taxize - taxonomic search and retrieval in R

Scott Chamberlain[1,*] and Eduard Szöcs[2,†]

[1] *Biology Department, Simon Fraser University, Canada.*
[2] *Institute for Environmental Sciences, University Koblenz-Landau, Forststraße 7, 76829 Landau, Germany*

Keywords: taxonomy; R; software; data; API

* E-mail: myrmecocystus@gmail.com
† E-mail: szoe8822@uni-landau.de

I. ABSTRACT

All species are hierarchically related to one another, and we use taxonomic names to label the nodes in this hierarchy. Taxonomic data is becoming easily available on the web, but scientists need a way to access taxonomic data on the web in a programmatic fashion that's easy and reproducible. We have developed taxize, an open-source software package (freely available from http://cran.r-project.org/web/packages/taxize/index.html) for the R language. taxize provides simple, programmatic access to taxonomic data for 13 data sources around the web. We discuss the need for a taxonomic toolbelt in R, and outline a suite of use cases for which taxize is ideally suited (including a full workflow as an appendix). The taxize package will facilitate open and reproducible science by allowing taxonomic data collection to be done in the open-source R platform.

II. INTRODUCTION

Evolution by natural selection has led to a hierarchical relationship among all living organisms. Thus, species are categorized using a taxonomic hierarchy, starting with the binomial species name (e.g, *Homo sapiens*), moving up to genus (*Homo*), then family (*Hominidae*), and on up to Domain (*Eukarya*). Biologists, whether studying organisms at the cell, organismal, or community level, can put their study taxa into taxonomic context, allowing them to know close and distant relatives, find relevant literature, and more. Discovering the correct taxonomic names is, unfortunately, not straightforward. Taxonomic names often change due to name changes at the generic or specific levels, lumping or splitting lower taxa (genera, species) among higher taxa (families), and name spelling changes. In addition, there is no one authoritative taxonomic names source. Instead, there are essentially competing sources (e.g., uBio, Tropicos, ITIS) that may have different accepted names for the same taxon. The goal of taxize, an R package in development, is to make all use cases having to do with retrieving and resolving taxonmic names easy and replicable.

Taxonomic data is getting easier to obtain through web interfaces (e.g., http://eol.org/). However, there are a number of good reasons to obtain taxonomic information programatically rather than through a web interface. First, if you have more than a few names to lookup on a website, it can take quite a long time to enter each name, get data, and repeat for each species. Second, programatically getting taxonomic names solves the first problem by looping over a list of names. In addition, doing taxonomic searching, etc. is reproducible. With increasing reports of irreproducibility in science [1, 2], it is extremely important to make science workflows repeatable. Science workflows can now easily incorporate text, code, and images in a single executable document [**?**].

The R language is the dominant language used by biologists (reference), and now has over 5,000 packages on the R package repository (CRAN) and more than 2,500 packages on other repositories to extend R. R is great for manipulating, visualizing and fitting statistical models to data. However, the key missing piece in R is the ability to get data from the internet within R. Getting data from the web will be increasingly common as more and more data gets moved to the cloud [**?**]. Increasingly, data is available from the web via API's, or application programming interfaces. These are bits of code that allow computers to talk to one another using code that is not human readable, but is machine readable. Web APIs often define a number of methods that allow users to search for a species name, or retrieve the synonyms for a species name, for example. A further strength of APIs is that they are language agnostic, meaning that data can be consumed in almost any computing context, allowing users to interact with the web API without having to know the details of the code. Whereas, if data are stored in an Excel file, for example, the file can only be opened in a few programs.

In taxize, we have written a suite of R functions that interact with many taxonomic data sources via their web APIs (Table I). The interface to each function is usually a simple list of species names, just as a user would do with a web API. Therefore, we hope moving from a web to R interface for taxonomic names will be relatively seamless (if one is already nominally familiar with R).

Here, we justify the need for taxize, discuss our data sources, and run through a suite of use cases to demonstrate the variety of ways that users can interact with taxize.

