PhyloJunction: a computational framework for simulating, developing, and teaching evolutionary models

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

[1] *Department of Biology, Washington University in St. Louis, St. Louis, MO*

[*] *Corresponding author: E-mail: f.mendes@wustl.edu*

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, through 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 paper 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.

Keywords: Evolutionary modeling, simulation, graphical model

Phylogenetic models of lineage diversification have been applied to a wide variety of evolutionary phenomena spanning the disciplines of paleobiology [9, 26, 66], historical biogeography [8, 22, 39, 58], macroecology [11, 81], epidemiology [13, 51, 64], cancer evolution [41], molecular evolution [20, 76], and linguistics [27]. 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 assumption, the so-called state-dependent speciation and extinction (SSE) processes, comprise a popular family of models [44] for the evolution of phylogenetic patterns.

In the recent past, many excellent methods for simulating under pure diversification models [e.g. 4, 24, 31, 43, 69] and SSE processes [e.g. 6, 19, 20, 43, 50] 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 [78]), (ii) support an intuitive model specification grammar (e.g., [14, 30]), (iii) be easily extended by others to include new models, and (iv) showcase a built-in graphical user interface for automatic visualization and summarization of synthetic data, streamlining user interaction with the software (but see [14]).

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 graphical user interface (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.

1 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). As illustrated in later sections and in the online documentation, our implementation can simulate arbitrarily complex SSE processes (all models in Table 1; validation against other software can be found in

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 incomplete sampling. Representative papers for each model are listed under references.

Model	A.k.a. or acronym	Reference(s)
Pure-birth	Yule	[67, 83]
skyline		
fossilized		
Birth-death		[36, 52]
skyline	BDSKY	[72]
fossilized	FBD	[26]
Binary SSE	BiSSE	[44]
skyline		
fossilized		
Multistate SSE	MuSSE	[19]
skyline		
fossilized		
Geographic SSE	GeoSSE	[17]
skyline		
fossilized		
Cladogenetic SSE	ClaSSE	[22]
skyline		
fossilized		
Multitype birth-death	MTBD	[71]
skyline		[64]
fossilized		

the supplement), and presents the user with a variety of textual and graphical outputs.

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., [12, 40, 45, 46, 59, 63, 68]).

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.

2 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; [29]). 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 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 [30], the behavior of a variable is determined by which assignment operator (`<-`, $\sim$, 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 '$\sim$' 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

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

```
# hyperprior
m <- 0.0 # log-normal mean
sd <- 0.1 # log-normal standard deviation

# rate values
d <- 1.0 # death
b ~ lognormal(mean=m, sd=sd) # birth

# deterministic rate containers
dr := sse_rate(name="death_rate", value=d, event="
        extinction")
br := sse_rate(name="birth_rate", value=b, event="
        speciation")

O <- 2.0 # origin age

# deterministic parameter stash
s := sse_stash(flat_rate_mat=[dr, br], n_states=1, n
        _epochs=1) # parameter stash

# phylogenetic tree
T ~ discrete_sse(stash=s, stop="age", stop_value=O,
        origin="true")
```

(b) Model built by listing in (a)

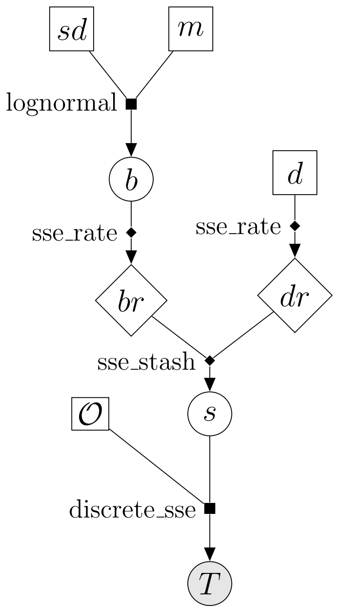


Figure 1. A birth-death phylogenetic model (a) as specified with `phylojunction`, PhyloJunction's eponymous programming language, and (b) shown as a factor graph, a generalization of a directed acyclic graph (DAG). A few of the symbols in (b) were introduced in the context of phylogenetics by [29]. Briefly, 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). 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 to a distribution characterized by a density function and from which values can be sampled, or (ii) filled diamonds, denoting each a deterministic function.

