

Package: taxadb (via r-universe)

September 29, 2026

Version 0.3.0

Title A High-Performance Taxonomic Database Interface

Description Provides fast access to many commonly used taxonomic authorities in a uniform Darwin Core format. Tables are read directly from versioned Parquet snapshots, streamed from cloud storage or from a local copy, and queried with familiar 'dplyr' verbs.

License MIT + file LICENSE

Encoding UTF-8

ByteCompile true

Depends R (>= 4.0)

Imports DBI, duckdb, tibble, dplyr, dbplyr, rlang, magrittr, stringi

Roxygen list(markdown = TRUE)

Suggests spelling, testthat, curl, knitr, rmarkdown, crayon, withr

Language en-US

VignetteBuilder knitr

URL <<https://docs.ropensci.org/taxadb/>>,
<<https://github.com/ropensci/taxadb>>

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

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev xz-utils

Repository <https://ropensci.r-universe.dev>

Date/Publication 2026-08-30 19:26:18 UTC

RemoteUrl <https://github.com/ropensci/taxadb>

RemoteRef master

RemoteSha e933f9567417c014dab0a6166d2f4f76f8f30409

Contents

available_providers	3
available_versions	3
build_col	4
build_dir	5
build_fishbase	5
build_gbif	7
build_itis	8
build_ncbi	9
build_ott	10
clean_names	11
common_contains	12
common_starts_with	13
filter_by	13
filter_common	15
filter_id	16
filter_name	17
filter_rank	19
fuzzy_filter	20
get_ids	21
get_names	23
latest_version	24
list_snapshots	25
name_contains	25
name_starts_with	26
taxa_tbl	27
taxadb_dir	28
taxadb_provider_info	28
taxadb_providers	29
taxadb_repo	30
taxadb_uri	30
td_build	31
td_connect	32
td_create	33
td_disconnect	34
td_download	35
td_manifest	36
td_validate	37
td_write_metadata	38

Index

40

available_providers	<i>Name providers available for a given version</i>
---------------------	---

Description

Name providers available for a given version

Usage

```
available_providers(version = latest_version(), db = td_connect())
```

Arguments

version	snapshot version, defaults to the latest available
db	a connection from td_connect()

Value

a data.frame of provider and the schemas published for it

Examples

```
## Not run:  
available_providers()  
  
## End(Not run)
```

available_versions	<i>Versions of the taxadb data available</i>
--------------------	--

Description

Versions of the taxadb data available

Usage

```
available_versions(db = td_connect())
```

Arguments

db	a connection from td_connect()
----	--

Value

a character vector of available snapshot versions

Examples

```
## Not run:
available_versions()

## End(Not run)
```

build_col	<i>Rebuild the Catalogue of Life snapshot</i>
-----------	---

Description

Rebuild the Catalogue of Life snapshot

Usage

```
build_col(
  version = format(Sys.Date(), "%Y"),
  archive = NULL,
  dir = build_dir(),
  db = td_connect()
)
```

Arguments

version	snapshot version to write, e.g. "2026"
archive	path to the COL Darwin Core Archive; downloaded if missing
dir	directory for build inputs and outputs
db	a duckdb connection

Details

COL publishes a Darwin Core Archive whose scientificName carries the authorship – Acanthocerataceae Crawford & R. taxadb wants the canonical name, since authorship abbreviations vary too much between providers to match on. COL also supplies scientificNameAuthorship separately, so the canonical name is the one with that suffix removed.

COL marks accepted names by leaving acceptedNameUsageID empty, and distinguishes accepted from provisionally accepted; both are accepted in the sense that matters here, that they are not a synonym of anything else, so both get acceptedNameUsageID set to their own taxonID.

Value

the paths written, invisibly

See Also

Other build: [build_fishbase\(\)](#), [build_gbif\(\)](#), [build_itis\(\)](#), [build_ncbi\(\)](#), [build_ott\(\)](#)

Examples

```
## Not run:  
build_col("2026")  
  
## End(Not run)
```

build_dir	<i>Where build inputs and outputs are kept</i>
-----------	--

Description

Where build inputs and outputs are kept

Usage

```
build_dir()
```

Details

Provider archives are large and slow to fetch, so they are cached here between builds. Override with the TAXADB_BUILD_DIR environment variable.

