# LinguaPhylo: a probabilistic model specification language for reproducible phylogenetic analyses

Alexei J. Drummond[1,2,3]*, Kylie Chen[1,2,3], Fábio K. Mendes[1,4], Dong Xie[1,2,3]

**1** Centre for Computational Evolution, University of Auckland, Auckland, New Zealand
**2** School of Biological Sciences, University of Auckland, Auckland, New Zealand
**3** School of Computer Science, University of Auckland, Auckland, New Zealand
**4** Department of Biology, Washington University in St. Louis, St. Louis, United States

* a.drummond@auckland.ac.nz

## Abstract

Phylogenetic models have become increasingly complex, and phylogenetic data sets ~~larger and richer. Yet~~ have expanded in both size and richness. However, current inference tools lack a model specification language that ~~succinctly describes a full phylogenetic analysis independently~~ can concisely describe a complete phylogenetic analysis while remaining independent of implementation details. We ~~present~~ introduce a new lightweight and concise model specification language, ~~called 'LPhy', that is~~ LPhy, which is designed to be both human and ~~machine readable. 'LPhy' is accompanied by a~~ machine-readable. A graphical user interface ~~for building modelsand simulating datausing this new language, as well as for creating~~ accompanies LPhy, allowing users to build models, simulate data, and create natural language narratives describing ~~such~~ the models. These narratives can ~~form the basis of~~ serve as the foundation for manuscript method sections. ~~We also introduce~~ Additionally, we present a command-line interface for converting LPhy-specified models into analysis specification files (in XML format) ~~to be used alongside~~ compatible with the BEAST2 software platform. ~~Together~~Collectively, these tools ~~will clarify the description~~ aim to enhance the clarity of descriptions and reporting of probabilistic models in phylogenetic studies, ~~and improve result reproducibility~~ ultimately promoting reproducibility of results.

## Author summary

~~We describe a succinct~~ In this study, we introduce a concise domain-specific language designed to accurately specify the details of a phylogenetic model~~for the purposes of reproducibility or reuse. In addition, we have~~, promoting reproducibility and reuse. We have also developed a graphical software package that ~~can be used~~ enables users to construct and simulate data from models described in this new language, ~~as well as create~~ while also generating natural language narratives that can ~~form the basis of a description of the model~~ serve as the foundation for the method section of a manuscript. ~~Finally, we report on~~ Furthermore, we present a command-line program that ~~can be used to generate~~ facilitates the creation of input files for the BEAST2 software package based on a model specified in ~~this new language. These tools together should aid in the goal~~ our novel language. Collectively, these tools contribute to the goals of reproducibility and reuse within the realm of probabilistic phylogenetic models.

# 1 Introduction

Transparency is a scientific ideal, and replicability and reproducibility lie at the heart of the scientific ~~endeavor~~ endeavour [1, 2]. Metaresearch efforts have uncovered the so-called ~~"reproducibility crisis"~~ 'reproducibility crisis' [3] in many scientific domains [3]. In recent years, the growing number of computational biology software packages available has enabled greater choice in data analyses, but at the cost of increased complexity in the data-preparation and analytical pipelines [4]. This increases the difficulty of accurately reporting and reproducing analyses. These barriers have been ~~recognized~~ recognised by the wider genomics research community [4] as well as within evolutionary biology [5].

In evolutionary biology, phylogenetics has become a highly technical discipline [5]. The most general phylogenetic tools are Bayesian methods (e.g., BEAST, BEAST 2, MrBayes and RevBayes; [6–9]) that can simultaneously reconstruct phylogenetic tree topology and divergence times, as well as estimate the related micro-evolutionary and macro-evolutionary parameters. Phylogenetic analyses often combine multiple models within a complex pipeline to answer questions in evolutionary biology such as species evolution [10–12], ancestral ~~bio-geographical~~ biogeographical ranges [13, 14], and epidemic dynamics [15, 16].

Reproducing, reusing and interpreting a phylogenetic model is not trivial, and requires an understanding of the input data, details of the model (i.e., its parameters and how they are ~~are~~ related and their priors), and inference methodology. The latter can include complex Markov chain Monte Carlo (MCMC) proposal distributions and sampling algorithms which are not part of the model. Currently, little research has been done on the readability, reproducibility and ~~re-usability~~ reusability of phylogenetic analyses employing phylogenetic models. Our paper presents a tool that aims to: (i) facilitate concise and exact communication of phylogenetic models, (ii) improve reproducibility, and (iii) increase re-usability of phylogenetic models and their variations on new datasets.