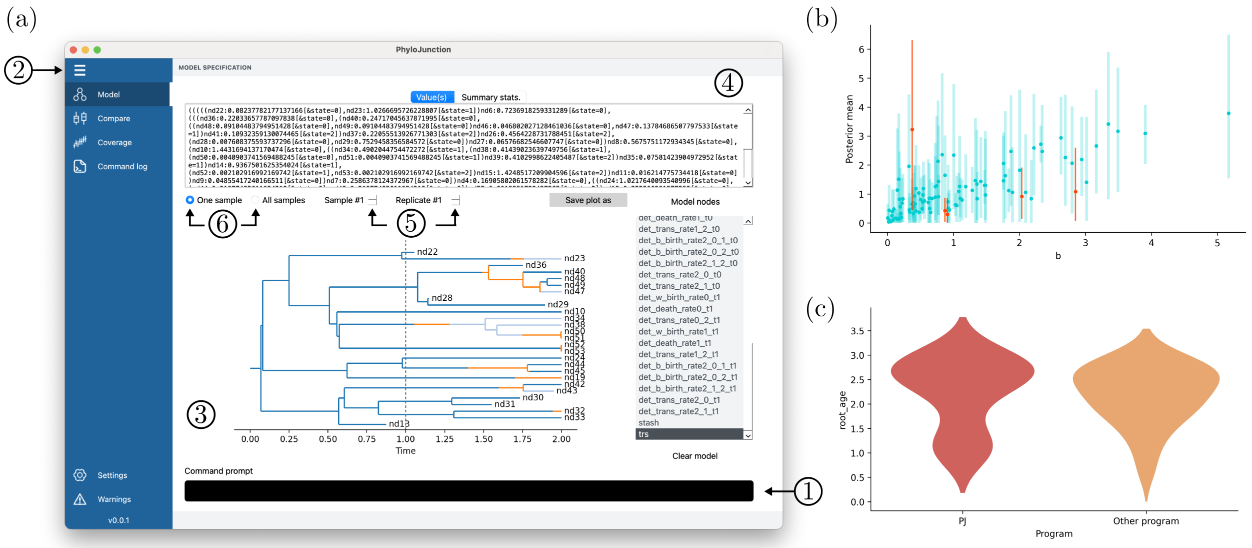
or more existing variables, and give users more control over model building. Without this class of explicit operations, such steps would instead take place out of sight in the backend, or alongside many other actions upon a single `pj` command, both of which can contribute to obscuring model structure.

Computer variables created with `pj` are nodes in the DAG that describes 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 [29, 30] 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 [29]) 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 a declarative language like XML. While commands in `pj` are Rev-like in syntax, and instantiate and store a DAG object in memory (the state of a PhyloJunction section), the similarities with Rev end here. In contrast to an imperative scripting language (e.g., R, Python, Rev), `pj` (i) is easier to learn, understand and write, (ii) enhances reproducibility, (iii) leaves less room for programming mistakes (e.g., variable overwriting, container indexing errors), and (iv) shifts the user's attention from how to specify a model to the structure of the model itself. 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, MCMC parameters, etc. All of these properties make `pj` a lightweight language that can be particularly useful in the classroom.



```
(((((nd22:0.08237782177137166[&state=0],nd23:1.0266695726228807[&state=1])nd6:0.7236918259331289[&state=0],
(((nd36:0.22033657787097838[&state=0],(nd40:0.24717045637871995[&state=0],
((nd48:0.09104483794951428[&state=0],nd49:0.09104483794951428[&state=0])nd46:0.046802027128461036[&state=0],nd47:0.13784686507797533[&state
=1])nd41:0.10932359130074465[&state=2])nd37:0.22055139267713O3[&state=2])nd26:0.4564228731788451[&state=2],
(nd28:0.007608375593737296[&state=0],nd29:0.7529458356584572[&state=0])nd27:0.06576682546607747[&state=0])nd8:0.567575117293434 5[&state=0],
(nd10:1.4431694137170474[&state=0],((nd34:0.4902044754472272[&state=1],(nd38:0.41439023639749756[&state=1],
(nd50:0.004090374156948 8245[&state=0],nd51:0.0040903741569488245[&state=1])nd39:0.4102998622405487[&state=2])nd35:0.07581423904972952[&stat
e=1])nd14:0.9367501625354024[&state=1],
(nd52:0.002102916992169742[&state=1],nd53:0.002102916992169742[&state=2])nd15:1.4248517209904596[&state=2])nd11:0.016214775734418[&state=0]
)nd9:0.04855417240166511[&state=0])nd7:0.2586378124372967[&state=0])nd4:0.16905802061578282[&state=0],((nd24:1.0217640093540996[&state=0],
```

