugBank and OnSIDES Database Objects*

Description

Creates an integrated dvobject object by linking DrugBank dvobject with OnSIDES dvobject using RxNorm CUIs as the bridge.

Usage

```
merge_drugbank_onsides(db_object, onsides_db)
```

Arguments

db_object	A dvobject from 'parseDrugBank()' OR an existing merged dvobject (containing '$drugbank').
onsides_db	A dvobject produced by 'dbparser::parseOnSIDES()'.

Details

This function performs the following key steps: 1. Creates a mapping table between DrugBank IDs and RxNorm CUIs from the DrugBank object. 2. Enriches the relevant OnSIDES tables ('vocab_rxnorm_ingredient' and optionally 'high_confidence') by adding a 'drugbank_id' column. 3. Assembles a new list object containing all original tables plus the enriched ones and the ID mapping table itself.

The resulting object allows for powerful queries that span both mechanistic data from DrugBank and clinical side-effect data from OnSIDES. Supports piping and chaining with other merge functions.

Value

A new dvobject containing the integrated data.

See Also

Other mergers: merge_drugbank_hpo(), merge_drugbank_twosides()

Examples

```
## Not run:
# First, parse the individual databases
drugbank <- parseDrugBank("path/to/drugbank.xml")
onsides <- parseOnSIDES("path/to/onsides_csvs/")

# Now, merge them into a single, powerful object
merged_db <- merge_drugbank_onsides(drugbank, onsides)

# --- Example Analysis: Find the protein targets of all drugs known to ---
```

```
# --- cause the side effect "Hepatitis" with high confidence.        ---

# 1. Find the MedDRA ID for "Hepatitis"
hepatitis_id <- merged_db$onsides$vocab_meddra_adverse_effect %>%
  filter(meddra_name == "Hepatitis") %>%
  pull(meddra_id)

# 2. Find all drug ingredients linked to this effect in the high_confidence table
drug_ids_causing_hepatitis <- merged_db$onsides$high_confidence_enriched %>%
  filter(effect_meddra_id == hepatitis_id) %>%
  pull(drugbank_id) %>%
  na.omit() %>%
  unique()

# 3. Look up the targets for these DrugBank IDs
targets_of_interest <- merged_db$targets %>%
  filter(parent_key %in% drug_ids_causing_hepatitis) %>%
  select(drug_id = parent_key, target_name = name, gene_name)

head(targets_of_interest)

## End(Not run)
```

`merge_drugbank_twosides` *Merge a DrugBank dvobject with a TWOSIDES dvobject*

Description

Integrates drug-drug interaction data from TWOSIDES with the rich mechanistic information from DrugBank. This function is chainable and can accept a raw DrugBank object or an already-merged dvobject.

Usage

```
merge_drugbank_twosides(db_object, twosides_db)
```

Arguments

`db_object`	A dvobject from ‘parseDrugBank()‘ or an existing merged dvobject.
`twosides_db`	A dvobject from ‘parseTWOSIDES()‘.

Value

A new, nested dvobject with the TWOSIDES data added.

See Also

Other mergers: `merge_drugbank_hpo()`, `merge_drugbank_onsides()`

parseDrugBank *parseDrugBank*

Description

parses given DrugBank XML database into a dvobject. dvobject is a list of data.frames in which each data.frame represents a part of parsed data (i.e drugs, prices, carriers, ...)

Usage

```
parseDrugBank(
  db_path,
  drug_options = NULL,
  parse_salts = FALSE,
  parse_products = FALSE,
  references_options = NULL,
  cett_options = NULL
)
```

Arguments

db_path	**string**, full path for the **DrugBank** xml or zip file.
drug_options	**character vector**, list of sub drug related nodes names options to parse (default = NULL). Check drug_node_options() for all available options. If its value is 'NULL' ONLY 'drug_general_information' will be placed in the returned dvobject.
parse_salts	**boolean**, parse salts info (default = FALSE)
parse_products	**boolean**, parse products info (default = FALSE)
references_options	**character vector**, list of sub references related nodes names options to parse (default = NULL). Check references_node_options() for all available options.
cett_options	**character vector**, list of sub cett related nodes names options to parse (default = NULL). Check cett_nodes_options() for all available options.

