SOFTWARE TOOL ARTICLE

REVISED Taxa: An R package implementing data standards and methods for taxonomic data [version 2; referees: 4 approved]

Zachary S.L. Foster[1], Scott Chamberlain[2], Niklaus J. Grünwald[3]

[1]Department of Botany and Plant Pathology, Oregon State University, Corvallis, OR, 97331, USA
[2]rOpenSci, University of California, Berkeley, CA, 94720, USA
[3]Horticultural Crops Research Laboratory, USDA Agricultural Research Service, Corvallis, OR, 97330, USA

v2 **First published:** 05 Mar 2018, **7**:272 (https://doi.org/10.12688/f1000research.14013.1)
Latest published: 11 Sep 2018, **7**:272 (https://doi.org/10.12688/f1000research.14013.2)

Abstract

The taxa R package provides a set of tools for defining and manipulating taxonomic data. The recent and widespread application of DNA sequencing to community composition studies is making large data sets with taxonomic information commonplace. However, compared to typical tabular data, this information is encoded in many different ways and the hierarchical nature of taxonomic classifications makes it difficult to work with. There are many R packages that use taxonomic data to varying degrees but there is currently no cross-package standard for how this information is encoded and manipulated. We developed the R package taxa to provide a robust and flexible solution to storing and manipulating taxonomic data in R and any application-specific information associated with it. Taxa provides parsers that can read common sources of taxonomic information (taxon IDs, sequence IDs, taxon names, and classifications) from nearly any format while preserving associated data. Once parsed, the taxonomic data and any associated data can be manipulated using a cohesive set of functions modeled after the popular R package dplyr. These functions take into account the hierarchical nature of taxa and can modify the taxonomy or associated data in such a way that both are kept in sync. Taxa is currently being used by the metacoder and taxize packages, which provide broadly useful functionality that we hope will speed adoption by users and developers.

Keywords

R language, taxonomy, taxa, R package, rOpenSci, metacoder, taxize

This article is included in the RPackage gateway.

Open Peer Review

Referee Status: ✓ ✓ ✓ ✓

	Invited Referees 1	2	3	4
REVISED version 2 published 11 Sep 2018	✓ report	✓ report		✓ report
version 1 published 05 Mar 2018	✓ report	? report	✓ report	? report

1 **Ethan P. White**, University of Florida, USA
Kristina Riemer, University of Florida, USA

2 **Titus Brown**, University of California, Davis, USA
Taylor Reiter, University of California, USA

3 **Damiano Oldoni**, Research Institute for Nature and Forest (INBO), Belgium
Peter Desmet, Research Institute for Nature and Forest (INBO), Belgium

4 **Holly M. Bik**, University of California, Riverside, USA

Discuss this article

Comments (3)

Corresponding author: Niklaus J. Grünwald (grunwaln@oregonstate.edu)

Author roles: Foster ZSL: Conceptualization, Methodology, Resources, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Chamberlain S**: Conceptualization, Methodology, Project Administration, Resources, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Grünwald NJ**: Conceptualization, Funding Acquisition, Methodology, Project Administration, Resources, Supervision, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: The authors have declared that no competing interests exist. The use of trade, firm, or corporation names in this publication is for the information and convenience of the reader. Such use does not constitute an official endorsement or approval by the United States Department of Agriculture or the Agricultural Research Service of any product or service to the exclusion of others that may be suitable.

Grant information: This work was supported in part by funds from USDA Agricultural Research Service Projects 2027-22000-039-00 and 2072-22000-039-15-S to NG and an rOpenSci grant to ZF.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Foster ZSL, Chamberlain S and Grünwald NJ. **Taxa: An R package implementing data standards and methods for taxonomic data [version 2; referees: 4 approved]** *F1000Research* 2018, **7**:272 (https://doi.org/10.12688/f1000research.14013.2)

First published: 05 Mar 2018, **7**:272 (https://doi.org/10.12688/f1000research.14013.1)

REVISED **Amendments from Version 1**

In this revision we addressed all the comments from the reviewers. Major points addressed include:

- We addressed the fact that various R packages handle taxonomic information differently and provide background on why taxa is useful as a low level taxonomy tool.
- We provide examples of how various databases (e.g., greengenes, SILVA, RDP) handle taxonomic information.
- We provide examples of how the database class handles database associated information using the example of NCBI.
- We provide examples of how the taxon_name, taxon_id and taxon_rank classes handle database associated information using the example of NCBI.
- We provide more detail on the hierarchy's class and taxonomy as a more memory-efficient alternative.
- We explain non-standard evaluation.
- We discuss use of the drop_obs and reassing_obs options.
- We discuss in detail how taxa can be used to implement flexible parsers for other higher level packages.
- We also added some issues to Github that are either resolved or pending future updates where feasible.
- We added reference for the databases included as examples.

See referee reports

Introduction

The R statistical computing language is rapidly becoming the leading tool for scientific data analysis in academic research programs (Tippmann, 2015). One of the reasons for R's popularity is how easy it is to develop and install extensions called R packages, relative to other programming languages. There are now more than 10,000 packages on the Comprehensive R Archive Network (CRAN), over 1,300 packages on Bioconductor (Gentleman *et al.*, 2004), and countless more on GitHub.

The recent increases in the affordability and effectiveness of high-throughput sequencing has led to a large number of ecological datasets of unprecedented size and complexity. The R community has responded with the creation of numerous packages for ecological data analysis and visualization, such as `vegan` (Oksanen *et al.*, 2013), `phyloseq` (McMurdie & Holmes, 2013), `taxize` (Chamberlain & Szöcs, 2013), and `metacoder` (Foster *et al.*, 2017). Taxonomic information is often associated with these large data sets and each package encodes this information differently. Some store taxonomic classification as a table with ranks as columns (e.g. `phyloseq`), some store it as simple character vectors (i.e. plain text) or column/row names, leaving it up to the user to decide on the details on how taxa in the classification are distinguished (e.g. `vegan`), and some store it as a list of tables with one classification in each table (e.g. `taxize`). Since each package tends to have a unique focus, it is common to use multiple packages on the same data set but converting between formats can be difficult. Considering how recently these large taxonomic data sets have become commonplace, it is likely that many more packages that use taxonomic information will be created.

Without a common data standard, using multiple packages with the same data set requires constant reformatting, which complicates analyses and increases the chance of errors. Package maintainers often add functions to convert between the formats of other popular packages, but this practice will become unsustainable as the number of packages dealing with taxonomic data increases. Even if a conversion function exists, doing the conversion can significantly increase the time needed to analyze very large data sets, like those generated by high-throughput sequencing. In addition, not all formats accommodate the same types of information, so conversion can force a loss of information.

The sources of taxonomic data, typically online databases, also vary in how they are encoded. Reference sequence databases used in ecology research often have taxon names in the headers separated by some character, but the details differ. For example, the popular Greengenes database (McDonald *et al.*, 2012) for prokaryotic 16S sequences encodes classifications as follows:

```
k_Bacteria; p_Cyanobacteria; c_Synechococcophycideae...
```

In contrast, the SILVA database (Yilmaz *et al.*, 2014) uses:

```
Bacteria;Proteobacteria;Gammaproteobacteria...
```

And the Ribosomal Database Project (RDP) (Cole *et al.*, 2014) has the ranks and taxon names intermixed with the same separator:

```
Root;rootrank;Fungi;domain;Ascomycota...
```

These minor differences, while not a problem for humans to understand, mean that different code must be used to read each type. Also, this information is often intermixed with other information in the same header, like the sequence ID or description of the organism further complicating parsing. In other cases, a classification might not be supplied at all, but just a taxon name (e.g. *Homo sapiens*), sequence ID, or taxon ID, as is done in sequences downloaded from GenBank:

```
>AC005336.1 Homo sapien chromosome 19
```

In this case the classification must be looked up using tools like the `taxize` package, but to do that the relevant information must be extracted from the rest of the header.

`Taxa` is a new R package that defines classes and functions for storing and manipulating taxonomic data. It is meant to provide a solid foundation on which to build an ecosystem of packages that will be able to interact seamlessly with minimal hassle for developers and users. It also provides highly flexible functions to read in data (i.e. parsers) from diverse formats, allowing it to be used with the ever-changing and proliferating selection of file formats used by biologists. The classes in `taxa`

are designed to be as flexible as possible so they can be used in all cases involving taxonomic information. Complexity ranges from low level classes used to store the names of taxa, ranks, and databases to high-level classes that can store multiple data sets associated with a taxonomy. In particular, the `taxmap` class is designed to hold any type of arbitrary, user-defined data associated with taxonomic information, making its applications limitless. In addition to the classes, there are associated functions for manipulating data based on the `dplyr` philosophy (Wickham & Francois, 2015). These functions provide an intuitive way of filtering and manipulating both taxonomic and user-defined data simultaneously. In combination with flexible parsers and classes, this allows for taxa to be used to subset complicated data/files based on their associated taxonomic information.

Methods

Implementation

The basic classes. `Taxa` defines some basic taxonomic classes and functions to manipulate them (Figure 1). The goal is to use these as low-level building blocks that other R packages can use. The `database` class stores the name of a database and any associated information, such as a description, its URL, and a regular expression matching the format of valid taxon identifiers (IDs):

```
taxon_database(
  name = "ncbi",
  url = "http://www.ncbi.nlm.nih.gov/taxonomy",
  description = "NCBI Taxonomy Database",
  id_regex = "*")
#> <database> ncbi
#>   url: http://www.ncbi.nlm.nih.gov/taxonomy
#>   description: NCBI Taxonomy Database
#>   id regex: *
```

The classes `taxon_name`, `taxon_id`, and `taxon_rank` store the names, IDs, and ranks of taxa and can include a `database` object indicating their source:

```
taxon_name("Poa", database = "ncbi")
#> <TaxonName> Poa
#>   database: ncbi
taxon_rank(name = "species", database = "ncbi")
#> <TaxonRank> species
#>   database: ncbi
taxon_id(12345, database = "ncbi")
#> <TaxonId> 12345
#>   database: ncbi
```

All of the classes mentioned so far can be replaced with character vectors in the higher-level classes that use them. This is convenient for users who do not have or need database information. However, using these classes allows for greater flexibility and rigor as the `taxa` develops; new kinds of information can be added to these classes without affecting backwards compatibility and the database objects stored in the `taxon_name`, `taxon_id`, and `taxon_rank` classes can be used to verify the integrity of data, even if data from multiple databases are combined. These classes are used to create the `taxon` class, which is the main building block of the package. It stores the name, ID, and rank of a taxon using the `taxon_name`, `taxon_id`, and `taxon_rank` classes. The `taxa` class is simply a list of `taxon` objects with a custom print method (i.e. the function controlling how it is displayed when printed to the console).

