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 according to continuous-time Markov models of sequence evolution. Pyvolve incorporates most standard models of nucleotide, amino-acid, and codon sequence evolution, and it allows users to fully customize all model parameters. Pyvolve additionally allows users to specify custom evolutionary models and incorporates several novel features, including a novel rate matrix scaling algorithm and branch-length perturbations. Easily incorporated into Python bioinformatics pipelines, Pyvolve represents a convenient and flexible alternative to third-party simulation softwares. Pyvolve is an open-source project available, along with a detailed user-manual, under a FreeBSD license from https://github.com/sjspielman/pyvolve. API documentation is available from http://sjspielman.org/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 (Arenas, 2012).

To this end, we introduce Pyvolve, a sequence simulation library written in Python [with dependencies BioPython (Cock *et al.*, 2009), SciPy, and NumPy (Oliphant, 2007)]. Pyvolve simulates sequences along a phylogeny using continuous-time Markov models of sequence evolution, according to standard approaches (Yang, 2006). Pyvolve supports a variety of standard modeling frameworks, as detailed in Table 1.

Similar to other simulation platforms (e.g. refs. Rambaut and Grassly (1997); Strope *et al.* (2007); Fletcher and Yang (2009)), 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. This release of Pyvolve does not include insertions and deletions (indels), although this functionality is planned for a future release.

Modeling Framework	**Available Models**
Nucleotide	GTR (Tavare, 1984) and nested variants [e.g. HKY85 (Hasegawa *et al.*, 1985) and TN93 (Tamura and Nei, 1993)]
Amino acid	JTT (Jones *et al.*, 1992), WAG (Whelan and Goldman, 2001), LG (Le and Gascuel, 2008)
Mechanistic codon	GY-style (Goldman and Yang, 1994; Nielsen and Yang, 1998) and MG-style (Muse and Gaut, 1994)
Empirical codon	ECM (restricted and unrestricted) (Kosiol *et al.*, 2007)
Mutation-selection	Halpern-Bruno model (Halpern and Bruno, 1998), for both nucleotides and codons

Table 1. Modeling frameworks included in Pyvolve. Note that Pyvolve also allows users to specify their own rate matrix instead of using a built-in framework.

