Dealing with taxonomic inconsistencies within and across datasets is a fundamental challenge of ecology and evolutionary biology. Accounting for species synonyms, taxa splitting and unification is especially important as aggregation of data across time and different data sources becomes increasingly common. One potentially powerful approach for addressing these issues is to resolve scientific names to taxonomic identifiers that follow a consistent taxonomic concept. In such a workflow, data from one of the many taxonomic providers (e.g. Integrated Taxonomic Information System [1], Catalogue of Life [2], National Center for Biological Information [3]) is integrated with biodiversity datasets to identify an accepted ID for each name. Multiple tools exist to facilitate this workflow, including R's taxize package [4], which provides an API interface to taxonomic databases. However, due to the nature of API queries which are slow, limited in scope, and dependent on the current state of the database, it remains difficult to resolve names to a taxonomic authority in quick, reproducible way. taxadb seeks to address these issues using a new approach for interfacing with taxonomic data via a local database of taxonomic providers.

The goal of this post is to illustrate the ease with which taxadb can be integrated into existing data munging workflows, as well as give a taste for the variety of other exploratory question that are facilitated by the database backend infrastructure.

Database backend

taxadb is built around a local database of taxonomic data from seven of the largest taxonomic providers. The tables of this database are standardized across providers and include information on accepted ID's, synonym mappings, and common names when available. The database is accessible by the user through a variety of database backends. Using a local database interface allows not only for quick queries to retrieve taxon ID's, but also queries across the whole-database. As taxonomic providers are constantly updating their data, databases will be time stamped and archived allowing for user selection of the desired release for reproducible results.

taxadb framework

taxadb has three main families of functions:

- queries that return vectors: `get_ids()` and it's complement, `get_names()`,
- queries that filter the underlying taxonomic data frames: `filter_name()`, `filter_rank()`, `filter_id()`, and `filter_common()`,
- database functions `td_create()`, `td_connect()` and `taxa_tbl()`

Query functions will trigger the automatic one-time set up of the local database for the chosen provider, but set up can also be triggered manually by `td_create()` for one or all providers.

taxadb workflow

taxadb is designed for relatively painless local database setup and easy integration of taxonomic ID's into existing workflows. For example, the common scenario of merging two different datasets with their own taxonomic approaches, such as matching trait data to data on IUCN status. Here we use snippets of data from the Elton Traits v1.0 database [5] and the IUCN Redlist [6].

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

iucn_name	category
Pipile pipile	CR
Pipile cumanensis	LC
Pipile cujubi	LC
Pipile jacutinga	EN
Megapodius decollatus	LC

iucn_name	category
Scleroptila gutturalis	LC
Margaroperdix madagarensis	LC
Falcipennis falcipennis	NT

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

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

The common approach in this scenario is to simply join by scientific name:

```
joined <- full_join(trait_data, status_data, by = c("elton_name" = "iucn_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	—	CR
Pipile cumanensis	—	LC
Pipile cujubi	—	LC
Pipile jacutinga	—	EN
Megapodius decollatus	—	LC
Scleroptila gutturalis	—	LC
Margaroperdix madagarensis	—	LC

This results in only one match between the two datasets, *Falcipennis falcipennis*. However, if we resolve names first to taxonomic identifiers, which account for synonyms and taxonomic changes, we see a different story.

First we get ID's for each dataset:

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

And join on the ID:

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

elton_name	iucn_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	—	LC	COL:35537166
—	Scleroptila gutturalis	—	LC	—

Now we see that there are many more matches between the datasets than we previously thought. In a workflow without taxonomic identifiers resolving these additional matches would require a significant investment of time as each name would need to be double checked and matched manually.

Database facilitated questions

The local database structure also allows us to ask general questions of the entire database, both across providers or across tables for one provider, that are not possible with the API interface. For example, which provider would be able to resolve the largest number of species names in our dataset?

```
provider_counts <- trait_data %>%
  select(elton_name) %>%
  mutate(
    gbif = get_ids(elton_name, "gbif"),
    col = get_ids(elton_name, "col"),
    itis = get_ids(elton_name, "itis"),
    ncbi = get_ids(elton_name, "ncbi"),
    wd = get_ids(elton_name, "wd"),
    iucn = get_ids(elton_name, "iucn"),
    ott = get_ids(elton_name, "ott")
  ) %>%
  purrr::map_dbl(function(x)
    sum(!is.na(x))) %>%
  tibble::enframe("provider", "ID_count")
```

provider	ID_count
gbif	10
col	10
itis	10
ncbi	1
wd	4
iucn	0
ott	10

Or even more generally which bird families have the most species?

```
bird_families <- filter_rank(name = "Aves", rank = "class", provider = "col")
%>%
  filter(taxonomicStatus == "accepted", taxonRank=="species") %>%
```

```
  group_by(family) %>%
  count(sort = TRUE) %>%
  head()
```

family	n
Tyrannidae	401
Thraupidae	374
Psittacidae	370
Trochilidae	338
Muscicapidae	314
Columbidae	312

And which species has the most synonyms?

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

acceptedNameUsageID	n
COL:43082445	456
COL:43081989	373
COL:43124375	329
COL:43353659	328
COL:43223150	322
COL:43337824	307

For the provider Catalogue of Life it is COL:43082445, or the mint species *Mentha longifolia*.

In addition to facilitating quick and easy incorporation of taxonomic identifiers into standard research workflows, taxadb provides direct access to the underlying database of taxonomic providers. Users can therefore use familiar syntax to ask important exploratory questions of the providers rather than being dependent upon the kinds of queries allowed by an API. By providing both a simple interface to ID's and the potential for more in depth exploration we hope to encourage improved inclusion and understanding of taxonomic data by the biodiversity community.