Value

path to the taxadb build directory

Examples

```
build_dir()
```

build_fishbase	<i>Rebuild the FishBase or SeaLifeBase snapshot</i>
----------------	---

Description

Rebuild the FishBase or SeaLifeBase snapshot

Usage

```
build_fishbase(  
  version = format(Sys.Date(), "%Y"),  
  provider = c("fb", "slb"),  
  fb_version = NULL,  
  dir = build_dir(),  
  db = td_connect()  
)
```

Arguments

version	snapshot version to write, e.g. "2026"
provider	"fb" for FishBase or "slb" for SeaLifeBase
fb_version	which FishBase snapshot to build from, e.g. "v26.07". Defaults to the most recent published.
dir	directory for build inputs and outputs
db	a duckdb connection

Details

FishBase and SeaLifeBase share a schema, and both are already published as Parquet alongside the taxadb snapshots, so this build reads them over the network and downloads nothing.

FishBase numbers accepted species (SpecCode) and synonyms (SynCode) in two independent sequences, so the same integer means different things in each: SpecCode 1 is *Scyris indica* while SynCode 1 is *Alausa coerulea*. Prefixing both as FB:1 would make one identifier name two taxa, which it did in the previously published table – 20,295 FishBase identifiers and 61,125 SeaLifeBase ones were ambiguous.

Only SpecCode is therefore used as the taxonID, and synonyms carry a NULL one exactly as they do for NCBI and OTT. Nothing is lost: the SynCode is published in its own synonymID column. A consequence is that a synonym whose SpecCode is 0 – not linked to any species record, 1,043 names in FishBase and 7,512 in SeaLifeBase – has nothing to resolve to and is dropped.

Classification comes from the families table. FishBase covers only fishes, so its phylum and kingdom are constant. SeaLifeBase spans some sixty phyla across several kingdoms and asserts no kingdom itself, so kingdom is left empty there rather than inferred.

FishBase data is CC-BY-NC (fishbase.org).

Value

the paths written, invisibly

See Also

Other build: [build_col\(\)](#), [build_gbif\(\)](#), [build_itis\(\)](#), [build_ncbi\(\)](#), [build_ott\(\)](#)

Examples

```
## Not run:
build_fishbase("2026", provider = "fb")

## End(Not run)
```

build_gbif*Rebuild the GBIF backbone snapshot*

Description

Rebuild the GBIF backbone snapshot

Usage

```
build_gbif(  
  version = format(Sys.Date(), "%Y"),  
  archive = NULL,  
  dir = build_dir(),  
  db = td_connect()  
)
```

Arguments

version	snapshot version to write, e.g. "2026"
archive	path to the GBIF backbone archive; downloaded if missing
dir	directory for build inputs and outputs
db	a duckdb connection

Details

GBIF supplies `canonicalName` – the name without authorship – alongside the full `scientificName`, so no name parsing is needed. It leaves `canonicalName` empty for names its parser cannot analyse, which includes the sequence-derived identifiers GBIF carries in quantity (BOLD BINs, UNITE species hypotheses, metagenome-assembled genomes) and hybrid formulas; for those the `scientificName` is the name, and is used.

GBIF leaves `acceptedNameUsageID` empty on accepted and on doubtful names alike, since it redirects neither. Both therefore become their own accepted name here, which is also what keeps GBIF's 40,895 synonyms of doubtful names resolvable.

Value

the paths written, invisibly

See Also

Other build: [build_col\(\)](#), [build_fishbase\(\)](#), [build_itis\(\)](#), [build_ncbi\(\)](#), [build_ott\(\)](#)

Examples

```
## Not run:  
build_gbif("2026")  
  
## End(Not run)
```

 build_itis

 Rebuild the ITIS snapshot

Description

Rebuild the ITIS snapshot

Usage

```
build_itis(
  version = format(Sys.Date(), "%Y"),
  archive = NULL,
  dir = build_dir(),
  db = td_connect()
)
```

Arguments

version	snapshot version to write, e.g. "2026"
archive	path to the ITIS SQLite archive; downloaded if missing
dir	directory for build inputs and outputs
db	a duckdb connection

Details

ITIS distributes a SQLite database, which duckdb reads directly. n_usage carries the accepted/synonym distinction under two vocabularies, zoological (valid/invalid) and botanical (accepted/not accepted); both map onto accepted and synonym.