III. WHY DO WE NEED TAXIZE?

There are a large suite of applications developed around the problem of searching for, resolving, and getting higher taxonomy for species names. For example, **Linnaeus** http://linnaeus.sourceforge.net/ provides the ability to search for taxonomic names in documents and normalize those names found. In addition, there are many web interfaces to search for and normalize names such as Encyclopedia of Life's Global Names Resolver http://resolver.globalnames.org/, uBio tools http://www.ubio.org/index.php?pagename=sample_tools, and iPlant's Taxonomic Name Resolution Service http://tnrs.iplantcollaborative.org/.

All of these tools provide great ways to search for taxonomic names and resolve them in some cases. However, scientists ideally need a tool that can be used programmatically, and thereby facilitate reproducible research. The goal of taxize is to make it easy to create reproducible and easy to use workflows for searching for taxonomic names, resolving them, getting higher taxonomic names, and other tasks related to research dealing with species.

One could argue that a different programming language would have been better than R. For example, Python performs many actions faster than R, and Ruby plays really nicely in a browser, facilitating web applications. However, our goal with taxize is to create a product for researchers primarily, and the most common programming language for researchers, at least in the life sciences, is R. Gentleman *et al.* [3] gives a detailed discussion of advantages of R in computational biology.

IV. DATA SOURCES

taxize uses many data sources (Table I), and more can easily be added. There are two common tasks provided by the data sources: name search and name resolution. Other functionality in taxize includes retrieving a classification tree for a species, or retrieving child taxa of a focal taxon. One of the data sources (Phylomat) returns phylogenies, while another (NCBI) returns genetic sequence data. However, there are other R packages that are focused solely on sequence data, such as rsnps [4], rentrez [5], BoSSA [6], and ape [7], so taxize will not venture deeply into these other domains.

Some of the data sources taxize interacts with require authentication. That is, in addition to the search terms the user provides (e.g., *Homo sapiens*), the data provider requires an alphanumeric identification key so that they can better manage their servers, collect analytics, and shut down users that abuse the API. The services that do require an API key in taxize are: Encyclopedia of Life (EOL), the Universal Biological Indexer and Organizer (uBio), Tropicos, and Plantminer. You can easily obtain an API key by visiting the website of each service (see (Table I) for links to each site). There are two ways of using your API keys. First, you can pass in your API key in a function call (e.g., *ubio_namebank(srchName='Ursus americanus', key='your_alphanumeric_key')*). Second, you can store your API keys in your .Rprofile file. On a Mac this file is at */yourhomefolder/.Rprofile*; on a Windows machine at */yourhomefolder/.Rprofile*; and on Linux at */yourhomefolder/.Rprofile*. This is an hidden file, so open up this file in your terminal (e.g., *open .Rprofile*), and add the API key as an entry like *options(myapikey = 'your_alphanumeric_key')*. We recommend the second option as it simplifies function calls.

One data source available in taxize is available in another R pacakge that solely interact with that data source. We provide a few convenience functions that wrap functions in *taxonstand*, an R interface to The Plant List http://www.theplantlist.org.

Table I. Data sources used in taxize, tasks available, and links to them

Source name	Name search	Name resolution	Phylogeny	Sequences	URL
Encyclopedia of Life	Yes	See GNR below	No	No	http://eol.org/
Integrated Taxonomic Information System	Yes	Synonyms	No	No	http://www.itis.gov/
iPlant Taxonomic Name Resolution Service	Yes	Yes	No	No	http://bit.ly/16dHkBy
Phylomatic	No	No	Yes	No	http://bit.ly/P0pjMz
uBio	Yes	Yes	No	No	http://www.ubio.org/
Global Names Resolver	Yes	Yes	No	No	http://bit.ly/11R3Pbr
Global Names Index	Yes	No	No	No	http://bit.ly/11R3RQB
IUCN Red List	Yes	No	No	No	http://bit.ly/11R3RQC
Tropicos	Yes	Yes	No	No	http://www.tropicos.org/
Plantminer	Yes	No	No	No	http://www.plantminer.com/
The Plant List	Yes	Yes	No	No	http://www.theplantlist.org/
Catalogue of Life	Yes	Yes	No	No	http://bit.ly/11R3S75
National Center for Biotechnology Information	Yes	X	Yes [a]	Yes	http://www.ncbi.nlm.nih.gov/

[a] Web only, no API, see http://1.usa.gov/11R446a

New data sources can be added; we may add the following sources: Wikispecies and The Tree of Life. Get in touch with one of the authors if you would like any data sources added.

