Pyvolve: A Flexible Python Module for Simulating Sequences along Phylogenies

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

† 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.
* Corresponding Author: stephanie.spielman@gmail.com

Abstract

We introduce Pyvolve, a flexible Python module for simulating genetic data along a phylogeny according to continuous-time Markov models of sequence evolution. Pyvolve is easily incorporated into Python bioinformatics pipelines, and it can simulate sequences according 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. Moreover, Pyvolve includes several novel sequence simulation features, including a new rate matrix scaling algorithm and branch-length perturbations. Pyvolve is an open-source project freely available, along with a detailed user-manual and example scripts, under a FreeBSD license from http://github.com/sjspielman/pyvolve.

Introduction

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 [1].

To this end, we introduce Pyvolve, a sequence simulation Python module (with dependencies Biopython [2], SciPy, and NumPy [3]). Pyvolve simulates sequences along a phylogeny using continuous-time Markov models of sequence evolution for nucloetides, amino acids, and codons, according to standard approaches [4]. Easily integrated into Python sequence analysis pipelines, Pyvolve offers a user-friendly and flexible interface that serves as a convenient alternative to third-party software packages.

Similar to other simulation platforms (e.g. refs. [5–7]), Pyvolve can simulate sequences in groups of *partitions*, such that different partitions can have unique evolutionary models and/or parameters. Pyvolve additionally supports both site-wise and branch (temporal) heterogeneity. Site-wise heterogeneity is modeled using either a discrete gamma distribution or a discrete user-specified rate distribution.

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 most other simulation platforms. For example, some of the most commonly used simulation tools that implement codon models, including Indelible [7], EVOLVER [8], and PhyloSim [9], 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 $dN/dS = 0.75$ and a κ (transition-tranversion mutational bias) of 3.25. By default, sequences will be outputted to a file called "simulated_alignment.fasta", although this file name can be changed, as described in Pyvolve's user manual.

Modeling frameworks in Pyvolve

Pyvolve supports a wide variety of standard modeling frameworks, as detailed in Table 1. In addition to these models, Pyvolve allows users to specify a custom rate matrix. This flexibility makes Pyvolve an ideal engine for novel model development and testing. Furthermore, users can also specify that Pyvolve evolve custom states rather than strictly nucleotides, amino acids, or codons. In other words, users can specify arbitrary models with states of their choosing, e.g. states 0, 1, and 2. This general framework enables users to evolve states according to models of character evolution, for instance the Mk model [10].

Table 1. Modeling frameworks included in Pyvolve.

Modeling Framework	Available Models
Nucleotide	GTR [11] and all nested variants (e.g. HKY85 [12] and TN93 [13])
Amino acid	JTT [14], WAG [15], LG [16], mtMAM [17], mtREV24 [18], DAYHOFF [19], AB [20]
Mechanistic codon	GY-style [21, 22] and MG-style [23]
Empirical codon	ECM (restricted and unrestricted) [24]
Mutation-selection	Halpern-Bruno model [25], for both nucleotides and codons

In addition, we highlight that Pyvolve incorporates the mutation-selection modeling framework. Mutation-selection models, first introduced over 15 years ago by Halpern and Bruno [25], are based on population genetics principles and model the fitness effects of all possible mutations via scaled selection coefficients. Inferring parameters of these models is computationally expensive, and thus these models have seen little use since their introduction. However, in the past few years, several computationally efficient mutation-selection inference frameworks have been released [26–29], allowing, for the first time, the potential for large-scale adoption of these models by the scientific community.

As this modeling framework continues to grow in popularity, it is critical that the community has robust tools to independently evaluate the behavior and performance of these models. Pyvolve provides one of the first open-source simulation tools to incorporate mutation-selection models. Indeed, the Pyvolve engine has already successfully been applied to investigate the relationship between mutation-selection and dN/dS modeling frameworks [30].

We note that the third-party simulation software SWGE [31] also includes mutation-selection models. However, SWGE has several additional software dependencies and is intended specifically for simulating genome and/or proteome sequences under complex evolutionary scenarios, including migration, recombination, and gene-tree discordance [31]. By contrast, Pyvolve provides a light-weight and convenient Python framework for more straight-forward simulation of sequences under mutation-selection models. In addition, although these models were originally developed for codon evolution [25, 32] Pyvolve implements mutation-selection models for both codons and nucleotides, which we expect will foster the continued development and use of this modeling framework.

