# Pyvolve: A Flexible Python Module for Simulating Sequences along Phylogenies

Stephanie J. Spielman[1,*] and Claus O. Wilke[1]

**1 Department of Integrative Biology, Center for Computational Biology and Bioinformatics, and Institute of Cellular and Molecular Biology. The University of Texas at Austin, Austin, TX 78712, USA.**

***stephanie.spielman@gmail.com**

## Abstract

We introduce Pyvolve, a flexible Python module for simulating genetic data along a phylogeny using continuous-time Markov models of sequence evolution. Easily incorporated into Python bioinformatics pipelines, Pyvolve can simulate sequences according to most standard models of nucleotide, amino-acid, and codon sequence evolution. All model parameters are fully customizable. Users can additionally specify custom evolutionary models, with custom rate matrices and/or states to evolve. This flexibility makes Pyvolve a convenient framework not only for simulating sequences under a wide variety of conditions, but also for developing and testing new evolutionary models. Pyvolve is an open-source project under a FreeBSD license, and it is available for download, along with a detailed user-manual and example scripts, from http://github.com/sjspielman/pyvolve.

## Introduction

The Python programming language has become a staple in biological computing. In particular, the molecular evolution community has widely embraced Python as standard tool, in part due to the development of powerful bioinformatics modules such as Biopython [1] and DendroPy [2]. However, Python lacks a robust platform for evolutionary sequence simulation.

In computational molecular evolution and phylogenetics, sequence simulation represents a fundamental aspect of model development and testing. Through simulating genetic data according to a particular evolutionary model, one can rigorously test hypotheses about the model, examine the utility of analytical methods or tools in a controlled setting, and assess the interactions of different biological processes [3].

To this end, we introduce Pyvolve, a sequence simulation Python module (with dependencies Biopython [1], SciPy, and NumPy [4]). Pyvolve simulates sequences along a phylogeny using continuous-time Markov models of sequence evolution for nucleotides, amino acids, and codons, according to standard approaches [5]. The primary purpose of Pyvolve is to provide a user-friendly and flexible sequence simulation platform that can easily be integrated into Python bioinformatics pipelines without necessitating the use of third-party software. Furthermore, Pyvolve allows users to specify and evolve custom evolutionary models and/or states, making Pyvolve an ideal engine for novel model development and testing.

## Substitution models and frameworks in Pyvolve

Pyvolve is specifically intended to simulate gene sequences along phylogenies according to Markov models of sequence evolution. Therefore, Pyvolve requires users to provide a fixed phylogeny along which sequences will evolve. Modeling frameworks which are included in Pyvolve are detailed in Table 1.

Pyvolve supports both site-wise and branch (temporal) heterogeneity. Site-wise heterogeneity can be modeled with $\Gamma$ or $\Gamma$+I rates, or users can specify a custom rate-distribution. Further, users can specify a custom rate matrix for a given simulation, and thus they can evolve sequences according to substitution models other than those shown in Table 1. Similarly, users have the option to specify a custom set of states to evolve, rather than being limited to nucleotide, amino-acid, or codon data. Therefore, it is possible to specify arbitrary models with corresponding custom states, e.g. states 0, 1, and 2. This general framework will enable users to evolve, for instance, states according to models of character evolution, such as the Mk model [6].

Similar to other simulation platforms (e.g. Seq-Gen [7], indel-Seq-Gen [8], and INDELible [9]), Pyvolve simulates sequences in groups of *partitions*, such that different partitions can have unique evolutionary models and/or parameters. Although Pyvolve enforces that all partitions within a single simulation evolve according to the same model family (e.g. nucleotide, amino-acid, or codon), Python's flexible scripting environment allows for straight-forward alignment concatenation. Therefore, it is readily possible to embed a series of Pyvolve simulations within a Python script to produce highly-heterogeneous alignments, for instance where coding sequences are interspersed with non-coding DNA sequences. Moreover, Pyvolve allows users to specify, for a given partition, the ancestral sequence (MRCA) to evolve.

