Syst. Biol. 73(6):1051–1060, 2024

https://doi.org/10.1093/sysbio/syae048
Advance Access Publication August 8, 2024

PhyloJunction: A Computational Framework for Simulating, Developing, and Teaching Evolutionary Models

Fábio K. Mendes[1,*] and Michael J. Landis[2]

[1]*Department of Biology, Louisiana State University, Baton Rouge, LA, USA*
[2]*Department of Biology, Washington University in St. Louis, Rebstock Hall, St. Louis, MO 63130, USA*
**Correspondence to be sent to: Department of Biology, Life Sciences Building, Louisiana State University, Baton Rouge, LA 70803, USA; E-mail: fmendes@lsu.edu.*

Received 15 February 2024; reviews returned 20 June 2024; accepted 05 August 2024
Associate Editor: Sebastian Höhna

Abstract.—We introduce PhyloJunction, a computational framework designed to facilitate the prototyping, testing, and characterization of evolutionary models. PhyloJunction is distributed as an open-source Python library that can be used to implement a variety of models, thanks to its flexible graphical modeling architecture and dedicated model specification language. Model design and use are exposed to users via command-line and graphical interfaces, which integrate the steps of simulating, summarizing, and visualizing data. This article describes the features of PhyloJunction—which include, but are not limited to, a general implementation of a popular family of phylogenetic diversification models—and, moving forward, how it may be expanded to not only include new models, but to also become a platform for conducting and teaching statistical learning. [Evolutionary modeling; graphical model; simulation.]

Phylogenetic models of lineage diversification have been applied to a wide variety of evolutionary phenomena spanning the disciplines of paleobiology (Heath et al. 2014; Silvestro et al. 2014b; Cantalapiedra et al. 2017), historical biogeography (Goldberg and Igić 2012; Caetano et al. 2018; Landis et al. 2022; Quintero et al. 2023), macroecology (Weber and Agrawal 2014; Condamine et al. 2019), epidemiology (Douglas et al. 2021; Nadeau et al. 2021; Sciré et al. 2022), cancer evolution (Lewinsohn et al. 2023), molecular evolution (Freyman and Höhna 2018), and linguistics (Heggarty et al. 2023). The evolutionary processes underlying these phenomena take place across a range of scales—from days to millions of years and from individual cells to the entire planet—and are known or hypothesized to operate under a similarly broad scope of tempos, modes, and spatial coordinates. Despite the heterogeneity in these biological phenomena, however, at the core of such phylogenetic models frequently lies the "state dependence" assumption: that the "states" of a lineage's characters—ecological, geographic, phenotypic, or genetic—may shape anagenetic and cladogenetic evolution. Stochastic processes that make such an assumption, the so-called state-dependent speciation and extinction (SSE) processes, comprise a popular family of models (Maddison et al. 2007) for the evolution of phylogenetic patterns.

In the recent past, many excellent methods for simulating under pure diversification models (e.g., Stadler 2011b; Höhna et al. 2016; Hagen and Stadler 2018; Louca and Doebeli 2018; Barido-Sottani et al. 2019) and SSE processes (e.g., Fitzjohn 2012; Morlon et al. 2015; Beaulieu and O'Meara 2016; Freyman and Höhna 2018; Louca and Doebeli 2018) have been published. While overlapping in their capabilities, each of those methods was developed and uniquely optimized given a specific intended application; hence, they differ in terms of their model assumptions, implementation details and documentation, and execution attributes (e.g., speed, ease-of-use, etc.). Amidst this variation, we are unaware of any methods that, within a single cohesive codebase, can simultaneously (i) simulate under arbitrarily complex SSE scenarios (but see Vaughan 2024), (ii) support an intuitive model specification grammar (e.g., Höhna et al. 2016; Drummond et al. 2022), (iii) be easily extended by others to include new models, and (iv) showcase a built-in graphical user interface (GUI) for automatic visualization and summarization of synthetic data, streamlining user interaction with the software (but see Drummond et al. 2022).

In the hope of filling this gap in the computational biology toolbox, we introduce a new, open-source computational framework for evolutionary modeling: PhyloJunction. PhyloJunction ships with a very general SSE model simulator and with additional functionalities for model validation and Bayesian analysis. Importantly, we designed PhyloJunction around a graphical modeling architecture, and equipped it with a dedicated probabilistic programming language. These features are forward-looking; they will make it easy to expand and integrate PhyloJunction's evolutionary model ecosystem in the future. PhyloJunction comes with a GUI that allows users to readily inspect and interact with simulation outputs, making this program amenable to classroom use. A command-line interface (CLI) is also available for running PhyloJunction remotely and in parallel.

Flexible Simulation: Prototyping, Testing, and Characterizing Evolutionary Models

PhyloJunction was created first and foremost as an evolutionary model simulator, more specifically a flexible simulator of SSE diversification models. A series of related diversification models (Table 1) have been implemented in multiple computational methods with varying foci and performance, each making different assumptions about how a process starts and ends, whether it flows backward or forward in time, and what output is processed and presented to the user. PhyloJunction was born out of the necessity of coalescing the strengths of these different implementations in a single, cohesive application with additional capabilities (see below), such as presenting the user with a variety of textual and graphical outputs. As illustrated in later sections and in the online documentation, our implementation can simulate arbitrarily complex birth–death processes with discrete state-dependent rates and sampling probabilities. Supported models are presented in Table 1 (validation against other software can be found in Supplementary Material).

Beyond its immediate goal of simulating SSE processes, however, it was evident early on that PhyloJunction could grow and serve more broadly as a computational framework for developing evolutionary models. This ultimate purpose manifests from PhyloJunction's graphical model architecture being written in Python—a design and language convenient for prototyping model code, on which we expand below—and from the critical role simulation plays in model testing and characterization, two key stages in a model's life cycle. A newly implemented model prototype is typically pitched against data simulated in simple scenarios, with the expectation that it returns acceptable parameter estimates given some truth (i.e., a value used in simulation). Upon failure, development loops back to implementation so bugs can be patched; this potentially iterative process is the testing (or validation) stage. If testing succeeds, the model is released to the public and enters a final characterization stage, in which its behavior and adequacy are thoroughly scrutinized by the scientific community, again via analysis of simulated data (e.g., Lanier and Knowles 2012; Davis et al. 2013; Rabosky and Goldberg 2015; Simpson et al. 2018; Mason et al. 2020; Matzke 2022).

In the following sections, we detail the different features that allow PhyloJunction to flexibly specify and simulate diversification processes, and to facilitate the different steps involved in computational evolutionary model development.

