Package ‘taxadb’

October 14, 2022

Version 0.1.6

Title A High-Performance Local Taxonomic Database Interface

Description Creates a local database of many commonly used taxonomic authorities and provides functions that can quickly query this data.

License MIT + file LICENSE

Encoding UTF-8

ByteCompile true

Depends R (>= 4.0)

Imports dplyr, DBI, arkdb (>= 0.0.16), tibble, readr, rappdirs, rlang, magrittr, stringi, progress, utils, dbplyr, curl, jsonlite, duckdb, contentid, R.utils

RoxygenNote 7.2.1

Suggests spelling, testthat, covr, knitr, rmarkdown, purrr, crayon, RSQLite

Language en-US

VignetteBuilder knitr


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

NeedsCompilation no

Author Carl Boettiger [aut, cre] (<https://orcid.org/0000-0002-1642-628X>), Kari Norman [aut] (<https://orcid.org/0000-0002-2029-2325>), Jorrit Poelen [aut] (<https://orcid.org/0000-0003-3138-4118>), Scott Chamberlain [aut] (<https://orcid.org/0000-0003-1444-9135>), Noam Ross [ctb] (<https://orcid.org/0000-0002-2136-0000>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2022-09-14 09:00:02 UTC
**R topics documented:**

- clean_names .................................................. 2
- common_contains .............................................. 3
- common_starts_with ......................................... 4
- filter_by ....................................................... 5
- filter_common ................................................. 6
- filter_id ......................................................... 7
- filter_name ..................................................... 9
- filter_rank .................................................... 10
- fuzzy_filter .................................................. 11
- get_ids .......................................................... 12
- get_names ....................................................... 14
- mutate_db ....................................................... 15
- name_contains ............................................... 16
- name_starts_with ............................................. 16
- taxadb_dir .................................................... 17
- taxa_tbl ........................................................ 18
- td_connect ..................................................... 19
- td_create ...................................................... 20
- td_disconnect ................................................ 21
- tl_import ....................................................... 22

**Index**

---

**clean_names**

*Clean taxonomic names*

**Description**

A utility to sanitize taxonomic names to increase probability of resolving names.

**Usage**

```r
clean_names(
  names,
  fix_delim = TRUE,
  binomial_only = TRUE,
  remove_sp = TRUE,
  ascii_only = TRUE,
  lowercase = TRUE,
  remove_punc = FALSE
)
```
common_contains

Arguments

- **names**: a character vector of taxonomic names (usually species names)
- **fix_delim**: Should we replace separators `., -` with spaces? e.g. 'Homo.sapiens' becomes 'Homo sapiens'. logical, default TRUE.
- **binomial_only**: Attempt to prune name to a binomial name, e.g. Genus and species (specific epithet), e.g. Homo sapiens sapiens becomes Homo sapiens. logical, default TRUE.
- **remove_sp**: Should we drop unspecified species epithet designations? e.g. Homo sp. becomes Homo (thus only matching against genus level ids). logical, default TRUE.
- **ascii_only**: Should we coerce strings to ascii characters? (see stringi::stri_trans_general())
- **lowercase**: Should names be coerced to lower-case to provide case-insensitive matching?
- **remove_punc**: replace all punctuation but apostrophes with a space, remove apostrophes

Details

Current implementation is limited to handling a few common cases. Additional extensions may be added later. A goal of the clean_names function is that any modification rule of the name strings be precise, atomic, and toggle-able, rather than relying on clever but more opaque rules and arbitrary scores. This utility should always be used with care, as indiscriminate modification of names may result in successful but inaccurate name matching. A good pattern is to only apply this function to the subset of names that cannot be directly matched.

Examples

```r
clean_names(c("Homo sapiens sapiens", "Homo.sapiens", "Homo sp."))
```

common_contains  common name starts with

Description

common name starts with

Usage

```r
common_contains(
  name, 
  provider = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  db = td_connect(),
  ignore_case = TRUE
)
```
common_starts_with

Arguments

name    vector of names (scientific or common, see by) to be matched against.
provider from which provider should the hierarchy be returned? Default is 'itis', which
can also be configured using options(default_taxadb_provider='...'). See
[td_create] for a list of recognized providers.
version Which version of the taxadb provider database should we use? defaults to latest.
See tl_import for details.
db      a connection to the taxadb database. See details.
ignore_case should we ignore case (capitalization) in matching names? default is TRUE.

