taxadb: A High-Performance Local Taxonomic Database Interface

Kari Norman[a], Jorrit Poelen[1], Scott Chamberlain[c], Carl Boettiger[a,1]

[a] *Dept of Environmental Science, Policy, and Management, University of California Berkeley, Berkeley CA 94720-3114, USA*
[b] *Independent consultant, Oakland, CA, USA*
[c] *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; **???**; **???**), 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, (**???**), the Catalogue of Life (COL; **???**), 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 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 impossible 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 compressed text files following a consistent, standardized layout or schema (discussed below). 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.

After installing the `taxadb` R package, users can create local copies of data from any of the providers using the `td_create()` function and specifying the provider abbreviation (see Table 1), or `all` to install all available providers:

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

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

This one-time download will download and import local copies of the provider data. 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, and data is stored in the appropriate location specified for applications by the operating sytstem (Ratnakumar, Mick, and Davis 2016). Users can customize this location, or opt for any alternative relational database backend by providing alternative arguments, though the default MonetDBLite system can be an order of magnitude faster than popular alternatives such as Postgres (Raasveldt and Mühleisen 2018). `taxadb` will automatically handle opening, caching, and closing the MonetDBLite connection, but note that one limitation of MonetDBLite is the restriction against concurrent access: two separate R sessions on the same machine cannot both access the database at the same time. Users who need concurrent access from multiple sessions may wish to use a standalone MonetDB server or other database server instead.

All `taxadb` functions can also operate in one-time-use mode without first installing a standalone server. If a function requests a provider which has not been installed, the relevant table can be downloaded and read directly into memory. This requires sufficient memory be available to hold the entire table (typically 1-2 GB) and will only be cached for the R session. This can be useful in special cases such as execution on remote servers where this will be faster than importing the data locally, or to allow other packages to use methods such as `get_ids()` internally without assuming users will have first created a local copy with `td_create()`. Specify `db=NULL` on any `taxadb` function to force this in-memory dispatch.

Functions in `taxadb` are organized into several families:

- database functions `td_create()`, `td_connect()` and `taxa_tbl()`
- 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()`,
- and helper utilities, such as `clean_names()`.

Returning 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 area 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 consise enough for for visual presentation we will focus on a subset involving just 10 species:

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

```
status_data
```

name_B	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

```
trait_data
```

name_A	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:

```
full_join(trait_data, status_data, by = c("name_A" = "name_B"))
```

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

```
traits <- trait_data %>% mutate(id = get_ids(name_A, "col"))
status <- status_data %>% mutate(id = get_ids(name_B, "col"))

full_join(traits, status, by = "id") %>%
  select(id, name_A, mass, category, name_B) # just reordering
```

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

The `get_ids()` function in `taxadb` can quickly resolve thousands of species names to unique identifiers.

Box: `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.

Notice that one species name could not be resolved to a COL identifier. Perhaps we can resolve this name against a different authority? As it has an IUCN classification already, the IUCN naming provider seems a natural place to start. We can use the `synonyms()` table to attempt to identify any other scientific names by which this species is known:

```
synonyms("Scleroptila gutturalis", "iucn")
```

acceptedNameUsage	synonym	taxonRank	acceptedNameUsageID
Scleroptila gutturalis	Francolinus gutturalis	species	IUCN:22678790
Scleroptila gutturalis	Francolinus levaillantoides	species	IUCN:22678790

Note that one of these synonyms is recognized in COL, and in fact matches a COL identifier of *Francolinus levalliantoides*, 376.69 grams, in our data set.

```
synonyms("Scleroptila gutturalis", "iucn") %>%
  pull(synonym) %>%
  get_ids("col")
```

```
## [1] NA             "COL:35518087"
```

Synonyms

Walking synonyms between providers should be done with care. An accepted name can have many synonyms:

```
most_synonyms <- taxa_tbl("col") %>% count(acceptedNameUsageID, sort = TRUE)
most_synonyms
```

```
## # Source:     lazy query [?? x 2]
## # Database:   MonetDBEmbeddedConnection
## # Ordered by: desc(n)
##    acceptedNameUsageID     n
##    <chr>                <dbl>
##  1 COL:43082445           456
##  2 COL:43081989           373
##  3 COL:43124375           329
##  4 COL:43353659           328
##  5 COL:43223150           322
##  6 COL:43337824           307
```

```
##  7 COL:43124158          302
##  8 COL:43081973          296
##  9 COL:43333057          253
## 10 COL:23162697          252
## # ... with more rows
```