Table 1. Phylogenetic models that can be simulated with PhyloJunction. Some of these models are nested within each other (e.g., BiSSE is a special case of MuSSE). "skyline" indicates time-heterogeneous rates varying in a piecewise-constant manner. "fossilized" means the addition of a fossilization parameter, which allows for direct ancestors in the reconstructed (sampled) tree. All models can be simulated with state-dependent incomplete sampling. Representative papers for each model are listed under references.

Model	A.k.a. or acronym	References
Pure-birth	Yule	Yule (1924) and Simon (1955)
Skyline		
Fossilized		
Birth–death		Kendall (1948) and Nee et al. (1994)
Skyline	BDSKY	Stadler et al. (2013)
Fossilized	FBD	Heath et al. (2014)
Binary SSE	BiSSE	Maddison et al. (2007)
Skyline		
Fossilized		
Multistate SSE	MuSSE	Fitzjohn (2012)
Skyline		
Fossilized		
Geographic SSE	GeoSSE	Goldberg et al. (2011)
Skyline		
Fossilized		
Cladogenetic SSE	ClaSSE	Goldberg and Igić (2012)
Skyline		
Fossilized		
Multitype birth–death	MTBD	Stadler and Bonhoeffer (2013)
Skyline		Sciré et al. (2022)
Fossilized		

A Graphical Model Architecture and Dedicated Language for Specifying Arbitrarily Complex Models

Any type of analytical or generative procedure involving statistical models requires some form of infrastructure for specifying such models. One example is the framework adopted by the BEAST, BEAST 2, and RevBayes platforms, whereby atomic model components can be combined into an arbitrarily large Bayesian network—a probabilistic graphical model whose structure can be represented by a directed acyclic graph (DAG; or more explicitly as a factor graph, e.g., Fig. 1b; Höhna et al. 2014). The popularity of these platforms is elevating graphical models to a modeling standard, although every one of these programs differs in how it allows users to specify models.

Here, we take a model specification approach that sits between those adopted by the BEAST and RevBayes community. PhyloJunction implements a programming language, `phylojunction` (written in lowercase and abbreviated as `pj`), together with an interactive development environment for specifying phylogenetic models (see the next section). `pj` is lightweight like popular markup languages (e.g., XML, BEAST's format of choice), but resembles model scripting languages (e.g., Rev, the language introduced by RevBayes) in its syntax, hence its retained human-readability.

Like the Rev language, `pj` commands can be read as mathematical statements, and are naturally interpreted as instructions for building a node in a DAG (see below). User commands instruct the application engine to take some form of user input, produce some value from it, and then store that value permanently in a new variable created on the spot. Every command string consists of an assignment operator placed between the variable

(a) Listing 1: `pj` script specifying a birth-death model

```
1  # hyperprior
2  m <- 0.0 # mean of log-b (for log-normal
           distribution)
3  sd <- 0.1 # std. deviation of log-b (for log-normal
           distribution)
4
5  # rate values
6  d <- 1.0 # death
7  b ~ lognormal(meanlog=m, sdlog=sd) # birth
8
9  # deterministic rate containers
10 dr := sse_rate(name="death_rate", value=d, event="
           extinction")
11 br := sse_rate(name="birth_rate", value=b, event="
           speciation")
12
13 O <- 2.0 # origin age
14
15 # deterministic parameter stash
16 s := sse_stash(flat_rate_mat=[dr, br], n_states=1, n
           _epochs=1) # parameter stash
17
18 # phylogenetic tree
19 T ~ discrete_sse(stash=s, stop="age", stop_value=O,
           origin="true")
```

(b) Model built by listing in (a)

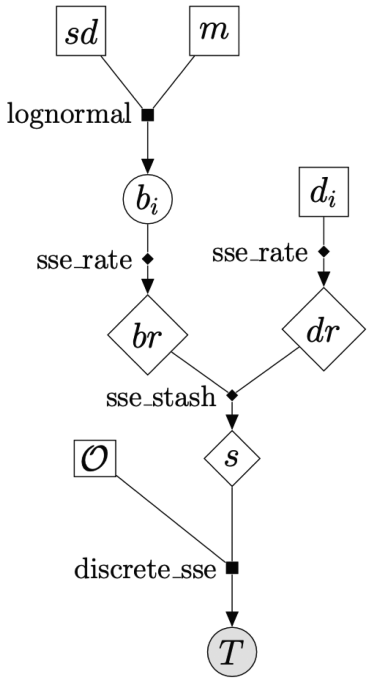

FIGURE 1. A birth–death phylogenetic model (a) as specified with `phylojunction`, PhyloJunction's eponymous programming language, and (b) shown as a factor graph, a generalization of a DAG. A few of the symbols in (b) were introduced in the context of phylogenetics by Höhna et al. (2014). Briefly, empty squares and empty circles drawn in continuous lines represent constant and stochastic nodes, respectively. Circles filled in gray represent stochastic nodes whose values are observed (i.e., data), while empty diamonds denote deterministic nodes (the output of deterministic functions). Factors capture the conditional dependencies between stochastic nodes and are either (i) filled squares, each associated with a distribution characterized by a density function and from which values can be sampled, or (ii) filled diamonds, each denoting a deterministic function.

being created (on its left side) and some user input (on its right side). Listing 1 (Fig. 1a) demonstrates the different ways in which this essential operation takes place as a time-homogeneous birth–death model is specified.

Following the grammar of Rev (Höhna et al. 2016), the behavior of a variable is determined by which assignment operator (`<-`, `~`, or `:=`) is used for assignment. For example, line 6 in listing 1 (Fig. 1a) creates a variable "`d`" (the death rate), which is then passed and henceforth stores an unmodified user input, constant value 1.0. This type of constant value assignment is carried out with the constant assignment operator, "`<-`". Line 7, in turn, shows how the stochastic assignment operator "`~`" is used to create a variable named "`b`" (the birth rate). This variable will then store a random value drawn from a user-specified distribution. Here, the user input consists of the moments of a log-normal distribution.

Finally, the deterministic assignment operator, "`:=`", is used to assign a value computed deterministically from existing variables (or other user input) to a new variable. This is illustrated by lines 10, 11, and 16 in listing 1 (Fig. 1a). The purpose of deterministic assignments is to transform, combine, or annotate one or more existing variables, and give users more control over model building. Without this class of explicit operations, such steps would instead take place out of sight in the backend, or alongside many other actions upon a single `pj` command, both of which can contribute to obscuring model structure.