Examples

common_contains("monkey")

common_starts_with common name starts with

Description

common name starts with

Usage

common_starts_with(
  name,
  provider =getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  db = td_connect(),
  ignore_case = TRUE
)

Arguments

name    vector of names (scientific or common, see by) to be matched against.
provider from which provider should the hierarchy be returned? Default is 'itis', which
can also be configured using options(default_taxadb_provider='...'). See
[td_create] for a list of recognized providers.
version Which version of the taxadb provider database should we use? defaults to latest.
See tl_import for details.
db      a connection to the taxadb database. See details.
ignore_case should we ignore case (capitalization) in matching names? default is TRUE.
**filter_by**

**Examples**

```r
common_starts_with("monkey")
```

---

**filter_by**  
Creates a data frame with column name given by `by`, and values given by the vector `x`, and then uses this table to do a filtering join, joining on the `by` column to return all rows matching the `x` values (scientific Names, taxonIDs, etc).

---

**Description**

Creates a data frame with column name given by `by`, and values given by the vector `x`, and then uses this table to do a filtering join, joining on the `by` column to return all rows matching the `x` values (scientific Names, taxonIDs, etc).

**Usage**

```r
filter_by(
  x,
  by,
  provider = getOption("taxadb_default_provider", "itis"),
  schema = c("dwc", "common"),
  version = latest_version(),
  collect = TRUE,
  db = td_connect(),
  ignore_case = TRUE
)
```

**Arguments**

- **x**  
a vector of values to filter on

- **by**  
a column name in the taxa_tbl (following Darwin Core Schema terms). The filtering join is executed with this column as the joining variable.

- **provider**  
from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using `options(default_taxadb_provider="...")`. See `td_create` for a list of recognized providers.

- **schema**  
One of "dwc" (for Darwin Core data) or "common" (for the Common names table.)

- **version**  
Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.

- **collect**  
logical, default TRUE. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.)
filter_common

db a connection to the taxadb database. See details.
ignore_case should we ignore case (capitalization) in matching names? default is TRUE.

Value

a data.frame in the Darwin Core tabular format containing the matching taxonomic entities.

See Also

Other filter_by: filter_common(), filter_id(), filter_name(), filter_rank()

Examples

sp <- c("Trochalopteron henrici gucenense", "Trochalopteron elliotii")
filter_by(sp, "scientificName")

filter_by(c("ITIS:1077358", "ITIS:175089"), "taxonID")

filter_by("Aves", "class")

filter_common

Look up taxonomic information by common name

Description

Look up taxonomic information by common name

Usage

filter_common(
  name,
  provider = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  collect = TRUE,
  ignore_case = TRUE,
  db = td_connect()
)
Arguments

name
provider
version
collect
ignore_case
db

Value

a data.frame in the Darwin Core tabular format containing the matching taxonomic entities.

See Also

Other filter_by: filter_by(), filter_id(), filter_name(), filter_rank()

Examples

filter_common("Pied Tamarin")

filter_id Return a taxonomic table matching the requested ids

Description

Return a taxonomic table matching the requested ids

Usage

filter_id(
  id,
  provider = getOption("taxadb_default_provider", "itis"),
  type = c("taxonID", "acceptedNameUsageID"),
  version = latest_version(),
)
filter_id

```r
collect = TRUE,
db = td_connect()
)
```

Arguments

- **id**: taxonomic id, in prefix format
- **provider**: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using `options(default_taxadb_provider=...)`. See [td_create] for a list of recognized providers.
- **type**: id type. Can be `taxonID` or `acceptedNameUsageID`, see details.
- **version**: Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.
- **collect**: logical, default TRUE. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.)
- **db**: a connection to the taxadb database. See details.

Details

Use `type="acceptedNameUsageID"` to return all rows for which this ID is the accepted ID, including both synonyms and accepted names (since both all synonyms of a name share the same acceptedNameUsageID.) Use `taxonID` (default) to only return those rows for which the Scientific name corresponds to the taxonID.

Some providers (e.g. ITIS) assign taxonIDs to synonyms, most others only assign IDs to accepted names. In the latter case, this means requesting taxonID will only match accepted names, while requesting matches to the acceptedNameUsageID will also return any known synonyms. See examples.

