

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, Département 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).

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



Last revision: *March 2, 2023*

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

11 In addition, taxonomic nomenclature changes regularly, with groups being split, merged, or moved to a new
12 position in the tree of life; often, taxonomic revisions lead to these events occurring simultaneously. This is,
13 notably, a common occurrence with viral taxonomy, each subsequent version of which can differ markedly from
14 the last; compare, *e.g.* Lefkowitz et al. (2018) to Walker et al. (2020), where entire viral sub-trees were split,
15 re-organized, and created within just two years. As a consequence any mapping of names to other biological
16 entities can become outdated, and therefore invalid. These taxonomic changes have profound implications for
17 the way we perceive biodiversity at global scales (Dikow et al. 2009), to the point where taxonomic revisions
18 should sometimes be actively conducted to improve *e.g.* conservation outcomes (Melville et al. 2021).

19 None of these issues, were they to happen in isolation, would be very difficult to deal with. Indeed, performing
20 the lookup for any text string in any database is a trivial operation. But to add to the complexity, one must also
21 consider that most taxa names are at some point manually typed, which has the potential to introduce additional
22 sources of variation in raw data; it is likely to expect that such mistakes may arise when attempting to write
23 down the (perfectly valid) names of the bacterial isolate known as *Myxococcus*
24 *llanfairpwllgwyngyllgogerychwyrndrobwlantysiliogogochensis*, or of the crowned slaty flycatcher
25 *Griseotyrannus aurantioatrocristatus*. These mistakes are more likely when dealing with hyper-diverse samples
26 (demanding to memorize more names), like plant census (Dauncey et al. 2016, Wagner 2016, Conti et al.
27 2021); when dealing with multiple investigators with different knowledge of the taxonomy; and as a result of the
28 estimated error in any data entry exercise, which other fields estimate at up to about 5% (Barchard and Pace
29 2011). As a result, the first question one needs to ask when confronted with a string of character that
30 purportedly points to a node in the tree of life is not “to which entry in the taxonomy database is it associated?”,

31 but “is there a mistake in this name that is likely to render a simple lookup invalid?”.

32 All these considerations become important when matching species names both within and across datasets. Let
33 us consider the hypothetical species survey of riverine fishes: European chub, *Cyprinus cephalus*, *Leuciscus*
34 *cephalus*, *Squalius cephalus*. All are the same species (*S. cephalus*), referred to as one of the vernacular
35 (European chub) and two formerly accepted names now classified as synonyms (but still present in the
36 literature). A simple estimate of diversity based on the user-supplied names would give $n = 4$ species, when
37 there is in fact only one. Some cases can be more difficult to catch; for example, the species *Isoetes minima* is
38 frequently mentioned as *Isætes minima*, because text processing use the “œ” grapheme to mark the “oe”
39 diphthong. When the size of biodiversity datasets increases, and notably when the taxonomic scope of these
40 datasets explodes, including organisms for which “names” are a fuzzier concept (for example, *Influenza A virus*
41 (*A/Sydney/05/97-like(H3N2)*)) is a valid name for a common influenza strain, although one that lacks a
42 taxonomic rank), the feasibility of manual curation decreases.

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

51 **Design principles and comparison to other tools**

52 Based on the author’s experience reconciling lists of thousands of biological names, `NCBITaxonomy.jl` is built
53 around a series of features that allow (i) maximum flexibility when handling names without a direct match, (ii) a
54 bespoke exception system to handle failures to match automatically, and (iii) limits to the pool of potential names
55 in order to achieve orders-of-magnitude speedups when the broad classification of the name to match is known.
56 Adhering to these design principles led to a number of choices. A comparison of the features of different
57 packages, as inferred from their public documentation, is presented in tbl. ??.

58 First, we specifically target programmatic (as opposed to command-line) based approaches, so that the

functionalities of the package can be accessed as part of a larger pipeline. Second, to speed up the queries, we work from a local version of the database, the installation of which is handled at build time by the package itself; each project using the package can use its own version of the taxonomy by specifying a folder where it is stored through an environmental variable. Third, because we *cannot* trust that the names as presented in the original data are correct, we offer case-insensitive search (at no time cost) and fuzzy-matching (at a significant time cost). Either of these strategies can be called only after a case-sensitive, non-fuzzy search yields an exception about the lack of a direct match. Finally, in order to achieve a good performance even when relying on fuzzy matching, we offer the ability to limit the search to specific parts of the taxonomy database. An example of the impact of this feature on the performance of the package is presented below.

