taxadb: A High-Performance Local Taxonomic Database Interface

Kari Norman[a], Scott Chamberlain[b], Carl Boettiger[a,1]

[a] *Dept of Environmental Science, Policy, and Management, University of California Berkeley, Berkeley CA 94720-3114, USA*
[b] *The rOpenSci Project, University of California Berkeley, Berkeley CA 94720-3114, USA*

Abstract

A familiar and growing challenge in ecological and evolutionary research is that of reconciling scientific names of relevant taxa when combining data from separate sources. While this problem is already well understood and numerous naming authorities have been created to address the issue, most researchers lack a fast, consistent, and intuitive way to reconcile taxonomic names. Here, we present `taxadb` R package to address this gap. In contrast to existing tools, `taxadb` provides the following:
1) `taxadb` accesses established naming authorities to resolve synonyms, IDs, and hiearchical classification.
2) `taxadb` creates a local database, managed automatically from within R, that provides fast operations on millions of taxonomic names.
3) `taxadb` provides a consistent and intuitive data format.
4) `taxadb` is built on a simple, extensible and language agnostic design that can easily accomodiate new authorities.

As ecologists and evolutionary biologists synthesize datasets across larger and larger assemblies of species, we face a continual challenge of reconciling taxonomic names. How many species are in the combined data? Do the studies use the same names for the same species, or do they use different synonyms for the same species? Failing to correct for such differences can lead to significant inflation of species counts and miss-aligned datasets (Figure 1). These challenges have become particularly acute as it becomes increasingly common for researchers to work across larger number and diversity of species in any given analysis, which may preclude the resources or substantative taxonomic expertise all clades needed to resolve scientific names (Patterson et al. 2010). While these issues have long been recognized in the literature [], and a growing number of databases and tools have emerged and grown over the past few decades (e.g. ITIS 2019; Biotechnology Information 2019; Roskov Y. 2018), it remains difficult to resolve taxonomic names to a common authority in a transparent, efficient, and automatable manner. Here, we present an R package, `taxadb`, which seeks to address this gap.

Databases of taxonomic names such as the Integrated Taxonomic Information System (ITIS; ITIS 2019), the National Center for Biological Information's (NCBI) Taxonomy database, (Biotechnology Information 2019), the Catalogue of Life (COL; Roskov Y. 2018), and over one hundred other providers have sought to address these problems by providing expert-curated lists of accepted taxonomic names, synonyms, associated taxonomic rank, hierarchical classification, and scientific authority (e.g. author and date) establishing a scientific name. The R language (R Core Team 2019) is widely used in ecology and evolution (Lai et al. 2019) and the `taxize` package (Chamberlain and Szöcs 2013) has become a popular way for R users to interact with naming providers and name resolution services. `taxize` implements bindings to the web APIs (Application Programming Interface) hosted by many popular taxonomic name providers. Unfortunately, this means that functions in the `taxize` are impacted by several major drawbacks that are inherent in the implementation of these central API servers, such as:

- Queries require internet access at all times
- Queries are slow and inefficient to implement and perform; frequently requiring separate API calls for each taxonomic name

- The type of query is highly limited by the API design. For instance, it is usually impossible to make queries across the entire corpus of names, such as "which accepted name has the most known synonyms?"
- Both query formats and responses differ substantially across different naming providers, making it difficult to apply a script designed for one provider to different provider.
- Most queries are not reproducible, as the results depend on the state of the central server (and potentially the quality of the internet connection). Many names providers update the server data either continuously or at regular intervals, including both revising existing names and adding new names.

Instead of binding existing web APIs, `taxadb` is built around a set of compressed text files following a consistent, standardized layout or schema following the Darwin Core Standard (Wieczorek et al. 2012). These files are automatically downloaded and imported and stored on a local database by `taxadb`. The largest of the taxonomic naming providers today contain under 6 million name records with uncompressed file sizes under a GB, and can be compressed to around 50 MB and downloaded in under a minute on a 1 MB/s connection. In contrast, querying a single name over the web API, requiring the server to respond, execute the query, and serialize the response, can take several seconds. Thus it does not take many taxa before transferring the entire data set to query locally is more efficient. Moreover, this local copy can be cached on the user's machine, requiring only the one-time setup, and enabling offline use and reproducible queries.

```r
library(tidyverse)
library(taxadb)
```

`taxadb` functions will automatically set up a local, persistent database if one has not yet been created.

```r
get_ids("Gadus morhua")
```

```
## [1] "ITIS:164712"
```

This one-time setup will download, uncompress, and import the data into persistant storage (using the appropriate location specified by the operating sytstem (see Ratnakumar, Mick, and Davis 2016), or configured using the environmental variable `TAXADB_HOME`). The example above searches for names in ITIS, the default provider, which can be configured using the `provider` argument. Any future function calls to this function or any other function using data from the same names provider will be able to access this data rapidly without the need for processing or an internet connection.