ITIS assigns a TSN to synonyms as well as accepted names, so every row here carries its own taxonID – which makes ITIS the reference for the taxadb rules checked by `td_validate()`.

scientificNameAuthorship is taken from ITIS's own author table, keyed on the author id together with the kingdom, since the id is only unique within one.

Value

the paths written, invisibly

See Also

Other build: [build_col\(\)](#), [build_fishbase\(\)](#), [build_gbif\(\)](#), [build_ncbi\(\)](#), [build_ott\(\)](#)

Examples

```
## Not run:
build_itis("2026")

## End(Not run)
```

build_ncbi

*Rebuild the NCBI Taxonomy snapshot***Description**

Rebuild the NCBI Taxonomy snapshot

Usage

```
build_ncbi(
  version = format(Sys.Date(), "%Y"),
  archive = NULL,
  dir = build_dir(),
  db = td_connect()
)
```

Arguments

version	snapshot version to write, e.g. "2026"
archive	path to the NCBI taxdump.tar.gz; downloaded if missing
dir	directory for build inputs and outputs
db	a duckdb connection

Details

NCBI distributes `nodes.dmp` (the hierarchy) and `names.dmp` (every name), in a format that claims to be tab-separated but delimits fields with `\\t|\\t`.

Every row of `names.dmp` carries the `tax_id` of the *accepted* taxon, whatever the name's class: NCBI mints no separate identifier for a synonym. So the scientific name rows become the accepted names, carrying a `taxonID` and pointing `acceptedNameUsageID` at themselves, and every other name class becomes a row for the same taxon with a `NULL taxonID` – which the `taxadb` rules permit, since there is no identifier to give.

Value

the paths written, invisibly

See Also

Other build: [build_col\(\)](#), [build_fishbase\(\)](#), [build_gbif\(\)](#), [build_itis\(\)](#), [build_ott\(\)](#)

Examples

```
## Not run:
build_ncbi("2026")

## End(Not run)
```

 build_ott

 Rebuild the Open Tree Taxonomy snapshot

Description

Rebuild the Open Tree Taxonomy snapshot

Usage

```
build_ott(
  version = format(Sys.Date(), "%Y"),
  archive = NULL,
  ott_version = "3.7.3",
  dir = build_dir(),
  db = td_connect()
)
```

Arguments

version	snapshot version to write, e.g. "2026"
archive	path to the OTT release archive; downloaded if missing
ott_version	the OTT release to build from, e.g. "3.7.3"
dir	directory for build inputs and outputs
db	a duckdb connection

Details

OTT ships `taxonomy.tsv` and `synonyms.tsv`, both delimited with `\\t|\\t`. `synonyms.tsv` keys each synonym to the uid of the name it is a synonym *of*: OTT mints no identifier for the synonym itself, so those rows carry a NULL `taxonID`, which the `taxadb` rules allow.

OTT no longer populates the `type` column of `synonyms.tsv` – it is empty for all 2.2 million rows in release 3.7.3 – so a synonym is recorded as synonym unless a type is given.

OTT publishes no vernacular names, so there is no common table for this provider; `filter_common()` warns accordingly.

Value

the paths written, invisibly

See Also

Other build: `build_col()`, `build_fishbase()`, `build_gbif()`, `build_itis()`, `build_ncbi()`

Examples

```
## Not run:
build_ott("2026")

## End(Not run)
```

clean_names

Clean taxonomic names

Description

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

Usage

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

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

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

common_contains	<i>common name starts with</i>
-----------------	--------------------------------

Description

common name starts with

Usage

```
common_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 available_versions() for details.
db	a connection to the taxadb database. See details.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.

Examples