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

3 Standalone command-line and graphical user interfaces

PhyloJunction integrates its multiple utilities for simulating, testing and characterizing evolutionary models via both command-line (CLI) and graphical user interfaces (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) 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 [13], 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., [28, 48]). 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., [21, 53, 84]). 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 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 [2, 3], and new pedagogical material [25, 62, 80]. 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.

4 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, 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 environment), 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., [4, 10, 19, 21, 50, 56, 61]). 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 [70] 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. Furthermore, Python has clear community standards and many tools (e.g., *mypy*, *Sphinx*, *pep8* [77], *pep20* [57]) 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* [34], *pandas*), statistics and Bayesian analysis (e.g., *scipy* [79], *PyMC3* [35], *ArviZ* [38]), and machine learning (e.g., *TensorFlow* [1], *scikit-learn* [55]). Of particular relevance to PhyloJunction's is PyPI's growing list of modules specifically aimed at phylogenetic or population genetic analysis (e.g., *DendroPy* [73], *PyRate* [65], *MESS* [54], *ete3* [33], *msprime* [5]), some of which have already been or may be integrated with PhyloJunction in the future. Below we suggest a few ways in which the latter may be done.

5 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`. Phylo-

Junction is licensed under GNU General Public License v3.0.

6 Future directions

We introduced PhyloJunction, an open-source package for simulating state-dependent speciation and extinction (SSE) processes, a large family of diversification models that has found success across a range of scientific domains [13, 27, 75]. 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 command-line and graphical interfaces, 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 distributions for different types of data models (e.g., DNA and protein sequences, [23, 74, 82]; discrete and continuous characters, [18, 42]), evolutionary clock models (e.g., [15, 16]), population-genetic and phylogeographic processes (e.g., [37, 60]), or models combining any of the above (e.g., [7, 32, 47]). 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 simulation and prototyping of evolutionary models, but we expect its future development to further take on the task of statistical inference. Moving in that direction may involve introducing subroutines for creating textual instructions for Bayesian inference, for example, as required by popular platforms (e.g., RevBayes, BEAST, BEAST 2). Bayesian inference is also possible within a Python environment, although it is unclear how immediately useful existing libraries (e.g., [35]) may be in terms of parameter estimation in phylogenetic space. Porting or implementing Bayesian inference utilities in our platform would in the very least allow synthetic data to be simulated under simple models, and immediately plotted. Such an extension would further empower PhyloJunction as a teaching tool. Alternatively, it should be straightforward to integrate PhyloJunction's functionalities for summarizing data together with Python machine learning libraries. There is increasing evidence [49] backing machine-learning methods as viable alternatives to frequentist and Bayesian evolutionary inference, especially when the latter is very onerous or impossible [75].

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.

7 Acknowledgements

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.

8 Funding

The research presented in this paper 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

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

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

[3] Joëlle Barido-Sottani, Joshua A. Justison, Rui Borges, Jeremy M. Brown, Wade Dismukes, Bruno do Rosário Petrucci, Luiza Guimarães Fabreti, Sebastian Höhna, Michael J. Landis, Paul O. Lewis, Michael R. May, Fáabio K. Mendes, Walker Pett, Benjamin D. Redelings, Carrie M. Tribble, April M.

Wright, Rosana Zenil-Ferguson, and Tracy A. Heath. Lessons learned from teaching virtual phylogenetics workshops. *BSSB*, 1:8245, 2022.