Users can also explicitly trigger this one-time setup using `td_create()` and specifying the provider abbreviation (see Table 1), or simply using `all` to install all available providers:

```r
td_create("all")
```

`taxadb` functions like `get_ids()` and `td_create()` take an optional argument, `db`, to an external database connection. `taxadb` will work with most DBI-compliant databases such as MySQL or Postgres (see **???**), but will be much faster when using a column-oriented database engine such as `duckdb` or `MonetDBLite`. These latter options are also much easier for most users, since each can be installed directly as an R package. `taxadb` will default to the fastest available option. `taxadb` can also run without a database backend by setting `db=NULL`, though some functions will require a lot (2-20 GB) of free RAM for this to work with many of the large names providers.

By default `taxadb` creates a MonetDBLite database instance (Raasveldt and Mühleisen 2018), a columnar-oriented relational database requiring no additional installation while also providing persistent disk-based storage. `taxadb` will automatically handle opening, caching, and closing the database connection.

Functions in `taxadb` are organized into several families:

- queries that return vectors: `get_ids()` and it's complement, `get_names()`,

- queries that filter the underlying taxonomic data frames: `by_name()`, `by_rank()`, `by_id()`, and `by_common()`,
- database functions `td_create()`, `td_connect()` and `taxa_tbl()`,
- and helper utilities, such as `clean_names()`.

Taxonomic IDs

Taxonomic identifiers provide a fundamental abstraction which lies at the heart of managing taxonomic names. For instance, by resolving scientific names to identifiers, we can idetnify which names are synonyms – different scientific names used to describe the same species – and which names are not recognized. Unmatched names may indicate an error in data entry or otherwise warrant further investigation. Taxon identifiers are also easily resolved to the original authority (scientific publication) establishing the name. (The historical practice of appending an author and year to a scientific name, e.g. *Poa annua ssp. annua* (Smith 1912), serves a valuable role in disambiguating different uses of the same name but can be notoriously harder to resolve to the appropriate reference, while variation in this convention creates many distinct versions of the same name (Patterson et al. 2010)).

These issues are best illustrated using a minimal example. We'll consider the task of combining data on bird extinction risk as assessed by the IUCN (International Union for Conservation of Nature and Natural Resources 2019) with data on average adult biomass, as estimated in the Elton Traits v1.0 database (Wilman et al. 2016.) To keep the example concise enough for for visual presentation we will focus on a subset involving just 10 species:

```r
trait_data <- read_tsv(system.file("extdata", "trait_data.tsv", package="taxadb"))
status_data <- read_tsv(system.file("extdata", "status_data.tsv", package="taxadb"))
```

redlist_name	category
Pipile pipile	CR
Pipile cumanensis	LC
Pipile cujubi	LC
Pipile jacutinga	EN
Megapodius decollatus	LC
Scleroptila gutturalis	LC
Margaroperdix madagarensis	LC
Falcipennis falcipennis	NT

elton_name	mass
Aburria pipile	1816.59
Aburria cumanensis	1239.22
Aburria cujubi	1195.82
Aburria jacutinga	1240.96
Megapodius reinwardt	666.34
Francolinus levalliantoides	376.69
Margaroperdix madagascariensis	245.00
Catreus wallichii	1436.88
Falcipennis falcipennis	685.61
Falcipennis canadensis	473.65

If we attempted to join these data directly on the species names provided by each table, we would find very little overlap, with only one species having both a body mass and an IUCN threat status resolved:

```r
joined <- full_join(trait_data, status_data, by = c("elton_name" = "redlist_name"))
```

elton_name	mass	category
Aburria pipile	1816.59	-
Aburria cumanensis	1239.22	-
Aburria cujubi	1195.82	-
Aburria jacutinga	1240.96	-
Megapodius reinwardt	666.34	-
Francolinus levalliantoides	376.69	-
Margaroperdix madagascariensis	245.00	-
Catreus wallichii	1436.88	-
Falcipennis falcipennis	685.61	NT
Falcipennis canadensis	473.65	-
Pipile pipile	NA	CR
Pipile cumanensis	NA	LC
Pipile cujubi	NA	LC
Pipile jacutinga	NA	EN
Megapodius decollatus	NA	LC
Scleroptila gutturalis	NA	LC
Margaroperdix madagarensis	NA	LC

If we first resolve names used in each data set into shared identifiers, (for instance, using the Catalogue of Life), we discover that there is far more overlap in the species coverage than we might have realized at first. First, we just add an ID column to each table by looking up the Catalog of Life identifier for the name provided:

```r
traits <- trait_data %>% mutate(id = get_ids(elton_name, "col"))
status <- status_data %>% mutate(id = get_ids(redlist_name, "col"))
```