In addition, we highlight that Pyvolve is among the first open-source simulation tools to include the mutation-selection modeling framework introduced by Halpern and Bruno in ref. [10] (we note that the simulation software SGWE [11] also includes this model). Importantly, although these models were originally developed for codon evolution [10, 12], Pyvolve implements mutation-selection models for both codons and nucleotides. We expect that this implementation will foster the continued development and use of this modeling framework, which has seen a surge of popularity in recent years [13–19].

**Table 1. Substitution models included in Pyvolve.**

| Modeling Framework | Available Models |
|---|---|
| Nucleotide | GTR [20] and all nested variants (e.g. HKY85 [21] and TN93 [22]) |
| Amino acid | JTT [23], WAG [24], LG [25], mtMAM [26], mtREV24 [27], DAYHOFF [28], AB [29] |
| Mechanistic codon | GY-style [30, 31] and MG-style [32] |
| Empirical codon | ECM (restricted and unrestricted) [33] |
| Mutation-selection | Halpern-Bruno model [10], for both nucleotides and codons |

## Basic Usage of Pyvolve

The basic framework for a simple simulation with the Pyvolve module is shown in Fig. 1. To simulate sequences, users should input the phylogeny along which sequences will evolve, define evolutionary model(s), and assign model(s) to partition(s). Pyvolve implements all evolutionary models in their most general forms, such that any parameter in the model may be customized. This behavior stands in contrast to several other simulation platforms of comparable scope to Pyvolve. For example, some of the most commonly used simulation tools that implement codon models, including

INDELible [9], EVOLVER [34], and PhyloSim [35], do not allow users to specify $dS$ rate variation in codon models. Pyvolve provides this option, among many others.

```
# Import the Pyvolve module
import pyvolve

# Read in phylogeny along which Pyvolve should simulate
my_tree = pyvolve.read_tree(file = "tree.tre")

# Define a mechanistic codon evolutionary model with the Model class
parameters = {"omega": 0.75, "kappa": 3.25}
my_model = pyvolve.Model("codon", parameters)

# Define partition(s) with the Partition class
my_partition = pyvolve.Partition(models = my_model, size = 100)

# Evolve partition(s) with the callable Evolver class
my_evolver = pyvolve.Evolver(tree = my_tree, partitions = my_partition)
my_evolver() # By default, the simulated alignment is saved to a file here
```

**Figure 1. Example code for a simple codon simulation in Pyvolve.** This example will simulate an alignment of 100 codons with a $dN/dS$ of 0.75 and a $\kappa$ (transition-tranversion mutational bias) of 3.25. By default, sequences will be output to a file called "simulated_alignment.fasta", although this file name can be changed, as described in Pyvolve's user manual.

In the example shown in Fig. 1, stationary frequencies are not explicitly specified. Under this circumstance, Pyvolve will assume equal frequencies, although they would normally be provided using the key `"state_freqs"` in the dictionary of parameters. Furthermore, Pyvolve contains a convenient module to help specify state frequencies. This module can read in frequencies from an existing sequence and/or alignment file (either globally or from specified alignment columns), generate random frequencies, or constrain frequencies to a given list of allowed states. In addition, this module will convert frequencies between alphabets, which is useful, for example, if one wishes to simulate amino-acid data using the state frequencies as read from a file of codon sequence data.

## Validating Pyvolve

We carefully assessed that Pyvolve accurately simulates sequences. To this end, we simulated several data sets under a variety of evolutionary models and conditions and compared the observed substitution rates with the simulated parameters.

To evaluate Pyvolve under the most basic of conditions, site-homogeneity, we simulated both nucleotide and codon data sets. We evolved nucleotide sequences under the JC69 model [36] across several phylogenies with varying branch lengths (representing the substitution rate), and we evolved codon sequences under a MG94-style model [32] with varying values of $dN/dS$. All alignments were simulated along a two-taxon tree and contained 100,000 positions. We simulated 50 replicates for each branch length and/or $dN/dS$ parameterization. As shown in Figs. 2A and B, the observed number of changes agreed precisely with the specified parameters.