Validating Pyvolve

We carefully assessed that Pyvolve accurately simulates sequences. To this end, we simulated several data sets under varying evolutionary models and conditions, and subsequently inferred evolutionary parameters using HyPhy [33].

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 HKY85 model [12] with varying values for the transition-transversion ratio parameter κ, and we evolved codon sequences under the MG94 model [23] with varying values of dN/dS. All alignments were simulated along a randomly-generated 25-taxon tree [34] and contained 500 positions. We simulated 50 replicates for each κ and/or dN/dS parameterization. As shown in Fig. 2A and Fig. 2B, inference with HyPhy confirmed that inferred parameters agreed with the simulated parameters.

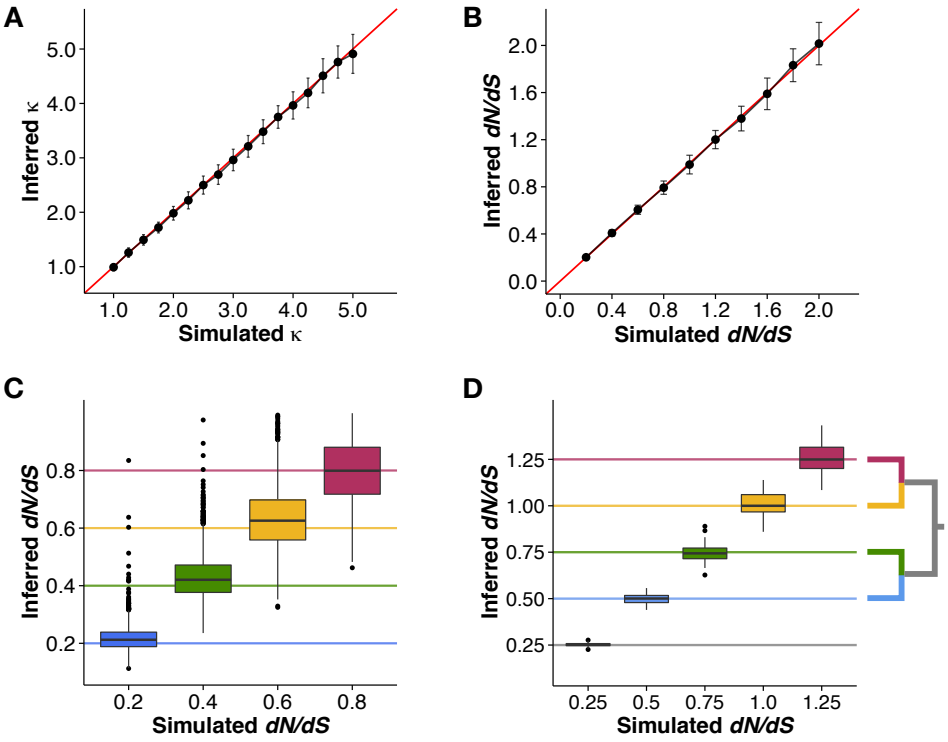

Figure 2. Pyvolve accurately evolves sequences under homogenous, site-wise rate heterogeneity, and branch-specific rate heterogeneity. A) Nucleotide alignments simulated under the HKY85 [12] model with varying values for the κ parameter. Points represent the average inferred κ value for the 50 alignment replicates simulated under the given value, and error bars represent standard deviation. The red line indicates the $x = y$ line. B) Codon alignments simulated under the MG94 [23] model with varying values for the dN/dS parameter. Points represent the average dN/dS inferred from the 50 alignment replicates simulated under the given dN/dS value, and error bars represent standard deviation. The red line indicates the $x = y$ line. C) Site-wise heterogeneity simulated with the MG94 [23] 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 the MG94 [23] model with each branch featuring a different 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 ($dN/dS = 0.25$) 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 **??**.