```
common_contains("monkey")
```

common_starts_with	<i>common name starts with</i>
--------------------	--------------------------------

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 available_versions() for details.
db	a connection to the taxadb database. See details.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.

Examples

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

filter_by	<i>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 (scientificNames, taxonIDs, etc).</i>
-----------	--

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 (scientificNames, taxonIDs, etc).

Usage

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

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 <code>options(default_taxadb_provider=...)</code> . See <code>[td_create]</code> 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 available_versions() 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.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.

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:180092", "ITIS:916116"), "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	a character vector of common (vernacular English) names, e.g. "Humans"
provider	from which provider should the hierarchy be returned? Default is 'itis', which can also be configured using <code>options(default_taxadb_provider=...)</code> . See <code>[td_create]</code> for a list of recognized providers.
version	Which version of the taxadb provider database should we use? defaults to latest. See available_versions() 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? Can be significantly slower to run.
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_id\(\)](#), [filter_name\(\)](#), [filter_rank\(\)](#)

Examples

```
filter_common("Pied Tamarin")
```

filter_id	<i>Return a taxonomic table matching the requested ids</i>
-----------	--

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(),  
  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 available_versions() 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

```
filter_id(c("ITIS:180092", "ITIS:916116"))
filter_id("ITIS:916116", type="acceptedNameUsageID")
```

filter_name	<i>Look up taxonomic information by scientific name</i>
-------------	---

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 = FALSE,
  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 available_versions() 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? Can be significantly slower to run.
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

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

filter_rank	<i>Get all members (descendants) of a given rank level</i>
-------------	--

Description

Get all members (descendants) of a given rank level

Usage

```
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 <code>options(default_taxadb_provider=...)</code> . See <code>[td_create]</code> for a list of recognized providers.
version	Which version of the taxadb provider database should we use? defaults to latest. See available_versions() 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? Can be significantly slower to run.
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	<i>Match names that start or contain a specified text string</i>
--------------	--

Description

Match names that start or contain a specified text string

Usage

```
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 available_versions() for details.
db	a connection to the taxadb database. See details.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.
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

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

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

get_ids

get_ids

Description

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

Usage

```
get_ids(
  names,
  provider = getOption("taxadb_default_provider", "itis"),
  format = c("prefix", "bare", "uri"),
  version = latest_version(),
  taxadb_db = td_connect(),
  ignore_case = FALSE,
  warn = TRUE,
  db = NULL,
  ...
)
```

Arguments

names	a list of scientific names (which may include higher-order ranks in most authorities).
-------	--

provider	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 <code>taxize::get_ids()</code>), • uri (e.g. <code>http://ncbi.nlm.nih.gov/taxonomy/9606</code>).
version	Which version of the taxadb provider database should we use? defaults to latest. see <code>[available_releases()]</code> for details.
taxadb_db	Connection to from <code>[td_connect()]</code> .
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 <code>filter_id()</code>)
db	previous name for provider argument, now deprecated
...	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()`

Examples

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

`get_names`*get_names*

Description

Translate identifiers into scientific names

Usage

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

Arguments

<code>id</code>	a list of taxonomic identifiers.
<code>provider</code>	abbreviation code for the provider. See details.
<code>version</code>	Which version of the taxadb provider database should we use? defaults to latest. see <code>[available_releases()]</code> for details.
<code>format</code>	Format for the returned identifier, one of • <code>prefix</code> (e.g. NCBI:9606, the default), or • <code>bare</code> (e.g. 9606, used in <code>taxize::get_ids()</code>), • <code>uri</code> (e.g. <code>http://ncbi.nlm.nih.gov/taxonomy/9606</code>).
<code>taxadb_db</code>	Connection to from <code>[td_connect()]</code> .
<code>db</code>	previous name for provider argument, now deprecated

Details

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 names, of the same length as the input ids. Any unmatched IDs will return as [NAs](#).

See Also

Other get: [get_ids\(\)](#)

Examples

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

latest_version	<i>The most recent taxadb snapshot version</i>
----------------	--

Description

The most recent taxadb snapshot version

Usage

```
latest_version(db = td_connect())
```

Arguments

db a connection from `td_connect()`

Details

Versions are ordered as version numbers, not as strings. This matters: as strings "22.12" sorts after "2026", so a plain `max()` would make an archival release from 2022 the default for every query once it was published.

Value

the latest available version, as a character string

Examples

```
## Not run:
latest_version()

## End(Not run)
```

list_snapshots	<i>List the taxonomic snapshots available from the taxadb repository</i>
----------------	--

Description

List the taxonomic snapshots available from the taxadb repository

Usage

```
list_snapshots(db = td_connect())
```

Arguments

db a connection from `td_connect()`

Details

Requires network access. Results are cached for the session.

Value

a data.frame with one row per published Parquet file, giving its version, schema, provider and uri.

Examples

```
## Not run:
list_snapshots()

## End(Not run)
```

name_contains	<i>return all taxa in which scientific name contains the text provided</i>
---------------	--

Description

return all taxa in which scientific name contains the text provided

Usage

```
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 available_versions() for details.
db	a connection to the taxadb database. See details.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.