We additionally validated Pyvolve's implementation of site-wise rate heterogeneity. We simulated an alignment of 400 codon positions, again under an MG94-style model [32], along a balanced tree of $2^{14}$ taxa with all branch lengths set to 0.01. This large number of taxa was necessary to achieve accurate estimates for site-specific measurements. To incorporate site-specific rate heterogeneity, we specified four $dN/dS$ values of 0.2, 0.4, 0.6, and 0.8, to be assigned in equal proportions to sites across this alignment. We counted the observed $dN/dS$ values for each resulting alignment column

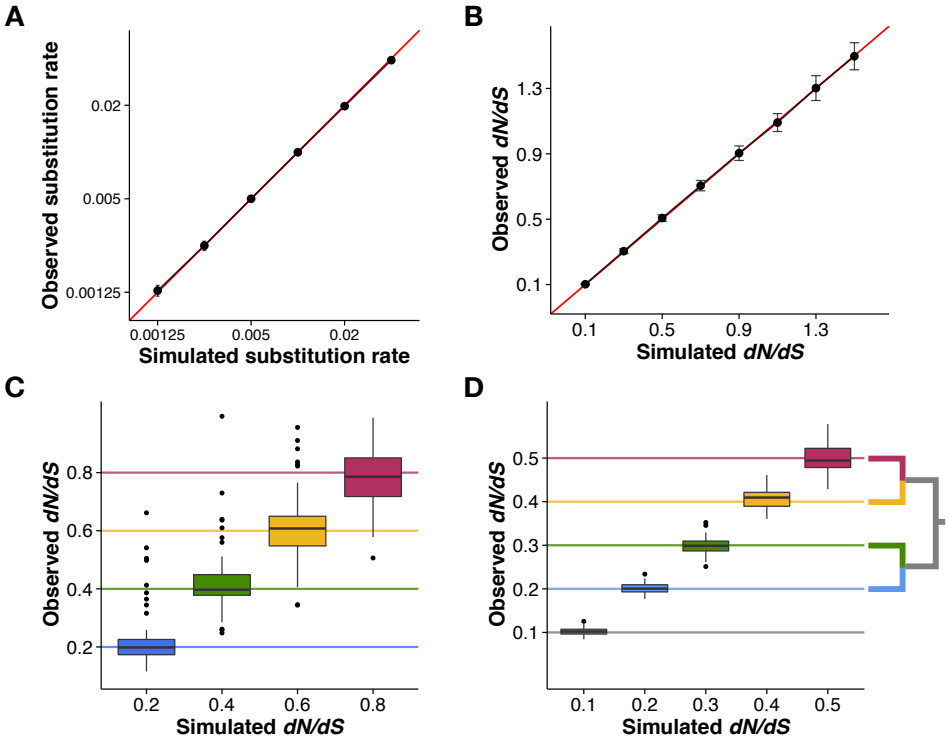


**Figure 2. Pyvolve accurately evolves sequences under homogenous, site-wise rate heterogeneity, and branch-specific rate heterogeneity.** A) Nucleotide alignments simulated under the JC69 [36] model along two-taxon trees with varying branch lengths, which represent the substitution rate. Points represent the mean observed substitution rate for the 50 alignment replicates simulated under the given value, and error bars represent standard deviations. The red line indicates the $x = y$ line. B) Codon alignments simulated under an MG94-style [32] model with varying values for the $dN/dS$ parameter. Points represent the mean $dN/dS$ inferred from the 50 alignment replicates simulated under the given $dN/dS$ value, and error bars represent standard deviations. The red line indicates the $x = y$ line. C) Site-wise heterogeneity simulated with an MG94-style [32] model with varying $dN/dS$ values across sites. Horizontal lines indicate the simulated $dN/dS$ value for each $dN/dS$ category. D) Branch-wise heterogeneity simulated with an MG94-style [32] model with each branch evolving according to a distinct $dN/dS$ value. Horizontal lines indicate the simulated $dN/dS$ value for each branch, as shown in the inset phylogeny. The lowest $dN/dS$ category ($dN/dS = 0.1$) was applied to the internal branch (shown in gray). All code and data used to validate Pyvolve's performance and generate this figure are available in File S1.