Value

dvobject

See Also

Other parsers: cett_nodes_options(), drug_node_options(), parseHGNC(), parseHPO(), parseHealthCanadaDPD(), parseOnSIDES(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

parseHealthCanadaDPD *Parse the Health Canada Drug Product Database (DPD) Data Extract*

Description

Reads Health Canada's Drug Product Database data extract and returns a `dvobject` with its 13 tables. Each of the four product sets (marketed, approved, cancelled, dormant) is read from its unzipped folder or straight from its zip; the rows of every set go in one table with an `extract` column.

Usage

```
parseHealthCanadaDPD(
  dpd_dir,
  extracts = c("marketed", "approved", "cancelled", "dormant"),
  release = NULL,
  release_date = NULL,
  retrieved_at = NULL
)
```

Arguments

`dpd_dir`	Character. Folder holding the extract's zips or unzipped folders.
`extracts`	Character. Product sets to read, any of `"marketed"`, `"approved"`, `"cancelled"` and `"dormant"` (all by default).
`release`	Character or NULL. Release label, e.g. `"2026-10"`. DPD publishes no release number.
`release_date`	Date, `"YYYY-MM-DD"` or NULL. Release date.
`retrieved_at`	Date, `"YYYY-MM-DD"` or NULL. Date the files were downloaded.

Details

`dpd_dir` holds the extract as Health Canada distributes it, each product set either as its zip or unzipped into a folder of the same name: `allfiles` (marketed), `allfiles_ap` (approved), `allfiles_ia` (cancelled) and `allfiles_dr` (dormant), plus `inactive` (the inactive products list). Zips are extracted to a temporary folder that is removed after reading. When neither `inactive` nor `inactive.zip` is there, `inactive.txt` is read from the cancelled set, which holds the same file.

The files have no header: column names are those of Health Canada's read-me ("Read Me File - Drug Product Database (DPD) Data Extract"), lower case, and table names are the read-me's without `QRYM_`. Every value is quoted and the files are UTF-8; empty values become `NA`. Code columns (`drug_code`, `active_ingredient_code`, `company_code`, `pharm_form_code`, `route_of_administration_code`, `si_code`) are integers and the `DD-MON-YYYY` dates are dates; every other column stays text as written. Dates are kept as written, including two lot expiry dates in the cancelled set written with years 0030 and 0031.

The therapeutic class files hold four fields where the read-me lists seven columns: the three certain ones are kept and the parser stops if the fourth ever holds a value.

The parser groups nothing: `status` holds every status row as written, with `current_status_flag` marking each product's current one.

DPD data are published under the Open Government Licence - Canada (version 2.0); the attribution it asks for is recorded in the object's metadata (see `get_source_metadata`). The DPD extract page also links to Health Canada's website terms; the dataset record on open.canada.ca states the Open Government Licence – Canada.

Value

A `dvobject` with 13 tibbles, each keyed by `drug_code`: `drug_product`, `active_ingredients`, `companies`, `form`, `packaging`, `pharmaceutical_std`, `route`, `schedule`, `status`, `therapeutic_class`, `veterinary_species`, `biosimilars` (each with `extract`) and `inactive_products`. `inactive_products` lists cancelled products, so it is built only when `extracts` includes `"cancelled"` and an inactive products file is found.

See Also

Other parsers: `cett_nodes_options()`, `drug_node_options()`, `parseDrugBank()`, `parseHGNC()`, `parseHPO()`, `parseOnSIDES()`, `parseOpenFDA()`, `parseTWOSIDES()`, `references_node_options()`

Examples

```
dpd_dir <- system.file("extdata", "health_canada_dpd", package = "dbparser")
dpd <- parseHealthCanadaDPD(dpd_dir, release = "2026-10",
                            retrieved_at = "2026-10-21")
dpd$drug_product[, c("drug_code", "brand_name", "extract")]
get_source_metadata(dpd)
```

---

parseHGNC *Parse the HGNC Gene Nomenclature Files*

Description

Reads the HUGO Gene Nomenclature Committee (HGNC) complete set (`hgnc_complete_set.txt`) and, optionally, the withdrawn entries (`withdrawn.txt`) and returns a `dvobject` with gene, gene name and withdrawn tables.

Usage

```
parseHGNC(
  complete_set_path,
  withdrawn_path = NULL,
  release = NULL,
  release_date = NULL,
  retrieved_at = NULL
)
```

Arguments

complete_set_path	Character. Path to hgnc_complete_set.txt.
withdrawn_path	Character or NULL. Path to withdrawn.txt. When NULL the withdrawn table is not built.
release	Character or NULL. Release label, e.g. "2026-10".
release_date	Date, "YYYY-MM-DD" or NULL. Release date.
retrieved_at	Date, "YYYY-MM-DD" or NULL. Date the files were downloaded.