Probabilistic graphical models (PGM) have previously been introduced to phylogenetic inference by Höhna et al. [8], where they are described in the Rev language of RevBayes [8]. Previous attempts to address model specification ~~and analysis setup~~ of Bayesian phylogenetic analyses include BEAST-style XMLs (eXtensible Markup Language) developed for the BEAST software [6, 7]~~and the~~, the Nexus-based language of MrBayes [9] and the aforementioned Rev programming language ~~for~~ used in RevBayes [8]. The extensibility of XMLs provides flexibility to developers allowing them to create new descriptive tags for specifying new models. ~~However, BEAST-style XMLs are~~ Unfortunately, BEAST XMLs can be hard to read due to their ~~verbose syntax and the usual complexity of the models being specified. Unsurprisingly, translating BEAST or BEAST 2 analyses from XMLs into text descriptions is difficult and error prone. Our experience suggests that most users are unable to verify if the XML analysis file matches the description in their manuscript.~~ XML-based, verbose syntax, which is unfamiliar to many users. The deeply nested structures and hierarchical organisation in BEAST XMLs make understanding model components and their relationships more challenging. In contrast, more general probabilistic programming languages such as found in JAGS, BUGS, and Stan employ more concise, linear structures that simplify comprehension and modification of models. Additionally, probabilistic programming languages are more accessible and widely applicable within the statistical modelling community, while BEAST XMLs are more domain-specific. The Rev language [8] ~~is~~ offers an alternative to XMLs, ~~and it incorporates~~ incorporating conventional notation from ~~more~~ general probabilistic programming languages~~. This feature makes Rev model specification more~~

recognisable to , making model specifications more recognizable and flexible for statistically literate usersand sometimes more flexible, but users must still contend . However, users still face challenges with verbose and prosaic implementation detailsextraneous to the model (extraneous implementation details, such as MCMC sampling settings, logging details information, and proposal distributions), as well as with the error-prone task of describing . Additionally, the REV model accurately task of accurately describing the Rev model in natural language when describing it in the methods section of a manuscriptfor a manuscript's methods section can be error-prone, further complicating the process of model specification and communication.

Here, we introduce In this study, we present LinguaPhylo (LPhy, pronounced 'el-fee'), an open-source model specification language aimed at improving designed to enhance the readability, reproducibility, and re-usability reusability of phylogenetic models. The LPhy language has LPhy boasts a simple syntax for that enables succinct specification of complex models, and is implemented in within a framework that generates precise text accurate textual descriptions and graphical diagrams of phylogenetic models from based on user input.

# 2 Methods

The LPhy language is designed to enable the specification of phylogenetic models using a concise and readable syntax. The reference implementation is built on top of the Java programming language and provides features for: (i) concise formal specification of phylogenetic models on real or synthetic data, (ii) data simulation from phylogenetic models, (iii) integration with the BEAST 2 phylogenetic inference framework, and (iv) an extensibility mechanism for adding new functionality and data types to the LPhy language.

## 2.1 Language features

The LPhy language provides a simple syntax for specifying and simulating under different evolutionary models models. These include tree models for phylogenies and genealogies (e.g., birth-death, coalescent) , substitution models for genomic sequences (e.g., GTR [17]), and parametric distributions for discrete and continuous parameters (e.g., Dirichlet, Normal).

In the context of simulation under a model specified with LPhy , "generative distributions" produce values for random variables, which in turn can be of different "datatypes": trees, continuous morphology, sequence alignments. These random variables constitute stochastic nodes in the probabilistic graphical model (PGM) that LPhy builds, but it is also possible to assign a fixed value to a node, in which case a constant node is created (see more on this below)is described by an Extended Backus–Naur form grammar (EBNF) [18]. We used ANTLR [19,20] to generate an LPhy parser in Java. This Java-based LPhy parser was used as the foundation for the development of LPhyStudio and LPhyBEAST software packages. The ANTLR parser generator can also be targeted to other general programming languages like Python, C++ and Javascript. This gives a relatively easy path to implementation of LPhy support in other phylogenetic software packages.