Value

a data.frame with id and name of all matching species

See Also

Other filter_by: `filter_by()`, `filter_common()`, `filter_name()`, `filter_rank()`

Examples

```r
filter_id(c("ITIS:1077358", "ITIS:175089"))
filter_id("ITIS:1077358", type="acceptedNameUsageID")
```
filter_name  

Look up taxonomic information by scientific name

Description

Look up taxonomic information by scientific name

Usage

filter_name(
  name, 
  provider = getOption("taxadb_default_provider", "itis"), 
  version = latest_version(), 
  collect = TRUE, 
  ignore_case = TRUE, 
  db = td_connect()
)

Arguments

name  a character vector of scientific names, e.g. "Homo sapiens"
provider  from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using options(default_taxadb_provider=...). See [td_create] for a list of recognized providers.
version  Which version of the taxadb provider database should we use? defaults to latest. See tl_import for details.
collect  logical, default TRUE. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.)
ignore_case  should we ignore case (capitalization) in matching names? default is TRUE.
db  a connection to the taxadb database. See details.

Details

Most but not all authorities can match against both species level and higher-level (or lower, e.g. subspecies or variety) taxonomic names. The rank level is indicated by taxonRank column.

Most authorities include both known synonyms and accepted names in the scientificName column, (with the status indicated by taxonomicStatus). This is convenient, as users will typically not know if the names they have are synonyms or accepted names, but will want to get the match to the accepted name and accepted ID in either case.

Value

a data.frame in the Darwin Core tabular format containing the matching taxonomic entities.
See Also

Other filter_by: filter_by(), filter_common(), filter_id(), filter_rank()

Examples

```r
sp <- c("Trochalopteron henrici gucenense",
       "Trochalopteron elliotii")
filter_name(sp)
```

---

**filter_rank**

Get all members (descendants) of a given rank level

**Description**

Get all members (descendants) of a given rank level

**Usage**

```r
filter_rank(
  name,
  rank,
  provider = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  collect = TRUE,
  ignore_case = TRUE,
  db = td_connect()
)
```

**Arguments**

- `name`: taxonomic scientific name (e.g. "Aves")
- `rank`: taxonomic rank name. (e.g. "class")
- `provider`: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using options(default_taxadb_provider=...). See [td_create] for a list of recognized providers.
- `version`: Which version of the taxadb provider database should we use? defaults to latest. See tl_import for details.
- `collect`: logical, default TRUE. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.)
- `ignore_case`: should we ignore case (capitalization) in matching names? default is TRUE.
- `db`: a connection to the taxadb database. See details.
Value

a data.frame in the Darwin Core tabular format containing the matching taxonomic entities.

See Also

Other filter_by: filter_by(), filter_common(), filter_id(), filter_name()

Examples

filter_rank("Aves", "class")

---

**fuzzy_filter**

*Match names that start or contain a specified text string*

Description

Match names that start or contain a specified text string

Usage

```r
fuzzy_filter(
  name,
  by = c("scientificName", "vernacularName"),
  provider = getOption("taxadb_default_provider", "itis"),
  match = c("contains", "starts_with"),
  version = latest_version(),
  db = td_connect(),
  ignore_case = TRUE,
  collect = TRUE
)
```

Arguments

- **name**: vector of names (scientific or common, see by) to be matched against.
- **by**: a column name in the taxa_tbl (following Darwin Core Schema terms). The filtering join is executed with this column as the joining variable.
- **provider**: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using options(default_taxadb_provider=...). See [td_create] for a list of recognized providers.
- **match**: should we match by names starting with the term or containing the term anywhere in the name?
version

Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.

db

a connection to the taxadb database. See details.

ignore_case

should we ignore case (capitalization) in matching names? default is `TRUE`.

collect

logical, default `TRUE`. Should we return an in-memory data.frame (default, usually the most convenient), or a reference to lazy-eval table on disk (useful for very large tables on which we may first perform subsequent filtering operations.)

Details

Note that fuzzy filter will be fast with an single or small number of names, but will be slower if given a very large vector of names to match, as unlike other `filter_` commands, fuzzy matching requires separate SQL calls for each name. As fuzzy matches should all be confirmed manually in any event, e.g. not every common name containing "monkey" belongs to a primate species.

This method utilizes the database operation `%like%` to filter tables without loading into memory. Note that this does not support the use of regular expressions at this time.