V. USE CASES

There are a variety of use cases for which taxize is ideally suited, and a few side cases in which taxize can be useful. We discuss five ideal use cases for taxize at length, and highlight the side cases in brief.

A. Installing taxize

First, we must install taxize. There are two versions of taxize, a) a stable release that can be installed from the R package repository, CRAN, and b) from GitHub https://github.com/ropensci/taxize_, where the code is developed.

Installing from CRAN or GitHub

```
## From CRAN
install.packages("taxize")

## From GitHub
install.packages("devtools")
require(devtools)
install_github("taxize_", "ropensci")
```

Loading taxize into your R session

```
library(taxize)
```

B. Resolve taxonomic names

This is a common task in biology. We often have a list of species names and we want to know if a) we have the most up to date names, b) our names are spelled correctly, and c) if we have common names, we likely need the scientific names. One way to resolve names is via the Global Names Resolver (GNR) service provided by the Encyclopedia of Life (http://resolver.globalnames.org/).

```
# Here, we are searching for two misspelled names
temp <- gnr_resolve(names = c("Helianthos annus", "Homo saapiens"), returndf = TRUE)

# Let's take a peek at the data, excluding the data source ID and score
# columns
temp[, -c(1, 4)]
```

It looks like the correct spellings are *Helianthus annuus* and *Homo sapiens*. Another approach uses the Taxonomic Name Resolution Service via the Taxosaurus API (http://taxosaurus.org/) developed by iPLant and the Phylotastic organization.

```
# A list of species names, some of which are misspelled
mynames <- c("Helianthus annuus", "Pinus contort", "Poa anua", "Abis magnifica",
    "Rosa california", "Festuca arundinace", "Sorbus occidentalos", "Madia sateva")

# And we'll call the API with the tnrs function, and remove a few columns
tnrs(query = mynames)[, -c(5:7)]

        submittedName        acceptedName    sourceId score
7   Helianthus annuus   Helianthus annuus iPlant_TNRS  1.00
4       Pinus contort      Pinus contorta iPlant_TNRS  0.98
5            Poa anua           Poa annua iPlant_TNRS  0.96
3      Abis magnifica     Abies magnifica iPlant_TNRS  0.96
8     Rosa california    Rosa californica iPlant_TNRS  0.99
```

```
2  Festuca arundinace Festuca arundinacea iPlant_TNRS  0.99
1 Sorbus occidentalos Sorbus occidentalis iPlant_TNRS  0.99
6        Madia sateva        Madia sativa iPlant_TNRS  0.97
```

It looks like there are a few corrections: e.g., *Madia sateva* should be *Madia sativa*, and *Rosa californica* should be *Rosa californica*. Note that this search worked because fuzzy matching was employed to retrieve names that were close, but not exact matches. Fuzzy matching is only available for plants in the TNRS service, so we advise using EOL's Global Names Resolver if you need to resolve animal names.

taxize takes the approach that the user should be able to make decisions about what resource to trust, rather than taxize making the decision. Both the EOL GNR and the TNRS services provide data from a variety of data sources. The user may trust a specific data source, thus may want to use the names from that data source. In the future, we may provide the ability for taxize to suggest the best match from a variety of sources, but since R is relatively inefficient in memmory management, etc., we would rather offload this sort of computationally intensive task.

C. Retrieve higher taxonomic names

Another task biologists often face is getting higher taxonomic names for a taxa list. Having the higher taxonomy allows you to put into context the relationships of your species list. For example, you may find out that species A and species B are in Family C, which may lead to some interesting insight, as opposed to not knowing that Species A and B are closely related. This also makes it easy to aggregate/standardize data to a specific taxonomic level (e.g., family level) or to match data to other databases with different taxonomic resolution (e.g., trait databases).