Computer variables created with `pj` are nodes in the DAG that describe all variable dependencies, distributions, functions, and values that comprise the full evolutionary model. With every `pj` command the DAG thus grows by a node, which is immediately assigned a value. The nature of the assignment (constant, stochastic, or deterministic) reflects which operator was used, as explained above. A thorough treatment of the grammar and usage of graphical models for evolutionary inference can be found in Höhna et al. (2014, 2016) and the tutorials therein.

Technical Remarks on the `phylojunction` Language

In PhyloJunction, models are specified through commands written in the eponymous custom language, `phylojunction` (`pj`). In the current version of `pj`, created variables are the sole, immutable output of every function—and this output depends exclusively on a function's arguments. Variable immutability has two consequences. First, it precludes loop control structures (e.g., *for* and *while* loops), with replication and "plating" (see Höhna et al. 2014) being achieved instead through vectorization, a concept R users should be familiar with. (`pj` also does not support structures such as *if-then-else* and *switch* statements, effectively abstracting control flow.) Second, apart from the logical dependencies between nested functions—which reflect dependencies among DAG nodes—command evaluation order does not affect model specification and simulation. For example, in listing 1 (Fig. 1a), commands on lines 2, 3, and 6 are order-interchangeable, and so are those on lines 10 and 11, but the command on line 7 must be executed before that on line 11.

The features described above make `pj` behave largely as BEAST and BEAST 2's adopted declarative language, XML, while `pj`'s syntax resembles that of RevBayes' Rev language. In addition to instantiating and storing a DAG object in memory, `pj` and the aforementioned languages all require the full DAG to be known at the onset of simulation or inference (i.e., none of them are universal probabilistic programming languages; but see Senderov et al. 2023); expressiveness is also comparable among these languages, at least in terms of what could be described as a "standard" analysis using each of these platforms. For example, empirical analyses conducted with these programs more commonly than not represent phylogenetic tree random variables as a single node in the DAG.

In contrast to imperative (e.g., R, Python, Rev) and possibly even other (powerful) declarative languages (e.g., TreePPL; Senderov et al. 2023), `pj` (i) enhances reproducibility, (ii) leaves less room for programming mistakes (e.g., variable overwriting, container indexing errors), and (iii) shifts the user's attention from how to specify a model to the structure of the model itself. While imposing a cost of reduced flexibility, the above should also alleviate user demand for question-and-answer online resources, making `pj` relatively quick to learn, understand, and write. The focus on model structure in PhyloJunction is further encouraged by `pj`'s grammar ignoring actions and settings unrelated to model building, such as dependency loading, input/output, Bayesian proposals, and Markov chain Monte Carlo (MCMC) parameters. All of these properties make `pj` a lightweight language that can be particularly useful in the classroom.

Standalone CLI and GUI

PhyloJunction integrates its multiple utilities for simulating, testing, and characterizing evolutionary models via both CLI and GUI (Fig. 2a). Through the CLI and GUI, users can provide PhyloJunction with a series of DAG-building instructions in the form of a `pj` script (e.g., listing 1, Fig. 1a). Users can also build a DAG by entering commands through the GUI's command prompt (Fig. 2a, number 1). Synthetic data is then generated while a `pj` script (or sequence of commands)

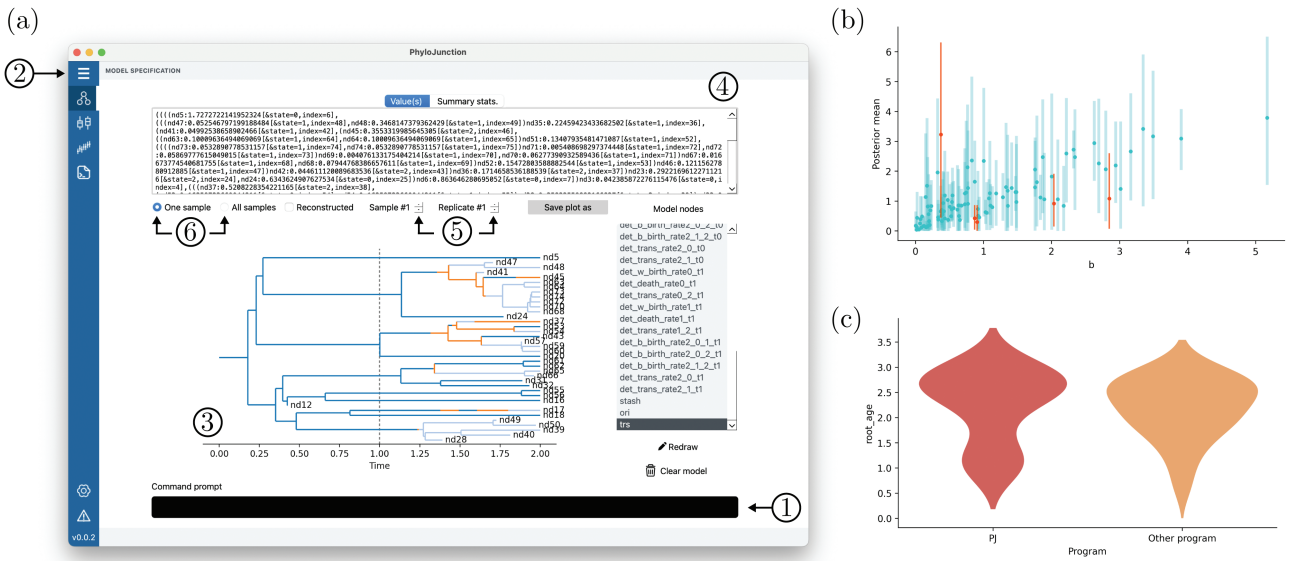

Figure 2. PhyloJunction's (a) GUI with different features indicated by numbers (see main text), and plots from (b) "Coverage" and (c) "Compare" graphical exploration functionalities.

is processed, and can later be exported as text files to a user-specified location. The interfaces can be further used to save and load a particular model instance as a serialized byte stream.

As any modern computer application, PhyloJunction's GUI exposes its features to users via a menu (Fig. 2a, number 2). On the main tab ("Model"), one can navigate the DAG and see its node values as a plot, text string, or both (Fig. 2a, numbers 3 and 4). Users can also cycle through replicated simulations (Fig. 2a, number 5), and examine node-value summaries computed for individual simulations or across replicates (Fig. 2a, number 6). Node-value summaries include the mean and standard deviation for scalar variables, and statistics like the root age and number of tips for phylogenetic trees.