Examples

```r
## match any common name containing:
name <- c("woodpecker", "monkey")
fuzzy_filter(name, "vernacularName")

## match scientific name
fuzzy_filter("Chera", "scientificName",
             match = "starts_with")
```

Description

A drop-in replacement for `taxize::get_ids()`

Usage

```r
get_ids(
  names,
  db = getOption("taxadb_default_provider", "itis"),
  format = c("prefix", "bare", "uri"),
  version = latest_version(),
  taxadb_db = td_connect(),
)```
get_ids

    ignore_case = TRUE,
    warn = TRUE,
    ...
)

Arguments

names  a list of scientific names (which may include higher-order ranks in most authorities).
db  abbreviation code for the provider. See details.
format  Format for the returned identifier, one of
   • prefix (e.g. NCBI:9606, the default), or
   • bare (e.g. 9606, used in taxize::get_ids()),
   • uri (e.g. http://ncbi.nlm.nih.gov/taxonomy/9606).
version  Which version of the taxadb provider database should we use? defaults to latest. see [available_releases()] for details.
taxadb_db  Connection to from [td_connect()].
ignore_case  should we ignore case (capitalization) in matching names? default is TRUE.
warn  should we display warnings on NAs resulting from multiply-resolved matches? (Unlike unmatched names, these NAs can usually be resolved manually via filter_id())
...  additional arguments (currently ignored)

Details

Note that some taxize authorities: nbn, tropicos, and eol, are not recognized by taxadb and will throw an error here. Meanwhile, taxadb recognizes several authorities not known to [taxize::get_ids()]. Both include itis, ncbi, col, and gbif.

Like all taxadb functions, this function will run fastest if a local copy of the provider is installed in advance using [td_create()].

Value

a vector of IDs, of the same length as the input names Any unmatched names or multiply-matched names will return as NAs. To resolve multi-matched names, use [filter_name()] instead to return a table with a separate row for each separate match of the input name.

See Also

filter_name

Other get: get_names()
get_names

Examples

```r
get_ids("Midas bicolor")
get_ids(c("Midas bicolor", "Homo sapiens"), format = "prefix")
get_ids("Midas bicolor", format = "uri")
```

get_names

Description

Translate identifiers into scientific names

Usage

```r
get_names(
  id,
  db = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  format = c("guess", "prefix", "bare", "uri"),
  taxadb_db = td_connect()
)
```

Arguments

- **id**: a list of taxonomic identifiers.
- **db**: abbreviation code for the provider. See details.
- **version**: Which version of the taxadb provider database should we use? defaults to latest. see [avialable_releases()] for details.
- **format**: Format for the returned identifier, one of
  - prefix (e.g. NCBI:9606, the default), or
  - bare (e.g. 9606, used in taxize::get_ids()),
- **taxadb_db**: Connection to from [td_connect()].

Details

Like all taxadb functions, this function will run fastest if a local copy of the provider is installed in advance using [td_create()].
**mutate_db**

**Value**

A vector of names, of the same length as the input ids. Any unmatched IDs will return as NAs.

**See Also**

Other get: `get_ids()`

**Examples**

```r
get_names(c("ITIS:1025094", "ITIS:1025103"), format = "prefix")
```

---

**mutate_db**

Add new variables to a database

**Description**

dplyr::mutate() cannot pass arbitrary R functions over a database connection. This function provides a way to work around this, by querying the data in chunks and applying the function to each chunk, which is then appended back out to a temporary table.

**Usage**

```r
mutate_db(.data, r_fn, col, new_column, n = 5000L, ...)
```

**Arguments**

- `.data` A dplyr::tbl that uses a database connection, tbl_dbi class.
- `r_fn` any R function that can be called on a vector (column) of the table
- `col` the name of the column to which the R function is applied. (Note, dplyr::mutate() can operate on an arbitrary list of columns, this function only operates on a single column at this time...)
- `new_column` column name for the new column.
- `n` the number of rows included in each chunk, see DBI::dbFetch()
- `...` named arguments to be passed to `r_fn`

**Value**

A dplyr tbl connection to the temporary table in the database
name_contains

*Description*

return all taxa in which scientific name contains the text provided

*Usage*

```r
name_contains(
  name,
  provider = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  db = td_connect(),
  ignore_case = TRUE
)
```

*Arguments*

- **name**: vector of names (scientific or common, see by) to be matched against.
- **provider**: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using `options(default_taxadb_provider=...)`. See [td_create] for a list of recognized providers.
- **version**: Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.
- **db**: a connection to the taxadb database. See details.
- **ignore_case**: should we ignore case (capitalization) in matching names? default is TRUE.

*Examples*