Details

Both files are tab separated and UTF-8. In the complete set a cell holding several values is pipe separated (a|b) and wrapped in double quotes; the parser strips the wrapper and keeps the pipes. Column names are lower-cased with . and - replaced by _ (pseudogene.org becomes pseudogene_org), and the date_* columns become dates. Every other column stays text as written.

HGNC publishes no release number or date for the current files, so release and release_date are NA unless given (e.g. the month of the download, or the date of an HGNC archive file).

HGNC data are released under CC0; HGNC recommends citing its latest paper, which is recorded in the object's metadata (see get_source_metadata).

Value

A dvobject with tibbles:

gene One row per approved gene, keyed by hgnc_id, with the file's columns. Multi-valued cells keep the file's a|b text.

gene_name One row per (hgnc_id, kind, name): each gene's approved symbol (kind "symbol") and name ("name"), and every alias and previous symbol and name ("alias_symbol", "prev_symbol", "alias_name", "prev_name"), split on | and trimmed. Empty values and exact duplicates are dropped; names that differ only in case are kept.

withdrawn Only when withdrawn_path is given. One row per withdrawn ID and the ID it was merged into or split to (merged_into_hgnc_id, merged_into_symbol, merged_into_status); an entry without a target has one row with these NA. merged_into_report keeps the original cell. Rows are as written in HGNC's file, including any repeated target.

See Also

Other parsers: cett_nodes_options(), drug_node_options(), parseDrugBank(), parseHPO(), parseHealthCanadaDPD(), parseOnSIDES(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

Examples

```
hgnc_dir <- system.file("extdata", "hgnc", package = "dbparser")
hgnc <- parseHGNC(file.path(hgnc_dir, "hgnc_complete_set.txt"),
                  withdrawn_path = file.path(hgnc_dir, "withdrawn.txt"),
                  release        = "2026-10",
                  retrieved_at   = "2026-10-16")
```

```
hgnc$gene_name[hgnc$gene_name$name == "PD1", ]
get_source_metadata(hgnc)
```

---

parseHPO *Parse HPO Gene-Phenotype Associations*

Description

Reads the Human Phenotype Ontology (HPO) gene-phenotype associations as distributed by Harmonizome (`gene_attribute_edges.txt`) and returns a `dvobject` with gene, phenotype and association tables.

Usage

```
parseHPO(
  filepath,
  minimum_weight = 0,
  db_version = NULL,
  db_exported_date = NULL
)
```

Arguments

`filepath`	Character. Path to the Harmonizome HPO `gene_attribute_edges.txt` file, plain or gzip-compressed.
`minimum_weight`	Numeric. Minimum association weight to keep. The default `0` keeps all associations.
`db_version`	Character or NULL. Dataset version, stored in the object metadata.
`db_exported_date`	Character or NULL. Dataset release date, stored in the object metadata.

Details

The Harmonizome file is tab separated and has two header lines: the column names (`source`, `source_desc`, `source_id`, `target`, `target_desc`, `target_id`, `weight`) followed by a descriptor row (`GeneSym`, `NA`, `GeneID`, `Phenotype`, `HPOID`, `NA`, `weight`). The parser drops the descriptor row and renames the columns to domain-specific names. Columns that carry no information in this dataset (`source_desc` and the `-666` placeholder in `target_id`) are not kept.

Gzip-compressed files (`.gz`) are read directly.

The resulting object can be linked to DrugBank target genes with `merge_drugbank_hpo`.

Value

A dvobject with three tibbles:

associations One row per gene-phenotype pair: gene_symbol, entrez_id, hpo_id, phenotype_name and association_weight.

genes Unique genes: gene_symbol, entrez_id and n_phenotypes, the number of associated phenotypes.

phenotypes Unique phenotypes: hpo_id, phenotype_name and n_genes, the number of associated genes.

See Also

Other parsers: cett_nodes_options(), drug_node_options(), parseDrugBank(), parseHGNC(), parseHealthCanadaDPD(), parseOnSIDES(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

Examples

```
hpo_path <- system.file("extdata", "hpo_gene_attribute_edges.txt",
                        package = "dbparser")
hpo_db <- parseHPO(hpo_path)
hpo_db$phenotypes
show_dvobject_metadata(hpo_db)
```

---

parseOnSIDES *Parse the OnSIDES Core Relational Database*

Description

Parses the core relational tables from the OnSIDES database.

