Ngesh, a tool for simulating random phylogenetic trees

DOI 10.5281/zenodo.2619311

`ngesh` is a Python library for simulating random phylogenetic trees and related data. It is intended for benchmarking phylogenetic methods and for providing dummy trees during the development or debugging of phylogenetic methods. The generation of random phylogenetic trees also goes by the name of "simulation methods for phylogenetic trees" or just "simulating phylogenetic trees".

Please note that this is both a work in progress and a library intended to be simple, for complex methods see the bibliography and the alternatives here listed

In detail, with ngesh:

- trees are returned either as strings in Newick representation or as Python's ETE tree objects
- trees will also have random branch-lengths, which you can remove if are only interested in the topology
- trees can be generated from user-provided seeds, so that the random generation can be reproduced across executions (while it cannot be guaranteed due to many technical reasons, the execution should be reproducible *also* in different machines and different version of Python)
- trees can be generated according to user-provided birth and death ratios (and the death ratio can be set to zero, resulting in a birth-only tree)
- non-extant leaves can be pruned from birth-death trees
- speciation events default to two descendants, but the number of descendants can be randomly drawn from a user-defined Poisson process (so that it is possible to model hard politomies)
- trees can be limited in terms of number of extant leaves, evolution time (as related to the birth and death parameters), or both
- nodes can optionally receive unique labels, either sequential ones (like "L01", "L02", and "L03"), random human-readable names (like "Sume", "Fekobir", and "Tukok"), or random biological names approximating the binomial nomenclature standard (like "Sburas wioris", "Zurbata ceglaces", and "Spellis spusso")

Installation

In any standard Python environment, `ngesh` can be installed with:

```
pip install ngesh
```

The `pip` installation will also fetch the dependencies `ete3` and `numpy`, if necessary.

How to use

You can test your installation from the command line with the `ngesh` command, which will return a different random small birth-death tree in Newick format each time it is called:

```
$ ngesh
(Saorus  getes:1.31562,((Voces  earas:1.07567,(Dallao  spettus:0.703609,Sburas
wioris:0.703609)1:0.372063)1:0.464667,(Zurbaza        ceglaces:0.527431,(Amduo
vizoris:0.345862,Uras wiurus:0.345862)1:0.18551)1:1.00897)1:2.1707);

$ ngesh
((Ollio             zavis:0.698453,(Spectuo             sicui:0.596731,((Ronis
mivulis:0.0431014,Vaporus             conomattas:0.0431014)1:0.413634,Rizarus
urrus:0.456735)1:0.139996)1:0.101722)1:3.17827,(Deses mepus:2.22061,(Ovegpuves
wiumoras:1.88469,(Easas                              ecdebus:0.201891,Muggas
lupas:0.201891)1:1.6828)1:0.335918)1:1.65611);
```

More complex illustrations can also be executed from the command line. If you have PyQt5 installed (which is listed as a dependency), the following code will pop up the ETE Tree Viewer on a random tree:

```
python3 -c "import ngesh ; ngesh.display_random_tree()"
```

