Generating heterogeneous data on gene trees

Martí Cortada Garcia, Adrià Diéguez Moscardó, Marta Casanellas

Abstract

We introduce GenPhylo, a Python module that simulates genetic data along a phylogeny avoiding the restriction of continuous-time Markov processes. GenPhylo uses directly a general Markov model and therefore naturally incorporates heterogeneity across lineages. We solve the challenge of generating transition matrices with a pre-given expected number of substitutions (the branch length information) by providing an algorithm that can be incorporated in other simulation software.

1 Introduction

When testing or training phylogenetic reconstruction methods, it is necessary to have synthetic data simulated by running a Markov process of state substitution on a phylogenetic (gene) tree. Usually, gene trees trees that represent the evolution of DNA sequences have branch lengths accounting for the expected number of nucleotide substitutions per site. When the Markov process of nucleotide substitution is assumed to be time-continuous, generating transition matrices with a pre-given number of substitutions is straightforward and it is what most phylogenetic software does [Nguyen et al., 2014, Rambaut and Grass, 1997, Schaller et al., 2022, Spielman and Wilke, 2015, Sjöstrand et al., 2013, Mallo et al., 2016]. However, assuming that the process is time-continuous might be very restrictive: less than 4% of Diagonal Largest in Column (DLC) matrices of size 4 are of continuous-type, see [Casanellas et al., 2023] (see also [Chang, 1996] for the importance of DLC in identifiability of substitution parameters). Another drawback arises when restricting to continuous-time processes: the simulation of heterogeneous rates across lineages has to be handled via some probabilistic model.

When one avoids the time-continuous assumption and considers a general Markov process, one has to face the difficult task of generating transition matrices with a pre-given number of substitutions. This was addressed in [Kedzierska and Casanellas, 2012] for very simple models of nucleotide substitution and a partial solution was proposed for the general Markov model (only working for uniform distribution of nucleotides).

In this paper we present a way of generating a random transition matrix M for an edge $e : u \longrightarrow v$ given an expected number b of substitutions per site and distribution π at u. According to Lake [Lake, 1994], b can be approximated by

$$b = -\frac{1}{4} \ln \frac{\sqrt{\det D_\pi} \det M}{\sqrt{\det D_{\pi'}}}, \tag{1}$$

where $\pi' = \pi M$ and D_π and $D_{\pi'}$ are the diagonal matrices with the corresponding node distributions.

We have implemented this method for DNA and in the software `GenPhylo` we use it to provide multiple sequence alignments evolving on a phylogenetic tree with given branch lengths. Our algorithm in python can also be imported to be used in larger simulation software such as [Spielman and Wilke, 2015].

2 Design and implementation

2.1 Generating DLC Markov matrices with a prefixed number of substitutions

Consider and edge between two nodes $u \longrightarrow v$. For a given distribution π at node u and for a given branch length b we want to generate a Markov matrix M such that $\pi' = \pi M$ (the distribution at node v) and M satisfy (1). As we need to control both the determinant of M and the distribution πM, we produce M as the product of two Markov matrices, $M = M_1 M_2$: M_1 will be a random Markov matrix satisfying mild constrains which will give the final distribution $\pi' = \pi M_1$, and M_2 will be a time-reversible Markov matrix with stationary distribution π' and whose determinant is adjusted such that (1) is satisfied. As the determinant of a Markov matrix is bounded above by one, one has to carefully check that the procedure can be executed. The full process is described in Algorithms 1 and 2 and further explained below.

We use a Dirichlet distribution $Dir(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ to generate the rows of random Markov matrices. One has to be careful when choosing the parameters for this distribution: on the one hand we want to generate DLC matrices but on the other hand we need (1) to hold. To this end, for each row i, parameters α_j are set to 1 if $j \neq i$ and parameter α_i is given as a function $\frac{ke^{-b}}{\sqrt{b}}$ where k is aimed to be the smallest value for which a matrix sampled from this Dirichlet distribution satisfies the desired determinant bounds. These parameters for the Dirichlet distribution in each row are referred to as $Dirichlet(b)$ in Algorithms 1 and 2.

In Algorithm 2, we first produce a random time-reversible matrix M_2 with stationary distribution π' using the Metropolis-Hastings method [Dunn and Shultis, 2011]. Then we seek for a new time-reversible matrix $\tilde{M}_2$ (with same stationary distribution π') such that its determinant equals

$$d_2 = \frac{e^{-4b}\sqrt{\det D_{\pi'}}}{\det(M_1)\sqrt{\det D_\pi}}. \tag{2}$$

To do so, we consider matrices $\tilde{M}_2 = (1-a)M_2 + aI$ and determine the value of a by imposing $\det \tilde{M}_2 = d_2$. With this process we obtain a Markov matrix with stationary distribution π' and with determinant d_2.

We check whether the final matrix $M = M_1 M_2$ is DLC and we repeat the process otherwise (we set a maximum of 5 repetitions as in most cases the algorithm already produces DLC matrices, even when using long branches as of 1.2).