using a version of the Suzuki-Gojobori algorithm [37] adapted for phylogenetic data [38]. Figure 2C demonstrates that Pyvolve accurately implements site-specific rate heterogeneity. The high variance seen in Figure 2C is an expected result of enumerating substitutions on a site-specific basis, which, as a relatively small data set, produces substantial noise.

Finally, we confirmed that Pyvolve accurately simulates branch heterogeneity. Using a four-taxon tree, we evolved codon sequences under an MG94-style model [32] and specified a distinct $dN/dS$ ratio for each branch. We simulated 50 replicate alignments of 100,000 positions, and we computed the observed $dN/dS$ value along each branch.

Figure 2D shows that observed branch $dN/dS$ values agreed with the simulated values.

## Conclusions

Because Pyvolve focuses on simulating the substitution processes using continuous-time Markov models along a fixed phylogeny, it is most suitable for simulating gene sequences, benchmarking inference frameworks, and for developing and testing novel Markov models of sequence evolution. For example, we see a primary application of Pyvolve as a convenient simulation platform to benchmark $dN/dS$ and mutation-selection model inference frameworks such as the ones provided by PAML [34], HyPhy [39], Phylobayes [17], or swMutSel [16]. Indeed, the Pyvolve engine has already successfully been applied to investigate the relationship between mutation-selection and $dN/dS$ modeling frameworks and to identify estimation biases in certain $dN/dS$ models [18]. Moreover, we believe that Pyvolve provides a convenient tool for easy incorporation of complex simulations, for instance those used in approximate Bayesian computation (ABC) or MCMC methods [40], into Python pipelines.

Importantly, Pyvolve is meant primarily as a convenient Python library for simulating simple Markov models of sequence evolution. For more complex evolutionary scenarios, including the simulation of entire genomes, population processes, or protein folding and energetics, we refer the reader to several more suitable platforms. For example, genomic process such as recombination, coalescent-based models, gene duplication, and migration, may be best simulated with softwares such as ALF [41], CoalEvol and SGWE [11], or EvolSimulator [42]. Simulators which consider the influence of structural and/or biophysical constraints in protein sequence evolution include CASS [43] or ProteinEvolver [44]. Similarly, the software REvolver [45] simulates protein sequences with structural domain constraints by recruiting profile hidden Markov models (pHMMs) to model site-specific substitution processes.

We additionally note that Pyvolve does not currently include the simulation of insertions and deletions (indels), although this functionality is planned for a future release. We refer readers to the softwares indel-Seq-Gen [8] and Dawg [46] for simulating nucleotide sequences, and we suggest platforms such as INDELible [9], phyloSim [35], or $\pi$Buss [47] for simulating coding sequences with indels.

In sum, we believe that Pyvolve's flexible platform and user-friendly interface will provide a helpful and convenient tool for the biocomputing Python community. Pyvolve is freely available from http://github.com/sjspielman/pyvolve, conveniently packaged with a comprehensive user manual and several example scripts demonstrating various simulation conditions. In addition, Pyvolve is distributed with two helpful Python scripts that complement Pyvolve simulations: one which implements the Suzuki-Gojobori [37] $dN/dS$ counting algorithm adapted for phylogenetic data [38], and one which calculates $dN/dS$ from a given set of mutation-selection model parameters as described in ref. [18]. Pyvolve is additionally available for download from the Python Package Index (e.g. via `pip`).