Usage

```
parseOnSIDES(
  dataDir,
  include_high_confidence = TRUE,
  db_version = NULL,
  db_exported_date = NULL
)
```

Arguments

dataDir	A string specifying the path to the directory containing the OnSIDES CSV files.
include_high_confidence	Logical. If TRUE (the default), the function will also parse the 'high_confidence.csv' file, which is a pre-aggregated summary of ingredient-to-effect relationships. If the file is not found, a warning is issued.

db_version	used onside version (default = NULL)
db_exported_date	used onside release date (default = NULL)

Value

dvobject

See Also

Other parsers: cett_nodes_options(), drug_node_options(), parseDrugBank(), parseHGNC(), parseHPO(), parseHealthCanadaDPD(), parseOpenFDA(), parseTWOSIDES(), references_node_options()

---

parseOpenFDA *Parse the openFDA Drugs@FDA and NDC Directory Downloads*

Description

Reads openFDA's bulk download files for Drugs@FDA (drug/drugsfda) and the NDC directory (drug/ndc) and returns a dvobject with one table per record level, field names as in the files.

Usage

```
parseOpenFDA(
  drugsfda_path = NULL,
  ndc_path = NULL,
  release = NULL,
  release_date = NULL,
  retrieved_at = NULL
)
```

Arguments

drugsfda_path	Character or NULL. Path to the Drugs@FDA download (zip or JSON).
ndc_path	Character or NULL. Path to the NDC directory download (zip or JSON).
release	Character or NULL. Release label, e.g. "2026-10".
release_date	Date, "YYYY-MM-DD" or NULL. Release date; by default the files' export date.
retrieved_at	Date, "YYYY-MM-DD" or NULL. Date the files were downloaded.

Details

Each download is one JSON object, {meta, results}, usually zipped (drug-drugsfda-0001-of-0001.json.zip, drug-ndc-0001-of-0001.json.zip). Zips are read directly, without extracting them to disk; an unzipped .json file works too. The parser checks the number of records against meta.results.total.

Drugs@FDA lists FDA application products (NDA, ANDA, BLA); the NDC directory lists products registered for sale, including products sold under an OTC monograph without an application. Every

application, product, submission and listing is kept as the files have it, including tentative approvals (`submission_status` "TA"); the parser does no status grouping and no matching across sources.

`YYYYMMDD` dates become dates. NDC packaging dates are written two ways (`20220525` and `04-DEC-09`) and are kept as text, as written. `finished` and `sample` are logical. Every other field stays text. The `openfda` block of each record is kept as (key, field, value) rows; it is openFDA's annotation, and it can belong to another drug (for example ANDA202111).

`release_date` defaults to the files' `meta.last_updated` (the export date) when every file given has the same one.

Parsing the full Drugs@FDA and NDC files takes about a minute and needs roughly 300 MB of memory.

openFDA data are released under CC0 1.0. openFDA asks for credit, which is recorded in the object's metadata (see `get_source_metadata`), and states: "Do not rely on openFDA to make decisions regarding medical care."

Value

A `dvobject` with tibbles. From Drugs@FDA: `drugsfda_application` (key `application_number`), `drugsfda_product` (key application and `product_number`), `drugsfda_active_ingredient`, `drugsfda_submission` (key application, `submission_type` and `submission_number`), `drugsfda_submission_property_type`, `drugsfda_application_doc` and `drugsfda_openfda`. From the NDC directory: `ndc_product` (key `product_id`), `ndc_active_ingredient`, `ndc_route`, `ndc_pharm_class`, `ndc_packaging` and `ndc_openfda`. Only the tables of the files given are built.

See Also

Other parsers: `cett_nodes_options()`, `drug_node_options()`, `parseDrugBank()`, `parseHGNC()`, `parseHPO()`, `parseHealthCanadaDPD()`, `parseOnSIDES()`, `parseTWOSIDES()`, `references_node_options()`

Examples

```
openfda_dir <- system.file("extdata", "openfda", package = "dbparser")
openfda <- parseOpenFDA(
  drugsfda_path = file.path(openfda_dir,
                            "drug-drugsfda-0001-of-0001.json.zip"),
  ndc_path      = file.path(openfda_dir, "drug-ndc-0001-of-0001.json.zip"),
  release       = "2026-10")
openfda$drugsfda_product[, c("application_number", "brand_name",
                             "marketing_status")]
get_source_metadata(openfda)
```

`parseTWOSIDES` *Parse the TWOSIDES Drug-Drug Interaction Database*

Description

Reads the TWOSIDES data file, which contains adverse event data for pairs of drugs taken concurrently (N=2 interactions).