The specification of generators and named variables is done through commands the user can type on LPhy 's command prompt or execute from a script file:

Listing 1 specifies a complete phylogenetic model using only five lines of code inside two blocks. Constant nodes with fixed values are shown in magenta. The first block specifies "200" nucleotide sites in "L" and ten different taxa named '1' to '10' in

~~"taxa". The second block declares the stochastic nodes or random variablesin green, and their generative distributions in blue.~~

~~The graphical representation of the probabilistic model defined in Listing 1.~~

The main components of the LPhy language grammar are variables, arrays of variables, and generators. There are three classes of generators: (i) generative ~~distrivutions~~ distributions which produce values for random variables, (ii) deterministic functions that produce the deterministic values given the same input values, and (iii) method calls. For deterministic functions and method calls, these generators produce deterministic nodes in the PGM. This is illustrated in Figure 1, which shows a graphical representation of the model specified in ~~Listing~~ Example 1. Deterministic nodes are shown as diamonds (e.g., the ~~"Q"~~ 'Q' matrix of the Jukes-Cantor model [21]). Stochastic nodes are represented by circles (e.g., $\Theta$, the population size governing the coalescent times generated by the Coalescent process [22]), and constant nodes are represented by squares (e.g., the mean of the log-normal generative distribution underlying $\Theta$).

An LPhy script is a text file with the '.lphy' file extension and is case-sensitive. In the reference implementation, syntax checking is performed during execution. Control flow structures are not allowed in order to promote simplicity and readability, facilitating a lower barrier to entry and a gentler learning curve. Instead of loop structures, we provide implicit vectorization similar to R [23]. Allowing all arguments of distributions and functions to be vectorised results in more compact and expressive model specifications. This can lead to clearer representations of the model's structure and relationships. Additionally an optional "replicates" argument allows users to generate vectors of IID random variables easily, further simplifying the model specification. When designing new generators, optional arguments in functions and generative distributions are allowed, these arguments may or may not have default values. Two distinct code blocks are used to differentiate between the part of the script describing the data (the data block), and part describing the model (the model block). The code block structure is inspired by the programming language Stan [24]. The syntax for the specification of random variables is similar to probabilistic programming languages such as JAGS [25], BUGS [26, 27] and Stan [24]. Scripts can be loaded into LPhyStudio via the menu or the toolbar. Additionally, we provide a console for executing LPhy commands line by line within LPhyStudio.

The LPhy language grammar does not specify any explicit types, nor does it specify any protected pre-defined generative distributions or functions, except for a very small number of mathematical functions that allow simple expressions. Accompanying the LPhy language grammar is the LPhy reference implementation, a Java implementation offering standard statistical and phylogenetic distributions, as well as supporting functions with specified Java types. Implementers of the LPhy language in other systems should support the reference distributions, using equivalent types in their respective languages.

In the reference Java implementation, variables can possess primitive or custom types. Primitive types include doubles, integers, booleans, and strings, while custom types, such as alignments, trees, and discrete traits, are created as Java Objects using the generator's constructor. Variables can be vectorised into an array of elements using the 'replicates' argument. Java-style overloading supports function overloading, and type checking for generator arguments is performed during execution.

### 2.1.1 Syntax

In LPhy, each line's syntax consists of a variable declaration on the left-hand side, a specification operator, and a generator on the right-hand side, with lines ending with a semicolon character. For instance:

```
b ~ Normal(mean= 0.0, sd=1.0);
```

In this example, variable $b$ is specified by the normal generative distribution `Normal()` with two arguments: the mean `mean` (0.0) and the standard deviation `sd` (1.0).

The left-hand side declares the name of a variable or an array of variables (case sensitive). The right-hand side specifies the values of the variable or array. This can be a constant value, array of constant values, deterministic function or stochastic generative distribution. Deterministic functions and generative distributions are matched by method signatures from constructors of their corresponding Java class in the reference implementation. A list of functions and generative distributions is shown in the LPhy reference implementation manual. Arguments inside functions or generative distributions follow the convention: (*argument name*) = (*value*).

### 2.1.2 Specification operators

An equal sign = is used to specify deterministic or constant values for variables, such as:

```
a = 2.0;
```

A tilde sign ~ is a specification operator which denotes the relationship between a stochastic random variable and its generator. In a Bayesian context, this is the prior distribution of the random variable. Or when given observed data, this specifies the likelihood function. For example, this specifies a prior for the variable $b$:

```
b ~ Normal(mean=0.0, sd=1.0);
```

The type of a variable is inferred from the return type of its generator and does not need to be declared. For arguments of functions or generative distributions, the types are defined in the LPhy reference implementation manual.