```r
name_contains("Chera")
```

name_starts_with

*Description*

scientific name starts with
Usage

```r
name_starts_with(
  name,
  provider = getOption("taxadb_default_provider", "itis"),
  version = latest_version(),
  db = td_connect(),
  ignore_case = TRUE
)
```

Arguments

- **name**: vector of names (scientific or common, see by) to be matched against.
- **provider**: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using `options(default_taxadb_provider=...)`. See `[td_create]` for a list of recognized providers.
- **version**: Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.
- **db**: a connection to the taxadb database. See details.
- **ignore_case**: should we ignore case (capitalization) in matching names? default is TRUE.

Examples

```r
name_starts_with("Chera")
```

---

**taxadb_dir**

*Show the taxadb directory*

**Description**

Show the taxadb directory

**Usage**

```r
taxadb_dir()
```

**Details**

NOTE: after upgrading duckdb, a user may need to delete any existing databases created with the previous version. An efficient way to do so is `unlink(taxadb::taxadb_dir(), TRUE)`.
Examples

```r
## show the directory
taxadb_dir()
## Purge the local db
unlink(taxadb::taxadb_dir(), TRUE)
```

---

taxa_tbl

Return a reference to a given table in the taxadb database

Description

Return a reference to a given table in the taxadb database

Usage

```r
taxa_tbl(
  provider = getOption("taxadb_default_provider", "itis"),
  schema = c("dwc", "common"),
  version = latest_version(),
  db = td_connect()
)
```

Arguments

- `provider` from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using `options(default_taxadbProvider=...)`. See [td_create] for a list of recognized providers.
- `schema` One of "dwc" (for Darwin Core data) or "common" (for the Common names table.)
- `version` Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.
- `db` a connection to the taxadb database. See details.

Examples

```r
## default schema is the dwc table
taxa_tbl()

## common names table
taxa_tbl(schema = "common")
```
td_connect

Connect to the taxadb database

Description

Connect to the taxadb database

Usage

```r
td_connect(
  dbdir = taxadb_dir(),
  driver = Sys.getenv("TAXADB_DRIVER", "duckdb"),
  read_only = FALSE
)
```

Arguments

- **dbdir**: Path to the database.
- **driver**: Default driver, one of "duckdb", "MonetDBLite", "RSQlite". taxadb will select the first one of those it finds available if a driver is not set. This fallback can be overwritten either by explicit argument or by setting the environmental variable `TAXADB_DRIVER`.
- **read_only**: logical, should the database be opened read_only? Prevents importing but will allow concurrent access from multiple sessions.

Details

This function provides a default database connection for taxadb. Note that you can use taxadb with any DBI-compatible database connection by passing the connection object directly to taxadb functions using the `db` argument. `td_connect()` exists only to provide reasonable automatic defaults based on what is available on your system.

duckdb or MonetDBLite will give the best performance, and regular users taxadb will work with the built-in RSQlite, and with other database connections such as Postgres or MariaDB, but queries (filtering joins) will be much slower on these non-columnar databases.

For performance reasons, this function will also cache and restore the existing database connection, making repeated calls to `td_connect()` much faster and more failsafe than repeated calls to `DBI::dbConnect`

Value

Returns a DBI connection to the default duckdb database
Examples

```r
## OPTIONAL: you can first set an alternative home location,
## such as a temporary directory:
Sys.setenv(TAXADB_HOME=file.path(tempdir(), "taxadb"))

## Connect to the database:
db <- td_connect()
```

---

**td_create**

create a local taxonomic database

**Description**

create a local taxonomic database

**Usage**