Examples

```
name_contains("Trochalop")
```

name_starts_with	<i>scientific name starts with</i>
------------------	------------------------------------

Description

scientific name starts with

Usage

```
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 available_versions() for details.
db	a connection to the taxadb database. See details.
ignore_case	should we ignore case (capitalization) in matching names? Can be significantly slower to run.

Examples

```
name_starts_with("Trochalop")
```

taxa_tbl	<i>Return a reference to a given table in the taxadb database</i>
----------	---

Description

Return a reference to a given table in the taxadb database

Usage

```
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 <code>options(default_taxadb_provider=...)</code> . See <code>[td_create]</code> 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 available_versions() for details.
db	a connection to the taxadb database. See details.

Details

The returned table is a duckdb view over Parquet, so it can be manipulated with any dplyr verb and is only ever read to the extent your query requires. Unless a local copy has been installed with [td_download\(\)](#), the data is streamed from remote storage on demand.

Value

a lazy dplyr table backed by the requested Parquet snapshot.

Examples

```
## default schema is the Darwin Core table
taxa_tbl()

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

taxadb_dir	Show the local taxadb directory
------------	---------------------------------

Description

Show the local taxadb directory

Usage

```
taxadb_dir()
```

Details

Local snapshots downloaded by [td_download\(\)](#) are stored here. Override with the TAXADB_HOME environment variable.

Value

path to the local taxadb data directory

Examples

```
taxadb_dir()
```

taxadb_provider_info	Describe the taxonomic name providers
----------------------	---------------------------------------

Description

Describe the taxonomic name providers

Usage

```
taxadb_provider_info(provider = taxadb_providers())
```

Arguments

provider one or more provider abbreviations; all by default

Details

Providers are not interchangeable. `col`, `gbif` and `ott` are synthesis projects that integrate other checklists, while `itis`, `ncbi`, `fb` and `slb` are primary authorities; the `synthesis` column records which is which. More importantly, providers disagree: the same name can be accepted by one and a synonym of something else in another, so a name resolved against one provider should not be mixed with names resolved against another. See `vignette("data-sources")`.

Redistribution terms differ too. `fb` and `slb` are CC BY-NC, so those two tables may not be used commercially.

Value

a data.frame with one row per provider giving its `title`, `url`, the source its data is taken from, its license and a citation.

Examples

```
taxadb_provider_info()
taxadb_provider_info("col")$citation
```

taxadb_providers	<i>Providers taxadb can rebuild</i>
------------------	-------------------------------------

Description

Providers taxadb can rebuild

Usage

```
taxadb_providers()
```

Details

Unlike `available_providers()`, which reports what is published, this reports what `td_build()` knows how to derive from the provider's own distribution.

Value

a character vector of provider abbreviations

Examples

```
taxadb_providers()
```

taxadb_repo	<i>The taxadb data repository</i>
-------------	-----------------------------------

Description

The taxadb data repository

Usage

```
taxadb_repo()
```

Details

Override with `options(taxadb_repo=)` or the `TAXADB_REPO` environment variable to read from a mirror or a staging repository.

Value

the object-store prefix holding taxadb snapshots.

Examples

```
taxadb_repo()
```

taxadb_uri	<i>Locate the Parquet files backing a taxadb table</i>
------------	--

Description

Locate the Parquet files backing a taxadb table

Usage

```
taxadb_uri(  
  provider = getOption("taxadb_default_provider", "itis"),  
  schema = c("dwc", "common"),  
  version = latest_version(),  
  local = NULL  
)
```

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 available_versions() for details.
local	should we return the path to a local snapshot? By default a local copy is used when one is present (see td_download()), and the remote snapshot is streamed otherwise.

Value

a glob pattern (or file path) that duckdb can read

Examples