[4] Joëlle Barido-Sottani, Walker Pett, Joseph E. O'Reilly, and Rachel C. M. Warnock. FossilSim: an R package for simulating fossil occurrence data under mechanistic models of preservation and recovery. *Methods Ecol. Evol.*, 10:835–840, 2019.

[5] Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, E Castedo Ellerman, Jared G Galloway, Ariella L Gladstein, Gregor Gorjanc, Bing Guo, Ben Jeffery, Warren W Kretzschumar, Konrad Lohse, Michael Matschiner, Dominic Nelson, Nathaniel S Pope, Consuelo D Quinto-Cortés, Murillo F Rodrigues, Kumar Saunack, Thibaut Sellinger, Kevin Thornton, Hugo van Kemenade, Anthony W Wohns, Yan Wong, Simon Gravel, Andrew D Kern, Jere Koskela, Peter L Ralph, and Jerome Kelleher. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3):iyab229, 2022.

[6] Jeremy M Beaulieu and Brian C O'Meara. Detecting hidden diversification shifts in models of trait-dependent speciation and extinction. *Syst. Biol.*, 65(4):583–601, 2016.

[7] James D Boyko, Brian C O'Meara, and Jeremy M Beaulieu. A novel method for jointly modeling the evolution of discrete and continuous traits. *Evolution*, 77:836–851, 2023.

[8] Daniel S Caetano, Brian C O'Meara, and Jeremy M Beaulieu. Hidden state models improve state-dependent diversification approaches, including biogeographical models. *Evolution*, 72(11):2308–2324, 2018.

[9] J. L. Cantalapiedra, J. L. Prado, M. Hernández Fernández, and M. T. Alberdi. Decoupled ecomorphological evolution and diversification in Neogene-Quaternary horses. *Science*, 355(6325):627–630, 2017.

[10] Julien Clavel, Gilles Escarguel, and Gildas Merceron. mvMORPH: an R package for fitting multivariate evolutionary models to morphometric data. *Methods Ecol. Evol.*, 6:1311–1319, 2015.

[11] Fabien L Condamine, Jonathan Rolland, and Héléne Morlon. Assessing the causes of diversification slowdowns: temperature-dependent and diversity-dependent models receive equivalent support. *Ecol. Lett.*, 22(11):1900–1912, 2019.

[12] Matthew P Davis, Peter E Midford, and Wayne Maddison. Exploring power and parameter estimation of the BiSSE method for analyzing species diversification. *BMC Evol. Biol.*, 13:1–11, 2013.

[13] Jordan Douglas, Fábio K. Mendes, Remco Bouckaert, Dong Xie, Cinthy L. Jimenez-Silva, C. Swanepoel, J. de Ligt, X. Ren, M. Storey, J. Hadfield, C. R. Simpson, J. L. Geoghegan, A. J. Drummond, and D. Welch. 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, 2021.

[14] Alexei J. Drummond, Kylie Chen, Fábio K. Mendes, and Dong Xie. LinguaPhylo: a probabilistic model specification language for reproducible phylogenetic analyses. *PLoS Comp. Biol.*, 19:e1011226, 2022.

[15] Alexei J Drummond, Simon Y W Ho, Matthew J Phillips, and Andrew Rambaut. Relaxed phylogenetics and dating with confidence. *PLoS Biol.*, 4:e88, 2006.

[16] Alexei J Drummond and Marc A Suchard. Bayesian random local clocks, or one rate to rule them all. *BMC Biol.*, 8, 2010.

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

[18] Joseph Felsenstein. Maximum-likelihood estimation of evolutionary trees from continuous characters. *Am. J. Hum. Genet.*, 25:471–492, 1973.

[19] Richard G. Fitzjohn. Diversitree: comparative phylogenetic analyses of diversification in R. *Methods Ecol. Evol.*, 3:1084–1092, 2012.

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

[21] Théo Gaboriau, Fábio K. Mendes, Simon Joly, Daniele Silvestro, and Nicolas Salamin. A multi-platform package for the analysis of intra- and interspecific trait evolution. *Methods Ecol. Evol.*, 11:1–9, 2020.