Automatic summarization and visual inspection of synthetic data expedites model testing and characterization, by helping researchers quickly determine if a model setup is appropriate. Empiricists can promptly examine the effect that prior choice may have during Bayesian inference, for example, depending on what simulated data sets look like. Under an SSE model (Douglas et al. 2021), high state-transition rates causing saturation would be immediately discernible in the state mappings coloring a phylogenetic tree (Fig. 2a, number 3).

Workflow Functionalities for Model Testing, Characterization, and Teaching

In addition to data simulation, the different stages of model development have a few common denominators. Researchers must usually contend with (i) parsing inference results, (ii) comparing parameter estimates and their true values, and (iii) displaying the results as graphs. These tasks will commonly be repeated across parameter space, and sometimes under different models altogether. Furthermore, testing and characterization pipelines are often built, executed, and described several times by multiple researchers—even when the procedures taking place are very similar (e.g., Mendes et al. 2018; Hibbins et al. 2022). This redundancy is not only an inefficient use of researchers' time, but also hinders reproducibility.

PhyloJunction introduces a suite of utilities for streamlining and automating model specification, testing, and characterization. These are meant to minimize scripting redundancy and maximize the reproducibility of in silico experiments. Different utilities are separately documented and can be invoked by the user from within custom Python scripts, as modules. Alternatively, users may access PhyloJunction's features via its standalone interfaces.

Validation utilities, for example, can be accessed via the GUI's "Coverage" tab. These were designed with Bayesian coverage validation in mind (e.g., Gaboriau et al. 2020; Ogilvie et al. 2022; Zhang et al. 2024). When a simulated data set is analyzed using a Bayesian platform, this tab can be used for loading raw or post-processed inference result files. True parameter values must be loaded as a table, or if data was simulated with PhyloJunction, users can load a model instance (saved previously as a byte stream). Parameter coverage can then be automatically computed, and Bayesian intervals plotted against true parameter values (Fig. 2b).

The GUI's "Compare" tab, in turn, exposes additional model exploration utilities to the user. Here, parameter values generated by PhyloJunction under a model can be visualized against theoretical expectations, or against values simulated by a different program. Because PhyloJunction automatically computes parameter summary statistics, those can also be displayed side-by-side with comparable quantities calculated elsewhere (Fig. 2c). These functionalities are useful in a Bayesian context, for example, whenever a model has been implemented for inference, but not for direct simulation. In such cases, one can use PhyloJunction to rapidly build a direct simulator for the first time, and then use the "Compare" tab to check it against Monte Carlo samples produced by the existing implementation.

Simulation functionalities as well as those available under the "Coverage" and "Compare" tabs were developed because of the ubiquitous (and repetitious) nature of certain tasks involved in validating and characterizing a model. In addition to methodological research, however, we anticipate that these features will find use in teaching settings—especially considering the growing availability and popularity of technical workshops (Barido-Sottani et al. 2018; Barido-Sottani et al. 2022), and new pedagogical material (Warnow 2017; Harmon 2019; Revell and Harmon 2022). While trying their hand at implementing simple evolutionary models, students could use PhyloJunction to validate said models or to obtain simulation benchmarks, and to immediately visualize results via the GUI. PhyloJunction's pedagogical impact will be further enhanced by its implementation in pure Python (see below), a cross-platform, user-friendly language that finds widespread use in the classroom.

Longevity Through an Extensible and User-Friendly Model Ecosystem

One hurdle that must be often overcome during model development is the steep learning curve of the low-level programming languages many software platforms are written in. RevBayes is written in C++, for example, while BEAST and BEAST 2 are developed in Java. This is a choice motivated by compiled programming languages generally outperforming interpreted languages (e.g., R, Python), and being preferred over the latter whenever speed is a priority, such as when a method is primarily used for inference from challenging data sets. Languages like C++ and Java also natively support object-oriented programming—a

programming paradigm that is critical for erecting vast, extensible, and maintainable codebases such as those living inside those platforms.

Despite being conversant in interpreted languages, many biologists with an enthusiasm for evolutionary modeling have little to no experience with the commonly abstruse syntax and features of low-level languages (e.g., memory management, abstraction, and typing). They also have rarely had to contend with the complicated pipelines for compiling large programs across different types of computers, and with the configuration of industry-grade IDEs (integrated development environments), used for navigating immense codebases. Unless working closely with developers of big software platforms, individual scientists are likely to struggle with (i) reverse-engineering complex code that may not have been written to be read by others, and (ii) adding new code that does not break the behavior of the original codebase.

One alternative that obviates some of these difficulties is to implement and release models as R or Python packages (e.g., Fitzjohn 2012; Revell 2012; Pennell et al. 2014; Clavel et al. 2015; Morlon et al. 2015; Barido-Sottani et al. 2019; Gaboriau et al. 2020). A package has a comparatively small codebase that can be written by anyone from scratch, is self-contained and thus easily maintainable, and can be integrated with other packages more or less readily, via the scripting language. Furthermore, public package archives such as CRAN (the Comprehensive R Archive Network) or PyPI (the Python Package Index) do not restrict how a package should be programmed; package source files are immensely variable in their coding language, conventions and documentation, and programming paradigm. The minimal package submission and code-design requirements of CRAN and PyPI allow researchers freedom and flexibility, both unquestionable advantages to this variety of method development.

Writing packages has its challenges. Developers who want to add to or combine existing packages will likely have to contend with code written in a mix of languages (e.g., R, Python, C, C++, FORTRAN), paradigms (e.g., functional, objected-oriented), and styles. Furthermore, CRAN and PyPI put the onus on the researcher to choose among (often multiple) packages for the same or different purposes. Packages may vary with respect to their underlying algorithms, modeling assumptions, and notation (see Stadler 2011b for an example). Lastly, every scientist will adopt a unique R scripting strategy when specifying a model. All of the above makes reproducibility of results harder, and leads to code that is often chimeric (in its style, paradigm, and language), single-use, or redundant.

The choice of platform for writing modeling software thus involves trade-offs related to technical difficulty, speed, distributability, and maintainability. PhyloJunction embodies our attempt at balancing the above considerations while introducing an alternative methodology for model development and characterization. The brunt of PhyloJunction's design effort involved conceiving a computational framework that could not merely be extended—among other things, our intention is to facilitate the early stages of model prototyping and testing—but extended with minimal refactoring and in the most developer-friendly way possible.