We additionally validated Pyvolve's implementation of site-wise rate heterogeneity. We simulated an alignment of 8000 codon positions, again under the MG94 model [23], along a 2048-taxon tree. To incorporate site-specific rate heterogeneity, we specified four dN/dS values of 0.25, 0.5, 0.75, and 1.0, to be assigned in equal proportions to sites across this alignment. We inferred site-specific dN/dS values using the SLAC

method [35] implemented in HyPhy [33]. Fig. 2C demonstrates that Pyvolve accurately implements site-specific rate heterogeneity.

Finally, we confirmed that Pyvolve accurately simulates branch heterogeneity. Using a four-taxon tree, we evolved codon sequences under the MG94 model [23] and specified a distinct dN/dS ratio for each branch. We simulated 50 replicate alignments of 10,000 positions, and we used the "TestBranchDNDS.bf" HyPhy batch file [33] to infer branch-specific dN/dS values. Figure 2D shows that inferred branch dN/dS values agreed with the simulated values.

Novel simulation features in Pyvolve

Pyvolve implements two simulation features which, to our knowledge, are not available in other open-source simulation platforms. First, it provides a novel approach to scaling rate matrices. Second, it allows for variation in branch lengths among sites.

Neutral rate matrix scaling approach

By convention, rate matrices used in models of sequence evolution are scaled such that the mean substitution rate is 1, e.g. $-\sum_{i=1} \pi_i q_{ii} = 1$, where π_i represents the equilibrium frequency of state i, and q_{ii} represents the diagonal element of the rate matrix. This standard approach, introduced by Yang [21,36], ensures that branch lengths explicitly represent the expected number of substitutions per unit (nucleotide, amino acid, or codon). However, this scaling scheme has some undesirable consequences when applied to modeling frameworks that contain explicit parameters representing selection strength, e.g. mechanistic codon and mutation-selection models.

Consider, for example, the case of dN/dS rate heterogeneity: due to the nature of mechanistic codon models, a different matrix is required for each dN/dS value. If each matrix is scaled according to Yang's approach, then the average number of substitutions will be the same for all matrices, regardless of dN/dS. In other words, sites with $dN/dS = 0.05$ would experience the same average number of substitutions as would sites with $dN/dS = 2.5$. From a biological perspective, this result is undesirable, as sites with low dN/dS values should evolve more slowly than sites with high dN/dS values, assuming the underlying mutation rate (and hence, dS) is the same across sites.

To overcome this issue, Pyvolve provides an option to scale matrices such that the mean *neutral* substitution rate is 1. For dN/dS codon models, this approach scales the matrix such that the mean number of substitutions is 1 when $dN/dS = 1$. For mutation-selection models (both nucleotide and codon), this approach scales the matrix such that the mean substitution rate is 1 when all states (nucleotides/codons) have equal fitness.

To evaluate our neutral scaling approach, we used Pyvolve to simulate alignments (50 replicates each) under the MG94 [23] codon model with a global dN/dS value ranging from 0.1 – 2.0, for each scaling approach. All alignments contained 200 positions and used the same randomly generated 25-taxon tree [34]. We inferred a maximum-likelihood phylogeny with RAxML [37], under the GTRGAMMA model, for each simulated alignment. For each inferred phylogeny, we used DendroPy [38] to calculate a tree length, indicating the expected number of substitutions in the full tree.

Fig. 3 displays how the resulting tree length differs between Yang's scaling approach and our neutral approach. In particular, when using Yang's approach, the tree length remains constant at roughly 0.806 across all dN/dS values. This value of 0.806, one-third of the length of the tree used in simulation (2.416), naturally results from the fact that the codon model used considers only single nucleotide instantaneous changes. Therefore, Yang's scaling approach enforces an average substitution rate of 1 per unit time, regardless of the dN/dS value.

Alternatively, under Pyvolve's neutral scaling approach, the tree length increases linearly with increasing dN/dS, as should be the case if dN is varied while dS is held fixed. Further, the Yang and neutral scaling approaches yield the same number of average substitutions when $dN/dS = 1$, as expected. We therefore find that using neutral scaling approach more reasonably reflects the strength of natural selection.