## Supporting Information

### File S1

**This file contains all scripts and data used to validate Pyvolve.**

## Acknowledgments

We thank Suyang Wan and Dariya Sydykova for helpful feedback during Pyvolve development and Rebecca Tarvin for her help designing the Pyvolve logo.

## References

1. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009;25:1422–1423.
2. Sukumaran J, Holder MT. DendroPy: A Python library for phylogenetic computing. Bioinformatics. 2010;26:1569–1571.
3. Arenas M. Simulation of Molecular Data under Diverse Evolutionary Scenarios. PLoS Comp Biol. 2012;8:e1002495.
4. Oliphant T. Python for Scientific Computing. IEEE Comput Sci Eng. 2007;9:10–20.
5. Yang Z. Computational Molecular Evolution. Oxford University Press; 2006.
6. Lewis PO. A Likelihood Approach to Estimating Phylogeny from Discrete Morphological Character Data. Syst Biol. 2001;50:913–925.
7. Rambaut A, Grassly N. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Comput Appl Biosci. 1997;13:235–238.
8. Strope C, Scott S, Moriyama E. indel-Seq-Gen: a new protein family simulator incorporating domains, motifs, and indels. Mol Biol Evol. 2007;24(3):640–649.
9. Fletcher W, Yang Z. INDELible: A Flexible Simulator of Biological Sequence Evolution. Mol Biol Evol. 2009;26(8):1879–1888.
10. Halpern A, Bruno W. Evolutionary distances for protein-coding sequences: modeling site-specific residue frequencies. Mol Biol Evol. 1998;15:910–917.
11. Arenas M, Posada D. Simulation of Genome-Wide Evolution under Heterogeneous Substitution Models and Complex Multispecies Coalescent Histories. Mol Biol Evol. 2014;31:1295–1301.
12. Yang Z, Nielsen R. Mutation-Selection Models of Codon Substitution and Their Use to Estimate Selective Strengths on Codon Usage. Mol Biol Evol. 2008 Jan;25(3):568–579.
13. Holder M, Zwickl D, Dessimoz C. Evaluating the robustness of phylogenetic methods to among-site variability in substitution processes. Phil Trans R Soc B. 2008;363:4013–4021.
14. Rodrigue N, Philippe H, Lartillot N. Mutation-selection models of coding sequence evolution with site-heterogeneous amino acid fitness profiles. Proc Natl Acad Sci USA. 2010;107(10):4629–4634.
15. Tamuri AU, dos Reis M, Goldstein RA. Estimating the distribution of selection coefficients from phylogenetic data using sitewise mutation-selection models. Genetics. 2012;190:1101–1115.

16. Tamuri AU, Goldman N, dos Reis M. A penalized-likelihood method to estimate the distribution of selection coefficients from phylogenetic data. Genetics. 2014;197:257–271.

17. Rodrigue N, Lartillot N. Site-heterogeneous mutation-selection models within the PhyloBayes-MPI Package. Bioinformatics. 2014;30:1020–1021.

18. Spielman S, Wilke C. The relationship between $dN/dS$ and scaled selection coefficients. Mol Biol Evol. 2015;32:1097–1108.

19. dos Reis M. How to calculate the non-synonymous to synonymous rate ratio of protein-coding genes under the Fisher–Wright mutation–selection framework. Biol Lett. 2015;11:20141031.

20. Tavare S. Lines of descent and genealogical processes, and their applications in population genetics models. Theor Popul Biol. 1984;26:119–164.

21. Hasegawa M, Kishino H, Yano T. Dating of human-ape splitting by a molecular clock of mitochondrial DNA. J Mol Evol. 1985;22(2):160–174.

22. Tamura K, Nei M. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. Mol Biol Evol. 1993;10:512—-526.

23. Jones D, Taylor W, Thornton J. The rapid generation of mutation data matrices from protein sequences. CABIOS. 1992;8:275–282.

24. Whelan S, Goldman N. A general empirical model of protein evolution derived from multiple protein families using a maximum likelihood approach. Mol Biol Evol. 2001;18:691–699.

