NCBITaxonomy.jl - rapid biological names finding and reconciliation

Timothée Poisot [1,2] Rory Gibb [3,4,5] Sadie J. Ryan [6,7,8] Colin J. Carlson [10,9]

[1] Université de Montréal, Départment de Sciences Biologiques, Montréal QC, Canada [10] Center for Global Health Science and Security, Georgetown University Medical Center, Georgetown University, Washington, D.C., United States of America [2] Québec Centre for Biodiversity Science, Montréal, QC, Canada [3] Centre on Climate Change and Planetary Health, London School of Hygiene and Tropical Medicine, London, UK [4] Centre for Mathematical Modelling of Infectious Diseases, London School of Hygiene and Tropical Medicine, London, UK [5] Current address: Centre for Biodiversity and Environment Research, University College London, London, UK [6] Emerging Pathogens Institute, University of Florida, Gainesville, FL, United States of America [7] School of Life Sciences, University of KwaZulu-Natal, Durban, South Africa [8] Department of Geography, University of Florida, Gainesville, FL, United States of America [9] Department of Microbiology and Immunology, Georgetown University Medical Center, Georgetown University, Washington, D.C., United States of America

Correspondance to:

Timothée Poisot — `timothee.poisot@umontreal.ca`

`NCBITaxonomy.jl` is a package designed to facilitate the reconciliation and cleaning of taxonomic names, using a local copy of the NCBI taxonomic backbone (Federhen 2012, Schoch et al. 2020); The basic search functions are coupled with quality-of-life functions including case-insensitive search and custom fuzzy string matching to facilitate the amount of information that can be extracted automatically while allowing efficient manual curation and inspection of results. `NCBITaxonomy.jl` works with version 1.6 of the Julia programming language (Bezanson et al. 2017), and relies on the Apache Arrow format to store a local copy of the NCBI raw taxonomy files. The design of `NCBITaxonomy.jl` has been inspired by similar efforts, like the R package `taxadb` (Norman et al. 2020), which provides an offline alternative to packages like `taxize` (Chamberlain and Szöcs 2013).

Last revision: *January 24, 2023*

Unambiguously identifying species is a far more challenging task than it may appear. There are a vast number of reasons for this. Different databases keep different taxonomic "backbones", *i.e.* different data structures in which names are mapped to species, and organised in a hierarchy. Not all names are unique identifiers to groups. For example, *Io* can either refer to a genus of plants from the aster family, or to a genus of molluscs; the genus *Mus* (of which the house mouse *Mus musculus* is a species), contains a sub-genus *also* named *Mus* (within which *Mus musculus* is located). Conversely, the same species can have several names, which are valid synonyms: for example, the domestic cow *Bos taurus* admits *Bos primigenius taurus* as a valid synonym. In addition to binomial names, the same species can be known by many vernacular (common) names, which are language or even region-specific: *Ovis aries*, for example, has valid English vernaculars including lamb, sheep, wild sheep, and domestic sheep.

Finally, taxonomic nomenclature changes regularly, with groups being split, merged, or moved to a new position in the tree of life; this is, notably, a common occurrence with viral taxonomy, each subsequent version of which can differ markedly from the last; compare, *e.g* Lefkowitz et al. (2018) to Walker et al. (2020), where entire viral sub-trees were split, re-organized, and created within just two years. These taxonomic changes have profound implications for the way we perceive biodiversity at global scales (Dikow et al. 2009), to the point were taxonomic revisions should sometimes be actively conducted to improve *e.g.* conservation outcomes (Melville et al. 2021).

To add to the complexity, one must also consider that most taxa names are at some point manually typed, which has the potential to introduce additional mistakes in raw data; it is likely to expect that such mistakes may arise when attempting to write down the (perfectly valid) names of the bacterial isolate known as *Myxococcus llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogochensis*, or of the crowned slaty flycatcher *Griseotyrannus aurantioatrocristatus*. These mistakes are more likely when dealing with hyper-diverse samples (demanding to memorize more names), like plant census (Dauncey et al. 2016, Wagner 2016, Conti et al. 2021); when dealing with multiple investigators with different knowledge of the taxonomy; and as a result of the estimated error in any data entry exercice, which other fields estimate at up to about 5% (Barchard and Pace 2011).