We chose to implement PhyloJunction in Python primarily because of its native support for object-oriented programming, a paradigm that aids codebase expansion and maintenance. (We also entertained the much younger Julia language, which despite its potential we opted against because of the greater implementation overhead it would impose on us at the moment of writing—mainly as a result of its comparatively scarcer and less mature scientific libraries.) Furthermore, Python has clear community standards and many tools (e.g., *mypy*, *Sphinx*; *pep8*, van Rossum et al. 2001; *pep20*, Peters 2004) for encouraging or enforcing conventions on coding style, type hinting and documentation—all of which further contribute to codebase clarity and consistency. A Python codebase can also be easily navigated with any of the various user-friendly IDEs with support for Python (e.g., Visual Studio Code, PyCharm, Spyder).

Finally, Python development can profit from a vast array of free, industry-grade scientific libraries for data manipulation (e.g., *matplotlib*, Hunter 2007; *pandas*), statistics and Bayesian analysis (e.g., *scipy*, Virtanen et al. 2020; *PyMC3*, Salvatier et al. 2016; *ArviZ*, Kumar et al. 2019), and machine learning (e.g., *TensorFlow*, Abadi et al. 2015; *scikit-learn*, Pedregosa et al. 2011). Of particular relevance to PhyloJunction is PyPI's growing list of modules specifically aimed at phylogenetic or population genetic analysis (e.g., *DendroPy*, Sukumaran and Holder 2010; *PyRate*, Silvestro et al. 2014a; *MESS*, Overcast et al. 2021; *ete3*, Huerta-Cepas et al. 2016; *msprime*, Baumdicker et al. 2022), some of which have already been or may be integrated into PhyloJunction in the future. Below we suggest a few ways in which the latter may be done.

Availability and Resources

PhyloJunction's source code is publicly available on https://github.com/fkmendes/PhyloJunction. Documentation on how to install and use the program can be found on https://phylojunction.org. PhyloJunction is licensed under GNU General Public License v3.0.

Future Directions

We introduced PhyloJunction, an open-source package for simulating SSE processes, a large family of diversification models that have found success across a range of scientific domains (Douglas et al. 2021; Heggarty et al. 2023; Thompson et al. 2024). Most implementations of SSE models have prioritized inference and

efficiency over simulation and generality; the latter is the relatively vacant niche PhyloJunction was designed to fill. In addition to model specification and simulation tools, our program ships with a series of utilities for summarizing and visualizing simulation outputs, as well as data-wrangling functions for model validation and characterization. These utilities are integrated and exposed to users by standalone CLI and GUI, which simplify the execution and reproduction of in silico experiments.

Models in PhyloJunction are embedded within a graphical modeling architecture, which also underlies the package's dedicated probabilistic programming language, `phylojunction`. These features make PhyloJunction's model ecosystem extensible beyond SSE processes, and allow its components to be promptly integrated. Future software releases are planned to include diversification models that incorporate lineage- or diversity-dependent, as well as protracted speciation (Rabosky 2014; Maliet et al. 2019; Barido-Sottani et al. 2020; Hua et al. 2022), mass speciation (birth) and extinction (death) events (Stadler 2011a; Höhna 2015), distributions for different types of data models (e.g., DNA and protein sequences; Tavaré 1986; Goldman et al. 1994; Whelan and Goldman 2001); discrete and continuous characters (Felsenstein 1973; Lewis 2001), evolutionary clock models (e.g., Drummond et al. 2006; Drummond and Suchard 2010), population genetic and phylogeographic processes (e.g., Kingman 1982; Rannala and Yang 2003), or models combining any of the above (e.g., May and Moore 2019; Hua et al. 2022; Boyko et al. 2023). A richer selection of evolutionary processes should widen the range of potential applications of PhyloJunction in research and teaching.

PhyloJunction was primarily designed to be a framework for the simulation and prototyping of evolutionary models, but we expect its future development to take on the task of statistical inference. Moving in the direction of Bayesian inference may involve introducing subroutines for writing textual instructions for analyses with popular platforms (e.g., RevBayes, BEAST, BEAST 2), or for carrying out approximate Bayesian computation. Bayesian inference is also possible within a Python environment thanks to mature statistical libraries (e.g., Salvatier et al. 2016; Tehrani et al. 2020) and should be uncomplicated as long as the phylogeny itself is not inferred and is instead treated as data—tree estimation is technically difficult due to the discrete and continuous nature of phylogenetic space (Gavryushkin and Drummond 2016), and would require the non-trivial integration of advanced Bayesian proposals into third-party libraries. Lastly, because maximum-likelihood optimization algorithms are available for Python, frequentist estimation of non-tree model parameters could be accomplished in much the same it has been done by popular R packages.

We predict that new visualization functionalities will further boost PhyloJunction's utility in research and teaching, especially if they support user interactivity (via, e.g., Plotly Technologies Inc. 2015). PhyloJunction could be used to plot multiple (simulated or empirical) trees (or trees within trees; Douglas 2021) annotated with branch-specific evolutionary parameters, as well as their lineage-through-time plots—while simultaneously displaying tree likelihoods, parsimony scores, or other tree statistics. Researchers and students alike could implement their own tree likelihood or summarizing functions, and use the interactive graphical environment to have a visually intuitive grasp of how competing hypotheses fare against each other (as done recently, Hawaiian Plant Biogeography Team 2024). Here, the limit will be set by the ambition and creativity of mentors fostering growth in their trainees' computational skills and evolutionary scholarship.

We conclude by emphasizing how easily PhyloJunction can be employed to train biologically relevant machine learning pipelines. There is increasing evidence (Mo et al. 2024) backing machine learning methods as viable alternatives to frequentist and Bayesian evolutionary inference, especially when the latter is very onerous or impossible (Thompson et al. 2024). So given PhyloJunction's core simulation and data summarization methods, as well as Python's ample artificial intelligence resources, it is likely a matter of time until our software is used in the context of likelihood-free inference. Supplementary Material contains an example in which PhyloJunction simulations train a neural network to estimate model parameters using `phyddle` (Landis and Thompson 2024), a phylogenetic deep learning pipeline.

It is our long-term hope for PhyloJunction that it not only increasingly facilitates research in evolutionary modeling, but that its capabilities can be diversified and enhanced by (and according to the needs of) the scientific community at large.

Acknowledgments

We thank Felipe Zapata, Isaac Lichter-Marck, Albert Soewongsono, Sarah Swiston, Sean McHugh, and Ammon Thompson for conversations that motivated the development of PhyloJunction, inspired its design, and improved its manuscript.

Data availability

Data available from the Dryad Digital Repository: https://doi.org/10.5061/dryad.kkwh70sbq.