We can now join on the `id` column instead of names directly:

```r
joined <- full_join(traits, status, by = "id")
```

elton_name	redlist_name	mass	category	id
Aburria pipile	Pipile pipile	1816.59	CR	COL:35517887
Aburria cumanensis	Pipile cumanensis	1239.22	LC	COL:35537158
Aburria cujubi	Pipile cujubi	1195.82	LC	COL:35537159
Aburria jacutinga	Pipile jacutinga	1240.96	EN	COL:35517886
Megapodius reinwardt	-	666.34	-	COL:35521309
Francolinus levalliantoides	-	376.69	-	COL:35518087
Margaroperdix madagascariensis	Margaroperdix madagarensis	245.00	LC	COL:35521355
Catreus wallichii	-	1436.88	-	COL:35518185
Falcipennis falcipennis	Falcipennis falcipennis	685.61	NT	COL:35521380
Falcipennis canadensis	-	473.65	-	COL:35521381
-	Megapodius decollatus	NA	LC	COL:35537166
-	Scleroptila gutturalis	NA	LC	NA

This results in many more matches, as different scientific names are recognized by the naming provider (Catalog of Life 2018 in this case), as *synonyms* for the same species, and thus resolve to the same taxonomic identifier. While we have focused on a small example for visual clarity here, the `get_ids()` function in `taxadb` can quickly resolve hundreds of thousands of species names to unique identifiers, thanks to the performance of fast joins in a local MonetDBLite database.

Box 1: Taxonomic Identifiers and Synonyms

`get_ids()` returns the `acceptedNameUsageID`, the identifier associated with the *accepted* name. Some naming providers, such as ITIS and NCBI, provide taxonomic identifiers to both synonyms and accepted names. Other providers, such as COL and GBIF, only provide identifiers for accepted names. Common practice in Darwin Core archives is to provide an `acceptedNameUsageID` only for names which are synonyms, and otherwise to provide a `taxonID`. For accepted names, the `acceptedNameUsageID` is then given as missing (`NA`), while for synonyms, the `taxonID` may be missing (`NA`). In contrast, `taxadb` lists the `acceptedNameUsageID` for accepted names (where it matches the `taxonID`), as well as known synonyms. This is semantically identical, but also more convenient for database interfaces, since it allows a name to mapped to it's accepted identifier (or an identifier to map to it's accepted name usage) without the additional logic. For consistency, we will use the term "identifier" to mean the `acceptedNameUsageID` rather than the more ambiguous `taxonID` (which is undefined for synonyms listed by many providers), unless explicitly stated otherwise.

Unresolved names

`get_ids` offers a first pass at matching scientifc names to id, but names may remain unresolved for a number of reasons. First, a name may match to multiple accepted names, as in the case of a species that has been split. By design, these cases are left to be resolved by the researcher using the 'by_' functions to filter underlying taxonomic tables for additional information. A name may also be unresolved due to typos or improper formatting. `clean_names` addresses common formatting issues such as the inclusion of missing species epithets (e.g. `Accipiter sp.`) that prevent matches to the Genus, or intraspecific epithets such as `Colaptes auratus cafer` that prevent matches to the bionmial name. These modifications are not appropriate in all settings and should be used with care (Language from vignette).

Names may also have an ambiguous resolution wherein a name may be resolved by a different provider than the one specified, either as an accepted name or a synonym. Mapping between providers represent a meaningful scientific statement requiring an understanding of the underlying taxonomic concepts of each provider. The spirit of taxadb is not to automate steps that require expert knowledge, but provide access to multiple potential "taxonomic theories".

`by_` functions for access to underlying tables

Underlying data tables can be accessed through the family of `by_` functions, which filter by certain attributes such as scientific name, id, common name, and rank. These functions allow us to ask general questions such as, how many bird species are there?

```r
by_rank("Aves", rank="class", provider = "col") %>%
  pull(taxonID) %>%
  n_distinct()
```

```
## [1] 32327
```

As well getting a detailed look at specific species or ids. For example, why can't `get_ids` resolve a seemingly common species?

```r
## example of sci name matching to more than one id.
by_name("Zosterops rendovae") %>%
  select(1:5)
```

sort	taxonID	scientificName	taxonRank	acceptedNameUsageID
1	NA	NA	NA	NA

We see that *Zosterops rendovae* matches two accepted names which the user will have to choose between.

The full taxonomic record in the MonetDBLite database can also be directly accessed by `taxa_tbl()`, allowing for whole-database queries that are not possible through the API or web interface of many providers (see Vignette for further details).

Box 2: Common Names

`taxizedb` can also resolve common names to their identifier by mapping common name to the accepted scientific name. Common names come with many of the same problems as scientific names but often more frequent, e.g. matching to more than one accepted name, non-standardized formatting (like capitalization) and spelling etc. Common names are accessed via `by_common` which takes a vector of common names. The user can then resolve discrepencies.