The hierarchy and taxonomy classes. The `taxon` class is used in the `hierarchy` and `taxonomy` classes, which store multiple taxa (Figure 1). The `hierarchy` class stores a taxonomic classification composed of nested taxa of different ranks (e.g. *Animalia*, *Chordata*, *Mammalia*, *Primates*, *Hominidae*, *Homo*, *sapiens*). Each taxon is stored as a `taxon` object in a list in the order they appear in the classification, from most inclusive to most specific. The `hierarchies` class is simply a list of `hierarchy` objects with a custom print method. The `hierarchies` class has the convenience of each hierarchy being independent, making it easy to subset by index or name, but it could also waste memory by storing multiple copies of the more coarse taxa (e.g. *Animalia*) that are likely to appear in many `hierarchy` objects. The `taxonomy` class is a more memory-efficient alternative that can store the same information.

The `taxonomy` class stores multiple taxa in a tree structure representing a taxonomy. The individual taxa are stored as a list of `taxon` objects and the tree structure is stored as an edge list representing subtaxa-supertaxa relationships. The edge list is a two-column table of taxon IDs that are automatically generated for each taxon. Using automatically generated taxon IDs, as opposed to taxon names, allows for multiple taxa with identical names. For example, *Achlya* is the name of an oomycete genus as well as a moth genus. It is also preferable to using taxon IDs from particular databases, since users might combine data from multiple databases and the same ID might correspond to different taxa in different databases. For example, "180092" is the ID for *Homo sapiens* in the Integrated Taxonomic Information System, but is the ID for *Acianthera teres* (an orchid) in the NCBI taxonomy database. The tree structure of the `taxonomy` class uses less memory than the same information saved as a table of ranks by taxa, since the information for each taxon occurs in only one instance. It also does not require explicit rank information (e.g. "genus" or "family").

The taxmap class. The `taxmap` class inherits the `taxonomy` class and is used to store any number of data sets associated with taxa in a taxonomy (Figure 1). A list called "data" stores any number of lists, tables, or vectors that are mapped to all or a subset of the taxa at any rank in the taxonomy. Therefore, the raw data used to make the object (and any other data associated with it) can be included in the `taxmap` object itself in its original form. In the case of tables, the presence of a "taxon_id" column containing unique taxon IDs indicates which rows correspond to which taxa. Lists and vectors can be named by taxon IDs to indicate which taxa their elements correspond to. When a `taxmap` object is subset or otherwise manipulated, these IDs allow for the taxonomy and associated data to remain in sync. The `taxmap` also contains a list called "funcs" that stores functions that return information based on the content of the `taxmap` object. In most functions that operate on `taxmap` objects, the results of built-in functions (e.g. `n_obs`), user-defined functions, and the user-defined content of lists, vectors, or columns of tables can

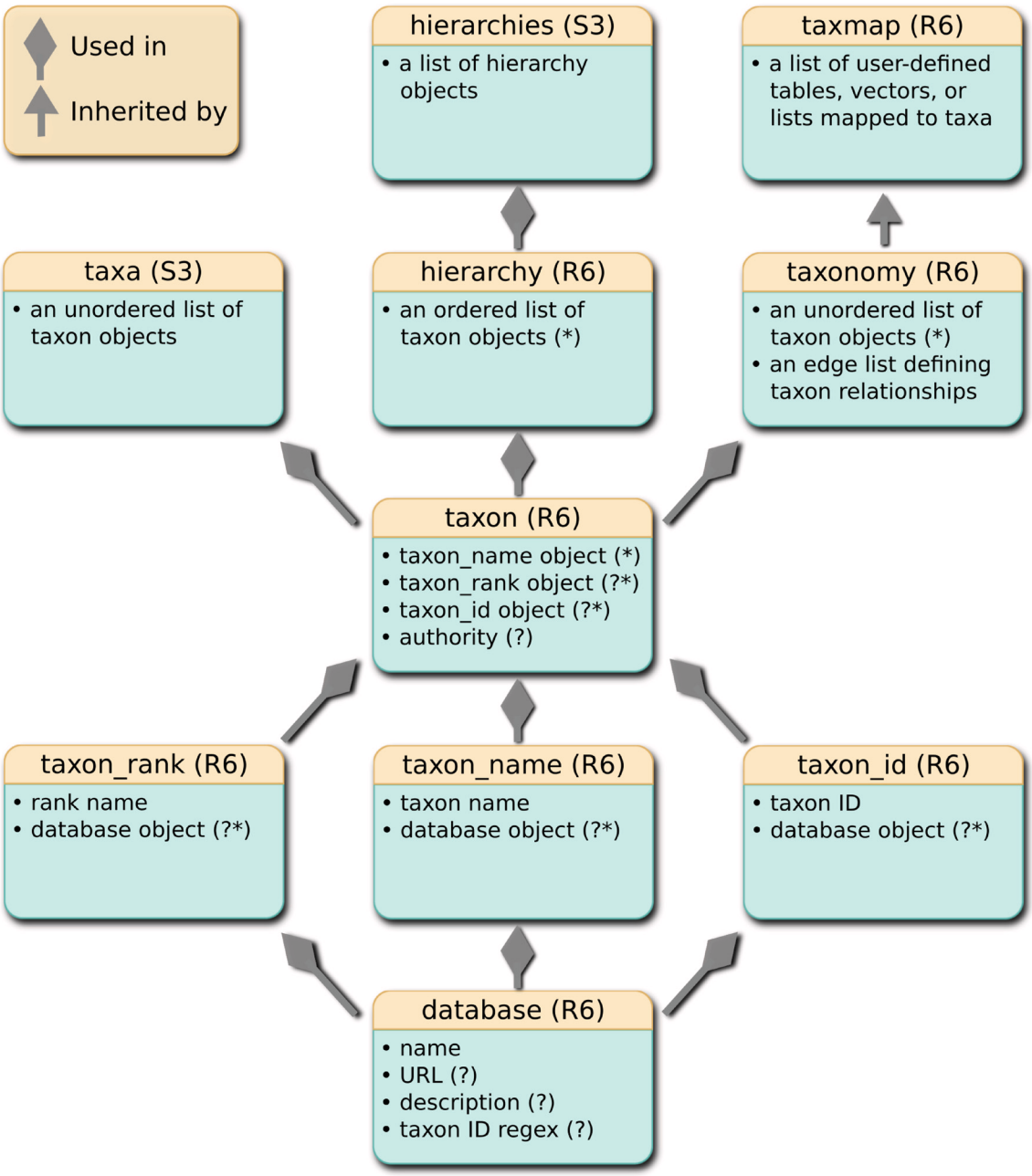

Figure 1. A class diagram representing the relationship between classes implemented in the `taxa` package. Diamond-tipped arrows indicate that objects of a lower class are used in a higher class. For example, a `database` object can be stored in the `taxon_rank`, `taxon_name`, or `taxon_id` `objects`. A standard arrow indicates that the lower class is inherited by the higher class. For example, the `taxmap` class inherits the `taxonomy` class. An asterisk indicates that an object (e.g. a `database` object) can be replaced by a simple character vector. A question mark indicates that the information is optional.

be referenced as if they are variables on their own, using non-standard evaluation (NSE). NSE is a technique used to make functions more convenient to use by interpreting things like variable names in a function call differently than they would be outside the function call or in other functions not using NSE. Any value returned by the `all_names` function can be used in this way. This greatly reduces the amount of typing needed and makes the code easier to read.

Manipulation functions. The `hierarchy`, `hierarchies`, and `taxa` classes have a relatively simple structure that is easily manipulated using standard indexing (i.e. using `[`, `[[`, or `$`),

but the `taxonomy` and `taxmap` classes are hierarchical, making them much harder to modify. To make manipulating these classes easier, we have developed a set of functions based on the `dplyr` data manipulation philosophy. The `dplyr` framework provides a consistent, intuitive, and chain-able set of commands that is easy for new users to understand. For example, `filter_taxa` and `filter_obs` are analogs of the `dplyr filter` function used to subset tables.

One aspect that makes `dplyr` convenient is the use of NSE to allow users to refer to column names as if they are variables on their own. The `taxa` package builds on this idea. Since `taxmap` objects can store any number of user-defined tables, vectors, lists, and functions, the values accessible by NSE are more diverse. All columns from any table and the contents of lists/vectors are available. There are also built-in and user-defined functions whose results are available via NSE. Referring to the name of the function as if it were an independent variable will run the function and return its results. This is useful for data that is dependent on the characteristics of other data and allows for convenient use of the `magrittr` `%>%` piping operator. For example, the built-in `n_subtaxa` function returns the number of subtaxa for each taxon. If this was run once and the result was stored in a static column, it would have to be updated each time taxa are filtered. If there are multiple filtering steps piped together using `%>%`, a static "n_subtaxa" column would have to be recalculated after each filtering to keep it up to date. Using a function that is automatically called when needed eliminates this hassle. The user still has the option of using a static column if it is preferable to avoid redundant calculations with large data sets.