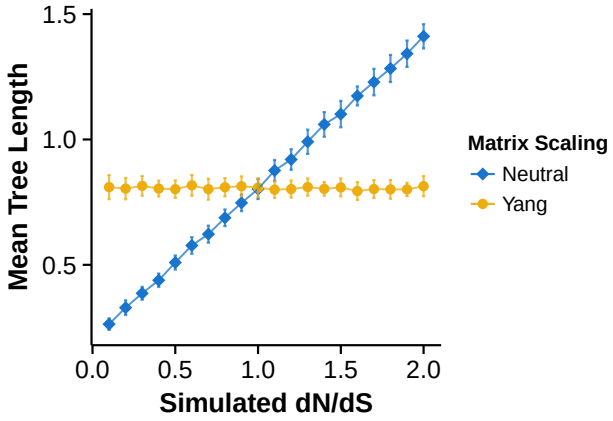

Figure 3. Neutral scaling approach yields more realistic expected numbers of substitutions, as represented by branch lengths. Under Yang's scaling approach, the tree length remains constant across dN/dS values, whereas under the neutral scaling approach, the tree length increases linearly with increasing dN/dS, as should be the case if dN is varied while dS is held fixed. Further, the Yang and neutral scaling approaches yield the same number of average substitutions when $dN/dS = 1$, as expected. Error bars represent standard deviations. All code and data used to generate this figure are available in **??**.

Branch length perturbations

Conventional sequence simulation algorithms apply a given branch length uniformly across all sites for a given branch. For example, if a given branch has a length of 0.1, then every site along that branch will evolve with a branch length of exactly 0.1. However, given that phylogenetic inference methods compute branch lengths effectively as an average value for all sites along that branch, there is no reasonable justification to apply the same branch length to all sites.

To address this issue, Pyvolve allows users to perturb branch lengths at individual sites. At each site i, for a given branch length t, Pyvolve draws a new branch length t_i from a user-specified distribution with a mean of t. Users can select from either a normal, gamma, or exponential distribution.

Conclusions

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. Pyvolve is conveniently packaged with a comprehensive user manual and several example scripts demonstrating various simulation conditions. Further, Pyvolve has been deposited in the Python Package Index and is therefore available for download and installation using Python package managers such as pip.

Acknowledgments

This work was supported in part by NIH grant F31GM113622 to SJS and by grants from the ARO (W911NF-12-1-0390), DTRA (HDTRA1-12-C-0007) and NSF (Cooperative Agreement No. DBI-0939454, BEACON Center) to COW. 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. Arenas M. Simulation of Molecular Data under Diverse Evolutionary Scenarios. PLoS Comp Biol. 2012;8:e1002495.

2. 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.

3. Oliphant T. Python for Scientific Computing. IEEE Comput Sci Eng. 2007;9:10–20.

4. Yang Z. Computational Molecular Evolution. Oxford University Press; 2006.

5. 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.

6. 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.

7. Fletcher W, Yang Z. INDELible: A Flexible Simulator of Biological Sequence Evolution. Mol Biol Evol. 2009;26(8):1879–1888.

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

9. 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).

10. Lewis PO. A Likelihood Approach to Estimating Phylogeny from Discrete Morphological Character Data. Syst Biol. 2001;50:913–925.

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

12. 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.

13. 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.

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

15. 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.

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

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

18. Yang N, Nielsen R, Hasegawa M. MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. Comput Sci Monogr. 19896;28:1–150.

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

20. 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.

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

22. 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.

23. 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.

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

25. Halpern A, Bruno W. Evolutionary distances for protein-coding sequences: modeling site-specific residue frequencies. Mol Biol Evol. 1998;15:910–917.

26. 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.

27. 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.

28. 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.

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

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

31. Arenas M, Posado D. Simulation of Genome-Wide Evolution under Heterogeneous Substitution Models and Complex Multispecies Coalescent Histories. Mol Biol Evol. 2014;31:1295 – 1301.

32. 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.

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

34. Paradis E, Claude J, Strimmer K. APE: analyses of phylogenetics and evolution in R language. Bioinformatics. 2004;20:289–290.

35. 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.

36. Yang Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J Mol Evol. 1994;39:306 – 314.

37. Stamatakis A. RAxML Version 8: A tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies. Bioinformatics. 2014;30:1312 – 1313.

38. Sukumaran J, Holder MT. DendroPy: A Python library for phylogenetic computing. Bioinformatics. 2010;26:1569–1571.