```
taxadb_uri("itis_test")
```

td_build	<i>Rebuild taxadb snapshots from the providers</i>
----------	--

Description

Runs a provider's preprocessing end to end: fetch the provider's own distribution, normalize it to the taxadb Darwin Core schema, and write the Parquet snapshot.

Usage

```
td_build(  
  provider = "itis",  
  version = format(Sys.Date(), "%Y"),  
  dir = build_dir(),  
  validate = TRUE,  
  db = td_connect(),  
  ...  
)
```

Arguments

provider	one or more providers to build. See taxadb_providers() .
version	the snapshot version to write, defaults to the year.
dir	directory for build inputs and outputs, see build_dir() .
validate	should each table be checked with td_validate() after it is written? Default TRUE.

db a duckdb connection

... passed to the individual provider builder, e.g. archive to use an already-downloaded copy.

Details

Snapshots are published for the providers so that most users never need to run this. It is here so that a user who needs a fresher snapshot than the published one, or who wants to check how a table was derived, can rebuild it themselves rather than asking someone to.

Builds are done entirely in duckdb, out of core, so they are bounded by disk rather than memory. The archives are large: COL and GBIF are around 500MB and 1GB compressed respectively, and are cached in dir between builds.

Value

a data.frame of the validation results, invisibly if validate is FALSE the paths written.

Examples

```
## Not run:
## rebuild one provider
td_build("itis")

## rebuild everything that can be built without credentials
td_build(taxadb_providers())

## End(Not run)
```

td_connect	<i>Connect to the taxadb database</i>
------------	---------------------------------------

Description

Connect to the taxadb database

Usage

```
td_connect(dbdir = NULL, driver = NULL, read_only = NULL)
```

Arguments

dbdir Deprecated, ignored.

driver Deprecated, ignored. The driver is always duckdb.

read_only Deprecated, ignored.

Details

taxadb reads Parquet snapshots directly from object storage (<https://source.coop>) using duckdb's httpfs extension, so no data import step is required. This function returns a connection with httpfs loaded and the S3 endpoint configured for anonymous access.

For performance reasons the connection is cached and reused, making repeated calls to `td_connect()` much faster and more failsafe than repeated calls to `DBI::dbConnect`.

The httpfs extension needed for remote reads is loaded on first use rather than at connect time, so a session that only reads local snapshots or the bundled test data never touches the network.

duckdb would otherwise scan with one thread per core and let its buffer pool grow to most of system RAM. For the selective scans taxadb makes that is the wrong trade: each scanning thread holds a decompressed Parquet row group, so memory grows with core count while the query gets no faster. On a 128-core machine, looking up one name in the GBIF table peaked at 1324 MB with the duckdb defaults and 322 MB capped at eight threads – and the capped run was faster (0.7s against 1.0s).

So the connection caps threads at `TAXADB_THREADS` (8) or the core count, whichever is lower. Raise it with `options(taxadb_threads=)` for bulk work – `td_build()` does this itself – and set `options(taxadb_memory_limit=)` to bound the buffer pool.

Value

a DBI connection to an in-process duckdb database, configured for anonymous streaming reads from the taxadb data repository.

Examples

```
db <- td_connect()
```

td_create	Create a local taxadb database
-----------	--------------------------------

Description

Superseded by `td_download()`.

Usage

```
td_create(
  provider = getOption("taxadb_default_provider", "itis"),
  schema = c("dwc", "common"),
  version = latest_version(),
  overwrite = FALSE,
  lines = NULL,
  dbdir = NULL,
  db = td_connect()
)
```

Arguments

provider	a character vector of provider(s) to download. See available_providers() for the providers published in a given version.
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 available_versions() for details.
overwrite	passed to td_download()
lines	deprecated, ignored.
dbdir	deprecated, ignored.
db	a connection to the taxadb database. See details.

Details

taxadb no longer needs to import data before querying it: tables are read directly from Parquet, streamed from remote storage or from a local copy. `td_create()` is retained as an alias for [td_download\(\)](#), which installs a local copy.

Value

the local paths of the downloaded files, invisibly.

Examples