Unlike `dplyr`'s `filter` function, `filter_taxa` works on a hierarchical structure and, optionally, on associated data simultaneously. By default, the hierarchical nature of the data is not considered; taxa that meet some criterion are preserved regardless of their place in the hierarchy. When the `subtaxa` option is `TRUE`, all of the subtaxa of taxa that pass the filter are also preserved and when `supertaxa` is `TRUE`, all of the supertaxa are likewise preserved. For example,

```
filter_taxa(my_taxmap, taxon_names == 'Fungi',
subtaxa = TRUE)
```

would remove any taxa that are not named "Fungi" or are not a subtaxon of a taxon named "Fungi". By default, steps are taken to ensure that the hierarchy remains intact when taxa are removed and that user-defined data are remapped to remaining taxa. When the `reassign_taxa` option is `TRUE` (the default), the subtaxa of removed taxa are reassigned to any supertaxa that were not removed, keeping the tree intact. When the `reassign_obs` option is `TRUE` (the default), any user-defined data assigned to removed taxa are reassigned to the closest supertaxa that passed the filter if such a taxon exists. This makes it easy to remove parts of the taxonomy without losing associated information. Finally, if the `drop_obs` option is `TRUE` (the default), any user-defined data assigned to removed taxa are also removed, allowing for subsetting of user-defined data based on taxon characteristics. The many combinations of these powerful options make `filter_taxa` a flexible tool and make it easier for new users to deal with the hierarchical nature of taxonomic data. For example, if the `drop_obs` option is `TRUE` (the default) and the `reassign_obs` option is `FALSE`, then any user-defined data assigned to taxa are removed even if a supertaxon is preserved. If the `drop_obs` option is `FALSE`, and the `reassign_obs` option is `FALSE`, then data associated with removed taxa is assigned a taxon ID placeholder of `NA`, but not removed. The function `sample_n_taxa` is a wrapper for `filter_taxa` that randomly samples some number of taxa. All of the options of `filter_taxa` can also be used for `sample_n_taxa`, in addition to options that influence the relative probability of each taxon being sampled.

Other dplyr analogs that help users manipulate their data include `filter_obs`, `sample_n_obs`, and `mutate_obs`.`filter_obs` is similar to running the `dplyr` function `filter` on a tabular, user-defined dataset, except that there are more values available to NSE and lists and vectors can also be subset. The `drop_taxa` option can be used to remove any taxa whose only observations have been removed during the filtering. The `sample_n_obs` function is a wrapper for `filter_obs` that randomly samples some number of observations. Like `sample_n_taxa`, there are options to weight the relative probability that each observation will be sampled. The `mutate_obs` function simply adds columns to tables of user-defined data.

Mapping functions. There are also a few functions that create mappings between different parts of the data contained in `taxmap` or `taxonomy` objects. These are heavily used internally in the functions described already, but are also useful for the user. The `subtaxa` and `supertaxa` functions return the taxon IDs (or other values) associated with all subtaxa or supertaxa of each taxon. They return one value per taxon. The `recursive` option controls how many ranks below or above each taxon are traversed. For example, `subtaxa(obj, recursive = 3)` will return information for all subtaxa and their immediate subtaxa for each taxon. The `recursive` option also accepts a simple `TRUE/FALSE`, with `TRUE` indicating all subtaxa of subtaxa, etc., and `FALSE` only returning immediate subtaxa, but not their descendants. By default, `subtaxa` and `supertaxa` return taxon IDs, but the `value` option allows the user to choose what information to return for each taxon. For example, `subtaxa(obj, value = "taxon_names")` will return the names of taxa instead of their IDs. Any data available to NSE (i.e. in the result of `all_names(obj)`) can be returned in this way.

The functions `roots`, `stems`, `branches`, and `leaves` are a conceptual set of functions that return different subsets of a taxonomy. A "root" is any taxon that does not have a supertaxon. A "stem" is a root plus all subtaxa before the first split in the tree. A "branch" is any taxon that has only one subtaxon and one supertaxon. Stems and branches are useful to identify since they can be removed without losing information on the relative relationship among the remaining taxa. "Leaves" are taxa with no subtaxa. By default, these options return taxon IDs, but also

have the `value` option like `subtaxa` and `supertaxa`, so they can return other information as well. For example, `leaves(obj, value = "taxon_names")` will return the names of taxa on the tips of the tree.

In the case of `taxmap` objects, the `obs` function returns information for observations associated with each taxon and its subtaxa. The observations could be rows in a table or elements in a list/vector that are named by taxon IDs. This is used to easily map between user-supplied information and taxa. For example, assuming a taxonomy with a single root, the value returned by `obs` for the root taxon will contain information for all observations, since they will all be assigned to a subtaxon of the root taxon. By default, row/element indices of observations will be returned, but the `obs` function also accepts the `value` option, so the contents of any column or other information associated with taxa can be returned as well.

The parsers. Taxonomic data appear in many different forms depending on the source of the data, making parsing a challenge. There are two main sources of variation in how taxonomic data are typically stored: the type of information supplied (e.g. a taxon name vs. a taxon ID) and how it is encoded (e.g. in a table vs. as part of a string). In addition, there might be additional user-specific data associated with the taxa that need to be parsed. These data might be associated with each taxon in a classification (e.g the taxon ranks) or might be associated with each classification (e.g. a sequence ID). In many cases, both types are present. This complexity makes implementing a generic parser for all types of taxonomic data difficult, so parsers are typically only available for specific formats. The `taxa` package introduces a set of three parsing functions that can parse the vast majority of taxonomic data as well as any associated data and return a `taxmap` object.

The `parse_tax_data` function is used to parse taxonomic classifications stored as vectors in tables that have already been read into R. In the case of tables, the classification can be spread over multiple columns or in a single column with character separators (e.g. "*Primates*;*Hominidae*;*Homo*;*sapiens*") or a combination of the two. Other columns are preserved in the output and the rows are mapped to the taxon IDs (e.g. the ID assigned to "sapiens" in the above example). For both tables and vectors, additional lists, vectors or tables can be included and are assigned taxon IDs based on some shared attribute with the source of the taxonomic data (e.g. a shared element ID or the same order). This makes it possible to parse many data sets at once and have them all mapped to the same taxonomy in the resultant `taxmap` object. Data associated with each taxon in each classification can also be parsed and included in the output using regular expressions with capture groups identifying the information to be stored and a key corresponding to the capture groups that identifies what each piece of information is. For example, `Hominidae_f_2;Homo_g_3;sapiens_s_4` would use the separator `";"`, the regular expression `"(.+)_(.+)_(.+)"`, and the key `c(my_taxon = "taxon_name", my_rank = "taxon_rank", my_id = "info")`. The values of the key indicate what the information is (a taxon name and two arbitrary pieces of information) and the names of the key (e.g. "my_rank") determine the names of columns in the output.

If only a taxon name (e.g. "Primates") or a taxon ID for a reference database (e.g. the NCBI taxon ID for *Homo sapiens* is "180092") is available in a table or vector, then the classification information must be queried from online databases and the function `lookup_tax_data` is used. `lookup_tax_data` has all the same functionality of `parse_taxa_data` in addition to being able to look up taxonomic classifications associated with taxon names, taxon IDs, and NCBI sequence IDs. If the data are embedded in a string (e.g. a FASTA header), then the function `extract_tax_data` is used instead. `extract_tax_data` has the functionality of `parse_tax_data` and `lookup_tax_data`, except that the information is extracted from raw strings using a regular expression and a corresponding key, the same way that data for each taxon in a classification is extracted by `parse_tax_data`. Together, these three parsing functions can handle every combination of data type and format presented in Figure 2 and many variations of those formats.

Operation

`Taxa` is an R package hosted on CRAN, so only an R installation and internet connection are needed to install and use `taxa`. Once installed, most of the functionality of the package can be used without an internet connection. R can be installed on nearly any operating system, including most UNIX systems, MacOS, and Windows. The minimum system requirements of R and the `taxa` package are easily met by most personal computers. The amount of resources needed will depend on the size of data being used and the complexity of analyses being conducted. The package can be installed by entering `install.packages("taxa")` in an interactive R session. The development version can be installed from GitHub using the `devtools` package:

```
library(devtools)
install_github("ropensci/taxa")
```

For users, the typical operation of the software will involve parsing some kind of input data into a `taxmap` object using a method demonstrated in Figure 2. Alternatively, a dependent package, such as `metacoder`, might provide a parser that wraps one of the taxa parsers or otherwise returns a `taxmap` object. Once the data is in a `taxmap` object, the majority of a user's interaction with the `taxa` package would typically involve filtering and manipulating the data using functions described in Table 1 and applying application-specific functions in other packages, such as `metacoder` (Figure 3).

Since `taxa` provides highly flexible parsers, it is usually possible to convert data from other packages to `taxa` classes, enabling manipulation of that data by `taxa` functions or packages that build upon `taxa`, like `metacoder`. For example, using the general-use parsers provided by the `taxa` package, `metacoder` supplies specialized and easy to use parsers for the following formats: taxonomy files produced by mothur,