Algorithm 1 Algorithm to compute M_1

```
Input: node distribution $\pi$, branch length $b$
  while number of iterations $< 50$ do
    $M_1 \leftarrow Dirichlet(b)$
    $\pi' \leftarrow \pi M_1$
    if $\frac{\exp(-4b)\sqrt{\det(D_{\pi'})}}{\sqrt{\det(D_\pi)}} < 1$ then
      if $\det(M_1) > \frac{\exp(-4b)\sqrt{\det(D_{\pi'})}}{\sqrt{\det(D_\pi)}}$ then
        break and return $M_1$
      end if
    else
      repeat
    end if
  end while
Output: $M_1$
```

Algorithm 2 Algorithm to compute M_2 using Metropolis-Hastings

```
Input: distribution $\pi_1$, $M_1$, and branch length $l$
  $P \leftarrow Dirichlet(b)$
  while number of iterations $< 50$ do
    repeat $(M_2)_{i,j} \leftarrow \begin{cases} P(i,j)\alpha(i,j) & \text{if } i \neq j \\ P(i,i) + \sum_{k\neq i} P(i,k)(1-\alpha(i,k)) & \text{otherwise} \end{cases}$
    until stationary distribution of $M_2$ approximates $\pi_1$.
  end while
  then
  Find $a \in (0,1)$ root of polynomial s.t. $\tilde{M_2} = (1-a)M_2 + aI$ satisfies (2)
  $M_2 = \tilde{M_2}$
Output: $M_2$
```

2.2 Sampling sequences from the Markov process on a phylogenetic tree

Given a Newick tree with branch lengths, we root the tree at an interior node and assign to it a distribution π from $Dir(1,1,1,1)$ (unless the distribution is specified by the user). Then for each edge incident to the root r we compute a Markov matrix with the desired branch length as indicated in the previous section. We compute the distribution at the adjacent nodes to the root and repeat the process until a Markov matrix M^e is assigned to each edge e.

We generate a sequence of a specific length L at the root r of the tree by sampling from the distribution at the root. Then, for each edge e directed from r and for each site in the sequence we read the nucleotide and we generate the nucleotide at the child node by sampling from the distribution at the corresponding row of the Markov matrix M^e. We iterate this process throughout the tree from deeper to outer nodes.

2.3 Implementation and execution time

These methods have been implemented in Python3 in our module `GenPhylo` [Cortada et al., 2024a], which has been released in the Python Package Index (PyPI) and can be installed using `pip install GenPhylo`. Additionally, we also offer the option to run the simulator without installing the package by simply cloning our repository [Cortada et al., 2024b] and running the `GenPhylo.py` script from the terminal with the appropriate parameters.

In Table 1 we provide the running time of our algorithm. Note that our algorithm encompasses a rich level of heterogeneity at the expense of speed.

Leaves	**Branch length**	**Alignment length**	
		1000	10000
4	0.1	1.20	1.51
	0.5	1.33	1.76
8	0.1	2.22	3.06
	0.5	2.64	3.53
16	0.1	4.15	6.17
	0.5	5.09	7.11

Table 1: Average execution time for generating 100 alignments on balanced trees with the given branch lengths (same at all branches) and alignment lengths. Computations run on a M1 Mac with an ARM-based processor.

Author's contributions

Last author conceived the idea and the first two authors equally contributed to develop and implement it; all authors wrote and revised the manuscript.

Acknowledgments

We thank Jesús Fernández-Sánchez for useful discussions on this topic. MC was partially supported by grants PID2019-103849GB-I00 and PID2023-146936NB-I00 funded by MICIU/AEI/ 10.13039/501100011033 and María de Maeztu Program for Centers and Units of Excellence in R&D (project CEX2020-001084-M) and the AGAUR project 2021 SGR 00603 Geometry of Manifolds and Applications, GEOMVAP.

References

[Casanellas et al., 2023] Casanellas, M., Fernández-Sánchez, J., and Roca-Lacostena, J. (2023). The embedding problem for Markov matrices. *Publicacions Matemàtiques*, 67:411–445.

[Chang, 1996] Chang, J. T. (1996). Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, 137(1):51–73.

[Cortada et al., 2024a] Cortada Garcia, M., Diéguez Moscardó, A., and Casanellas, M. (2024). GenPhylo module, https://pypi.org/project/GenPhylo/

[Cortada et al., 2024b] Cortada Garcia, M., Diéguez Moscardó, A., and Casanellas, M. (2024). GenPhylo Project repository. https://github.com/GenPhyloProject/GenPhylo

[Dunn and Shultis, 2011] Dunn, W. and Shultis, J. (2011). *Exploring Monte Carlo Methods*. Elsevier Science & Technology.

[Kedzierska and Casanellas, 2012] Kedzierska, A. M. and Casanellas, M. (2012). GenNon-h: Generating multiple sequence alignments on nonhomogeneous phylogenetic trees. *BMC Bioinformatics*, 13(1):216.

[Lake, 1994] Lake, J. A. (1994). Reconstructing evolutionary trees from DNA and protein sequences: paralinear distances. *Proceedings of the National Academy of Sciences*, 91(4):1455–1459.

[Mallo et al., 2016] Mallo, D., De Oliveira Martins, L., and Posada, D. (2016). Simphy: Phylogenomic simulation of gene, locus, and species trees. *Systematic Biology*, 65(2):334–344.

[Nguyen et al., 2014] Nguyen, L.-T., Schmidt, H. A., von Haeseler, A., and Minh, B. Q. (2014). IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies. *Molecular Biology and Evolution*, 32(1):268–274.

[Rambaut and Grass, 1997] Rambaut, A. and Grass, N. C. (1997). Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Bioinformatics*, 13(3):235–238.

[Schaller et al., 2022] Schaller, D., Hellmuth, M., and Stadler, P. F. (2022). Asymmetree: A flexible python package for the simulation of complex gene family histories. *Software*, 1(3):276–298.

[Sjöstrand et al., 2013] Sjöstrand, J., Arvestad, L., Lagergren, J., and Sennblad, B. (2013). Genphylodata: realistic simulation of gene family evolution. *BMC Bioinformatics*, 14(209).

[Spielman and Wilke, 2015] Spielman, S. J. and Wilke, C. O. (2015). Pyvolve: A flexible python module for simulating sequences along phylogenies. *PLOS ONE*, 10(9):1–7.