```r
td_create(
    provider = getOption("taxadb_default_provider", "itis"),
    schema = c("dwc", "common"),
    version = latest_version(),
    overwrite = TRUE,
    lines = 1e+05,
    dbdir = taxadb_dir(),
    db = td_connect(dbdir)
)
```

**Arguments**

- `provider`  a list (character vector) of provider to be included in the database. By default, will install itis. See details for a list of recognized provider. Use `provider="all"` to install all available provider automatically.
- `schema`  One of "dwc" (for Darwin Core data) or "common" (for the Common names table.)
- `version`  Which version of the taxadb provider database should we use? defaults to latest. See tl_import for details.
- `overwrite`  Should we overwrite existing tables? Default is TRUE. Change to "ask" for interactive interface, or TRUE to force overwrite (i.e. updating a local database upon new release.)
- `lines`  number of lines that can be safely read in to memory at once. Leave at default or increase for faster importing if you have plenty of spare RAM.
- `dbdir`  a location on your computer where the database should be installed. Defaults to user data directory given by [rappdirs::user_data_dir].
- `db`  connection to a database. By default, taxadb will set up its own fast database connection.
Details

Authorities currently recognized by taxadb are:

- **itis**: Integrated Taxonomic Information System, [https://www.itis.gov](https://www.itis.gov)
- **tpl**: The Plant List, [http://www.theplantlist.org/](http://www.theplantlist.org/)
- **gbif**: Global Biodiversity Information Facility, [https://www.gbif.org/](https://www.gbif.org/)
- **fb**: FishBase, [http://fishbase.org](http://fishbase.org)
- **slb**: SeaLifeBase, [http://sealifebase.org](http://sealifebase.org)
- **wd**: Wikidata: [https://www.wikidata.org](https://www.wikidata.org)
- **ott**: OpenTree Taxonomy: [https://github.com/OpenTreeOfLife/reference-taxonomy](https://github.com/OpenTreeOfLife/reference-taxonomy)
- **iucn**: IUCN Red List, [https://iucnredlist.org](https://iucnredlist.org)
- **itis_test**: a small subset of ITIS, cached locally with the package for testing purposes only

Value

path where database has been installed (invisibly)

Examples

```r
## Install the ITIS database
td_create()

## force re-install:
td_create( overwrite = TRUE)
```

---

**td_disconnect**

Disconnect from the taxadb database.

Description

Disconnect from the taxadb database.

Usage

```
td_disconnect(db = td_connect())
```

Arguments

- **db** database connection
Details

This function manually closes a connection to the taxadb database.

Examples

```r
## Disconnect from the database:
td_disconnect()
```

---

**tl_import**

Import taxonomic database tables

Description

Downloads the requested taxonomic data tables and return a local path to the data in tsv.gz format. Downloads are cached and identified by content hash so that `tl_import` will not attempt to download the same file multiple times.

Usage

```r
tl_import(
  provider = getOption("tl_default_provider", "itis"),
  schema = c("dwc", "common"),
  version = latest_version(),
  prov = paste0("https://raw.githubusercontent.com/",
                "boettiger-lab/taxadb-cache/master/prov.json")
)
```

Arguments

- **provider**: from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using options(default_taxadb_provider=...). See [td_create] for a list of recognized providers.
- **schema**: One of "dwc" (for Darwin Core data) or "common" (for the Common names table.)
- **version**: Which version of the taxadb provider database should we use? defaults to latest. See `tl_import` for details.
- **prov**: Address (URL) to provenance record

Details

`tl_import` parses a DCAT2/PROV-O record to determine the correct version to download. If offline, `tl_import` will attempt to resolve against it’s own provenance cache. Users can also examine `/parse` the prov JSON-LD file directly to determine the provenance of the data products used.
Value

path(s) to the downloaded files in the cache
Index

* filter_by
  - filter_by, 5
  - filter_common, 6
  - filter_id, 7
  - filter_name, 9
  - filter_rank, 10

* get
  - get_ids, 12
  - get_names, 14

  clean_names, 2
  common_contains, 3
  common_starts_with, 4

  DBI::dbConnect, 19
  DBI::dbFetch(), 15
  dplyr::mutate(), 15
  dplyr::tbl, 15

  filter_by, 5, 7, 8, 10, 11
  filter_common, 6, 6, 8, 10, 11
  filter_id, 6, 7, 7, 10, 11
  filter_id(), 13
  filter_name, 6–8, 9, 11
  filter_rank, 6–8, 10, 10
  fuzzy_filter, 11

  get_ids, 12, 15
  get_names, 13, 14

  mutate_db, 15

  NA, 13, 15
  name_contains, 16
  name_starts_with, 16

  stringi::stri_trans_general(), 3

  taxa_tbl, 18
  taxadb_dir, 17
  td_connect, 19
  td_create, 20
  td_disconnect, 21
  tl_import, 4, 5, 7–10, 12, 16–18, 20, 22, 22
  TRUE, 3