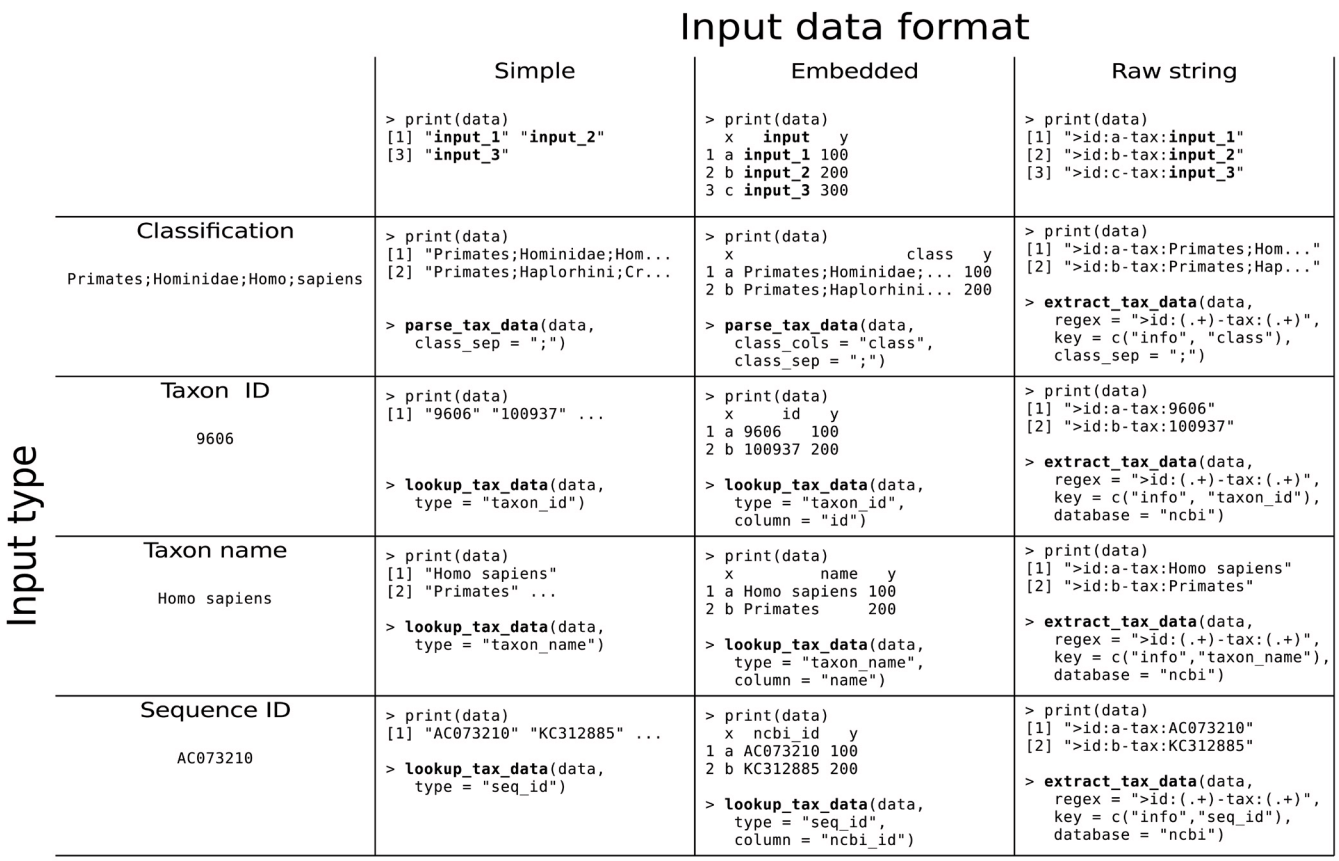

Figure 2. A table for determining how to parse different sources of taxonomic information using the `taxa` package. The rows correspond to the common sources of taxonomic information: full taxonomic classifications encoded in text, taxon IDs from a database, taxon names (a single rank), and NCBI sequence IDs. The columns correspond to the different formats the information can be encoded in: as a simple vector, as columns in a table, and as a piece of a complex string (e.g. a FASTA header). In the case of tables and complex strings, other information associated with the taxa can be preserved in the parsed result, as is done in the "use cases" example below. Any one cell in the table shows how to parse a given taxonomic information source in a given format using one of the three parsing functions: `parse_tax_data`, `lookup_tax_data`, `extract_tax_data`.

biom files produced by QIIME and MEGAN, newick files, objects from the `phyloseq` package, `phylo` objects from the `ape` package, and fasta files from the Greengenes (McDonald *et al.*, 2012), RDP (Cole *et al.*, 2014), SILVA (Yilmaz *et al.*, 2014), and UNITE databases (Kõljalg *et al.*, 2013). We have not encountered any text-based file format containing taxonomic information that can be described using regular expressions that the taxa parsers cannot read. For classes from other packages that inherit `list`, `vector`, or `data.frame`, conversion is not needed to include that information in a taxmap object, since the manipulation functions such as `filter_taxa` will handle them correctly as is.

Use case

`Taxa` is currently being used by `metacoder` and we are working on refactoring parts of `taxize` to work seamlessly with `taxa` as well. Both `taxize` and `metacoder` provide broadly useful functions such as querying databases with taxonomic information and plotting taxonomic information, respectively. We hope that having these two packages adopt the `taxa` framework will encourage developers of new packages to do so as well. Regardless, the flexible parsers implemented in `taxa` (Figure 2) allow for data from nearly any source to be used. The example analysis below uses data from the package `rgbif` (Chamberlain, 2017; Chamberlain & Boettiger, 2017), even though `rgbif` was not designed to work with `taxa`. This example shows a few of the benefits of using `taxa`. The function `occ_data` from the `rgbif` package returns a `data.frame` (i.e. table) of occurrence data for species from the Global Biodiversity Information Facility (GBIF) with one row per occurrence. The table has one column per taxonomic rank from kingdom to species.

```
# Look up plant occurrence data for Oregon
library(rgbif)
occ <- rgbif::occ_data(stateProvince = "Oregon",
                       scientificName ="Plantae")
```

This format returned by `rgbif::occ_data` is a variant on the format described in Figure 2, row 1, column 2, except

Table 1. Primary classes and functions found in `taxa`.

Function	Description
• `taxon`	A class that combines the classes containing the name, rank, and ID for a taxon.
• `taxa`	A simple list of taxon objects in an arbitrary order.
• `hierarchy`	A class that stores a list of nested taxa constituting a classification.
• `hierarchies`	A simple list of hierarchy objects in an arbitrary order.
• `taxonomy`	A class that stores a list of unique taxon objects and a tree structure.
• `taxmap`	A class that combines a taxonomy with user-defined, tables, lists, or vectors associated with taxa in the taxonomy. The taxonomic tree and the associated data can then be manipulated such that the two remain in sync.
• `supertaxa` • `subtaxa`	A "supertaxon" is a taxon of a coarser rank that encompasses the taxon of interest (e.g. *Homo* is a supertaxon of *Homo sapiens*). The "subtaxa" of a taxon are all those of a finer rank encompassed by that taxon. For example, *Homo sapiens* is a subtaxon of *Homo*. The supertaxa/subtaxa function returns the supertaxa/subtaxa of all or a subset of the taxa in a taxonomy object. By default, these functions return taxon IDs, but they can also return any data associated with taxa.
• `roots` • `leaves` • `stems` • `branches`	Roots are taxa that lack a supertaxon. Likewise, leaves are taxa that lack a subtaxon. Stems are those taxa from the roots to the first split in the tree. Branches are taxa with exactly one supertaxon and one subtaxon. In general, stems and branches can be filtered out without changing the relative relationship between the remaining taxa. By default, these functions return taxon IDs, but they can also return any data associated with taxa.
• `obs`	Returns the information about every observation from an user-defined data set for each taxon and their subtaxa. By default, indices of a list, vector, or table mapped to taxa are returned.
• `filter_taxa` • `filter_obs`	Subset taxa or associated data in `taxmap` objects based on arbitrary conditions. Hierarchical relationships among taxa and mappings between taxa and observations are taken into account.
• `arrange_taxa` • `arrange_obs`	Order taxon or observation data in `taxmap` objects.
• `sample_n_taxa` • `sample_n_obs` • `sample_frac_taxa` • `sample_frac_obs`	Randomly sample taxa or observation data in `taxmap` objects. Weights can be applied that take into account the taxonomic hierarchy and associated data. Hierarchical relationships among taxa and mappings between taxa and associated data are taken into account.

that there is only one rank per column instead of all ranks being concatenated in the same column (the parser accepts any number of columns, each of which could contain multiple ranks delineated by a separator).

```
# Parse data with taxa
library(taxa)
obj <- parse_tax_data(occ$data, class_cols = c(22:26, 28),
                      named_by_rank = TRUE)
```

In the `taxmap` object returned by `parse_tax_data`, the original table returned by `occ_data` is stored as `obj$data$tax_data`, but an extra column with taxon IDs for each row is prepended.

```
> print(obj)
<Taxmap>
626 taxa: aab. Plantae ... ayc. NA
626 edges: NA->aab, aab->aac ... aml->ayc
1 data sets:
  tax_data:
    # A tibble: 500 x 103
    taxon_id name          key     decimalLatitude
       <chr>  <chr>         <int>  <dbl>
     1 amm    Racomitriu... 1.70e9 44.2
     2 amn    Orthotrich... 1.68e9 NA
     3 amo    Didymodon ... 1.67e9 45.7
# ... with 497 more rows, and 99 more
# <<< List of additional columns ommited >>>
```

The data are then passed through a series of filters piped together. The `filter_obs` command removes rows from the occurrence data table not corresponding to preserved specimens, as well as any corresponding taxa that no longer have occurrences due to this filtering. The multiple calls to `filter_taxa` that follow demonstrate some of the different parameterizations of this powerful function. By default, taxa that don't pass the filter are simply removed and any occurrences assigned to them are reassigned to supertaxa that did pass the filter (e.g. occurrences for a deleted species would be assigned to the species' genus). When the `supertaxa` option is set to `TRUE`, all the supertaxa of taxa that pass the filter will also be preserved. The `subtaxa` option works the same way. Finally, the filtered data are passed to a plotting function from the `metacoder`

