Ngesh: a Python library for phylogenetic simulation

Tiago Tresoldi[1, 2]

1 Department of Linguistics and Philology, Uppsala University **2** Department of Linguistic and Cultural Evolution, Max Planck Institute for the Science of Human History

DOI:

Software

- Review
- Repository
- Archive

Editor:

Submitted:
Published:

Summary

This work presents `ngesh`, a Python library for simulating phylogenetic trees and data designed for research in Historical Linguistics and Stemmatics. It generates reproducible stochastic simulations of evolution according to various criteria, including character mutation rates and lateral transmission probability. Many output formats are supported and results can simulate inadequate data compilation. The library is designed for usage in development, debugging, and benchmarking of software pipelines and methods for phylogenetic inference.

Background

Computational phylogenetics is being increasingly accepted in fields beyond biology, such as historical linguistics (Bouckaert et al., 2012) and stemmatics (Robinson, 2016). Stochastic simulations, long advocated for natural sciences in general (Bailey, 1964) and genetics in specific (Foote, Hunter, Janis, & Sepkoski, 1999; Harmon, 2019), should be extended to these new approaches. This way, evolutionary analogies and methods' performance can be evaluated through vast amounts of simulated histories, without limits imposed by data availability and collection time, along with quantifiable precision of results. Simulations also allow basic fuzzy testing of software and support studies on which evolutionary models, processes, and evolutionary parameters better match observed phenomena.

The `ngesh` library is a tool for setting up these kind of simulations, designed for easy integration into phylogenetic pipelines. It can generate reproducible trees and correlated data from random seeds, following both user-established parameters, such as rates of birth and death, and constrains, such as branch length and the node number. Character evolution related to the tree topology can likewise be simulated, including *ex novo* mutations and horizontal gene transfers. Results can be manipulated in diverse manners, for example by pruning extinct leaves or simulating uneven sampling. The library can label taxa in progression or randomly with either names easy to pronounce (e.g. "Sume" and "Fekobir") or binominal nomenclature-like (e.g. "Sburas wioris" and "Zurbata pusso"). The simulated trees are standard ETE3 objects (Huerta-Cepas, Serra, & Bork, 2016) and may be exported into different formats like Newick trees, ASCII-art representation, and tabular lists.

The library is proposed as a building block for evaluating software pipelines. It is a curated alternative to the basic technique of randomizing taxa placement in existing cladograms, and to simpler tools such as the one by Noutahi (2017) or the `populate()` method of ETE3's `Tree` class (Huerta-Cepas et al., 2016). Within its intended scope, it compares favorably to popular alternatives, including the R `TreeSim` (Stadler, 2011) and `geiger` packages (Pennell et al., 2014) and the `rtree()` function of `ape` (Paradis & Schliep, 2018), because of its specific support for historical linguistics and stemmatics, as well of its availability as a stand-alone tool.

Installation, Usage, & Examples

Users can install the library with the standard pip tool for managing Python packages. Trees can be generated from the command-line, defaulting to small phylogenies in Newick format:

```
$ ngesh
(Ukis:1.11985,(Koge:0.880823,(Rozkob:0.789548,(Meu:0.706601,
(((Felbuh:0.189693,Kefa:0.189693)1:0.117347,((Epib:0.153782,
Vugog:0.153782)1:0.0884745,Puluk:0.242256)1:0.0647836)1:0.0469885,
Efam:0.354028)1:0.352573)1:0.0829465)1:0.0912757)1:0.23903);
```

The tool supports both configuration files and command-line flags that take precedence over the former. Here we specify a model to generate Nexus data for a reproducible Yule tree, with a birth rate of 0.75, at least 5 leaves, "human" labels, and 20 presence/absence features:

```
$ cat my_tree.conf
[Config]
labels=human
birth=0.75
death=0.0
output=nexus
min_leaves=5
num_chars=20
$ ngesh -c my_tree.conf --seed 12345
begin data;
  dimensions ntax=6 nchar=33;
  format datatype=standard missing=? gap=-;
  matrix
Buza      111110110111011011010101000100110
Lenlar    111111010110111101100010010011001
Mukom     111110111011011011101001000100110
Pagil     111110110111011011100100100100110
Suglu     111110110111011011100011001001010
Wite      111110110111011011100101000100110
  ;
end;
```

Despite the benefit of a stand-alone tool, the package is designed to be run as a library. The two primary functions are `gen_tree()`, which returns a random tree, and `add_characters()`, which adds character evolution data to a tree. Users can therefore generate random trees without character information or simulate character evolution within existing trees, including non-simulated ones.

```
>>> import ngesh
>>> tree = ngesh.gen_tree(1.0, 0.5, max_time=0.3, labels="bio",
                          seed="135")
>>> print(tree)

   /-Lubedsas larpes
--|
  |   /-Rasso wimapudda
   \-|
      \-Sbaes rapis
>>> print(tree.write())
(Lubedsas larpes:0.201311,(Rasso wimapudda:0.0894405,Sbaes rapis:0.0894405)
1:0.11187);
>>> tree = ngesh.add_characters(tree, 15, 2.0, 0.5)
```

Besides the `write()` method of the example above, which outputs Newick trees, results can be exported in NEXUS format with the `tree2nexus()` function and in tabular output, appropriate for BEASTling (Maurits, Forkel, Kaiping, & Atkinson, 2017), with `tree2wordlist()`.

Code and Documentation Availability

The `ngesh` source code is available on GitHub at https://github.com/evotext/ngesh.

The full user documentation is available at https://ngesh.readthedocs.io/.

Acknowledgements

The author has received funding from the Riksbankens Jubileumsfond (ID: MXM19-1087:1, "Cultural Evolution of Texts") and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (No. ERC #715618, "Computer-Assisted Language Comparison").

References

Bailey, N. T. (1964). *The elements of stochastic processes with applications to the natural sciences*. New York, London, Sydney: John Wiley & Soins.

Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S. J., Alekseyenko, A. V., Drummond, A. J., Gray, R. D., et al. (2012). Mapping the origins and expansion of the indo-european language family. *Science*, *337*(6097), 957–960.

Foote, M., Hunter, J. P., Janis, C. M., & Sepkoski, J. J. (1999). Evolutionary and preservational constraints on origins of biologic groups: Divergence times of eutherian mammals. *Science*, *283*(5406), 1310–1314. doi:10.1126/science.283.5406.1310

Harmon, L. J. (2019). *Phylogenetic comparative methods*. University of Idaho.

Huerta-Cepas, J., Serra, F., & Bork, P. (2016). ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data. *Molecular Biology and Evolution*, *33*(6), 1635–1638. doi:10.1093/molbev/msw046

Maurits, L., Forkel, R., Kaiping, G. A., & Atkinson, Q. D. (2017). BEASTling: A software tool for linguistic phylogenetics using beast 2. *PloS One*, *12*(8).

Noutahi, M.-R. (2017). How to simulate a phylogenetic tree? Retrieved from https://mrnoutahi.com/2017/12/05/How-to-simulate-a-tree/

Paradis, E., & Schliep, K. (2018). Ape 5.0: An environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics*, *35*, 526–528.

Pennell, M., Eastman, J., Slater, G., Brown, J., Uyeda, J., FitzJohn, R., Alfaro, M., et al. (2014). Geiger v2.0: An expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. *Bioinformatics*, *30*, 2216–2218.

Robinson, P. (2016). The digital revolution in scholarly editing. *Ars Edendi Lecture Series*, *4*, 181–207.

Stadler, T. (2011). Simulating trees with a fixed number of extant species. *Systematic Biology*, *60*(5), 676–684. doi:10.1093/sysbio/syr029