### 2.1.3 Arrays

Arrays can be defined using square brackets with elements delimited by comma separators. For sequences of consecutive integers, we allow a more compact notation using the colon ':' to define a range. For example:

```
c =[1, 2, 3, 4, 5];
d =[2:10];
```

The variable c has an array with values (1, 2, 3, 4, 5). The variable d has an array with values 2 to 10.

### 2.1.4 Code blocks

The **data** and **model** keywords are reserved to specify code blocks inside curly brackets. The data block is used to read in and store input data, which are used by the model. Within the data block, we can read in alignment data via the NEXUS or FASTA parsers, specify constant values, and store metadata about the dataset. The model block is used to define the models and parameters in a Bayesian phylogenetic analysis.

```
data {
  L = 200;
  taxa = taxa(names=1:10);
}
model {
  Θ ~ LogNormal(meanlog=3.0, sdlog=1.0);
  ψ ~ Coalescent(theta=Θ, taxa=taxa);
  D ~ PhyloCTMC(L=L, Q=jukesCantor(), tree=ψ);
}
```

**Example 1.** An LPhy script defining a constant-size coalescent tree prior with log-normally distributed population sizes, a strict clock model, and a Jukes-Cantor model on 10 nucleotide sequences with 200 sites (base pairs).

Example 1 specifies a complete phylogenetic model using only five lines of code inside two blocks. The first block specifies '200' nucleotide sites in 'L' and ten taxa named from 1 to 10 in 'taxa'. Taxa can be declared as strings or as numbers. In this example, the taxa names are numbered. The second block declares stochastic nodes or random variables highlighted in bluishgreen, and their generative distributions highlighted in blue. Constant nodes with fixed values are shown in vermillion.

Figure 1 shows this model specification represented as a probabilistic graphical model. Stochastic nodes are shown as circles, deterministic functions are shown as diamonds, and constants are shown as squares.

### 2.1.5 Variable vectorization

~~Named variables in LPhy~~ Named variables can be scalars or vectors. Any generator can be vectorized to produce a vector of independent and identically distributed (i.i.d.~~values by~~) random variables using the `replicates` keyword:

```
κ ~ LogNormal(meanlog=0.5, sdlog=1.0, replicates=3);
```

In the example above, $\kappa$ is a random vector of three log-normally distributed i.i.d. values.

Vectorization can also be applied to a generative distribution that ~~already~~ produces vectors. In ~~which~~ this case, the output will be a matrix~~as~~, as seen in the following example~~, where the major dimension has size 3, with each element of the $\pi$ random variable being a vector of base frequencies~~.

```
π ~ Dirichlet(conc=[2.0, 2.0, 2.0, 2.0], replicates=3);
```

Here, $\pi$ contains 3 vectors, where each vector represents nucleotide base frequencies. So the resultant matrix will be 3 x 4 (major dimension of 3 and a minor dimension of 4).

~~In the example above, $\kappa$ is a random vector of three log-normally distributed i. i.d. values.~~

~~Finally,~~ A second mechanism for vectorization can be ~~coerced simply~~ used by passing a vector ~~input~~ of elements instead of a ~~scalar input to one or more of the inputs of~~ single element as an input argument of a generator:

```
κ ~ LogNormal(meanlog=0.5, sdlog=1.0, replicates=3);
π ~ Dirichlet(conc=[2.0, 2.0, 2.0, 2.0], replicates=3);
Q = hky(kappa=κ, freq=π);
```