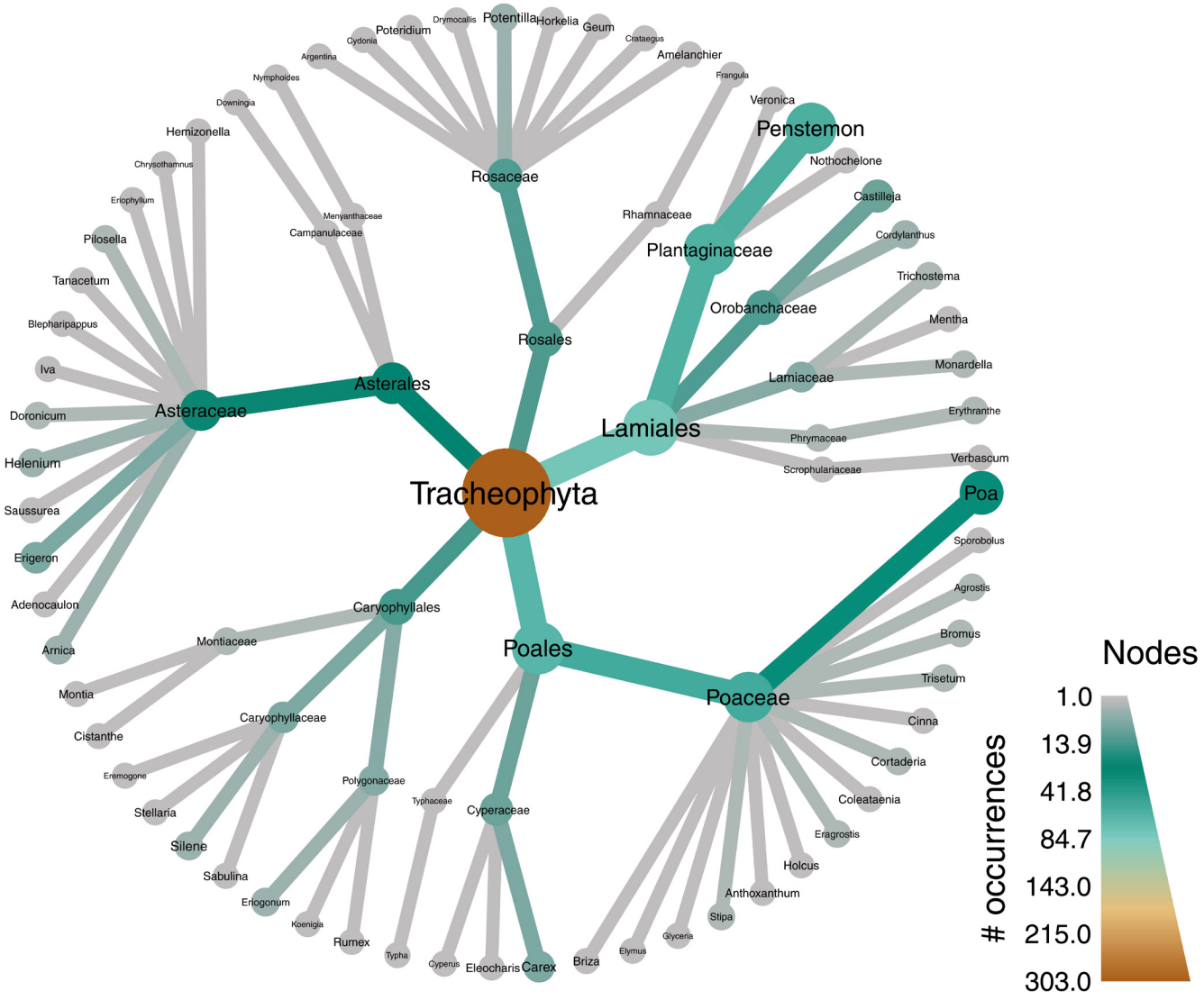

Figure 3. The result of the example analysis shown in the text. Records of plant species occurrences in Oregon are downloaded from the Global Biodiversity Information Facility (GBIF) using the `rgbif` package (Chamberlain, 2017). Then a `taxa` parser is used to parse the table of GBIF data into a `taxmap` object. A series of filters are then applied. First, all occurrences that are not from preserved specimens as well any `taxa` that have no occurrences from preserved specimens are removed. Then, all taxa at the species level are removed, but their occurrences are reassigned to the genus level. All taxa without names are then removed. In the final two filters, only orders within Tracheophyta with greater than 10 subtaxa are preserved. The `metacoder` package is then used to create a heat tree (i.e. taxonomic tree) with color and size used to display the number of occurrences associated with each taxon at each level of the hierarchy.

package that accepts the `taxmap` format. The plot is a taxonomic tree with color and size used to display the number of occurrences associated with each taxon (Figure 3).

```
# Plot number of occurrences for each taxon
library(metacoder)
obj %>%
  filter_obs("tax_data",
             basisOfRecord == "PRESERVED_SPECIMEN",
             drop_taxa = TRUE) %>%
  filter_taxa(taxon_ranks != "specificEpithet") %>%
  filter_taxa(! is.na(taxon_names)) %>%
  filter_taxa(taxon_names == "Tracheophyta",
              subtaxa = TRUE) %>%
  filter_taxa(taxon_ranks == "order",
              n_subtaxa > 10, subtaxa = TRUE,
              supertaxa = TRUE) %>%
heat_tree(node_label = taxon_names,
          node_color = n_obs,
          node_size = n_obs,
          node_color_axis_label = "# occurrences")
```

Note the use of columns in the original input table like `basisOfRecord` being used as if they were independent variables. This is implemented by NSE as a convenience to users, but they could also have been included by typing the full path to the variable (e.g. `obj$data$tax_data$basisOfRecord`

`or occ$data$basisOfRecord`). This is similar to the use of `taxon_ranks` and `taxon_names`, which are actually functions included in the class (e.g. `obj$taxon_ranks()`). The benefit of using NSE is that they are reevaluated each time their name is referenced. This means that the first time `taxon_ranks` is referenced in the example code it returns a different value than the second time it is referenced, because some taxa were filtered out. If `obj$taxon_ranks()` is used instead, it would fail on the second call because it would return information for taxa that have been filtered out already.

Conclusions

While `taxa` is useful on its own, its full potential will be realized after being adopted by the community as a standard for interacting with taxonomic information in R. A robust standard for the commonplace problems of data parsing and manipulation will free developers to focus on specific novel functionality. The `taxa` package already serves as the foundation of another package called `metacoder`, which provides functions for plotting taxonomic information and parsing common file formats used in metagenomics research. `Taxize`, the primary package for querying taxonomic information from internet sources, is also being refactored to be compatible with `taxa`. We hope the broadly useful functionality of these two packages will jump start adoption of `taxa` as the standard for taxonomic data manipulation in R.

Data and software availability

Install in R as `install.packages("taxa")`

Software available from: https://cran.r-project.org/web/packages/taxa/index.html

Source code available from: https://github.com/ropensci/taxa

Archived source code available from: https://doi.org/10.5281/zenodo.1183667 (Foster *et al.*, 2017)

License: MIT

Grant information

This work was supported in part by funds from USDA Agricultural Research Service Projects 2027-22000-039-00 and 2072-22000-039-15-S to NG and an rOpenSci grant to ZF.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

Chamberlain S: **rgbif: Interface to the Global 'Biodiversity' Information Facility 'API'**. R package version 0.9.8. 2017.
Reference Source

Chamberlain SA, Boettiger C: **R Python, and Ruby clients for GBIF species occurrence data.** *PeerJ Preprints.* 2017; **5**: e3304v1.
Publisher Full Text

Chamberlain SA, Szöcs E: **taxize: taxonomic search and retrieval in R [Version 1; Referees: 3 Approved].** *F1000Res.* 2013; **2**: 191.
PubMed Abstract | Publisher Full Text | Free Full Text

Cole JR, Wang Q, Fish JA, *et al.*: **Ribosomal Database Project: data and tools for high throughput rRNA analysis.** *Nucleic Acids Res.* 2014; **42**(Database issue): D633–42.
PubMed Abstract | Publisher Full Text | Free Full Text

Gentleman RC, Carey VJ, Bates DM, *et al.*: **Bioconductor: open software development for computational biology and bioinformatics.** *Genome Biol.* 2004; **5**(10): R80.
PubMed Abstract | Publisher Full Text | Free Full Text

Foster Z, Chamberlain S, Grunwald N: **taxa v0.2.0 (Version 0.2.0).** *Zenodo.* 2017.
http://www.doi.org/10.5281/zenodo.1183667

Foster ZS, Sharpton TJ, Grünwald NJ: **Metacoder: An R package for visualization and manipulation of community taxonomic diversity data.** *PLoS Comput Biol.* 2017; **13**(2): e1005404.
PubMed Abstract | Publisher Full Text | Free Full Text

Kõljalg U, Nilsson RH, Abarenkov K, *et al.*: **Towards a unified paradigm for sequence-based identification of Fungi.** *Mol Ecol.* 2013; **22**(21): 5271–7.
PubMed Abstract | Publisher Full Text

McMurdie PJ, Holmes S: **phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data.** *PLoS One.* 2013; **8**(4): e61217.
PubMed Abstract | Publisher Full Text | Free Full Text

McDonald D, Price MN, Goodrich J, *et al.*: **An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea.** *ISME J.* 2012; **6**(3): 610–8.
PubMed Abstract | Publisher Full Text | Free Full Text

Oksanen J, Blanchet FG, Kindt R, *et al.*: **Package 'Vegan'**. Community Ecology Package, Version. 2013; **2**(9).

Tippmann S: **Programming Tools: Adventures with R.** *Nature.* 2015; **517**(7532): 109–10.
PubMed Abstract | Publisher Full Text

Wickham H, Francois R: **"Dplyr: A Grammar of Data Manipulation"**. R Package Version 0.4. 2015; **1**: 20.

Yilmaz P, Parfrey LW, Yarza P, *et al.*: **The SILVA and "All-species Living Tree Project (LTP)" taxonomic frameworks.** *Nucleic Acids Res.* 2014; **42**(Database issue): D643–8.
PubMed Abstract | Publisher Full Text | Free Full Text

Open Peer Review

Current Referee Status: ✔ ✔ ✔ ✔

Version 2

Referee Report 12 November 2018

https://doi.org/10.5256/f1000research.17689.r38171

✔ **Titus Brown**
Department of Population Health and Reproduction, University of California, Davis, Davis, CA, USA

This version addresses all of my concerns. Thank you!

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Referee Report 20 September 2018

https://doi.org/10.5256/f1000research.17689.r38170

✔ **Holly M. Bik**
Department of Nematology, University of California, Riverside, Riverside, CA, USA

The authors have satisfactorily clarified all of my outstanding questions and added useful improvements to the text - additional explanations/discussions that makes this article informative for a broader audience were much appreciated.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Referee Report 11 September 2018

https://doi.org/10.5256/f1000research.17689.r38173

✔ **Ethan P. White** [1], **Kristina Riemer** [2]
[1] Department of Wildlife Ecology and Conservation and Informatics Institute, University of Florida, Gainesville, FL, USA
[2] Department of Wildlife Ecology and Conservation, University of Florida, Gainesville, FL, USA

This revision addresses all of our suggestions. Thanks!

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Referee Report 19 April 2018

https://doi.org/10.5256/f1000research.15231.r31496

Holly M. Bik
Department of Nematology, University of California, Riverside, Riverside, CA, USA

The authors present a framework named "taxa", designed to serve as a new standard package for interacting with taxonomy data in R. This package aims to address the ongoing difficulties in dealing with hierarchical taxonomy strings and numerical IDs in R, and I commend the authors on developing an exciting new framework that will simplify the manipulation and filtering of taxonomic data.

Overall, I thought this was a well written manuscript that did a fairly comprehensive job at explaining the functions and classes within the taxa package. However, I have a few comments that would further clarify the package functionality and inputs, and help make this manuscript accessible to a more general audience of computational biologists and ecologists (e.g. with novice to intermediate knowledge of R).

- Currently, this manuscript is geared towards a technical audience who are experts in R programming and package development. I would incorporate some more generalized explanations of the taxa package and its purpose (e.g. that assume a novice level of knowledge in R). For example, the use case using GBIF data frames assumes that readers are familiar with the field of biodiversity informatics and the format/information content of GBIF species occurrence data.