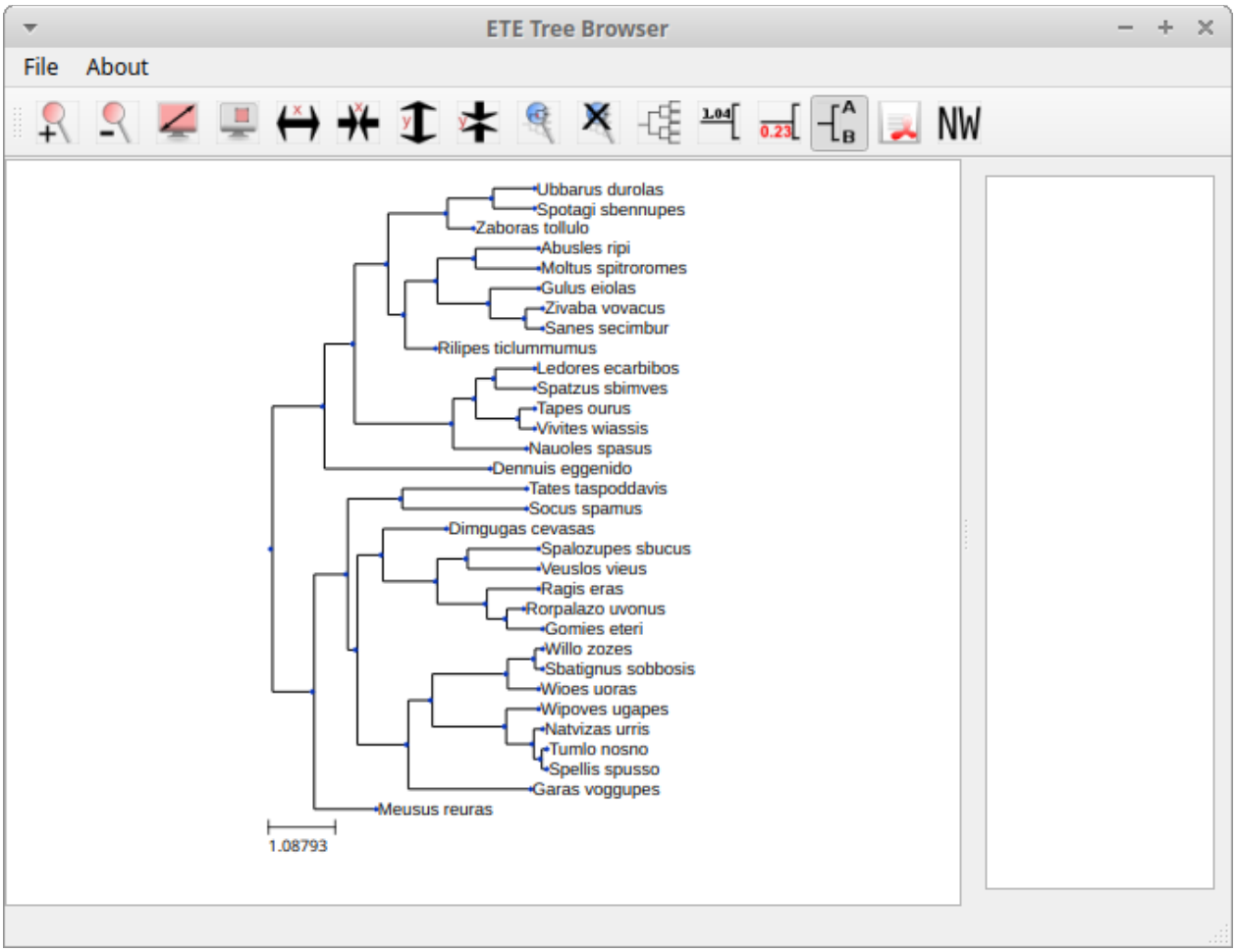


Figure 1: random tree

The package is designed to be used a library. The main function is the `gen_tree()` function, which will return an ETE tree object, which can be printed either in ASCII art or as a Newick tree.

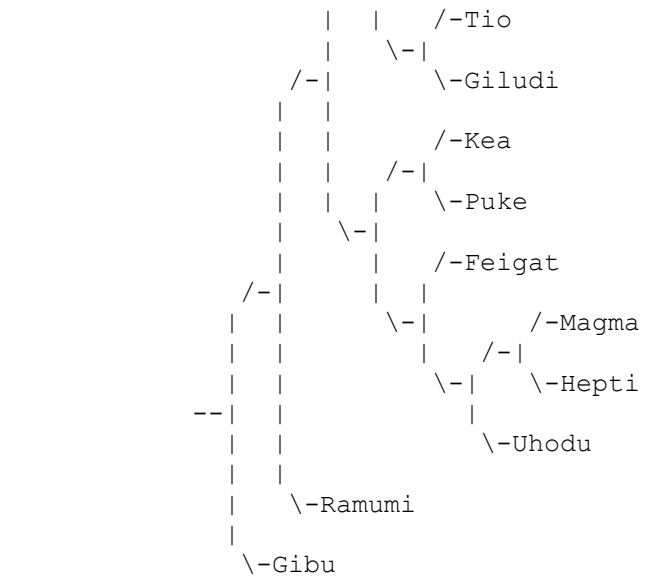
```
$ ipython3

In [1]: import ngesh

In [2]: tree = ngesh.gen_tree(1.0, 0.5, max_time=3.0, labels="human")

In [3]: print(tree)

                    /-Gupe
                /-|
            /-|   \-Ui
           |  |
         /-|   \-Otevze
        |  |
```




```
In [4]: print(tree.write())
((((((Gupe:0.221592,Ui:0.221592)1:0.274564,Otevze:0.130366)1:0.208385,
(Tio:0.114998,Giludi:0.102135)1:0.589543)1:0.979981,
((Kea:0.223773,Puke:0.901625)1:0.115261,(Feigat:0.944765,
((Magma:0.541031,Hepti:0.541031)1:0.120214,Uhodu:0.661245)1:0.28352)1:0.072120
7)1:0.667637)1:0.119319,Ramumi:1.24512)1:1.03792,Gibu:0.663448);
```