Usage

```
parseTWOSIDES(twosides_file_path, db_version = NULL, db_exported_date = NULL)
```

Arguments

twosides_file_path	Path to the TWOSIDES data file (e.g., 'TWOSIDES.csv.gz').
db_version	used twoside version (default = NULL)
db_exported_date	used twoside release date (default = NULL)

Details

TWOSIDES is a database of drug-drug interaction safety signals mined from the FDA's Adverse Event Reporting System using the same approach as is used to generate OffSIDES.

Database fields as follow:

drug_1_rxnorn_id RxNORM identifier for drug 1

drug_1_concept_name RxNORM name string for drug 1

drug_2_rxnorm_id RxNORM identifier for drug 2

drug_2_concept_name RxNORM name string for drug 3

condition_meddra_id MedDRA identifier for the side effect

condition_concpet_name MedDRA name string for the side effect

A The number of reports for the pair of drugs that report the side effect

B The number of reports for the pair of drugs that do not report the side effect

C The number of reports for other PSM matched drugs (including perhaps the single versions of drug 1 or drug 2) that report the side effect

D The number of reports for other PSM matched drugs and other side effects

PRR Proportional reporting ratio, PRR=(A/(A+B))/(C/(C+D))

PRR_error Error estimate of the PRR

mean_reporting_frequency Proportion of reports for the drug that report the side effect, A/(A+B)

Value

A dvobject of class 'TWOSIDESDB' containing the 'drug_drug_interactions' data frame and associated metadata.

See Also

Other parsers: `cett_nodes_options()`, `drug_node_options()`, `parseDrugBank()`, `parseHGNC()`, `parseHPO()`, `parseHealthCanadaDPD()`, `parseOnSIDES()`, `parseOpenFDA()`, `references_node_options()`

`references_node_options` *returns references node valid options.*

Description

returns references node valid options.

Usage

```
references_node_options()
```

Value

list of references valid options

See Also

Other parsers: `cett_nodes_options()`, `drug_node_options()`, `parseDrugBank()`, `parseHGNC()`, `parseHPO()`, `parseHealthCanadaDPD()`, `parseOnSIDES()`, `parseOpenFDA()`, `parseTWOSIDES()`

`search_phenotypes` *Search HPO Phenotypes by Keyword*

Description

Finds phenotypes whose name contains a keyword and lists their associated genes. Matching is case-insensitive and treats `keyword` as a plain string, not a regular expression.

Usage

```
search_phenotypes(hpo_db, keyword, max_results = 50L)
```

Arguments

`hpo_db`	An HPO `dvobject` returned by `parseHPO`.
`keyword`	Character. Search term, e.g. `"seizure"`.
`max_results`	Integer. Maximum number of phenotypes to return.

Value

A tibble with columns `hpo_id`, `phenotype_name`, `n_genes` and `genes` (comma-separated gene symbols), ordered by decreasing `n_genes`.

See Also

Other utility: add_database_info(), get_genes_for_phenotype(), get_source_metadata(), show_dvobject_metadata(), subset_drugbank_dvobject(), subset_onsides_dvobject()

Examples

```
hpo_db <- parseHPO(system.file("extdata", "hpo_gene_attribute_edges.txt",
                               package = "dbparser"))
search_phenotypes(hpo_db, "seizure")
```

---

show_dvobject_metadata

Display dvobject Metadata

Description

Displays information about passed dbobject object including basic info, database metadata, and all data.frames contained within nested lists.

Usage

```
show_dvobject_metadata(obj, return_df = FALSE)
```

Arguments

obj	A dvobject
return_df	Logical. If TRUE, returns metadata data.frame without printing. Default is FALSE.

Value

Invisibly returns a data.frame containing dvobject metadata

See Also

Other utility: add_database_info(), get_genes_for_phenotype(), get_source_metadata(), search_phenotypes(), subset_drugbank_dvobject(), subset_onsides_dvobject()

Examples

```
drugbank <- parseDrugBank(
  system.file("extdata", "drugbank_record_biotech.xml",
              package = "dbparser"))
show_dvobject_metadata(drugbank)
metadata <- show_dvobject_metadata(drugbank, return_df = TRUE)
```

`subset_drugbank_dvobject` *Subset a DrugBank dvobject by a vector of DrugBank IDs*

Description

Subset a DrugBank dvobject by a vector of DrugBank IDs

Usage

```
subset_drugbank_dvobject(dvobject, drug_ids)
```

Arguments

`dvobject`	The dvobject from ‘parseDrugBank()‘.
`drug_ids`	A character vector of ‘drugbank_id‘ values to keep.