A number of data sources in taxize provide the capability to retrieve higher taxonomic names, but we will highlight two of the more useful ones: ITIS and NCBI. The principle in both is the same - first you need to get a numeric identifier for the queried species, then retrieve the higher taxonomy with this identifier. First, we'll explore the user of ITIS, by searching for two species, *Abies procera* and *Pinus contorta*.

```
specieslist <- c("Abies procera", "Pinus contorta")
classification(get_tsn(specieslist, "sciname"))

[[1]]
        parentName parentTsn      rankName       taxonName    tsn
1                                  Kingdom         Plantae 202422
2          Plantae    202422    Subkingdom  Viridaeplantae 846492
3   Viridaeplantae    846492  Infrakingdom    Streptophyta 846494
4     Streptophyta    846494      Division    Tracheophyta 846496
5     Tracheophyta    846496   Subdivision Spermatophytina 846504
6  Spermatophytina    846504 Infradivision    Gymnospermae 846506
7     Gymnospermae    846506         Class       Pinopsida 500009
8        Pinopsida    500009         Order         Pinales 500028
9          Pinales    500028        Family        Pinaceae  18030
10        Pinaceae     18030         Genus           Abies  18031
11           Abies     18031       Species   Abies procera 181835

[[2]]
        parentName parentTsn      rankName       taxonName    tsn
1                                  Kingdom         Plantae 202422
2          Plantae    202422    Subkingdom  Viridaeplantae 846492
3   Viridaeplantae    846492  Infrakingdom    Streptophyta 846494
4     Streptophyta    846494      Division    Tracheophyta 846496
5     Tracheophyta    846496   Subdivision Spermatophytina 846504
6  Spermatophytina    846504 Infradivision    Gymnospermae 846506
7     Gymnospermae    846506         Class       Pinopsida 500009
8        Pinopsida    500009         Order         Pinales 500028
9          Pinales    500028        Family        Pinaceae  18030
10        Pinaceae     18030         Genus           Pinus  18035
11           Pinus     18035       Species  Pinus contorta 183327
```

It turns out both species are in the family Pinaceae. You can also get this type of information from the NCBI by doing *classification(get_uid(specieslist))*.

Instead of a full classification, you may only want a single name, say a family name for your species of interest. The function *tax_name* is built just for this purpose. And you can specify the data source you retrieve the taxonomic name from with the *db* parameter.

```
# Using ITIS
tax_name(query = "Helianthus annuus", get = "family", db = "itis")

      family
1 Asteraceae


# Using NCBI
tax_name(query = "Helianthus annuus", get = "family", db = "ncbi")

      family
1 Asteraceae
```

D. Interactive name selection

In the previous section we used the function *get_tsn* to get a classification for two plant species. Below are a few examples. When you run these examples in R, you are presented with a command prompt asking for the row that contains the name you would like back; that output is not printed below fro brevity.

```
# In this example, this search term has many matches. The function returns
# a data.frame of the matches, and asks for user input. Enter the row
# number of the name you want.
get_tsn(searchterm = "Elisabethae", searchtype = "sciname")

# You can pass in a long list of taxonomic names
splist <- c("annona cherimola", "annona muricata", "quercus robur", "shorea robusta",
    "pandanus patina", "oryza sativa", "durio zibethinus")
get_tsn(searchterm = splist, searchtype = "sciname")

# In this example, note that no match at all returns an NA
get_uid(sciname = c("Chironomus riparius", "aaa vva"))
```

E. Retrieve a phylogeny

Ecologists are increasingly taking a phylogenetic approach to ecology, applying phylogenies to topics such as the study of community structure [**?**], ecological networks [**?**], functional trait ecology [**?**]. Yet, Many biologists are not adequately trained in reconstructing phylogenies. Fortunately, there are some sources for getting a phylogeny without having to know how to build one; one of these is for angiosperms, called Phylomatic [8]. We have created a workflow in taxize that accepts a species list, and taxize works behind the scenes to get higher taxonomic names, which are required by Phylomatic to get a phylogeny. Here is a short example.