Table 1: Comparison of core features of packages offering access to the NCBI taxonomic backbone. “Library”: ability to be called from code. “CLI”: ability to work as a command-line tool. “Local DB”: ability to store a copy of the database locally. “Fuzzy”: ability to perform fuzzy matching on inputs. “Case”: ability to perform case-insensitive search. “Subsets”: ability to limit the search to a subset of the raw database. “Ranks”: ability to limit the search to specific taxonomic ranks. The features of the various packages have been determined from reading their documentation. {tbl. ??}

Tool	Lang.	Library	CLI	Local					Reference
				DB	Fuzzy	Case	Subsets	Ranks	
NCBITaxonomy.jl	Julia	+		+	+	+	+	+	
taxadb	R	+		+			+	+	
taxopy	Python	+		+		+			
rentrez	R	+						+	
Taxonkit	Python		+	+					
NCBI-taxonomist	Python		+	+					

Overview of functionalities

An up-to-date version of the documentation for NCBITaxonomy.jl can be found in the package’s *GitHub* repository ([PoisotLab/NCBITaxonomy.jl](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, all of which can be consulted publicly. Alternatively, the package can be downloaded from its Zenodo page (ID [5825828](https://zenodo.org/record/5825828)), along with a versioned DOI.

74 Local file storage

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

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

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

```
using Pkg  
Pkg.add("NCBITaxonomy")
```

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

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

97 Improved name matching

98 Name finding, *i.e.* the matching of an arbitrary string to a taxonomic identifier, is primarily done through the
99 `taxon` function, which admits either a unique NCBI identifier (*e.g.* `taxon(36219)` for the bogue *Boops boops*),
100 a string (`taxon("Boops boops")`), or a data frame with a restricted list of names in order to create a name finder
101 function (see the next section). The `taxon` method has additional arguments to perform fuzzy matching in order
102 to catch possible typos (`taxon("Boops bops"; strict=false)`), to perform a lowercase search (useful when
103 alphanumeric codes are part of the taxon name, like for some viruses), and to restrict the the search to a specific
104 taxonomic rank. The `taxon` function also accepts a `preferscientificname` keyword, to prevent matching
105 vernacular names; the use of this keyword ought to be informed by knowledge about how the data were entered.
106 The lowercase search can be a preferable alternative to fuzzy string matching. Consider the string
107 `Adeno-associated virus 3b` - it has three names with equal distance (under the Levenstein string distance
108 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
```

109 Depending on the operating system (and specifically whether it is case-sensitive), either of these three names
110 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)
```

111 This returns the correct name.

112 Name matching output and error handling

113 When it succeeds, `taxon` will return a `NCBITaxon` object (made of a name string field, and an id numerical
114 field). That being said, the package is designed under the assumption that ambiguities should yield an error for

115 the user to handle. There are two such errors: `NameHasNoDirectMatch` (with instructions about how to possible
116 solve it, using the `similarnames` function), or a `NameHasMultipleMatches` (listing the possible valid matches,
117 and suggesting to use `alternativetaxa` to find the correct one). Therefore, the common way to work with the
118 `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
```

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

124 Name filtering functions

125 As the full NCBI names table has over 3 million entries at the time of writing, we have provided a number of
126 functions to restrict the scope of names that are searched. These are driven by the NCBI *divisions*. For example
127 `nf = mammalfilter(true)` will return a data frame containing the names of mammals, inclusive of rodents
128 and primates, and can be used with *e.g.* `taxon(nf, "Pan")`. This has the dual advantage of making search
129 faster, but also of avoiding matching on names that are shared by another taxonomic group (which is not an

130 issue with *Pan*, but is an issue with *e.g. Io* as mentioned in the introduction, or with the common name *Lizard*,
 131 which fuzzy-matches on the hemipteran genus *Lisarda* rather than the class *Lepidosauria*).

132 Note that the use of a restricted list of names can have significant performance consequences: compare, for
 133 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
	yes	0.3	92	27 KiB

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

140 **Quality of life functions**

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

145 **Taxonomic lineages navigation**

146 The `children` function will return all nodes that are directly descended from a taxon; the `descendants` function
 147 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 taxonomic levels is as in Shimatani (2001) (*i.e.* species have a distance of 0, genus of 1, family of 2, sub-classes of 3, and everything else 4), but specific scores can be passed for *any* taxonomic level known 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.

174 Bezanson, J. et al. 2017. [Julia: A Fresh Approach to Numerical Computing](#). - SIAM Review 59: 65–98.

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

177 Chamberlain, S. A. and Szöcs, E. 2013. [Taxize: Taxonomic search and retrieval in R](#). - F1000Research 2: 191.

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

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

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

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

185 Gibb, R. et al. 2021. [Data Proliferation, Reconciliation, and Synthesis in Viral Ecology](#). - BioScience in press.

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

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

190 Norman, K. E. A. et al. 2020. [Taxadb: A high-performance local taxonomic database interface](#). - Methods in
 191 Ecology and Evolution 11: 1153–1159.

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

194 Shimatani, K. 2001. [On the Measurement of Species Diversity Incorporating Species Differences](#). - Oikos 93:
 195 135–147.

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

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