

NCBITaxonomy.jl - rapid biological names finding and reconciliation

Timothée Poisot^{1,2} Rory Gibb^{3,4} Sadie J. Ryan^{5,6,7} Colin J. Carlson^{8,9}

¹ Université de Montréal, Département de Sciences Biologiques, Montréal QC, Canada 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 Emerging Pathogens Institute, University of Florida, Gainesville, FL, United States of America ⁶ School of Life Sciences, University of KwaZulu-Natal, Durban, South Africa ⁷ Department of Geography, University of Florida, Gainesville, FL, United States of America ⁸ Department of Microbiology and Immunology, Georgetown University Medical Center, Georgetown University, Washington, D.C., United States of America ⁹ Center for Global Health Science and Security, Georgetown University Medical Center, Georgetown University, Washington, D.C., United States of America

Correspondance to:

Timothée Poisot — timothee.poisot@umontreal.ca

This work is released by its authors under a CC-BY 4.0 license



Last revision: *June 27, 2022*

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).

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

11 Finally, taxonomic nomenclature changes regularly, with groups being split, merged, or moved to a new
12 position in the tree of life; this is, notably, a common occurrence with viral taxonomy, each subsequent
13 version of which can differ markedly from the last; compare, *e.g.* Lefkowitz et al. (2018) to Walker et al.
14 (2020), where entire viral sub-trees were split, re-organized, and created within just two years. These
15 taxonomic changes have profound implications for the way we perceive biodiversity at global scales
16 (Meier 2009), to the point where taxonomic revisions should sometimes be actively conducted to improve
17 *e.g.* conservation outcomes (Melville et al. 2021).

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

27 All these considerations become important when matching species names both within and across
28 datasets. Let us consider the hypothetical species survey of riverine fishes: European chub, *Cyprinus*
29 *cephalus*, *Leuciscus cephalus*, *Squalius cephalus*. All are the same species (*S. cephalus*), referred to as one of
30 the vernacular (European chub) and two formerly accepted names now classified as synonyms (but still

present in the literature). 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 host-mammal 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 online from the EcoJulia documentation hub <https://docs.ecojulia.org/>, 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 Julia package repository, which is not an ideal solution, and therefore we recommend that users set an environment variable to specify where the data will be loaded from (this path will be created if it doesn't exist):

```
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") # Downloading the files may take a long time
```

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

62 Improved name matching

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

71 The lowercase search can be a preferable alternative to fuzzy string matching. Consider the string
72 `Adeno-associated virus 3b` - it has three names with equal distance (under the Levenstein string
73 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
```

74 Depending on the operating system (and specifically whether it is case-sensitive), either of these three
75 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)
```

76 This returns the correct name.

77 **Name matching output and error handling**

78 When it succeeds, `taxon` will return a `NCBITaxon` object (made of a name string field, and an id numerical
79 field). That being said, the package is designed under the assumption that ambiguities should yield an
80 error for the user to handle. There are two such errors: `NameHasNoDirectMatch` (with instructions about
81 how to possibly solve it, using the `similarnames` function), or a `NameHasMultipleMatches` (listing the
82 possible valid matches, and suggesting to use `alternativetaxa` to find the correct one). Therefore, the
83 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
```

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

89 **Name filtering functions**

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

98 Note that the use of a restricted list of names can have significant performance consequences: compare,
99 for example, the time taken to return the taxon *Pan* in the entire database, in all mammals, and in all
100 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
	yes	0.3	92	27 KiB

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

108 **Quality of life functions**

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

113 **Taxonomic lineages navigation**

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

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

126 **Acknowledgements:** This work was supported by funding to the Viral Emergence Research Initiative
127 (VERENA) consortium including NSF BII 2021909 and a grant from Institut de Valorisation des Données
128 (IVADO), by the NSERC Discovery Grants and Discovery Acceleration Supplement programs, and by a
129 donation from the Courtois Foundation. Benchmarking of this package on distributed systems was
130 enabled by support provided by Calcul Québec (www.calculquebec.ca) and Compute Canada
131 (www.computecanada.ca). TP wrote the initial code, TP and CJC contributed to API design, and all
132 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.
- 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.
- Meier, R. 2009. Biodiversity Research Based on Taxonomic Revisions - A Tale of Unrealized Opportunities -. - In: pp. 323–345.
- 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.

- 160 Wagner, V. 2016. A review of software tools for spell-checking taxon names in vegetation databases. -
161 Journal of Vegetation Science 27: 1323–1327.
- 162 Walker, P. J. et al. 2020. Changes to virus taxonomy and the Statutes ratified by the International
163 Committee on Taxonomy of Viruses (2020). - Archives of Virology 165: 2737–2748.