[22] Emma E. Goldberg and Boris Igić. Tempo and mode in plant breeding system evolution. *Evolution*, 66:3701–3709, 2012.

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

[24] Oskar Hagen and Tanja Stadler. TreeSim GM: Simulating phylogenetic trees under general Bellman–Harris models with lineage-specific shifts of speciation and extinction in R. *Methods Ecol. Evol.*, 9(3):754–760, 2018.

[25] Luke J. Harmon. Phylogenetic comparative methods: learning from trees. *EcoEvoRxiv*, 2019.

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

[27] Paul Heggarty, Cormac Anderson, Matthew Scarborough, Benedict King, Remco Bouckaert, LechosŁaw Jocz, Martin Joachim Kümmel, Thomas Jügel, Britta Irslinger, Roland Pooth, Henrik Liljegren, Richard F. Strand, Geoffrey Haig, Martin Macák, Ronald I. Kim, Erik Anonby, Tijmen Pronk, Oleg Belyaev, Tonya Kim Dewey-Findell, Matthew Boutilier, Cassandra Freiberg, Robert Tegethoff, Matilde Serangeli, Nikos Liosis, Krzysztof Stroński, Kim Schulte, Ganesh Kumar Gupta, Wolfgang Haak, Johannes Krause, Quentin D. Atkinson, Simon J. Greenhill, Denise Kühnert, and Russell D. Gray. Language trees with sampled ancestors support a hybrid model for the origin of Indo-European languages. *Science*, 381(6656):eabg0818, 2023.

[28] Mark S. Hibbins, Lara C. Breithaupt, and Matthew W. Hahn. Phylogenomic comparative methods: accurate evolutionary inferences in the presence of gene tree discordance. *Proc. Natl. Acad. Sci. U.S.A*, 230:e2220389120, 2022.

[29] Sebastian Höhna, Bastien Boussau, Michael J. Landis, Fredrik Ronquist, and John P. Huelsenbeck. Probabilistc graphical model representation in phylogenetics. *Syst. Biol.*, 63:753–771, 2014.

[30] Sebastian Höhna, Michael J. Landis, Tracy A. Heath, Bastien Boussau, Nicolas Lartillot, Brian R. Moore, John P. Huelsenbeck, and Fredrik Ronquist. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Syst. Biol.*, 65:726–736, 2016.

[31] Sebastian Höhna, Michael R May, and Brian R Moore. TESS: an R package for efficiently simulating phylogenetic trees and performing Bayesian inference of lineage diversification rates. *Bioinformatics*, 32(5):789–791, 2016.

[32] Xia Hua, T Herdha, and C J Burden. Protracted speciation under the state-dependent speciation and extinction approach. *Syst. Biol.*, 71:1362–1377, 2022.

[33] Jaime Huerta-Cepas, Francois Serra, and Peer Bork. ETE 3: Reconstruction, analysis and visualization of phylogenomic data. *Mol. Biol. Evol.*, 33:1635–1638, 2016.

[34] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[35] Salvatier J., Wiecki T.V., and Fonnesbeck C. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[36] D. G. Kendall. On the generalized “birth-and-death” process. *Ann. Math. Statist.*, 19:1–15, 1948.

[37] J F C Kingman. On the genealogy of large populations. *J. Appl. Probab.*, 19:27–43, 1982.

[38] Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. Arviz a unified library for exploratory analysis of bayesian models in python. *JOSS*, 4(33):1143, 2019.

[39] Michael J. Landis, Ignacio Quintero, Martha M. Mu noz, and Michael J. Donoghue. Phylogenetic inference of where species spread or split across barriers. *Proc. Natl. Acad. Sci. U.S.A.*, 119:e2116948119, 2022.

[40] Hayley C. Lanier and L. Lacey Knowles. Is recombination a problem for species-tree analyses? *Syst. Biol.*, 61:691–701, 2012.

[41] Maya A Lewinsohn, T Bedford, N F Müller, and A F Feder. State-dependent evolutionary models reveal modes of solid tumour growth. *Nat. Ecol. Evol.*, 7:581–596, 2023.