- What is the ideal input file for the taxa package? A basic tab-delimited taxonomy mapping file (e.g. with Accession IDs and taxonomic hierarchies only), a metabarcoding OTU table (e.g. JSON formatted or tab-delimited from QIIME where taxonomy strings are embedded along with study-specific data), or a full database with accessions and associated taxonomic information such as SILVA or NCBI? This package seems like it offers powerful tools for parsing and manipulating taxonomic information but it is not entirely clear what end users could (or should) be using as input files.

- It would be useful to explain how the "taxa" package can be integrated and linked to the other ecological R packages. Specific explanations or use cases involving vegan or phyloseq would be useful here. The link to metacoder and taxize is much more clearly laid out, probably due to the fact that the authors also developed these packages.

- Related to the previous point, how would you use taxa as a standalone package? The use case examples presented here make it seem like the "taxa" package is much more useful when used in

conjunction with metacoder or taxize. However, given the diverse functionality it seems like there are many other (very common) use cases for taxa that are not clearly presented here.

- How does "taxa" deal with (or allow manipulation / correction of) taxonomic hierarchies with non-homologous taxonomic levels. For example, a set of input hierarchies where level 4 represents "Order" level in Fungi but "Subclass" level in protists. This is a very common scenario for metabarcoding datasets - ideally you want to introduce gaps/placeholders for hierarchies that do not contain a certain level, so that users can automatically or manually standardize their taxonomic levels all rows in a dataset (e.g. making Level 7 correspond to "Family" level across all taxa).

- Does "taxa" (or related packages like taxize) contain any Taxonomic Name Resolution Service (TNRS) functionality? If not, is this planned for future releases?

- Page 5, paragraph 3: I found the description of the "reassign_taxa" option to be confusing. It was not clear to me what the purpose or result of this reassignment function would be. Clarifying the wording and adding a real world example would be useful here.

- Table 1: The description of "arrange_taxa" and "arrange_obs" is fairly vague. Do these functions rearrange data within a file or object (e.g. sorting or filtering)? If so, what are the options for ordering data (e.g. by abundance, alphabetical sorting, etc.)

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 31 Aug 2018

Niklaus Grunwald, USDA ARS, USA

Thank very much for your detailed, constructive review that much improved this manuscript. We addressed all your comments as follows:

- “Currently, this manuscript is geared towards a technical audience who are experts in R programming and package development. I would incorporate some more generalized explanations of the taxa package and its purpose (e.g. that assume a novice level of knowledge in R). For example, the use case using GBIF data frames assumes that readers are familiar with the field of biodiversity informatics and the format/information content of GBIF species occurrence data.”

The paper was written mostly to package developers, but we agree that it would be valuable to make it more accessible. We added some more explanation of technical concepts, including the GBIF data.

- “What is the ideal input file for the taxa package? A basic tab-delimited taxonomy mapping file (e.g. with Accession IDs and taxonomic hierarchies only), a metabarcoding OTU table (e.g. JSON formatted or tab-delimited from QIIME where taxonomy strings are embedded along with study-specific data), or a full database with accessions and associated taxonomic information such as SILVA or NCBI? This package seems like it offers powerful tools for parsing and manipulating taxonomic information but it is not entirely clear what end users could (or should) be using as input files.”

The taxa package has no ideal input format and provides flexibility for many formats, but there are some formats that are much easier to parse than others. Tabular data is usually easy to read, as are delimited taxonomy strings (e.g. taxa separated by ;). The taxa parsers do not read files directly. Instead they parse data already in R, so the different transformations/subsets of data from the same file could be parsed differently. For this reason, taxa does not read JSON/BIOM files from QIIME. Instead, it provides highly abstracted parsers to handle most formats and provides the foundation for more specialized (and easier to use) parsers, like those found in metacoder, which wrap the taxa parsers. There is a parser for JSON/BIOM QIIME files (parse_qiime_biom) in metacoder that uses the taxa parsers internally. The other examples you listed are easily handled by the taxa parsers after reading the file into R using something like read.table.

- “It would be useful to explain how the "taxa" package can be integrated and linked to the other ecological R packages. Specific explanations or use cases involving vegan or phyloseq would be useful here. The link to metacoder and taxize is much more clearly laid out, probably due to the fact that the authors also developed these packages.”

`taxa` is primarily intended as a foundation for future packages rather than a way of interacting with existing packages, but it can use the data from other packages in some cases. For data structures from other packages that inherit lists, vectors, or data.frames, the taxa filtering functions should be able to manipulate them correctly as is. For other data structures with one or two dimensions with arbitrary row/column/item names, like vegan’s distance matrix or ape’s DNAbin objects, these can be included as is in the taxmap object’s `data` list, the same way standard data.frames/lists are, although the filtering functions do not currently support these and they will be ignored. We would like to add the ability to natively handle these classes in the future, so that, for example, a DNAbin could be included in a `taxmap` object and filtered using `filter_taxa` the same way a list can be now. This should not be too hard to implement, but we have not gotten to it yet. For complicated objects like phyloseq objects that hold many fields themselves, the best solution would be to convert them to `taxmap` objects, manipulate them with the taxa functions, and convert them back. The conversion should be lossless since the `taxmap` class should be able to store all the information in a phyloseq object. There is a function in metacoder called `parse_phyloseq` to convert a phyloseq object to a taxmap object and this uses the `taxa` parsers internally.

- “Related to the previous point, how would you use taxa as a standalone package? The use case examples presented here make it seem like the "taxa" package is much more useful

when used in conjunction with metacoder or taxize. However, given the diverse functionality it seems like there are many other (very common) use cases for taxa that are not clearly presented here."

Good point! Yes, `taxa` can be quite useful on its own. A few examples we can think of include:

- Looking up taxonomic classifications from sequence IDs, taxon IDs, and taxon names from a variety of databases (using taxize internally).
- Subsetting data to a specific taxon
- Removing ranks or specific taxa from classification strings
- Combining taxonomic data from multiple sources into the same taxonomy
- Getting lists of all the subtaxa/supertaxa for each taxon or other data associated with all of the subtaxa/supertaxa.

We added a few of these examples to the paper.

- "How does "taxa" deal with (or allow manipulation / correction of) taxonomic hierarchies with non-homologous taxonomic levels. For example, a set of input hierarchies where level 4 represents "Order" level in Fungi but "Subclass" level in protists. This is a very common scenario for metabarcoding datasets - ideally you want to introduce gaps/placeholders for hierarchies that do not contain a certain level, so that users can automatically or manually standardize their taxonomic levels all rows in a dataset (e.g. making Level 7 correspond to "Family" level across all taxa)."

In the `taxonomy` and `taxmap` classes, the taxonomy is stored as a tree structure, not a table, so rank information is not needed, although it is supported when present. The absence of a rank has no placeholder, it simply does not exist in the tree. If the user wanted to subset the tree to only taxa of a specific set of ranks, they could do something like `filter_taxa(obj, taxon_ranks %in% c("family", "genus", "species"))` and the tree would remain intact, although there would be missing levels in the tree if some tips did not have a "family" supertaxon, for example. There is not currently a way to enforce that each rank exists at a fixed depth/level from the root of the tree, but we could add a function to add placeholder taxa to force that to the case (we added an issue to github at: https://github.com/ropensci/taxa/issues/169).

- "Does "taxa" (or related packages like taxize) contain any Taxonomic Name Resolution Service (TNRS) functionality? If not, is this planned for future releases?"

Yes, the parsing functions can optionally preprocess taxon names using the "Global Names Resolver" service via `taxize::gnr_resolve`. You just set the `type` option in `lookup_tax_data` or `extract_tax_data` to `"fuzzy_name"` instead of `"taxon_name"` to make this happen. This was a recent addition so it was not in the paper, but added it now.

- "Page 5, paragraph 3: I found the description of the "reassign_taxa" option to be confusing. It was not clear to me what the purpose or result of this reassignment function would be. Clarifying the wording and adding a real world example would be useful here."

Ok, we added some more explanation. The basic idea is that if you remove a taxon in the middle of the tree (say a family), it will assign any genera below that family to the order the family was in if reassign_taxa is set to TRUE (the default).

- "Table 1: The description of "arrange_taxa" and "arrange_obs" is fairly vague. Do these functions rearrange data within a file or object (e.g. sorting or filtering)? If so, what are the options for ordering data (e.g. by abundance, alphabetical sorting, etc.)"

"arrange_taxa" sorts the order of the taxa stored in `taxonomy` or `taxmap` objects. The order of taxa has little effect on most operations on these objects besides ordering the results of functions that return per-taxon information, like `supertaxa`. "arrange_obs" orders data stored in a `taxmap` objects (e.g. the row in an OTU table) based on some characteristics of that data. The options for ordering the data are therefore any piece of information associated with elements in that data set,

such as the contents of columns in that data set (e.g. the name of OTUs, the counts of OTUs, etc). We added some more descriptions of this to the paper.

Competing Interests: No competing interests were disclosed.

Referee Report 04 April 2018

https://doi.org/10.5256/f1000research.15231.r32711

Damiano Oldoni , **Peter Desmet**
Research Institute for Nature and Forest (INBO), Brussels, Belgium

The article is well written. Its structure is clear and the goals are well defined. Here below some minor issues.

General Issue

- What is the reason taxa functionalities are not implemented in taxize, which already seems a general purpose package to work with taxonomic information?

Introduction

- We found citation numbers for R to be very low. Does “its extensions” include all packages? Overall, it would be better to drop the sentence with citation numbers and keep it to R + easy development of packages, thus going fast to paragraph 2 which is more important to provide context for the taxa package
- “Database” is a very generic (technical) term. Would have expected “source”, or similar for source of taxonomic information, cf. **http://dublincore.org/documents/dcmi-terms/#elements-source**

Methods

Implementation

- Figure 1: we found ourselves drawing examples of the classes presented in figure 1. Would maybe be useful to add those to figure? If it is not possible for graphic issues, maybe could be useful to add them in the text, more or less as done in vignette of the package.
- In “manipulation functions” : “Finally, if the drop_obs option is TRUE (the default), any user-defined data assigned to removed taxa are also removed, ...” With the reassign_taxa and reassign_obs discussed above, it wasn’t immediately clear how taxa can be removed. Maybe update to “... data assigned to removed taxa (those without supertaxa matching the criteria) are also removed ...”