The general framework for a simple simulation with the Pyvolve module is shown in Figure 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 other simulation frameworks; for instance, the simulation platform Indelible (Fletcher and Yang, 2009) does not allow users to specify dS rate variation in codon models, but 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()
```

Figure 1: Example code for a simple Pyvolve simulation. The code shown here will simulate an alignment of 100 codons with a $dN/dS = 0.75$ and a κ (transition-tranversion mutational bias) of 3.25. Additional parameters, such as equilibrium frequencies or other mutation rates, or indeed dN and dS separately, may be incorporated into the parameters dictionary.

The following sections describe novel simulation features in Pyvolve.

Inclusion of mutation-selection models

Pyvolve is, to our knowledge, the only open-source simulation tool that accommodates mutation-selection models. These models, first introduced over 15 years ago by Halpern and Bruno (1998), are based on population genetics principles and use scaled selection coefficients to model fitness effects of all possible mutations. Due to their high computational expense, mutation-selection models have seen little use, and consequently the properties of these models remain poorly understood. However, in the past few years, several computationally efficient mutation-selection model implementations have been released (Tamuri *et al.*, 2012, 2014; Rodrigue *et al.*, 2010; Rodrigue and Lartillot, 2014), allowing, for the first time, the potential for large-scale adoption by the scientific community. Pyvolve's inclusion of mutation-selection models, therefore, provides the first open-source simulation platform for independently evaluating the behavior and performance of these models. Indeed, the Pyvolve engine has already successfully been applied to investigate the relationship between mutation-selection and dN/dS modeling frameworks (Spielman and Wilke, 2015). Moreover, although the original mutation-selection model framework was developed in the context of coding sequence evolution (Halpern and Bruno, 1998; Yang and Nielsen, 2008), Pyvolve implements mutation-selection models for both codons and nucleotides.

Novel 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 elements of the rate matrix. This standard approach, introduced by Yang (Goldman and Yang, 1994; Yang, 1994), 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. We show, in Figure 2, how our neutral scaling approach more reasonably reflects selective pressure.

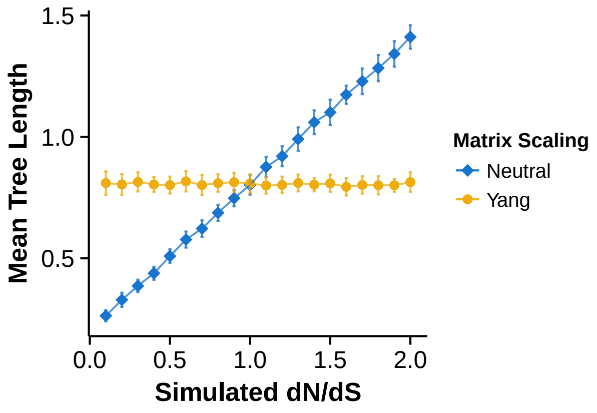

Figure 2: Neutral scaling approach yields more realistic expected numbers of substitutions, as represented by branch lengths. We simulated alignments of 200 codons each under the GY94 (Goldman and Yang, 1994) mechanistic codon model. All simulations were performed along the same, randomly generated 25-taxon tree (Paradis *et al.*, 2004). We simulated alignments (50 replicates each) with a global dN/dS value ranging from 0.1 – 2.0, for each scaling approach. We inferred a maximum-likelihood phylogeny with RAxML (Stamatakis, 2014), under the GTRGAMMA model, for each simulated alignment. We then calculated a tree length, indicating the expected number of substitutions in the full tree, for each inferred phylogeny using DendroPy (Sukumaran and Holder, 2010). When using Yang's scaling approach, the tree length remains constant across dN/dS values, whereas 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. Error bars represent standard deviations.

Perturbing branch lengths

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.

Acknowledgements

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

References

Arenas, M. (2012). Simulation of molecular data under diverse evolutionary scenarios. *PLoS Comp Biol*, **8**, e1002495.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Fletcher, W. and Yang, Z. (2009). INDELible: A flexible simulator of biological sequence evolution. *Mol Biol Evol*, **26**(8), 1879–1888.

Goldman, N. and Yang, Z. (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol Biol Evol*, **11**, 725–736.

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

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

Jones, D., Taylor, W., and Thornton, J. (1992). The rapid generation of mutation data matrices from protein sequences. *CABIOS*, **8**, 275–282.

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

Le, S. and Gascuel, O. (2008). An improved general amino acid replacement matrix. *Mol Biol Evol*, **25**, 1307–1320.

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

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

Oliphant, T. (2007). Python for scientific computing. *IEEE Comput. Sci. Eng.*, **9**, 10–20.

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

Rambaut, A. and Grassly, N. (1997). Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput Appl Biosci*, **13**, 235 – 238.

Rodrigue, N. and Lartillot, N. (2014). Site-heterogeneous mutation-selection models within the PhyloBayes-MPI package. *Bioinformatics*, **30**, 1020 – 1021.

Rodrigue, N., Philippe, H., and Lartillot, N. (2010). Mutation-selection models of coding sequence evolution with site-heterogeneous amino acid fitness profiles. *Proc Natl Acad Sci USA*, **107**(10), 4629–4634.

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

Stamatakis, A. (2014). RAxML Version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, **30**, 1312 – 1313.

Strope, C., Scott, S., and Moriyama, E. (2007). indel-Seq-Gen: a new protein family simulator incorporating domains, motifs, and indels. *Mol Biol Evol*, **24**(3), 640–649.

Sukumaran, J. and Holder, M. T. (2010). DendroPy: A Python library for phylogenetic computing. *Bioinformatics*, **26**, 1569–1571.

Tamura, K. and Nei, M. (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol*, **10**, 512–526.

Tamuri, A. U., dos Reis, M., and Goldstein, R. A. (2012). Estimating the distribution of selection coefficients from phylogenetic data using sitewise mutation-selection models. *Genetics*, **190**, 1101 – 1115.

Tamuri, A. U., Goldman, N., and dos Reis, M. (2014). A penalized-likelihood method to estimate the distribution of selection coefficients from phylogenetic data. *Genetics*, **197**, 257 – 271.

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

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

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

Yang, Z. (2006). *Computational Molecular Evolution*. Oxford University Press.

Yang, Z. and Nielsen, R. (2008). Mutation-Selection Models of Codon Substitution and Their Use to Estimate Selective Strengths on Codon Usage. *Mol Biol Evol*, **25**(3), 568–579.