Supported providers

Table version of below - citation, number of unique identifier

- the Integrated Taxonomic Information System (ITIS; ITIS 2019), original formed to standardize taxonomic name usage across many agencies in the United States federal government,
- the National Center for Biological Information's (NCBI) Taxonomy database, (Biotechnology Information 2019)
- Catalogue of Life (COL) Annual Species list. (2018 version at present)
- Global Biodiversity Information Facility Taxonomic Backbone (GBIF; **???**).

- FishBase (**???**) taxonomic names
- SeaLifeBase (**???**) taxonomic names
- Wikidata (**???**)
- Open Tree Taxonomy, OTT
- International Union for Conservation of Nature and Natural Resources (IUCN) Red list International Union for Conservation of Nature and Natural Resources (2019)

provider	abbreviation	total_identifiers
Integrated Taxonomic Information System	itis	580304
National Center for Biological Information's Taxonomy database	ncbi	146153
Catalogue of Life	col	1875378
Global Biodiversity Information Facility Taxonomic Backbone	gbif	1666859
FishBase	fb	33125
Wikidata	wd	9204
Open Tree Taxonomy	ott	3516484
International Union for Conservation of Nature and Natural Resources	iucn	12583

Discussion

Taxonomic indentifiers are a powerful tool for maintaing taxonomic consistency, an essential task for a wide variety of applications. Despite multiple taxonomic standardization efforts, resolving names to taxonomic identifiers is often not a standard step in the research work flow due to difficulty in accessing providers and the time consuming API queries necessary fo resolving even moderately sized data sets. `taxadb` fills an important gap between existing tools and typical research patterns by providing a fast, reproducible approach for matching names to taxonomic identifiers. `taxadb` is not intended as an improvement or replacement for any existing approaches to taxonomic name resolution.In particular, `taxadb` is not a replacement for the APIs or databases provided, but merely an interface to taxonomic naming information contained within that data.

`taxadb` establishes a local database of provider allowing for consistent and reproducible queries. The set of functions operate consistently accross providers returning data in a standardized format.

References

Biotechnology Information, National Center for. 2019. “NCBI Taxonomy.” 8600 Rockville Pike, Bethesda MD, 20894 USA: U.S. National Library of Medicine. https://www.itis.gov.

Chamberlain, Scott A., and Eduard Szöcs. 2013. “Taxize: Taxonomic Search and Retrieval in R.” *F1000Research* 2 (October): 191. https://doi.org/10.12688/f1000research.2-191.v2.

International Union for Conservation of Nature and Natural Resources. 2019. “The IUCN Red List of Threatened Species.” https://www.iucnredlist.org.

ITIS. 2019. “Integrated Taxonomic Information System.” https://www.itis.gov.

Lai, Jiangshan, Christopher J. Lortie, Robert A. Muenchen, Jian Yang, and Keping Ma. 2019. “Evaluating the Popularity of R in Ecology.” *Ecosphere* 10 (1): e02567. https://doi.org/10.1002/ecs2.2567.

Patterson, D.J., J. Cooper, P.M. Kirk, R.L. Pyle, and D.P. Remsen. 2010. “Names Are Key to the Big New Biology.” *Trends in Ecology & Evolution* 25 (12): 686–91. https://doi.org/10.1016/j.tree.2010.09.004.

Raasveldt, Mark, and Hannes Mühleisen. 2018. “MonetDBLite : An Embedded Analytical Database.” In *Proceedings of Cikm 2018 International Conference on Information and Knowledge Management (Cikm’18)*. New York, New York, USA: ACM. https://doi.org/10.475/123_4.

Ratnakumar, Sridhar, Trent Mick, and Trevor Davis. 2016. *Rappdirs: Application Directories: Determine Where to Save Data, Caches, and Logs.* https://CRAN.R-project.org/package=rappdirs.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Roskov Y., Orrell T., Abucay L. 2018. “Species 2000 & ITIS Catalogue of Life, 2018 Annual Checklist.” Leiden, the Netherlands.: Species 2000: Naturalis. www.catalogueoflife.org/annual-checklist/2018.

Wieczorek, John, David Bloom, Robert Guralnick, Stan Blum, Markus Döring, Renato Giovanni, Tim Robertson, and David Vieglais. 2012. “Darwin Core: An Evolving Community-Developed Biodiversity Data Standard.” *PLoS ONE* 7 (1). https://doi.org/10.1371/journal.pone.0029715.

Wilman, Hamish, Jonathan Belmaker, Jennifer Simpson, Carolina De La Rosa, Marcelo M. Rivadeneira, and Walter Jetz. 2016. “EltonTraits 1.0: Species-Level Foraging Attributes of the World’s Birds and Mammals.” Figshare. https://doi.org/10.6084/m9.figshare.c.3306933.v1.