Funding

The research presented in this article was funded by the National Science Foundation (NSF Award DEB-2040347), the Fogarty International Center at the National Institutes of Health (Award Number R01 TW012704) as part of the joint NIH-NSF-NIFA Ecology

and Evolution of Infectious Disease program, and the Washington University Incubator for Transdisciplinary Research.

References

Hawaiian Plant Biogeography Team 2024. Phylogenetic Biogeography Workshop @ WUSTL. Available from: URL https://sites.wustl.edu/hawaiianplantbiogeography/phylogenetic-biogeography-workshop-wustl/

Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., Davis A., Dean J., Devin M., Ghemawat S., Goodfellow I., Harp A., Irving G., Isard M., Jia Y., Jozefowicz R., Kaiser L., Kudlur M., Levenberg J., Mané D., Monga R., Moore S., Murray D., Olah C., Schuster M., Shlens J., Steiner B., Sutskever I., Talwar K., Tucker P., Vanhoucke V., Vasudevan V., Viégas F., Vinyals O., Warden P., Wattenberg M., Wicke M., Yu Y., Zheng X. 2015. TensorFlow: large-scale machine learning on heterogeneous systems. Available from: URL https://www.tensorflow.org/.

Barido-Sottani J., Stadler T. 2020. A multitype birth–death model for Bayesian inference of lineage-specific birth and death rates. Syst. Biol. 69:973–986.

Barido-Sottani J., Bošková V., Plessis L.D., Kühnert D., Magnus C., Mitov V., Müller N.F., PecErska J., Rasmussen D.A., Zhang C., Drummond A.J., Heath T.A., Pybus O.G., Vaughan T.G., Stadler T. 2018. Taming the BEAST – a community teaching material resource for BEAST 2. Syst. Biol. 67:170–174.

Barido-Sottani J., Justison J.A., Borges R., Brown J.M., Dismukes W., do Rosário Petrucci B., Fabreti L.G., Höhna S., Landis M.J., Lewis P.O., May M.R., Mendes F.K., Pett W., Redelings B.D., Tribble C.M., Wright A.M., Zenil-Ferguson R., Heath T.A. 2022. Lessons learned from teaching virtual phylogenetics workshops. BSSB 1:8245.

Barido-Sottani, J., Pett W., O'Reilly J.E., Warnock R.C.M. 2019. FossilSim: an R package for simulating fossil occurrence data under mechanistic models of preservation and recovery. Methods Ecol. Evol. 10:835–840.

Baumdicker F., Bisschop G., Goldstein D., Gower G., Ragsdale A.P., Tsambos G., Zhu S., Eldon B., Ellerman E.C., Galloway J.G., Gladstein A.L., Gorjanc G., Guo B., Jeffery B., Kretzschumar W.W., Lohse K., Matschiner M., Nelson D., Pope N.S., Quinto-Cortés C.D., Rodrigues M.F., Saunack K., Sellinger T., Thornton K., van Kemenade H., Wohns A.W., Wong Y., Gravel S., Kern A.D., Koskela J., Ralph P.L., Kelleher J. 2022. Efficient ancestry and mutation simulation with msprime 1.0. Genetics 220:iyab229.

Beaulieu J.M., O'Meara B.C. 2016. Detecting hidden diversification shifts in models of trait-dependent speciation and extinction. Syst. Biol. 65:583–601.

Boyko J.D., O'Meara B.C., Beaulieu J.M. 2023. A novel method for jointly modeling the evolution of discrete and continuous traits. Evolution 77:836–851.

Caetano D.S., O'Meara B.C., Beaulieu J.M. 2018. Hidden state models improve state-dependent diversification approaches, including biogeographical models. Evolution 72:2308–2324.

Cantalapiedra J.L., Prado J.L., Fernández M.H., Alberdi M.T. 2017. Decoupled ecomorphological evolution and diversification in Neogene-Quaternary horses. Science 355:627–630.

Clavel J., Escarguel G., Merceron G. 2015. mvMORPH: an R package for fitting multivariate evolutionary models to morphometric data. Methods Ecol. Evol. 6:1311–1319.

Condamine F.L., Rolland J., Morlon H. 2019. Assessing the causes of diversification slowdowns: temperature-dependent and diversity-dependent models receive equivalent support. Ecol. Lett. 22:1900–1912.

Davis M.P., Midford P.E., Maddison W. 2013. Exploring power and parameter estimation of the BiSSE method for analyzing species diversification. BMC Evol. Biol. 13:1–11.

Douglas J. 2021. UglyTrees: a browser-based multispecies coalescent tree visualizer. Bioinformatics 37:268–269.

Douglas J., Mendes F.K., Bouckaert R., Xie D., Jimenez-Silva C.L., Swanepoel C., de Ligt J., Ren X., Storey M., Hadfield J.,Simpson C.R., Geoghegan J.L., Drummond A.J., Welch D. 2021. Phylodynamics reveals the role of human travel and contact tracing in controlling the first wave of COVID-19 in four island nations. Virus Evol. 7:veab052.

Drummond A.J., Chen K., Mendes F.K., Xie D. 2022. LinguaPhylo: a probabilistic model specification language for reproducible phylogenetic analyses. PLoS Comput. Biol. 19:e1011226.

Drummond A.J., Ho S.Y.W., Phillips M.J., Rambaut A. 2006. Relaxed phylogenetics and dating with confidence. PLoS Biol. 4:e88.

Drummond A.J., Suchard M.A. 2010. Bayesian random local clocks, or one rate to rule them all. BMC Biol. 8:114.

Felsenstein J. 1973. Maximum-likelihood estimation of evolutionary trees from continuous characters. Am. J. Hum. Genet. 25:471–492.

Fitzjohn R.G. 2012. Diversitree: comparative phylogenetic analyses of diversification in R. Methods Ecol. Evol. 3:1084–1092.

Freyman W.A., Höhna S. 2018. Cladogenetic and anagenetic models of chromosome number evolution: a Bayesian model averaging approach. Syst. Biol. 67:195–215.

Gaboriau T., Mendes F.K., Joly S., Silvestro D., Salamin N. 2020. A multi-platform package for the analysis of intra- and interspecific trait evolution. Methods Ecol. Evol. 11:1–9.

Gavryushkin A., Drummond A.J. 2016. The space of ultrametric phylogenetic trees. J. Theor. Biol. 403:197–208.

Goldberg E.E., Igić B. 2012. Tempo and mode in plant breeding system evolution. Evolution 66:3701–3709.