The full documentation for the `gen_tree()` function is available in its docstring (which can be visualized with `print(ngesh.gen_tree.__doc__)` or directly in the source code. The most important parameters are:

- `birth`, the birth rate (i.e., lambda)
- `death`, the death rate (i.e., mu)
- `num_leaves` and `max_time`, which are stopping parameters indicating the minimum number of extant leaves or the maximum evolutionary time
- `label`, which indicates which kind of random labels should be given to the nodes (`None`, `"enum"` for a simple enumeration, `"human"` for randomly generated names, and `"bio"` for randomly generated specie names)

Other parameters allow to tweak the birth-process, allowing polytomies, prune the tree for extinct lineages, and a seed for the random generator that guarantees reproducibility.

How to cite

If you use `ngesh`, please cite it as:

> Tresoldi, Tiago (2019). Ngesh, a tool for simulating random phylogenetic trees. Version 0.1.1. Jena. Available at: https://github.com/tresoldi/ngesh

In BibTeX:

```
@misc{Tresoldi2019ngesh,
```

```
  author = {Tresoldi, Tiago},
   title = {Ngesh, a tool for simulating random phylogenetic trees. Version
0.1.2},
  howpublished = {\url{https://github.com/tresoldi/ngesh}},
  address = {Jena},
  year = {2019},
  doi = {10.5281/zenodo.2619311},
}
```

How does ngesh work?

For each tree, an `event_rate` is computed from the sum of the `birth` and `death` rates. At each iteration, which takes place after an random expovariant time from the `event_rate`, one of the extant nodes is selected for an "event": either a birth or a death from the proportion of each rate. All other extant leaves have their distances updated with the event time.

The random labels follow the expected methods for random text generation from a set of patterns, taking care to generate names as universally readable (if not pronounceable) as possible.

What does "ngesh" mean?

Technically it is just an unique name, but it was derived from one of the Sumerian words for "tree", ĝeš, albeit with an uncommon transcription. The name comes from the library once being a module of a larger system for simulating language evolution and benchmarking related tools, called Enki after the Sumerian god of (among many other things) language and mischief.

The intended pronounciation, as in the most accepted reconstructions, is /ŋeʃ/. But don't strees over it: it is just a unique name.

Alternatives

There are many tools for simulating phylogenetic processes in order to obtain random phylogenetic trees. The most complete is probably the R package `TreeSim` by Tanja Stadler, which includes many flexible tree simulation functions. In R, one can also use the `rtree()` function from package `ape` and the `birthdeath.tree()` one from package `geiger`, as well as manually randomizing taxon placement in cladograms.

In Python, some code similar to `ngesh` and which served as initial inspiration is provided by Marc-Rolland Noutahi on the blog post How to simulate a phylogenetic tree ? (part 1).