Details

Intelligently filters a DrugBank dvobject to retain only the data associated with a specified list of drugbank_ids. It correctly handles the deep, multi-level nested structure of the entire object, including the complex relationships within the ‘cett‘ list.

Value

A new, smaller dvobject with the same structure and attributes.

See Also

Other utility: `add_database_info()`, `get_genes_for_phenotype()`, `get_source_metadata()`, `search_phenotypes()`, `show_dvobject_metadata()`, `subset_onsides_dvobject()`

Examples

```
## Not run:
library(dbparser)
one_drug <- subset_drugbank_dvobject(dvobject = dbdataset::drugbank,
                                     drug_ids = "DB00001")

## End(Not run)
```

`subset_onsides_dvobject` *Subset an OnSIDES dvobject by a vector of RxNorm Ingredient IDs (Schema-Aware)*

Description

Intelligently filters an OnSIDES dvobject by cascading filters through the relational tables, ensuring the final subset is self-consistent.

Usage

```
subset_onsides_dvobject(dvobject, ingredient_ids)
```

Arguments

`dvobject`	A dvobject from 'parseOnSIDES()'.
`ingredient_ids`	A character vector of RxNorm CUIs (ingredients) to keep.

Value

A new, smaller dvobject with the same structure.

See Also

Other utility: `add_database_info()`, `get_genes_for_phenotype()`, `get_source_metadata()`, `search_phenotypes()`, `show_dvobject_metadata()`, `subset_drugbank_dvobject()`

Index

∗ **mergers**
  merge_drugbank_hpo, 5
  merge_drugbank_onsides, 7
  merge_drugbank_twosides, 8
∗ **parsers**
  cett_nodes_options, 3
  drug_node_options, 3
  parseDrugBank, 9
  parseHealthCanadaDPD, 10
  parseHGNC, 11
  parseHPO, 13
  parseOnSIDES, 14
  parseOpenFDA, 15
  parseTWOSIDES, 16
  references_node_options, 18
∗ **utility**
  add_database_info, 2
  get_genes_for_phenotype, 4
  get_source_metadata, 5
  search_phenotypes, 18
  show_dvobject_metadata, 19
  subset_drugbank_dvobject, 20
  subset_onsides_dvobject, 21

add_database_info, 2
add_database_info(), *4, 5*, *19–21*

cett_nodes_options, 3
cett_nodes_options(), *3*, *9*, *11, 12*, *14–18*

drug_node_options, 3
drug_node_options(), *3*, *9*, *11, 12*, *14–18*

get_genes_for_phenotype, 4
get_genes_for_phenotype(), *3*, *5*, *19–21*
get_source_metadata, 5, *11, 12*, *16*
get_source_metadata(), *3, 4*, *19–21*

merge_drugbank_hpo, 5, *13*
merge_drugbank_hpo(), *7, 8*
merge_drugbank_onsides, 7
merge_drugbank_onsides(), *6*, *8*
merge_drugbank_twosides, 8
merge_drugbank_twosides(), *6, 7*

parseDrugBank, 9
parseDrugBank(), *3*, *11, 12*, *14–18*
parseHealthCanadaDPD, 10
parseHealthCanadaDPD(), *3*, *9*, *12*, *14–18*
parseHGNC, 11
parseHGNC(), *3*, *9*, *11*, *14–18*
parseHPO, *4*, 13, *18*
parseHPO(), *3*, *9*, *11, 12*, *15–18*
parseOnSIDES, 14
parseOnSIDES(), *3*, *9*, *11, 12*, *14*, *16–18*
parseOpenFDA, 15
parseOpenFDA(), *3*, *9*, *11, 12*, *14, 15*, *17, 18*
parseTWOSIDES, 16
parseTWOSIDES(), *3*, *9*, *11, 12*, *14–16*, *18*

references_node_options, 18
references_node_options(), *3*, *9*, *11, 12*, *14–17*

search_phenotypes, 18
search_phenotypes(), *3–5*, *19–21*
show_dvobject_metadata, 19
show_dvobject_metadata(), *3–5*, *19–21*
subset_drugbank_dvobject, 20
subset_drugbank_dvobject(), *3–5*, *19*, *21*
subset_onsides_dvobject, 21
subset_onsides_dvobject(), *3–5*, *19, 20*