[42] Paul O Lewis. A likelihood approach to estimating phylogeny from discrete morphological character data. *Syst. Biol.*, 50(6):913–925, 2001.

[43] Stilianos Louca and Michael Doebeli. Efficient comparative phylogenetics on large trees. *Bioinformatics*, 34(6):1053–1055, 2018.

[44] Wayne P. Maddison, Peter E. Midford, and Sarah P. Otto. Estimating a binary character’s effect on speciation and extinction. *Syst. Biol.*, 56:701–10, 2007.

[45] Nicholas A. Mason, Nicholas K. Fletcher, Brian A. Gill, W. Chris Funk, and Kelly R. Zamudio. Coalescent-based species delimitation is sensitive to geographic sampling and isolation by distance. *Syst. Biodivers.*, 18:269–280, 2020.

[46] Nicholas J. Matzke. Statistical comparison of DEC and DEC+J is identical to comparison of two ClaSSE submodels, and is therefore valid. *J. Biogeogr.*, 49:1805–1824, 2022.

[47] Michael R May and Brian R Moore. 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(3):530–544, 2019.

[48] Fábio K. Mendes, Jesualdo A. Fuentes-González, Joshua G. Schraiber, and Matthew W. Hahn. A multispecies coalescent model for quantitative traits. *eLife*, 7:e36482, 2018.

[49] Yu K. Mo, Matthew W. Hahn, and Megan L. Smith. Applications of machine learning in phylogenetics. *EcoEvoRxiv*, 2023.

[50] Hélène Morlon, Eric Lewitus, Fabien L. Condamine, Marc Manceau, Julien Clavel, and Jonathan Drury. RPANDA: an R package for macroevolutionary analyses on phylogenetic trees. *Methods Ecol. Evol.*, 7:589–597, 2015.

[51] Sarah A Nadeau, Timothy G Vaughan, Jérémie Scire, Jana S Huisman, and Tanja Stadler. The origin and early spread of SARS-CoV-2 in Europe. *Proc. Natl. Acad. Sci. U.S.A.*, 118(9):e2012008118, 2021.

[52] S. Nee, R. M. May, and P. H. Harvey. The reconstructed evolutionary process. *Philos Trans. R. Soc. Lond B. Biol. Sci.*, 344:305–311, 1994.

[53] Huw A. Ogilvie, Fábio K. Mendes, Timothy G. Vaughan, Nicholas J. Matzke, Tanja Stadler, David Welch, and Alexei J. Drummond. Novel integrative modeling of molecules and morphology across evolutionary timescales. *Syst. Biol.*, 71:208–220, 2022.

[54] Isaac Overcast, Megan Ruffley, James Rosindell, Luke Harmon, Paulo A. V. Borges, Brent C. Emerson, Rampal S. Etienne, Rosemary Gillespie, Henrik Krehenwinkel, D. Luke Mahler, Francois Massol, Christine E. Parent, Jairo Patiño, Ben Peter, Bob Week, Catherine Wagner, Michael J. Hickerson, and Andrew Rominger. A unified model of species abundance, genetic diversity, and functional diversity reveals the mechanisms structuring ecological communities. *Mol. Ecol. Resour.*, 21(8):2782–2800, 2021.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[56] Matthew W. Pennell, Jonathan M. Eastman, Graham J. Slater, Jeremy W. Brown, J. C. Uyeda, R. G. Fitzjohn, Michael E. Alfaro, and Luke J. Harmon. geiger v2.0: an expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. *Bioinformatics*, 30:2216–2218, 2014.

[57] Tim Peters. The Zen of Python. PEP 20, 2004.

[58] Ignacio Quintero, Michael J Landis, Walter Jetz, and Hélène Morlon. The build-up of the present-day tropical diversity of tetrapods. *Proc. Natl. Acad. Sci. U.S.A.*, 120(20):e2220672120, 2023.

[59] Dan L Rabosky and Emma E Goldberg. Model inadequacy and mistaken inferences of trait-dependent speciation. *Syst. Biol.*, 64:340–355, 2015.

[60] Bruce Rannala and Ziheng Yang. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics*, 164:1645–1656, 2003.