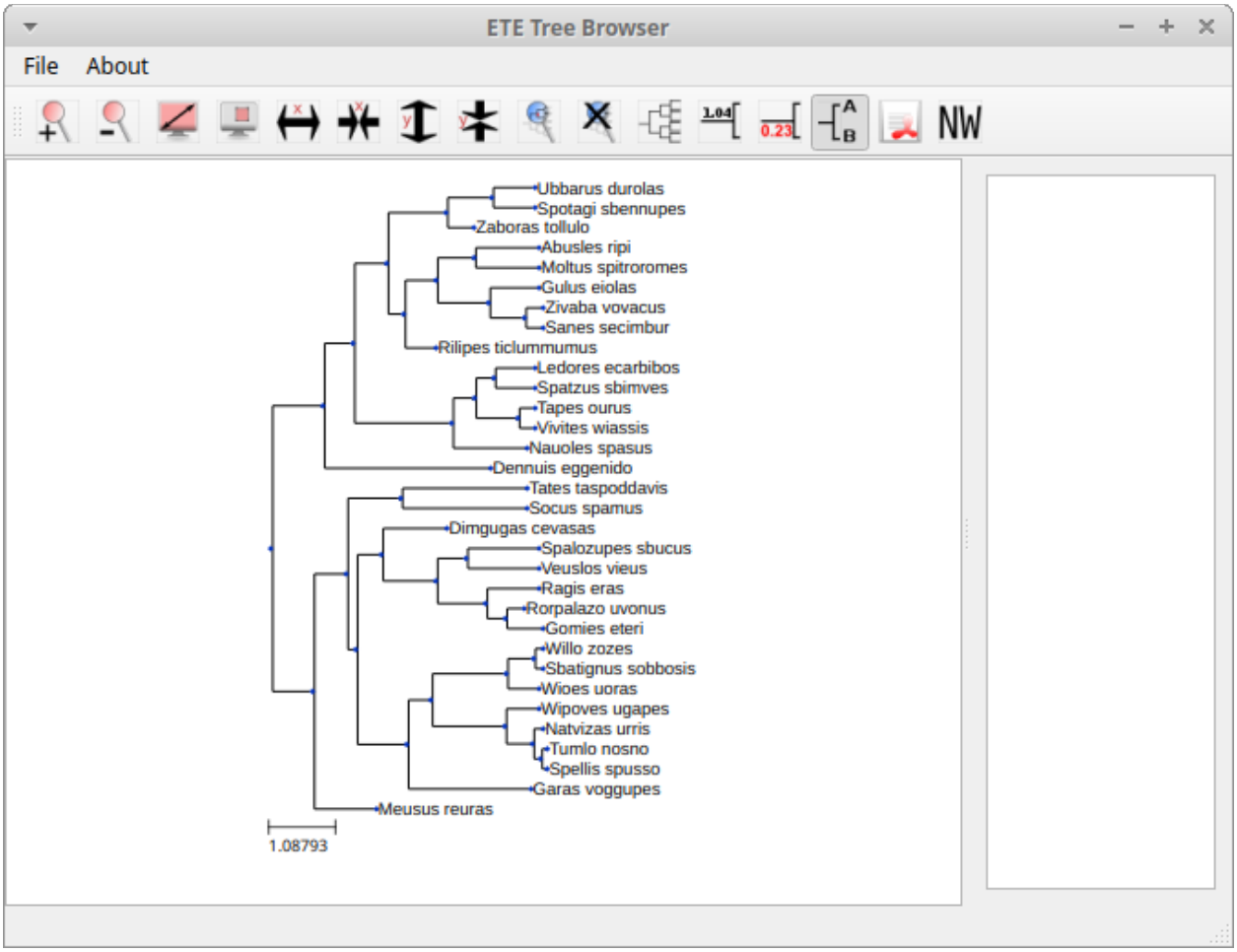
A number of on-line tools are also available at the time of writing:

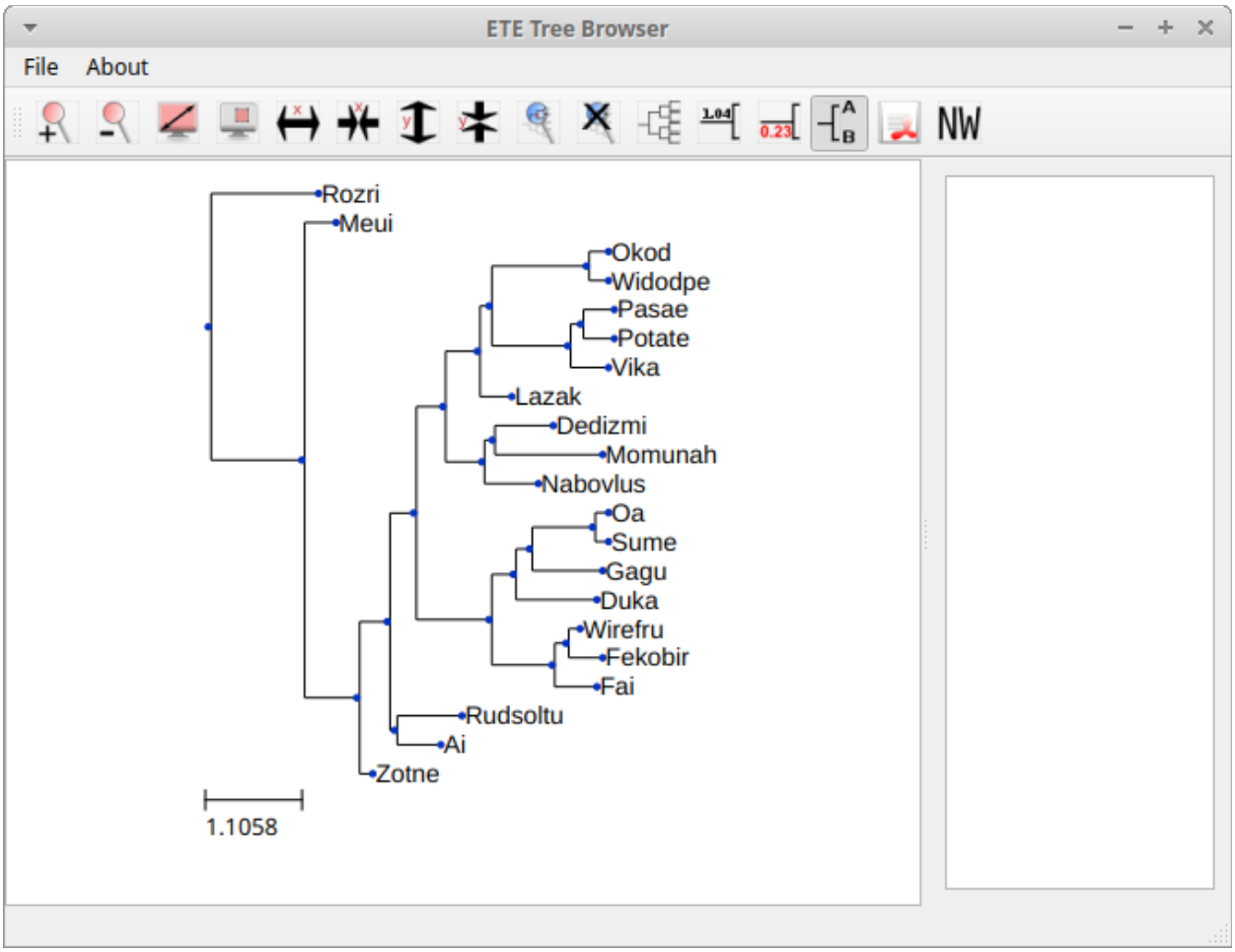
- T-Rex (Tree and reticulogram REConstruction at the Université du Québec à Montréal (UQAM)
- Anvi'o Server can be used on-line as a wrapper to T-Rex above
- phyloT, which by randomly sampling taxonomic names, identifiers or protein accessions can be used for the same purpose