Goldberg E.E., Lancaster L.T., Ree R.H. 2011. Phylogenetic inference of reciprocal effects between geographic range evolution and diversification. Syst. Biol. 60:451–465.

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

Hagen O., Stadler T. 2018. TreeSim GM: simulating phylogenetic trees under general Bellman–Harris models with lineage-specific shifts of speciation and extinction in R. Methods Ecol. Evol. 9:754–760.

Harmon L.J. 2019. Phylogenetic comparative methods: learning from trees. Available from: URL: https://lukejharmon.github.io/pcm/

Heath T.A., Huelsenbeck J.P., Stadler T. 2014. The fossilized birth–death process for coherent calibration of divergence-time estimates. Proc. Natl. Acad. Sci. U.S.A. 111:E2957–E2966.

Heggarty P., Anderson C., Scarborough M., King B., Bouckaert R., Jocz L., Kümmel M.J., Jügel T., Irslinger B., Pooth R., Liljegren H., Strand R.F., Haig G., Macák M., Kim R.I., Anonby E., Pronk T., Belyaev O., Dewey-Findell T.K., Boutilier M., Freiberg C., Tegethoff R., Serangeli M., Liosis N., Stroński K., Schulte K., Gupta G.K., Haak W., Krause J., Atkinson Q.D., Greenhill S.J., Kühnert D., Gray R.D. 2023. Language trees with sampled ancestors support a hybrid model for the origin of Indo-European languages. Science 381:eabg0818.

Hibbins M.S., Breithaupt L.C., Hahn M.W. 2022. Phylogenomic comparative methods: accurate evolutionary inferences in the presence of gene tree discordance. Proc. Natl. Acad. Sci. U.S.A. 230:e2220389120.

Höhna S. 2015. The time-dependent reconstructed evolutionary process with a key-role for mass-extinction events. J. Theor. Biol. 380:321–331.

Höhna S., Boussau B., Landis M.J., Ronquist F., Huelsenbeck J.P. 2014. Probabilistic graphical model representation in phylogenetics. Syst. Biol. 63:753–771.

Höhna S., Landis M.J., Heath T.A., Boussau B., Lartillot N., Moore B.R., Huelsenbeck J.P., Ronquist F. 2016. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. Syst. Biol. 65:726–736.

Höhna S., May M.R., Moore B.R. 2016. TESS: an R package for efficiently simulating phylogenetic trees and performing Bayesian inference of lineage diversification rates. Bioinformatics 32:789–791.

Hua X., Herdha T., Burden C. 2022. Protracted speciation under the state-dependent speciation and extinction approach. Syst. Biol. 71:1362–1377.

Huerta-Cepas J., Serra F., Bork P. 2016. ETE 3: reconstruction, analysis and visualization of phylogenomic data. Mol. Biol. Evol. 33:1635–1638.