[61] Liam J. Revell. phytools: an R package for phylogenetic comparative biology (and other things). *Methods Ecol. Evol.*, 3:217–223, 2012.

[62] Liam J. Revell and Luke J. Harmon. *Phylogenetic comparative methods in R.* Princeton University Press, 2022.

[63] Orlando Schwery, William A Freyman, and Emma E Goldberg. adequaSSE: model adequacy testing for trait-dependent diversification models. *bioRxiv*, pages 2023–03, 2023.

[64] Jéremie Sciré, Joëlle Barido-Sottani, Denise Kühnert, Timothy G. Vaughan, and Tanja Stadler. Robust phylodynamic analysis of genetic sequencing data from structured populations. *Viruses*, 14:1–18, 2022.

[65] Daniele Silvestro, Nicolas Salamin, and Jan Schnitzler. PyRate: a new program to estimate speciation and extinction rates from incomplete fossil data. *Methods Ecol. Evol.*, 5:1126–1131, 2014.

[66] Daniele Silvestro, Jan Schnitzler, Lee Hsiang Liow, Alexandre Antonelli, and Nicolas Salamin. Bayesian estimation of speciation and extinction from incomplete fossil occurrence data. *Syst. Biol.*, 63(3):349–367, 2014.

[67] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.

[68] Andrew G Simpson, Peter J Wagner, Scott L Wing, and Charles B Fenster. Binary-state speciation and extinction method is conditionally robust to realistic violations of its assumptions. *BMC Evol. Biol.*, 18(1):1–11, 2018.

[69] Tanja Stadler. TreeSim. Available from `http://cran.r-project.org/web/packages/TreeSim/index.html`. [Internet]: 2010.

[70] Tanja Stadler. Simulating trees with a fixed number of extant species. *Syst. Biol.*, 60:676–684, 2011.

[71] Tanja Stadler and Sebastian Bonhoeffer. Uncovering epidemiological dynamics in heterogeneous host populations using phylogenetic methods. *Philos Trans. R. Soc. Lond B. Biol. Sci.*, 368:20120198, 2013.

[72] Tanja Stadler, Denise Kühnert, Sebastian Bonhoeffer, , and Alexei J. Drummond. Birthdeath skyline plot reveals temporal changes of epidemic spread in HIV and hepatitis C virus (HCV). *Proc. Natl. Acad. Sci. U.S.A.*, 110:228–33, 2013.

[73] J. Sukumaran and Mark T. Holder. DendroPy: A Python library for phylogenetic computing. *Bioinformatics*, 26:1569–1571, 2010.

[74] Simon Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. In Robert M Miura, editor, *Some Mathematical Questions in Biology - DNA Sequence Analysis*, volume 17, pages 57–86. Bantam, 1986.

[75] Ammon Thompson, Benjamin Liebeskind, Erik J. Scully, and Michael Landis. Deep learning and likelihood approaches for viral phylogeography converge on the same answers whether the inference model is right or wrong. *bioRxiv*, 2023.

[76] Carrie M Tribble, José Ignacio Márquez-Corro, Michael R May, Andrew L Hipp, Marcial Escudero, and Rosana Zenil-Ferguson. Detecting shifts in the mode of chromosomal speciation across the cosmopolitan plant lineage *Carex*. *bioRxiv*, pages 2023–09, 2023.

[77] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Style guide for Python code. PEP 8, 2001.

[78] Timothy G. Vaughan. ReMASTER: improved phylodynamic simulation for BEAST 2.7. *bioRxiv*, 2023.

[79] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[80] Tandy Warnow. *Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation*. Cambridge University Press, 2017.

[81] Marjorie G. Weber and Anurag A. Agrawal. Defense mutualisms enhance plant diversification. *Proc. Natl. Acad. Sci. U.S.A.*, 111:16442–16447, 2014.

[82] Simon Whelan and Nick Goldman. A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach. *Mol. Biol. Evol.*, 18(5):691–699, 2001.

[83] G. U. Yule. 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, 1924.

[84] Rong Zhang, Alexei J. Drummond, and Fábio K. Mendes. Fast Bayesian inference of phylogenies from multiple continuous characters. *Syst. Biol.*, syad067, 2023.