All these considerations become important when matching species names both within and across datasets. Let us consider the hypothetical species survey of riverine fishes: European chub, *Cyprinus cephalus*, *Leuciscus cephalus*, *Squalius cephalus*. All are the same species (*S. cephalus*), referred to as one of the vernacular (European chub) and two formerly accepted names now classified as synonyms (but still present in the litterature). A cautious estimate of diversity based on the user-supplied names would give $n = 4$ species, when there is in fact only one. When the size of biodiversity datasets increases, and notably when the taxonomic scope of these datasets explodes, including organisms for which "names" are a fuzzier concept (for example, *Influenza A virus (A/Sydney/05/97-like(H3N2))* is a valid name for a common influenza strain, although one that lacks a taxonomic rank), the feasibility of manual curation decreases.

In this manuscript, we describe `NCBITaxonomy.jl`, a Julia package that provides advanced name matching and error handling capacities for the reconciliation of taxonomic names to the NCBI database. This package was used to facilitate the development of the *CLOVER* (Gibb et al. 2021) database of host-virus associations, by reconciling the names of viruses and mammals from four different sources, where all of the issues described above were present. More recently, it has become part of the automated curation of data for the *VIRION* (Carlson et al. 2022) database, which automatically curates an up-to-date, authoritative virome network from dozens of heterogeneous sources. We describe the core capacities of this package, and highlight how it enables safe, high-performance name reconciliation.

Overview of functionalities

An up-to-date version of the documentation for `NCBITaxonomy.jl` can be found in the package `git` repository (`https://github.com/PoisotLab/NCBITaxonomy.jl`), including examples and in-line documentation of every method. The package is released under the MIT license. Contributions can be made in the form of issues (bug reports, questions, features suggestions) and pull requests.

In order to achieve good performance, the package will first retrieve the latest (as validated by its checksum) NCBI taxonomy backbone, store it locally, and pre-process it as a set of Julia data tables. By default, the taxonomy will be downloaded to the user's hoome directory, which is not an ideal solution, and therefore we recommend that users set an environment variable to specificy where the data will be loaded from (this path will be created if it doesn't exist):

```julia
ENV["NCBITAXONOMY_PATH"] = joinpath(homedir(), "data", "NCBITaxonomy.jl")
```

Note that this location can be different for different projects, as the package is able to update the taxonomic backbone (and will indeed prompt the user to do so if the taxonomy is more than 90 days old). The package can then be checked out and installed anonymously from the central Julia repository:

```
using Pkg
Pkg.add("NCBITaxonomy") # Dowloading the files may take a long time
```

As long as the package is not re-built, the local set of tables downloaded from NCBI will not change; this way, users can re-run an analysis with a guarantee that the underlying taxonomic backbone has not changed, which is not the case when relying on API queries. In order to update the taxonomic backbone, users can call the `build` function of Julia's package manager (`]build NCBITaxonomy`), which will download the most recent version of all files.

This software note describes version `v0.3.0` of the package (we follow semantic versioning), which works on Julia 1.5 upwards. The dependencies are all resolved by the package manager at installation, and (on the user-facing side) include the `StringDistances.jl` package, allowing users to experiment with different string matching methods. As is best practices for Julia packages, a `Project.toml` file specifying compatible dependencies versions is distributed with the package. The code is covered by unit-tests (with about 98% coverage), as well as integration tests as part of the documentation (specifically, a use-case detailing how to clean data from a biodiversity survey, and a use-case aiming to reconstruct a taxonomic tree for the Lemuriformes).

Improved name matching

Name finding, *i.e.* the matching of an arbitrary string to a taxonomic identifier, is primarily done through the `taxon` function, which admits either a unique NCBI identifier (*e.g.* `taxon(36219)` for the bogue *Boops boops*), a string (`taxon("Boops boops")`), or a data frame with a restricted list of names in order to create a name finder function (see the next section). The `taxon` method has additional arguments to perform fuzzy matching in order to catch possible typos (`taxon("Boops bops"; strict=false)`), to perform a lowercase search (useful when alphanumeric codes are part of the taxon name, like for some viruses), and to restrict the the search to a specific taxonomic rank. The `taxon` function also accepts a `preferscientificname` keyword, to prevent matching vernacular names; the use of this keyword ought to be informed by knowledge about how the data were entered.