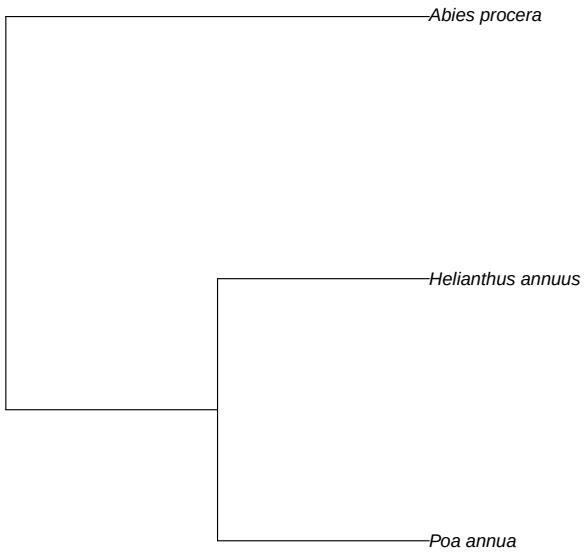
```
# Input the taxonomic names
taxa <- c("Poa annua", "Abies procera", "Helianthus annuus")

# Fetch the tree - the formatting of names and higher taxonmy is done
# within the function
tree <- phylomatic_tree(taxa = taxa, get = "POST", informat = "newick", method = "phylomatic",
    storedtree = "R20120829", taxaformat = "slashpath", outformat = "newick",
```

```
    clean = "true")

# Captilize the species names
tree$tip.label <- capwords(tree$tip.label)

# Plot the tree
plot(tree, cex = 1)
```



Behind the scenes the function *phylomatic_tree* retrieves a Taxonomic Serial Number (TSN) from ITIS for each species name, then a string is created for each species like this *poaceae/oryza/oryza_sativa* (with format 'family/genus/genus_epithet'). These strings are submitted to the Phylomatic API, and if no errors occur, a phylogeny in newick format is returned. The *phylomatic_tree()* function also cleans up the newick string and converts it to an **ape** *phylo* object. The output from *phylomatic_tree()* is a *phylo* object, which can be used for plotting and phylogenetic analyses.

There are currently no resources for getting a phylogeny of animals simply from species names. However, a few projects are working on this problem, including the Open Tree of Life URL. We will incorporate these resources when the appropriate APIs are available.

F. What taxa are the children of my taxon of interest?

If you aren't a taxonomic specialist on a particular taxon you likely don't know what children taxa are within a family, or within a genus. This task becomes especially unwieldy when there are a large number of taxa downstream. You can of course go to a website like Wikispecies (http://species.wikimedia.org/wiki/Main_Page) or Encyclopedia of Life (http://eol.org/) to get downstream names. However, taxize provides an easy way to programatically search for downstream taxa, both for the Catalogue of Life (CoL; http://www.catalogueoflife.org/) and the Integrated Taxonomic Information System (http://www.itis.gov/). Here is a short example using the CoL in which we want to find all the species within the genus *Apis* (honey bees).

```
# Search for all species downstream from Apis
col_downstream(name = "Apis", downto = "Species")[[1]]

  childtaxa_id     childtaxa_name childtaxa_rank
```

```
1      6971712 Apis andreniformis        Species
2      6971713        Apis cerana        Species
3      6971714       Apis dorsata        Species
4      6971715        Apis florea        Species
5      6971716 Apis koschevnikovi        Species
6      6845885     Apis mellifera        Species
7      6971717   Apis nigrocincta        Species
```

The result from the above call to *col_downstream()* is a data.frame that gives a number of columns of different information.

VI. SIDE USE CASES

A. IUCN Status

There are a number of things we can do once we have the correct taxonomic names. One thing we can do is ask about the conservation status of a species. We have provided a set of functions, *iucn_summary* and *iucn_status*, to search for species names, and extract the status information, respectively. Here, we search for the Panther and Lynx.

```
ia <- iucn_summary(c("Panthera uncia", "Lynx lynx"))
iucn_status(ia)

Panthera uncia      Lynx lynx
          "EN"           "LC"
```

It turns out that the panther has a statu of endangered (EN) and the lynx has a status of least concern (LN). (ED, WHAT ELSE DO YOU WANT TO SAY HERE?)