25. Le S, Gascuel O. An improved general amino acid replacement matrix. Mol Biol Evol. 2008;25:1307–1320.

26. Yang N, Nielsen R, Hasegawa M. Models of Amino Acid Substitution and Applications to Mitochondrial Protein Evolution. Mol Biol Evol. 1998;15:1600–1611.

27. Adachi J, Hasegawa M. MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. Comput Sci Monogr. 19896;28:1–150.

28. Dayhoff M, Schwartz R, Orcutt B. A model of evolutionary change in proteins. Atlas of Protein Sequence and Structure. 1978;5(3):345–352.

29. Mirsky A, Kazandjian L, Anisimova M. Antibody-Specific Model of Amino Acid Substitution for Immunological Inferences from Alignments of Antibody Sequences. Mol Biol Evol. 2015;32:806–819.

30. Goldman N, Yang Z. A codon-based model of nucleotide substitution for protein-coding DNA sequences. Mol Biol Evol. 1994;11:725–736.

31. Nielsen R, Yang Z. Likelihood models for detecting positive selected amino acid sites and applications to the HIV-1 envelope gene. Genetics. 1998;148:929–936.

32. Muse S, Gaut B. A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. Mol Biol Evol. 1994;11:715–724.

33. Kosiol C, Holmes I, Goldman N. An empirical codon model for protein sequence evolution. Mol Biol Evol. 2007;24:1464–1479.

34. Yang Z. PAML 4: Phylogenetic Analysis by Maximum Likelihood. Molecular Biology and Evolution. 2007;24:1586–1591.

35. Sipos B, Massingham T, Jordan GE, Goldman N. PhyloSim - Monte Carlo simulation of sequence evolution in the R statistical computing environment. BMC Bioinform. 2011;12(104).

36. Jukes T, Cantor C. Evolution of protein molecules. In: Munro H, editor. Mammalian protein metabolism. New York: Academic Press; 1969. .

37. Suzuki Y, Gojobori T. A method for detecting positive selection at single amino acid sites. Mol Biol Evol. 1999;16:1315–1328.

38. Kosakovsky Pond S, Frost S. Not So Different After All: A Comparison of Methods for Detecting Amino Acid Sites Under Selection. Mol Biol Evol. 2005;22:1208–1222.

39. Kosakovsky Pond S, Frost S, Muse S. HyPhy: hypothesis testing using phylogenies. Bioinformatics. 2005;12:676–679.

40. Arenas M. Advances in Computer Simulation of Genome Evolution: Toward More Realistic Evolutionary Genomics Analysis by Approximate Bayesian Computation. J Mol Evol. 2015;8:189–192.

41. Dalquen D, Anisimova M, Gonnet G, Dessimoz C. ALF–a simulation framework for genome evolution. Mol Biol Evol. 2012;29:1115–1123.

42. Beiko R, Charlebois R. A simulation test bed for hypotheses of genome evolution. Bioinformatics. 2007;23:825–831.

43. Grahnen J, Liberles D. CASS: Protein sequence simulation with explicit genotype-phenotype mapping. Trends in Evolutionary Biology;4:e9.

44. Arenas M, Dos Santos H, Posada D, Bastolla U. Protein evolution along phylogenetic histories under structurally constrained substitution models. Bioinformatics. 2013;29:3020–3028.

45. Koestler T, von Haeseler A, Ebersberger I. REvolver: modeling sequence evolution under domain constraints. Mol Biol Evol. 2012;29:2133–2145.

46. Cartwright R. DNA assembly with gaps (Dawg): simulating sequence evolution. Bioinformatics. 2005;21:iii31–iii38.

47. Bielejec F, Lemey P, Carvalho L, Baele G, Rambaut A, Suchard M. piBUSS: a parallel BEAST/BEAGLE utility for sequence simulation under complex evolutionary scenarios. BMC Bioinformatics. 2014;15:133.