Here we see that some accepted identifiers, such as `COL:43082445`, are known by as many as 456 different synonyms! `get_ids()` function seamlessly handles the resolution of all synonyms to their accepted name usage identifier, since we can unambiguously map from many-to-one.

This query uses the `taxa_tbl()` function to directly access the full taxonomic record in the MonetDBLite database. This approach is fast and powerful, but note that we are restricted to using `dplyr` functions, which can be directly translated into SQL, for such queries to work successfully. Though the result looks like a data frame, most base R operations will not work as expected:

```
most_synonyms$n
```

```
## NULL
```

Such queries are also not possible through the API or web interface to Catalogue of Life. To use custom R functions or functions from other packages we must first call the `dplyr` function `collect()` to return the table into R's memory. This should be done with care as an unfiltered table may require several gigabytes of memory to load.

Some synonyms may also be associated with more than one unique identifier. This situation is more difficult, as it is impossible to know which accepted name should be used for such synonyms without more context. The `get_ids()` function will thus return `NA` for any name that matches more than one identifier, just as it will return `NA` for a name that does not match any identifier at all. Other functions in `taxadb` allow us to explore these names further. We have already seen this strategy above

Having access to a local database makes it easy to explore such issues as "identify all synonyms that resolve to more than one accepted scientific name ID", like so:

```
most_ambiguous <-
taxa_tbl("col") %>%
  filter(taxonomicStatus != "accepted") %>%
  select(scientificName, acceptedNameUsageID) %>%
  distinct() %>%
  count(scientificName, sort=TRUE) %>%
  filter(n > 1)
most_ambiguous
```

```
## # Source:     lazy query [?? x 2]
## # Database:   MonetDBEmbeddedConnection
## # Ordered by: desc(n)
##    scientificName             n
##    <chr>                  <dbl>
##  1 Mabuya bistriata          32
##  2 Mabuya sloanii            29
##  3 Zygaena confluens         27
##  4 Mabuya mabouia            24
##  5 Mabuya mabouya mabouya    21
##  6 Mabuya mabouya            19
##  7 Zygaena cingulata         18
##  8 Tilapia melanopleura      17
```

```
##  9 Phimenes flavopictum      16
## 10 Carabus viridis           15
## # ... with more rows
```

Here we see the name *Mabuya bistriata* is a synonym which may be associated with no fewer than 32 different accepted scientific names!

For many applications, it is easier to work with the `by_*` class of functions, that first perform the relevant filtering join in the MonetDBLite database before returning the table in memory. This means that the `data.frame` objects returned by these functions behave as familiar objects. For instance, we can take the top 5 ambiguous names and see all the taxa to which those names may refer:

```
top_5 <- most_ambiguous %>%
  head(5) %>%
  pull(scientificName)


by_name(top_5, "col") %>%
  select(scientificName, acceptedNameUsageID) %>%
  distinct() %>%
  mutate(acceptedNameUsage = get_names(acceptedNameUsageID, "col")) %>%
  head(5)
```

scientificName	acceptedNameUsageID	acceptedNameUsage
Mabuya bistriata	COL:28889395	Capitellum mariagalantae
Mabuya bistriata	COL:28900904	Spondylurus caicosae
Mabuya bistriata	COL:28900903	Spondylurus anegadae
Mabuya bistriata	COL:28900176	Spondylurus turksae
Mabuya bistriata	COL:28900175	Spondylurus spilonotus

```
by_rank("Aves", rank="class", provider = "col") %>% head(5)
```

taxonID	scientificName	acceptedNameUsageID	taxonomicStatus	taxonRank	kingdom	phy
COL:35516814	Struthio camelus	COL:35516814	accepted	species	Animalia	Ch
COL:35516815	Rhea americana	COL:35516815	accepted	species	Animalia	Ch
COL:35516817	Dromaius novaehollandiae	COL:35516817	accepted	species	Animalia	Ch
COL:35516818	Casuarius bennetti	COL:35516818	accepted	species	Animalia	Ch
COL:35516819	Casuarius unappendiculatus	COL:35516819	accepted	species	Animalia	Ch

Darwin Core schema.