```
## Not run:
td_create("itis")

## End(Not run)
```

td_disconnect	<i>Disconnect from the taxadb database.</i>
---------------	---

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.

Value

invisible TRUE

Examples

```
td_disconnect()
```

td_download	<i>Install a local copy of a taxadb snapshot</i>
-------------	--

Description

Downloads the Parquet files for the requested provider(s) into `taxadb_dir()`, so that subsequent queries read from local disk instead of streaming from remote storage.

Usage

```
td_download(
  provider = getOption("taxadb_default_provider", "itis"),
  schema = c("dwc", "common"),
  version = latest_version(),
  overwrite = FALSE,
  db = td_connect()
)
```

Arguments

provider	a character vector of provider(s) to download. See available_providers() for the providers published in a given version.
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 available_versions() for details.
overwrite	should we re-download files that are already present? Default FALSE.
db	a connection to the taxadb database. See details.

Details

Streaming is fast enough for most interactive use and requires no setup, so a local copy is optional. Install one when you will make many queries against the same table, when you need to work offline, or when you want a snapshot pinned on disk for reproducibility.

Snapshots are large: the Darwin Core tables for col and gbif are each several hundred MB. Use `available_providers()` to see what is published, and delete a local copy with `unlink(taxadb_dir(), recursive = TRUE)`.

Value

the local paths of the downloaded files, invisibly.

Examples

```
## Not run:
## Install a local copy of ITIS. Writes to taxadb_dir() and downloads
# tens of MB, so this is never run unattended.
td_download("itis")

## End(Not run)
```

td_manifest	<i>Describe a built snapshot</i>
-------------	----------------------------------

Description

Describe a built snapshot

Usage

```
td_manifest(
  version = format(Sys.Date(), "%Y"),
  dir = build_dir(),
  validate = TRUE,
  db = td_connect()
)
```

Arguments

version	the snapshot version to describe
dir	the build output directory, see build_dir()
validate	should each table be checked with td_validate() and the result recorded? Default TRUE. Archival snapshots predate the schema rules and will report violations; recording them is the point.
db	a duckdb connection

Details

Run after [td_build\(\)](#). The row counts and checksums are read back off the written files rather than carried over from the build, so the manifest describes what was actually published.

Value

a data.frame with one row per published table, giving its provider, schema, row count, columns, file sizes and checksum, and the upstream release it was derived from.

Examples

```
## Not run:
td_manifest("2026")

## End(Not run)
```

td_validate

Check a taxadb table against the taxadb Darwin Core rules

Description

Check a taxadb table against the taxadb Darwin Core rules

Usage

```
td_validate(
  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 <code>options(default_taxadb_provider=...)</code> . See <code>[td_create]</code> 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 available_versions() for details.
db	a connection to the taxadb database. See details.

Details

The rules checked are:

- **columns** – the required Darwin Core terms are present, spelled in Darwin Core camelCase.
- **types** – identifier and name columns are character. A column that is entirely NA will often be typed as integer or logical by mistake, which this catches.
- **scientificName** – never NA. Every row names something, at every rank: *Animalia* is a scientificName just as *Homo sapiens* is.
- **taxonRank**, **taxonomicStatus** – never NA.
- **acceptedNameUsageID** – never NA, on synonyms *and* on accepted names. This is where taxadb is stricter than Darwin Core.

- **accepted_has_id** – a row labelled accepted is its own accepted name: taxonID is present and equals acceptedNameUsageID. (taxonID may be NA on a synonym, where the provider mints no identifier for it – OTT and NCBI, for instance, do not.)
- **accepted_resolves** – every acceptedNameUsageID matches the taxonID of a self-referencing row. No dangling references.
- **synonym_not_self** – a row labelled a synonym points somewhere else, never at itself.
- **taxonID_one_name** – a taxonID always names the same scientificName. An identifier may appear on more than one row: ITIS records 255 synonyms that are ambiguous between two accepted taxa, and a row for each is the honest representation. What must not happen is one identifier naming two *different* names, which is what results from a provider numbering its accepted names and its synonyms in separate sequences and both being given the same prefix.
- **accepted_unique** – no duplicate taxonID among accepted names.
- **id_prefix** – identifiers are the provider's identifier prefixed by the provider abbreviation in capitals, e.g. ITIS:180092.

taxonomicStatus is deliberately *not* checked against a controlled vocabulary: providers draw real distinctions (homotypic synonym, provisionally accepted, doubtful, misapplied) that are worth preserving. The rules are therefore phrased structurally. A name the provider does not redirect to another name is its own accepted name whatever confidence it expresses about it, so doubtful and provisionally accepted rows self-reference exactly as accepted ones do; only the two terms whose meaning taxadb actually relies on, accepted and synonym, are given a required shape.