B. Search for available genes in GenBank

Another use case available in taxize deals with genetic sequences. taxize has three functions to interact with GenBank to search for available genes (*get_genes_avail*), download genes by GenBank ID (*get_genes*), and download genes via taxonomic name search, including retrieving a congeneric if the searched taxon does not exist in the databas (*get_seqs*). In this example, we search for gene sequences for *Umbra limi*.

```
# Serch for available genes in GenBank.
out <- get_genes_avail(taxon_name = "Umbra limi", seqrange = "1:2000", getrelated = FALSE)

# Get a list of available genes, removing non-unique records.
# unique(out$genesavail)

# Does the string 'RAG1' exist in any of the gene names?
out[grep("RAG1", out$genesavail, ignore.case = T), ]

        spused length
414 Umbra limi    732
427 Umbra limi    959
434 Umbra limi   1631
                                                                          genesavail
414 isolate UlimA recombinase activating protein 1 (rag1) gene, exon 3 and partial cds
427           recombination-activating protein 1 (RAG1) gene, intron 2 and partial cds
434                         recombination-activating protein 1 (RAG1) gene, partial cds
    access_num       ids
414   JX190826 394772608
427   AY459526  45479841
434   AY380548  38858304
```

It turns out that there are XX different unique records found. However, this doesn't mean that there are XX different genes found as the API does not provide metadata to classify genes. However, at the end of the example, we showed that you can use regular expressions via (e.g., via *grep*) to search for the gene of interest.

VII. CONCLUSION

Taxonomic information is increasingly sought out by biologists as we take phylogenetic and taxonomic approaches to science. Taxonomic data is quickly becoming available on the web, yet scientists require programmatic access to this data to create reproducible workflows. taxize was created to bridge this gap - to bring taxonomic data on the web into R, where the data can be easily manipulated, visualized, and analyzed in a reproducible workflow.

We have outlined a suite of use cases in taxize that will likely fit real use cases of many biologists. Of course we have not thought of all possible use cases, so we hope that the biology community can give us feedback on what use cases they want to see available in taxize. One thing we could change in the future is to make functions that fit use cases, and then allow users to select the data source as a parameter in the function. This could possibly make the user interface easier to understand.

taxize is currently under development and will be for some time given the large number of data sources kitted together in the package, and the fact that APIs for each data source can change, requiring changes in taxize code. Contributions to taxize are strongly encouraged, and can be easily done using GitHub here `http://github.com/ropensci/taxize_`. We hope taxize will be taking up by the community and developed collaboratively, making it progressively better through time as new use cases arise, bug reports are squashed, and pull requests are merged.

VIII. ACKNOWLEDGEMENTS

The taxize package is part of the rOpenSci project `http://ropensci.org/`. We thank X, Y, and Z for comments on previous versions of this manuscript. SAC is supported by CANPOLIN of Canada, grant number XXXXXX.

IX. APPENDICES

- Appendix A. A complete reproducible workflow, from a species list to a phylogeny.

---

[1] Victoria C Stodden, "Reproducible research: Addressing the need for data and code sharing in computational science," Computing in Science & Engineering **12**, 8–12 (2010).

[2] Carl Zimmer, "A sharp rise in retractions prompts calls for reform," New York Times (2012).

[3] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean YH Yang, and Jianhua Zhang, "Bioconductor: open software development for computational biology and bioinformatics," Genome Biology **5**, R80 (2004), PMID: 15461798.

[4] Scott Chamberlain and Kevin Ushey, *rsnps: Interface to SNP data on the web.* (2013), r package version 0.0.4.

[5] David Winter, *rentrez: Entrez in R* (2013), r package version 0.2.1.

[6] Pierre Lefeuvre, *BoSSA: a Bunch of Structure and Sequence Analysis* (2010), r package version 1.2.

[7] E. Paradis, J. Claude, and K. Strimmer, "APE: analyses of phylogenetics and evolution in R language," Bioinformatics **20**, 289–290 (2004).

[8] Campbell O Webb and Michael J Donoghue, "Phylomatic: tree assembly for applied phylogenetics," Molecular Ecology Notes **5**, 181–183 (2005).