Supported providers

- 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, (**???**)
- Catalogue of Life annual Species list

Discussion

- Resolve names to identifiers

- Resolve synonyms to accepted names
- Associate names with taxonomic rank and heirarchical classification

discuss use Scientific names vs identifiers. taxonConcepts?

- `taxadb` is not intended as an improvement or replacement for any existing approaches to taxonomic name resolution. Instead, it fills an important gap between existing tools and typical research patterns. 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` works from versioned snapshots of the data providers.
- All `taxadb` functions are specific to a provider, while operating consistently across different providers.
- `taxadb` does not attempt to present any unified taxonomic backbone or synthesis from existing naming providers, or make any assertions, numerical scores or other inferences about the data or matches to the data – it is merely a tool for accessing this infromation. This contrasts from other approaches such as the Global Names Resolver, which

There is an important objection to this approach that must also be addressed.
Franz and Sterner (2018)

> no self-respecting biodiversity researcher would or should trust aggregated data blindly, that indeed careful data cleaning is almost always necessary and expected to render the downloaded data fit for purpose, and that therefore aggregator services need to be understood mainly or merely as data discovery tools.
>
> We reject this deflationary view for four reasons. For one, aggregators frequently blur the lines between advertizing their services just as a data discovery tool or as a more powerful data signal tool. Second, the biases inherent in using unitary backbones remain in place even if users are only interested in discovering all relevant data for their research purpose. If the backbone-based data record modulations are not easily retrievable through primary on-line interfaces, then users are significantly constrained in their ability to design search queries with high rates of precision and recall (81). Third, let us assume that labor-intensive off-line data quality review and correction efforts are indeed the norm, prior to publishing. Then why must the fruits of these efforts remain outside of the aggregator's environment? Why can they not immediately flow back into the same aggregation domain, while recording the provenance of expert changes? In other words, if the workflow of rendering data fit for purpose flows only in one direction, i.e. from the on-line aggregate to the off-line quality review and publication, then our criticism of the design stands.

The inability to make automate use and manipulation
(Figure 1: schematic example of data join?)
Assemble data across multiple studies
How many species of Chameleons are known? There are over

```
taxa_tbl("col") %>% count(taxonomicStatus, sort = TRUE)
```

```
## # Source:     lazy query [?? x 2]
## # Database:   MonetDBEmbeddedConnection
## # Ordered by: desc(n)
##   taxonomicStatus         n
##   <chr>               <dbl>
## 1 accepted          1873640
## 2 synonym           1598290
## 3 ambiguous synonym   28410
## 4 misapplied name     11820
```

The extent of mismatch between providers can be startling. Catalogue of Life uses ITIS data as one of it's many input sources, and yet we can quickly calculate how many of the (2018) ITIS accepted species names are not in the Catalogue of Life records:

```
itis_col <-
taxa_tbl("itis") %>%
  filter(taxonomicStatus == "accepted", taxonRank == "species") %>%
  select(scientificName) %>%
  left_join(taxa_tbl("col"), by="scientificName") %>%
  filter(is.na(acceptedNameUsageID)) %>%
  count() %>%
  pull(n)


itis_accepted <- taxa_tbl("itis") %>%
  filter(taxonomicStatus == "accepted", taxonRank == "species") %>%
  count() %>% pull(n)
```

7.1341×10^4 of the 4.39589×10^5 of *accepted names* in ITIS (16.229023%) are not recognized by the (2018) Catalogue of Life as either accepted names or known synonyms.

References

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.

Franz, Nico M, and Beckett W Sterner. 2018. "To Increase Trust, Change the Social Design Behind Aggregated Biodiversity Data." *Database* 2018 (January). https://doi.org/10.1093/database/bax100.

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/.

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.