Value

a data.frame with one row per rule, giving whether the table passes, how many rows violations were found, and a note.

Examples

```
td_validate("itis_test")
```

td_write_metadata	<i>Write the metadata published with a snapshot</i>
-------------------	---

Description

Writes manifest.csv and README.md into the snapshot directory, ready to be uploaded alongside the Parquet files.

Usage

```
td_write_metadata(  
  version = format(Sys.Date(), "%Y"),  
  dir = build_dir(),  
  repo = taxadb_repo(),  
  archival = FALSE,  
  db = td_connect()  
)
```

Arguments

version	the snapshot version to describe
dir	the build output directory, see build_dir()
repo	the data repository the snapshot will be published to
archival	is this a republication of a historical release rather than a fresh build? Archival snapshots are byte-identical to what that version originally contained, so they predate the current schema rules and the README says so.
db	a duckdb connection

Details

The README states what the tables are, what the schema means, where each provider's data came from and under what licence, so that someone who finds the data without the package can still use it.

Value

the paths written, invisibly

Examples

```
## Not run:  
td_write_metadata("2026")  
  
## End(Not run)
```

Index

* build

- build_col, 4
- build_fishbase, 5
- build_gbif, 7
- build_itis, 8
- build_ncbi, 9
- build_ott, 10

* filter_by

- filter_by, 13
- filter_common, 15
- filter_id, 16
- filter_name, 17
- filter_rank, 19

* get

- get_ids, 21
- get_names, 23

- available_providers, 3
- available_providers(), 29, 34, 35
- available_versions, 3
- available_versions(), 12–16, 18–20, 26, 27, 31, 34, 35, 37

- build_col, 4
- build_col(), 6–10
- build_dir, 5
- build_dir(), 31, 36, 39
- build_fishbase, 5
- build_fishbase(), 4, 7–10
- build_gbif, 7
- build_gbif(), 4, 6, 8–10
- build_itis, 8
- build_itis(), 4, 6, 7, 9, 10
- build_ncbi, 9
- build_ncbi(), 4, 6–8, 10
- build_ott, 10
- build_ott(), 4, 6–9

- clean_names, 11
- common_contains, 12

- common_starts_with, 13

- DBI::dbConnect, 33

- filter_by, 13
- filter_by(), 15, 17–19
- filter_common, 15
- filter_common(), 10, 14, 17–19
- filter_id, 16
- filter_id(), 14, 15, 18, 19, 22
- filter_name, 17
- filter_name(), 14, 15, 17, 19
- filter_rank, 19
- filter_rank(), 14, 15, 17, 18
- fuzzy_filter, 20

- get_ids, 21
- get_ids(), 23
- get_names, 23
- get_names(), 22

- latest_version, 24
- list_snapshots, 25

- NA, 22, 23
- name_contains, 25
- name_starts_with, 26

- stringi::stri_trans_general(), 11

- taxa_tbl, 27
- taxadb_dir, 28
- taxadb_dir(), 35
- taxadb_provider_info, 28
- taxadb_providers, 29
- taxadb_providers(), 31
- taxadb_repo, 30
- taxadb_uri, 30
- td_build, 31
- td_build(), 29, 33, 36
- td_connect, 32

td_connect(), [3](#), [24](#), [25](#)
td_create, [33](#)
td_disconnect, [34](#)
td_download, [35](#)
td_download(), [27](#), [28](#), [31](#), [33](#), [34](#)
td_manifest, [36](#)
td_validate, [37](#)
td_validate(), [8](#), [31](#), [36](#)
td_write_metadata, [38](#)
TRUE, [11](#)