The lowercase search can be a preferable alternative to fuzzy string matching. Consider the string `Adeno-associated virus 3b` - it has three names with equal distance (under the Levensthein string distance function):

```
julia> similarnames("Adeno-associated virus 3b"; threshold=0.95)
3-element Vector{Pair{NCBITaxon, Float64}}:
  Adeno-associated virus - 3 (ncbi:46350) => 0.96
   Adeno-associated virus 3B (ncbi:68742) => 0.96
 Adeno-associated virus 3A (ncbi:1406223) => 0.96
```

Depending on the operating system (and specifically whether it is case-sensitive), either of these three names can be returned; compare to the output of a case insensitive name search:

```
julia> taxon("Adeno-associated virus 3b"; casesensitive=false)
Adeno-associated virus 3B (ncbi:68742)
```

This returns the correct name.

Name matching output and error handling

When it succeeds, `taxon` will return a `NCBITaxon` object (made of a `name` string field, and an `id` numerical field). That being said, the package is designed under the assumption that ambiguities should yield an error for the user to handle. There are two such errors: `NameHasNoDirectMatch` (with instructions about how to possible solve it, using the `similarnames` function), or a `NameHasMultipleMatches` (listing the possible valid matches, and suggesting to use `alternativetaxa` to find the correct one). Therefore, the common way to work with the `taxon` function would be to wrap it in a `try/catch` statement:

```
try
  taxon(name)
  # Additional operations with the matched name
catch err
  if isa(err, NameHasNoDirectMatch)
    # What to do if no match is found
  elseif isa(err, NameHasMultipleMatches)
    # What to do if there are multiple matches
  else
```

```
    # What to do in case of another error that is not NCBITaxonomy specific
  end
end
```

These functions will not demand any user input in the form of key presses (though they can be wrapped in additional code to allow it), as they are intended to run on clusters or virtual machines without supervision. The `taxon` function has good scaling using muliple threads. For convenience in rapidly getting a taxon for demonstration purposes, we also provide a string macro, whereby *e.g.* `ncbi"Procyon lotor"` will return the taxon object for the raccoon.

Name filtering functions

As the full NCBI names table has over 3 million entries at the time of writing, we have provided a number of functions to restrict the scope of names that are searched. These are driven by the NCBI *divisions*. For example `nf = mammalfilter(true)` will return a data frame containing the names of mammals, inclusive of rodents and primates, and can be used with *e.g.* `taxon(nf, "Pan")`. This has the dual advantage of making search faster, but also of avoiding matching on names that are shared by another taxonomic group (which is not an issue with *Pan*, but is an issue with *e.g.* *Io* as mentioned in the introduction, or with the common name *Lizard*, which fuzzy-matches on the hemipteran genus *Lisarda* rather than the class *Lepidosauria*).

Note that the use of a restricted list of names can have significant performance consequences: compare, for example, the time taken to return the taxon *Pan* in the entire database, in all mammals, and in all primates:

Names list	Fuzzy matching	Time (ms)	Allocations	Memory allocated
all	no	23	34	2 KiB
	yes	105	2580	25 MiB
`mammalfilter(true)`	no	0.55	32	2 KiB
	yes	1.9	551	286 KiB
`primatefilter()`	no	0.15	33	2 KiB

Names list	Fuzzy matching	Time (ms)	Allocations	Memory allocated
	yes	0.3	92	27 KiB

Clearly, the optimal search strategy is to (i) rely on name filters to ensure that search are conducted within the appropriate NCBI division, and (ii) only rely on fuzzy matching when the strict or lowercase match fails to return a name, as fuzzy matching can result in order of magnitude more run time and memory footprint. These numbers were obtained on a single Intel i7-8665U CPU (@ (1.90GHz). Using `"chimpanzees"` as the search string (one of the NCBI recognized vernaculars for *Pan*) gave qualitatively similar results, suggesting that there is no performance cost associated with working with synonyms or verncular input data.

Quality of life functions

In order to facilitate working with names, we provide the `authority` function (gives the full taxonomic authority for a name), `synonyms` (to get alternative valid names), `vernacular` (for English common names), and `rank` (for the taxonomic rank). These functions are not used in name matching, but are often useful in the post-processing of results.

Taxonomic lineages navigation

The `children` function will return all nodes that are directly descended from a taxon; the `descendants` function will recursively apply this function to all descendants of these nodes, until only terminal leaves are reached. The `parent` function is an "upwards" equivalent, giving the taxon from which a taxon descends; the `lineage` function chains calls to `parent` until either `taxon(1)` (the taxonomy root) or an arbitrary ancestor is reached.