Hunter J.D. 2007. Matplotlib: a 2d graphics environment. Comput. Sci. Eng. 9:90–95. doi:10.1109/MCSE.2007.55.
Kendall, D.G. 1948. On the generalized "birth-and-death" process. Ann. Math. Stat. 19:1–15.
Kingman J.F.C. 1982. On the genealogy of large populations. J. Appl. Probab. 19:27–43.
Kumar R., Carroll C., Hartikainen A., Martin O. 2019. Arviz a unified library for exploratory analysis of Bayesian models in python. JOSS 4:1143.
Landis M.J., Quintero I., Muñoz M.M., Donoghue M.J. 2022. Phylogenetic inference of where species spread or split across barriers. Proc. Natl. Acad. Sci. U.S.A. 119:e2116948119.
Landis M.J., Thompson A. 2024. phyddle: software to fit phylogenetic models with deep learning. Available from: URL https://github.com/mlandis/phyddle
Lanier H.C., Knowles L.L. 2012. Is recombination a problem for species-tree analyses? Syst. Biol. 61:691–701.
Lewinsohn M.A., Bedford T., Müller N.F., Feder A.F. 2023. State-dependent evolutionary models reveal modes of solid tumour growth. Nat. Ecol. Evol. 7:581–596.
Lewis P.O. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. Syst. Biol. 50: 913–925.
Louca S., Doebeli M. 2018. Efficient comparative phylogenetics on large trees. Bioinformatics 34:1053–1055.
Maddison W.P., Midford P.E., Otto S.P. 2007. Estimating a binary character's effect on speciation and extinction. Syst. Biol. 56:701–710.
Maliet O., Hartig F., Morlon H. 2019. A model with many small shifts for estimating species-specific diversification rates. Nat. Ecol. Evol. 3:1086–1092.
Mason N.A., Fletcher N.K., Gill B.A., Funk W.C., Zamudio K.R. 2020. Coalescent-based species delimitation is sensitive to geographic sampling and isolation by distance. Syst. Biodivers. 18:269–280.
Matzke N.J. 2022. Statistical comparison of DEC and DEC+J is identical to comparison of two ClaSSE submodels, and is therefore valid. J. Biogeogr. 49:1805–1824.
May M.R., Moore B.R. 2019. A Bayesian approach for inferring the impact of a discrete character on rates of continuous-character evolution in the presence of background-rate variation. Syst. Biol. 69:530–544.
Mendes F.K., Fuentes-González J.A., Schraiber J.G., Hahn M.W. 2018. A multispecies coalescent model for quantitative traits. eLife 7:e36482.
Mo Y.K., Hahn M.W., Smith M.L. 2024. Applications of machine learning in phylogenetics. Mol. Phylogenet. Evol. 196:108066.
Morlon H., Lewitus E., Condamine F.L., Manceau M., Clavel J., Drury J. 2015. RPANDA: an R package for macroevolutionary analyses on phylogenetic trees. Methods Ecol. Evol. 7:589–597.
Nadeau S.A., Vaughan T.G., Scire J., Huisman J.S., Stadler T. 2021. The origin and early spread of SARS-CoV-2 in Europe. Proc. Natl. Acad. Sci. U.S.A. 118:e2012008118.
Nee S., May R.M., Harvey P.H. 1994. The reconstructed evolutionary process. Philos Trans. R. Soc. Lond. B: Biol. Sci. 344:305–311.
Ogilvie H.A., Mendes F.K., Vaughan T.G., Matzke N.J., Stadler T., Welch D., Drummond A.J. 2022. Novel integrative modeling of molecules and morphology across evolutionary timescales. Syst. Biol. 71:208–220.
Overcast I., Ruffley M., Rosindell J., Harmon L., Borges P.A.V., Emerson B.C., Etienne R.S., Gillespie R., Krehenwinkel H., Mahler D.L., Massol F., Parent C.E., Patiño J., Peter B., Week B., Wagner C., Hickerson M.J., Rominger A. 2021. A unified model of species abundance, genetic diversity, and functional diversity reveals the mechanisms structuring ecological communities. Mol. Ecol. Resour. 21:2782–2800.
Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. 2011. Scikit-learn: machine learning in Python. JMLR 12:2825–2830.
Pennell M.W., Eastman J.M., Slater G.J., Brown J.W., Uyeda J.C., Fitzjohn R.G., Alfaro M.E., Harmon L.J. 2014. geiger v2.0: an expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. Bioinformatics 30:2216–2218.
Peters T. 2004. The Zen of Python. PEP 20. Available from: URL: https://www.python.org/dev/peps/pep-0020/
Plotly Technologies Inc. 2015. Collaborative data science. Available from: URL: https://plot.ly
Quintero I., Landis M.J., Jetz W., Morlon H. 2023. The build-up of the present-day tropical diversity of tetrapods. Proc. Natl. Acad. Sci. U.S.A. 120:e2220672120.
Rabosky D.L. 2014. Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. PLoS One 9:e89543.
Rabosky D.L., Goldberg E.E. 2015. Model inadequacy and mistaken inferences of trait-dependent speciation. Syst. Biol. 64:340–355.
Rannala B., Yang, Z. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. Genetics 164:1645–1656.
Revell L.J. 2012. phytools: an R package for phylogenetic comparative biology (and other things). Methods Ecol. Evol. 3:217–223.
Revell L.J., Harmon L.J. 2022. Phylogenetic comparative methods in R. Princeton, New Jersey: Princeton University Press.
Salvatier J., Wiecki T.V., Fonnesbeck C. 2016. Probabilistic programming in Python using PyMC3. PeerJ Comput. Sci. 2:e55.
Sciré J., Barido-Sottani J., Kühnert D., Vaughan T.G., Stadler T. 2022. Robust phylodynamic analysis of genetic sequencing data from structured populations. Viruses 14:1–18.
Senderov V., Kudlicka J., Lundén D., Palmkvist V., Braga M.P., Granqvist E., Broman D., Ronquist F. 2023. TreePPL: a universal probabilistic programming language for phylogenetics. Available from: URL: https://treeppl.org/
Silvestro D., Salamin N., Schnitzler J. 2014a. PyRate: a new program to estimate speciation and extinction rates from incomplete fossil data. Methods Ecol. Evol. 5:1126–1131.
Silvestro D., Schnitzler J., Liow L.H., Antonelli A., Salamin N. 2014b. Bayesian estimation of speciation and extinction from incomplete fossil occurrence data. Syst. Biol. 63:349–367.
Simon H.A. 1955. On a class of skew distribution functions. Biometrika 42:425–440.
Simpson A.G., Wagner P.J., Wing S.L., Fenster C.B. 2018. Binary-state speciation and extinction method is conditionally robust to realistic violations of its assumptions. BMC Evol. Biol. 18:1–11.
Stadler T. TreeSim. 2010 Available from: URL http://cran.r-project.org/web/packages/TreeSim/index.html. [Internet] .
Stadler T. 2011a. Mammalian phylogeny reveals recent diversification rate shifts. Proc. Natl. Acad. Sci. U.S.A. 108:6187–6192.
Stadler T. 2011b. Simulating trees with a fixed number of extant species. Syst. Biol. 60:676–684.
Stadler T., Bonhoeffer S. 2013. Uncovering epidemiological dynamics in heterogeneous host populations using phylogenetic methods. Philos Trans. R. Soc. Lond B: Biol. Sci. 368:20120198.
Stadler T., Kühnert D., Bonhoeffer S., Drummond A.J. 2013. Birthdeath skyline plot reveals temporal changes of epidemic spread in HIV and hepatitis C virus (HCV). Proc. Natl. Acad. Sci. U.S.A. 110:228–233.
Sukumaran J., Holder M.T. 2010. DendroPy: a Python library for phylogenetic computing. Bioinformatics 26:1569–1571.
Tavaré S. 1986. Some probabilistic and statistical problems in the analysis of DNA sequences. In: Miura R.M., editor. Some mathematical questions in biology - DNA sequence analysis. New York, United States: The American Mathematical Society. p. 57–86. URL: https://api.semanticscholar.org/CorpusID:82212051.
Tehrani N., Arora N.S., Noursi D., Tingley M., Torabi N., Lippert E. 2020. Bean machine: a declarative probabilistic programming language for efficient programmable inference. European Workshop on Probabilistic Graphical Models .
Thompson A., Liebeskind B., Scully E.J., Landis M. 2024. Deep learning and likelihood approaches for viral phylogeography converge

on the same answers whether the inference model is right or wrong. Syst. Biol. 73:183–206.
van Rossum G., Warsaw B., Coghlan N. 2001. Style guide for Python code. PEP 8. Available from: URL: https://www.python.org/dev/peps/pep-0008/.
Vaughan, T.G. 2024. ReMASTER: improved phylodynamic simulation for BEAST 2.7. Bioinformatics. 40:btae015.
Virtanen P., Gommers R., Oliphant T.E., Haberland M., Reddy T., Cournapeau D., Burovski E., Peterson P., Weckesser W., Bright J., van der Walt S.J., Brett M., Wilson J., Millman K.J., Mayorov N., Nelson A.R.J., Jones E., Kern R., Larson E., Carey C.J., Polat İ., Feng Y., Moore E.W., VanderPlas J., Laxalde D., Perktold J., Cimrman R., Henriksen I., Quintero E.A., Harris C.R., Archibald A.M., Ribeiro A.H., Pedregosa F., van Mulbregt P., SciPy 1.0 Contributors 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17:261–272.
Warnow T., 2017. Computational phylogenetics: an introduction to designing methods for phylogeny estimation. Cambridge, UK: Cambridge University Press.
Weber M.G. Agrawal A.A., 2014. Defense mutualisms enhance plant diversification. Proc. Natl. Acad. Sci. U.S.A. 111: 16442–16447.
Whelan S. 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.
Yule G.U. 1924. A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F.R.S. Philos Trans. R. Soc. Lond B: Biol. Sci. 213:21–87.
Zhang R., Drummond A.J., Mendes F.K. 2024. Fast Bayesian inference of phylogenies from multiple continuous characters. Syst. Biol. 73:102–124.