Use Cases

- Use cases: one use case presented. Update title to “Use case”. The presented use case is very informative, no need to add more use cases
- Use case might have been stronger if taxonomic information from 2 sources was combined (e.g. GBIF and …)

Here below some minor issues about the package:

- Consider moving CONDUCT.md to .github directory, as that directory is already used for CONTRIBUTING.md
- Add proper MIT License in LICENSE file

- README.md is now a combination of https://github.com/ropensci/taxa/blob/master/vignettes/taxa-introduction.Rmd and https://github.com/ropensci/taxa/blob/master/README.Rmd. Would keep README shorter (based on README.Rmd), with links to vignettes instead.
- Consider adding a pkgdown website, with references for functions + the two vignettes. Site can be build in docs/ folder and hosted on GitHub pages, cf. https://inbo.github.io/wateRinfo/
- Add website to repo description in repo settings.
- Consider moving vignette figures to “vignettes/figures” subdirectory for clarity

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 31 Aug 2018

Niklaus Grunwald, USDA ARS, USA

Thank very much for your detailed, constructive review that much improved this manuscript. We addressed all your comments as follows:

- “What is the reason taxa functionalities are not implemented in taxize, which already seems a general purpose package to work with taxonomic information?”

The main focus of taxize is to download taxonomic information from online databases. Since there are many sources of taxonomic data, taxize is already a large and complicated package. The R community provides restrictions on the size of packages. Also, not all applications where a class system for taxonomic data is needed require the ability to download taxonomic information.

- “We found citation numbers for R to be very low. Does “its extensions” include all packages? Overall, it would be better to drop the sentence with citation numbers and keep it to R + easy development of packages, thus going fast to paragraph 2 which is more important to provide context for the taxa package”

It should include all packages. It does not seem low to us, considering that many people do not cite software, but we see your point. We removed it.

- ""Database" is a very generic (technical) term. Would have expected "source", or similar for source of taxonomic information, cf. http://dublincore.org/documents/dcmi-terms/#elements-source"

We agree that "Source" would be a good term, although we hesitate to change the code at this point. We would have to rename the `taxon_database` to `taxon_source` (it is shorter, which is nice) and many option names that have some reference to "database". Changing our references to "database" to "source" in the paper is easy enough, but then the different words used for the same thing in the paper and the code might confuse some people.

- "Figure 1: we found ourselves drawing examples of the classes presented in figure 1. Would maybe be useful to add those to figure? If it is not possible for graphic issues, maybe could be useful to add them in the text, more or less as done in vignette of the package."

Good idea. We assume you are talking about the classes' print methods. Some would fit well in Figure 1, but the `taxmap` and `taxonomy` print methods would be too big. We added some examples to the body of the paper.

- "In "manipulation functions" : "Finally, if the drop_obs option is TRUE (the default), any user-defined data assigned to removed taxa are also removed, ..." With the reassign_taxa and reassign_obs discussed above, it wasn't immediately clear how taxa can be removed. Maybe update to "... data assigned to removed taxa (those without supertaxa matching the criteria) are also removed ..."

Yes, we see why that is confusing; thanks for the suggestion. Observations are only removed if they cannot be reassigned to something else. That could happen when "reassign_obs" is FALSE or there are no taxa left they could be reassigned to (as you say). We added some clarification about this.

- "Use cases: one use case presented. Update title to "Use case". The presented use case is very informative, no need to add more use cases"

Good point, thanks!

- "Use case might have been stronger if taxonomic information from 2 sources was combined (e.g. GBIF and …)"

We like that idea, but we can't think of a way to do it that would keep the example simple. We could look up the taxonomic hierarchy from NCBI or ITIS using the species binomial, but it would be an odd thing to do when the full classification is already available, so it might make the example confusing.

- "Consider moving CONDUCT.md to .github directory, as that directory is already used for CONTRIBUTING.md"

Done, see: https://github.com/ropensci/taxa/issues/149

- "Add proper MIT License in LICENSE file"

To conform with CRAN guidelines we could not do this. See: https://github.com/ropensci/taxa/issues/150

- "README.md is now a combination of https://github.com/ropensci/taxa/blob/master/vignettes/taxa-introduction.Rmd and https://github.com/ropensci/taxa/blob/master/README.Rmd. Would keep README shorter (based on README.Rmd), with links to vignettes instead." and "Consider adding a pkgdown website, with references for functions + the two vignettes. Site can be build in docs/ folder and hosted on GitHub pages, cf. https://inbo.github.io/wateRinfo/"

Good idea! We would like to add a website, but we will probably wait until we have one or two more vignettes done (the second is still being worked on). Any additional vignettes will be added with links. We would be ok with reducing the readme once we have a website up with documentation up that we can link to.
https://github.com/ropensci/taxa/issues/151

- "Consider moving vignette figures to "vignettes/figures" subdirectory for clarity"

We will do so and have pending issue on github: https://github.com/ropensci/taxa/issues/152

Competing Interests: No competing interests were disclosed.

Referee Report 26 March 2018

https://doi.org/10.5256/f1000research.15231.r31493

? **Titus Brown** [1], **Taylor Reiter** [2]
[1] Department of Population Health and Reproduction, University of California, Davis, Davis, CA, USA
[2] Food Science and Technology, University of California, Davis, CA, USA

The authors present the R package taxa, which provides a set of datatypes and functions for working with taxonomic data. The authors hope that they have contributed a strong base from which the taxonomic data ecosystem can build in R. The authors have also included a particularly useful set of parsers, and dplyr-like functionality within their package. The packages metacoder and rgbif are included as being compatible with taxa, and the authors mention that the popular package taxize is being refactored for compatibility.

Major points

- There is no discussion of the limitations of the software, or an specific discussion of incompatibility issues. If the authors have never encountered incompatibility issues, it would be helpful if they stated other packages or formats for which they have not encountered issues with.
- The introduction does not provide a concrete discussion of the challenges that the package taxa addresses.

Minor points

- In paragraph one, the authors note the ease with which one can develop an R package. I recommend adding "relative" somewhere in there.
- In paragraph two, it's not clear what is meant by "each package encodes this information differently."
- In paragraph four, "Complexity ranges from simple," "simple" is perhaps not the right word
- In paragraph five, "However, using these classes allows for greater flexibility and rigor as the package develops," it is not clear what is meant by "the package."

- In paragraph six, "(e.g. Animalia, Chordata, Mammalia, Primates, Hominidae, Homo, sapiens)" and “Achlya” should be italicized.
- I paragraph eight, "for the average user" should be removed. The clause, "that is easier for new users to understand than equivalent base R commands, which have accumulated some idiosyncrasies over the last 40 years" should also be rephrased to celebrate dplyr without cutting down base R.
- In paragraph 10, "The many combinations of these powerful options make filter_taxa a flexible tool and make it easier for new users to deal with the hierarchical nature of taxonomic data," "make" should be "makes."
- In paragraph 11, the sentence "Other dplyr analogs that help users manipulate their data include filter_obs, sample_n_obs, and mutate_obs, filter_obs is similar to running the dplyr function filter on a tabular, user-defined dataset, except that there are more values available to NSE and lists and vectors can also be subset," is confusing.
- In paragraph 15, sentence 1, "for many users" should be removed.
- In paragraph 16, “Primates;Hominidae;Homo;sapiens,” “sapiens,” and "Primates" should be italicized
- In paragraph 17, "Together, these three parsing functions can handle every combination of data type and format (Figure 2)," every is a strong assertion.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.

Author Response 31 Aug 2018

Niklaus Grunwald, USDA ARS, USA

Thank very much for your detailed, constructive review that much improved this manuscript. We addressed all your comments as follows:

- “There is no discussion of the limitations of the software, or an specific discussion of incompatibility issues. If the authors have never encountered incompatibility issues, it would be helpful if they stated other packages or formats for which they have not encountered issues with.”

We think adding some information of limitations of the software is a good idea, but we are not sure what you had in mind exactly. In regards to limitations on data set size and speed, we have not explored this systematically yet, although we plan to identify parts of the code to port to C++ to increase speed where needed. We are also not sure what you mean specifically by “incompatibility issues”. It is certainly true that few packages are designed to work seamlessly with `taxa` currently, but `taxa` was designed with this in mind and the parsers can be used to import data from other formats not designed for use with `taxa`, as was done in the use case. We added some examples of packages we have used with `taxa` to demonstrate compatibility and will mention if we find any that are not compatible in some way.

In regards to formats, in our own work and when helping others, we have used the taxa parsers with numerous (> 20 maybe) different formats of taxonomic data and have never encountered a raw-text-based format that the current parsers cannot handle, but of course there might be some cases we have have not encountered/considered. We like the idea of adding a list of formats for which we have used taxa to read. We added this to the paper.

- “The introduction does not provide a concrete discussion of the challenges that the package taxa addresses.”

We do mention the lack of a standard set of classes for packages to build on, which is the main challenge `taxa` is trying to address, and we added some more background on data parsing and manipulation, which are the other goals taxa` tries to address.

- “In paragraph one, the authors note the ease with which one can develop an R package. I recommend adding "relative" somewhere in there.”

Good point! We did that. We remember it did not appear easy when we started.

- “In paragraph two, it's not clear what is meant by "each package encodes this information differently.””

Ok, we added some examples.

- “In paragraph four, "Complexity ranges from simple," "simple" is perhaps not the right word”

Agreed. The low-level classes are quite simple currently, little more than containers for a few variables, but we removed the word “simple”, since it is not all that descriptive anyway.

- “In paragraph five, "However, using these classes allows for greater flexibility and rigor as the package develops," it is not clear what is meant by "the package.""

We meant `taxa`. We reworded that, thanks!

- “In paragraph six, "(e.g. Animalia, Chordata, Mammalia, Primates, Hominidae, Homo, sapiens)" and “Achlya” should be italicized.”

Agreed. Done.

- “I paragraph eight, "for the average user" should be removed. The clause, "that is easier for new users to understand than equivalent base R commands, which have accumulated some idiosyncrasies over the last 40 years" should also be rephrased to celebrate dplyr without cutting down base R.”