The `taxonomicdistance` function (and its in-place equivalent, `taxonomicdistance!`, which uses memory-efficient re-allocation if the user needs to change the distance between taxonomic ranks) uses the Shimatani (2001) approach to reconstruct a matrix of distances based on taxonomy, which can serve as a rough proxy when no phylogenies are available. This allows coarse estimations of taxonomic diversity based on species lists. The default distance between taxoonmic levels is as in Shimatani (2001) (*i.e.* species have a distance of 0, genus of 1, family of 2, sbu-classes of 3, and everything else 4), but specific scores can be passed for *any* taxonomic level know to the NCBI name table.

Conclusion

`NCBITaxonomy.jl` enables rapid, taxonomically-restricted, adaptive matching for taxonomic names. By implementing various combinations of search strategies, it allows users to (i) optimize the speed of their queries and (ii) avoid usual caveats of simple string matching. Through explicit exceptions, it allows to write code that will handle the possible edge cases that cannot be solved automatically in a way that does not interrupt execution, or requires manual input by the user. Given the breadth of the NCBI taxonomy database, `NCBITaxonomy.jl` is particularly suited to the name cleaning of large datasets of names.

Acknowledgements: This work was supported by funding to the Viral Emergence Research Initiative (VERENA) consortium including NSF BII 2021909 and 2213854 and a grant from Institut de Valorisation des Données (IVADO), by the NSERC Discovery Grants and Discovery Acceleration Supplement programs, and by a donation from the Courtois Foundation. Benchmarking of this package on distributed systems was enabled by support provided by Calcul Québec (`www.calculquebec.ca`) and Compute Canada (`www.computecanada.ca`). TP wrote the initial code, TP and CJC contributed to API design, and all authors contributed to functionalities and usability testing.

References

Barchard, K. and Pace, L. 2011. Preventing human error: The impact of data entry methods on data accuracy and statistical results. - Computers in Human Behavior 27: 1834–1839.

Bezanson, J. et al. 2017. Julia: A Fresh Approach to Numerical Computing. - SIAM Review 59: 65–98.

Carlson, C. J. et al. 2022. The Global Virome in One Network (VIRION): An Atlas of Vertebrate-Virus Associations. - mBio in press.

Chamberlain, S. A. and Szöcs, E. 2013. Taxize: Taxonomic search and retrieval in R. - F1000Research 2: 191.

Conti, M. et al. 2021. Match Algorithms for Scientific Names in FlorItaly, the Portal to the Flora of Italy. - Plants 10: 974.

Dauncey, E. A. et al. 2016. Common mistakes when using plant names and how to avoid them. - European Journal of Integrative Medicine 8: 597–601.

Dikow, T. et al. 2009. Biodiversity Research Based on Taxonomic Revisions - A Tale of Unrealized Opportunities. - In: Diptera Diversity: Status, Challenges and Tools. Brill, pp. 323–346.

Federhen, S. 2012. The NCBI taxonomy database. - Nucleic acids research 40: D136–D143.

Gibb, R. et al. 2021. Data Proliferation, Reconciliation, and Synthesis in Viral Ecology. - BioScience in press.

Lefkowitz, E. J. et al. 2018. Virus taxonomy: The database of the International Committee on Taxonomy of Viruses (ICTV). - Nucleic Acids Research 46: D708–D717.

Melville, J. et al. 2021. A return-on-investment approach for prioritization of rigorous taxonomic research needed to inform responses to the biodiversity crisis. - PLOS Biology 19: e3001210.

Norman, K. E. A. et al. 2020. Taxadb: A high-performance local taxonomic database interface. - Methods in Ecology and Evolution 11: 1153–1159.

Schoch, C. L. et al. 2020. NCBI Taxonomy: A comprehensive update on curation, resources and tools. - Database in press.

Shimatani, K. 2001. On the Measurement of Species Diversity Incorporating Species Differences. - Oikos 93: 135–147.

Wagner, V. 2016. A review of software tools for spell-checking taxon names in vegetation databases. - Journal of Vegetation Science 27: 1323–1327.

Walker, P. J. et al. 2020. Changes to virus taxonomy and the Statutes ratified by the International Committee on Taxonomy of Viruses (2020). - Archives of Virology 165: 2737–2748.