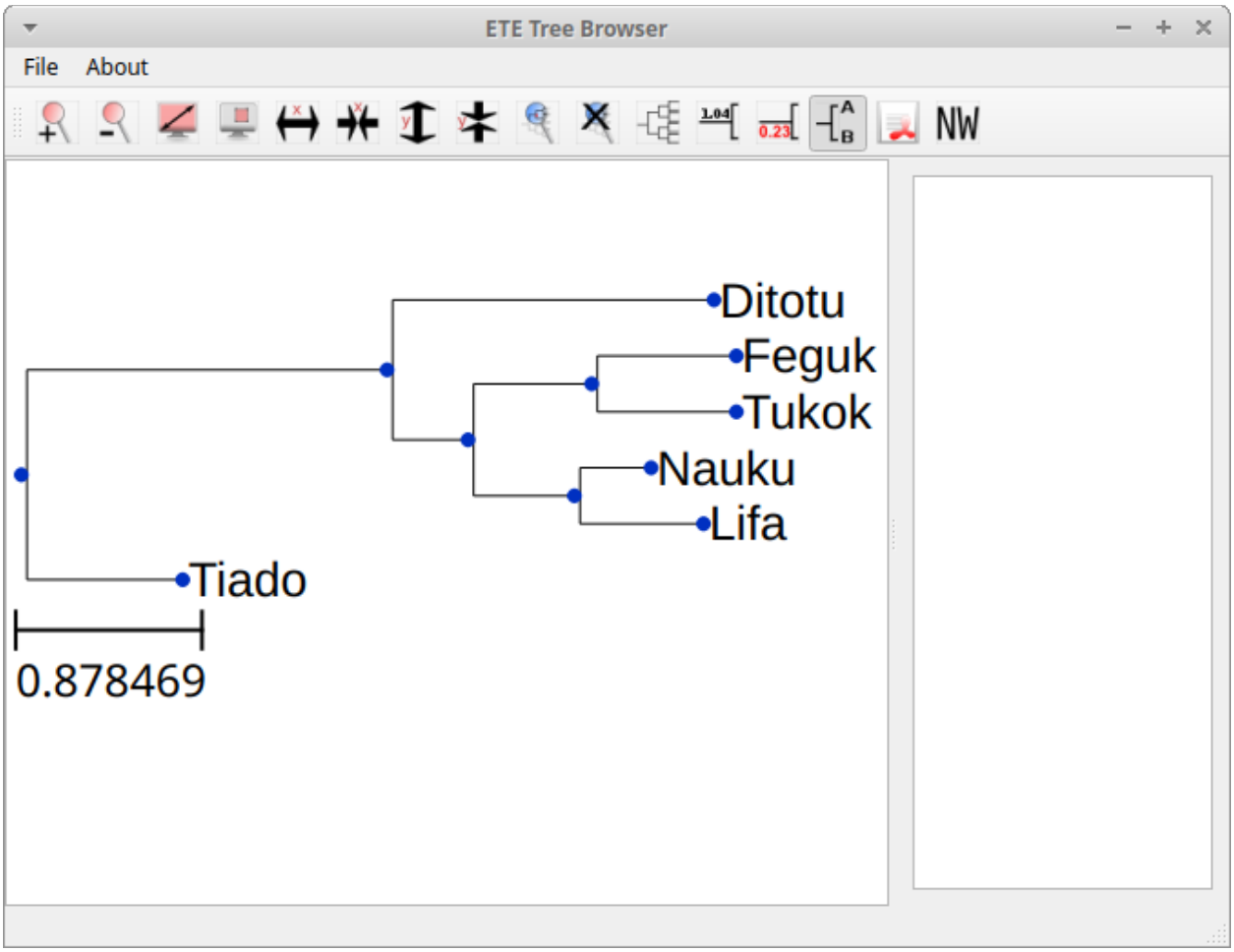
TODO

- Add stoping criterion of total number of nodes (in complement to the number of *extant* nodes currently available)
- Rewrite the random label generators into actual Python generators
- Better implementation of the scientific name generator, exporting the function and guaranteeing there are no duplicates
- Import the text generators from Enki or similar methods
- Consider replacing `expovariate` with an actual random Poisson sampling, or allow multiple models
- Build a simple website and link to automatically generated documentation

Gallery




ETE Tree Browser
File
About
NW
Rozri
Meui
Okod
Widodpe
Pasae
Potate
Vika
Lazak
Dedizmi
Momunah
Nabovlus
Oa
Sume
Gagu
Duka
Wirefru
Fekobir
Fai
Rudsoltu
Ai
Zotne
1.1058



References

- Bailey, N. T. J. (1964). *The elements of stochastic processes with applications to the natural sciences*. John Wiley & Sons.
- Foote, M., J. P. Hunter, C. M. Janis, and J. J. Sepkoski Jr. (1999). *Evolutionary and preservational constraints on origins of biologic groups: Divergence times of eutherian mammals*. Science 283:1310–1314.
- Harmon, Luke J (2019). *Phylogenetic Comparative Methods – learning from trees*. Available at: https://lukejharmon.github.io/pcm/chapter10_birthdeath/. Access date: 2019-03-31.
- Noutahi, Marc-Rolland (2017). *How to simulate a phylogenetic tree? (part 1)*. Available at: https://mrnoutahi.com/2017/12/05/How-to-simulate-a-tree/. Access date: 2019-03-31
- Stadler, Tanja (2011). *Simulating Trees with a Fixed Number of Extant Species*. Systematic Biology 60.5:676-684. DOI: https://doi.org/10.1093/sysbio/syr029

Author

Tiago Tresoldi (tresoldi@shh.mpg.de)

The author was supported during development by the ERC Grant #715618 for the project CALC (Computer-Assisted Language Comparison: Reconciling Computational and Classical Approaches in Historical Linguistics), led by Johann-Mattis List