Agreed, we made those changes. We did not mean to berate base R, but rather point out a lack of consistency relative to dplyr, but we can see how people could get that impression.

- “In paragraph 10, "The many combinations of these powerful options make filter_taxa a flexible tool and make it easier for new users to deal with the hierarchical nature of taxonomic data," "make" should be "makes.""

“The many combinations” is plural, so we think it should be “make”.

- "In paragraph 11, the sentence "Other dplyr analogs that help users manipulate their data include filter_obs, sample_n_obs, and mutate_obs, filter_obs is similar to running the dplyr function filter on a tabular, user-defined dataset, except that there are more values available to NSE and lists and vectors can also be subset," is confusing."

Thanks for catching that! That comma between "mutate_obs" and "filter_obs" was supposed to be a period, which we think makes it significantly less confusing.

- "In paragraph 15, sentence 1, "for many users" should be removed."

Ok, we did that.

- "In paragraph 16, "Primates;Hominidae;Homo;sapiens," "sapiens," and "Primates" should be italicized"

Agreed.

- "In paragraph 17, "Together, these three parsing functions can handle every combination of data type and format (Figure 2)," every is a strong assertion."

We meant every combination of data type and format covered in the preceding paragraphs and the figure, and we clarified that further.

Competing Interests: No competing interests were disclosed.

Referee Report 26 March 2018

https://doi.org/10.5256/f1000research.15231.r31494

Ethan P. White [1], **Kristina Riemer** [2]

[1] Department of Wildlife Ecology and Conservation and Informatics Institute, University of Florida, Gainesville, FL, USA

[2] Department of Wildlife Ecology and Conservation, University of Florida, Gainesville, FL, USA

The software described in this paper provides useful tools for working with taxonomic data in R by providing a standard approach for storing and manipulating this hierarchically structured data. Taxonomic data is prevalent in many biological disciplines. As result, this package fills an important niche and has the potential to become widely used by other packages dealing with biological data.

The software itself follows good development practices including modularization, documentation, version control, and automated testing. The package is available through CRAN – the main repository for R packages. Both the CRAN release and the development version of the package install smoothly. The use case examples given in the paper all run as expected on the development version, but they include functionality that is not present in the most recent release. This means that readers of the paper who have not installed the development version will encounter issues with the examples. We recommend a new minor version release so that the existing functionality is reflected in the latest release. Alternatively in-development functionality could be removed from the examples in the paper.

The paper does a nice job of motivating the need for the package and the use case section nicely demonstrates some of the core functionality. However, there are improvements that could be made to help the paper communicate with a broader audience. Specifically we recommend changes to the Introduction and Methods sections.

In the Introduction we suggest either expanding the second paragraph or adding a new paragraph to describe the other kinds of datasets that this will be helpful for. There are many large and small ecological

and evolutionary datasets beyond high-throughput sequencing that involve lots of taxonomic data (e.g., museum records, citizen science projects, compilations of literature data) and broadening the context will help more readers understand why this package might be useful to them. We also suggest adding an additional paragraph, following the third paragraph, that describes typical taxonomic data, including an example, and that mentions the specific challenges of this kind of hierarchical data. This will help readers less familiar with these issues understand the value of the software and help set up the technical details in the last paragraph of the Introduction. To make room for these additions we suggest removing the first paragraph, which currently states that R is "becoming the leading tool for scientific data analysis in academic research." This specific interpretation isn't justified by the associated citation and it is broadly understood that R is an important language so a paragraph explaining this isn't really necessary.

In the Methods we suggest moving the parsing section to the beginning, and using the examples from that section throughout the descriptions of classes. This will help ground the descriptions of the classes and how they are related, which currently reads as somewhat abstract. The current second paragraph ("The hierarchy and taxonomy class") would benefit from having the hierarchy class defined more and the differences between the hierarchy and taxonomy classes clarified. For example, it is stated later that the hierarchy class is simpler and the taxonomy class is more hierarchical; it would be helpful to include this information earlier. Moving the last two sentences of this paragraph to the beginning might address this issue. The taxon IDs information could be its own paragraph, starting with "Using automatically generated taxon IDs". The examples in that section are really helpful. In the beginning of the third paragraph of the methods ("The taxmap class"), it would be helpful to emphasize that this class combines the rest of the original data (including an example of original data, e.g., mass) back with the taxon class. Finally, a figure of an example taxonomic hierarchy that illustrates the operation of the filtering, mapping, and roots/stems/branches/leaves functions would be useful.

Minor suggestions

- Define the following phrases to broaden communication
  - "character vectors", first paragraph of methods
  - "custom print method", first paragraph of methods
  - "non-standard evaluation", third paragraph of methods
  - "parsing", paragraph 11 of methods
- Cite Wickham & Francois (2015) for the dplyr philosophy in the fourth Methods paragraph
- Consider color coding the boxes in Fig. 1 to match the three classes paragraphs
- Define R6 and S3 in the Fig. 1 legend

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 31 Aug 2018

Niklaus Grunwald, USDA ARS, USA

Thank very much for your detailed, constructive review that much improved this manuscript. We addressed all your comments as follows:

- 'The use case examples given in the paper all run as expected on the development version, but they include functionality that is not present in the most recent release. This means that readers of the paper who have not installed the development version will encounter issues with the examples. We recommend a new minor version release so that the existing functionality is reflected in the latest release."

Thanks for catching this! We have released a version on CRAN with this functionality.

- "In the Introduction we suggest either expanding the second paragraph or adding a new paragraph to describe the other kinds of datasets that this will be helpful for. There are many large and small ecological and evolutionary datasets beyond high-throughput sequencing that involve lots of taxonomic data (e.g., museum records, citizen science projects, compilations of literature data) and broadening the context will help more readers understand why this package might be useful to them"

Good idea! We added a section on this.

- "We also suggest adding an additional paragraph, following the third paragraph, that describes typical taxonomic data, including an example, and that mentions the specific challenges of this kind of hierarchical data. This will help readers less familiar with these issues understand the value of the software and help set up the technical details in the last paragraph of the Introduction"

Ok, good idea, we added some examples of diverse formats that we have used.

- "To make room for these additions we suggest removing the first paragraph, which currently states that R is "becoming the leading tool for scientific data analysis in academic research." This specific interpretation isn't justified by the associated citation and it is broadly understood that R is an important language so a paragraph explaining this isn't really necessary."

Yes, we had a similar comment from another reviewer, so we removed part of this.

- "In the Methods we suggest moving the parsing section to the beginning, and using the examples from that section throughout the descriptions of classes. This will help ground the descriptions of the classes and how they are related, which currently reads as somewhat abstract."

The parsers only returns `taxmap` objects so far, and `taxmap` is built upon the previous classes, so that is why we ordered it that way. However, we agree that it is not immediately clear what the importance of the first classes described are.

- "The current second paragraph ("The hierarchy and taxonomy class") would benefit from having the hierarchy class defined more and the differences between the hierarchy and taxonomy classes clarified. "

Agreed, we will clarify this.

- "In the beginning of the third paragraph of the methods ("The taxmap class"), it would be helpful to emphasize that this class combines the rest of the original data (including an example of original data, e.g., mass) back with the taxon class. "

Good point. This is a key aspect of the class.

- "Finally, a figure of an example taxonomic hierarchy that illustrates the operation of the filtering, mapping, and roots/stems/branches/leaves functions would be useful."

We like this idea. This is something we have considered in the past. We will add this to one of our vignettes in the future:
https://github.com/ropensci/taxa/issues/170

- "Define the following phrases to broaden communication
- "character vectors", first paragraph of methods
- "custom print method", first paragraph of methods
- "non-standard evaluation", third paragraph of methods
- "parsing", paragraph 11 of methods

Agreed, we made those changes.

- "Cite Wickham & Francois (2015) for the dplyr philosophy in the fourth Methods paragraph"

We cited it in the introduction.

- "Consider color coding the boxes in Fig. 1 to match the three classes paragraphs"

We are not sure what you mean here.

- "Define R6 and S3 in the Fig. 1 legend"

Agreed.

Competing Interests: No competing interests were disclosed.

Discuss this Article

Version 2

Reader Comment (*Member of the F1000 Faculty*) 10 Oct 2018

Peter Uetz, Center for the Study of Biological Complexity, Virginia Commonwealth University, USA

Thanks Niklaus,

Sorry about my nitpicking, but if you want to get these tools used in the community clarification in the abstract is important (editors and reviewers should remember that too ;-): I had to look up the kind of taxon IDs you are refering to in the taxize paper. The only sequence ID I can find in your paper is "NCBI sequence ID", so why not just mention "NCBI sequence ID" in the abstract? What do you mean by "manipulation" and "modification"? There are all kinds of ways to "manipulate" and "modify" taxonomic data, so an upfront clarification (in the abstract) would be useful. Thanks again.

Competing Interests: none

Author Response 08 Oct 2018

Niklaus Grunwald, USDA ARS, USA

Dear Peter, thanks very much for your comment. We believe that the abstract does mention (albeit concisely) what the package does:

*We developed the R package taxa to provide a robust and flexible s**olution to storing and manipulating taxonomic data in R** and any application-specific information associated with it. **Taxa provides parsers that can read common sources of taxonomic information** (taxon IDs, sequence IDs, taxon names, and classifications) from nearly any format while preserving associated data. **Once parsed, the taxonomic data and any associated data can be manipulated** using a cohesive set of functions modeled after the popular R package dplyr. These functions take into account the hierarchical nature of taxa and **can modify the taxonomy or associated data** in such a way that both are kept in sync.*

We also mention examples which are in turn provided in the main body of the MS:

***Taxa is currently being used by the metacoder and taxize packages,** which provide broadly useful functionality that we hope will speed adoption by users and developers.*

Please keep in mind that this is a low-level package that provides tools for end-user packages like taxize and metacoder. This package is only useful to colleagues doing analyses in the R language.

Competing Interests: No competing interests were disclosed.

Reader Comment (*Member of the F1000 Faculty*) 14 Sep 2018

Peter Uetz, Center for the Study of Biological Complexity, Virginia Commonwealth University, USA

As the curator of a taxonomic database and as a daily user of taxonomic data, it is completely unclear from the abstract (at least to me) what this package does. The authors should provide at least an example of some concrete usages of the package, otherwise people (like me) will stop reading after the abstract.

Competing Interests: none