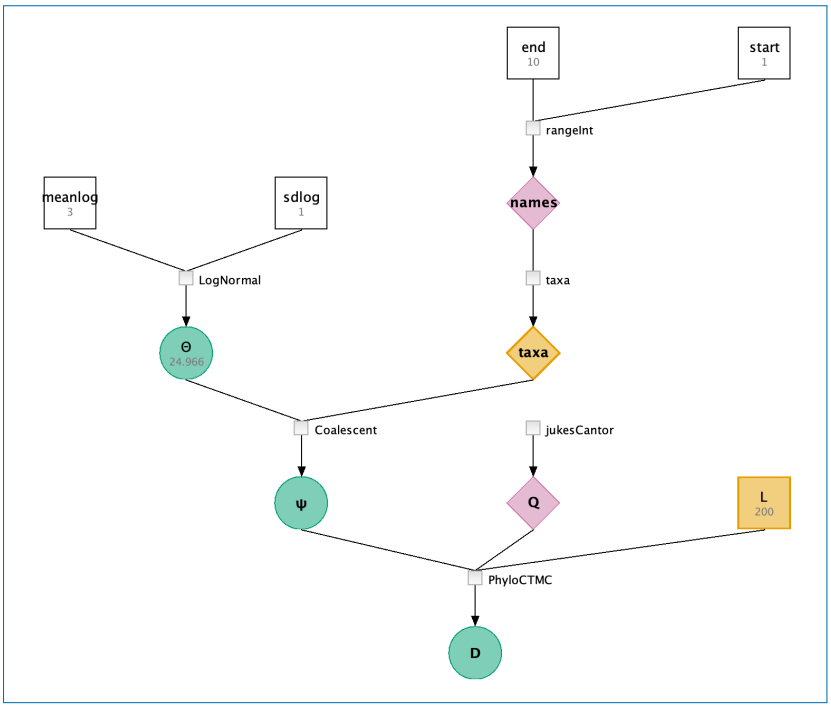


**Fig 1.** The graphical representation of the probabilistic model defined in Example 1.

Here, ~~since both~~ $\kappa$ and $\pi$ are ~~random vectorswith the same major dimension length (3), we can assign them as values of~~ vectors, both with a major dimension of 3. These are passed as arguments into the `hky` deterministic function, which ~~in turn outputs a vector of~~ returns a vector containing three instantaneous rate matrices ~~,~~ stored in $Q$.

### 2.1.6 Extension mechanism

LPhy is designed to be modular and extensible to encourage integrative software development and adoption by method developers in the field. LPhy core and its extensions are built using Java 17 with long-term support (LTS) and Gradle 4. The LPhy extension mechanism is implemented using the Java Platform Module System (JPMS) and the Java Service Provider Interface (SPI). This modular extension framework allows developers greater flexibility in code development and software releases. New functionality such as generative distributions, data types, and deterministic functions can be developed within LPhy extension modules. Additionally, software releases can be done independently to core LPhy releases.

### 2.1.7 Parametric distributions

~~LPhy implements~~ The LPhy reference implementation comes with a series of parametric distributions commonly used in evolutionary models, ~~such as Uniform, Normal, Lognormal, Gamma, Exponential, and Dirichlet. Specifying parametric distributions~~ including uniform, normal, log-normal, gamma, exponential, and dirichlet distributions. Parametric distributions can be specified as generative distributions for model parameters ~~can be achieved~~ by:

```
μ ~ LogNormal(meanlog=-5.0, sdlog=1.25);
```

Each parametric distribution is characterized by its own parameters. In the example

above, the extinction rate parameter $\mu$ is drawn from a ~~Lognormal~~ log-normal distribution with mean -5 and standard deviation 1.25 in log space.

### 2.1.8 Tree models

Tree models are used to generate phylogenetic trees, and are central components in phylogenetic simulation and analysis~~, and are used to generate phylogenetic trees. Below we briefly expand on~~. We briefly describe some of the main tree models implemented in LPhy ~~.~~ below. This includes coalescent models and birth-death models.

#### ~~Coalescent models~~Serially sampled coalescent model

~~*Serially sampled coalescent*~~

The simplest coalescent model LPhy implements is the constant-population size coalescent, which can be extended to generate serially sampled (heterochronous) data [28]:

```
Θ ~ LogNormal(meanlog=3.0, sdlog=1.0);
ψ ~ Coalescent(theta=Θ, taxa=taxa(names=["a", "b", "c", "d"],
  ages=[0.0, 1.0, 2.0, 3.0]));
```

The script above specifies a serially sampled constant-population size coalescent (a generative distribution) for tree $\psi$ with four taxa, ~~"a", "b", "c", and "d"~~ 'a', 'b', 'c', 'd' sampled at 0.0, 1.0, 2.0 and 3.0 time points, respectively. Here, sample time is defined as the age of a sample, ~~with~~ where 0.0 ~~meaning~~ represents the present moment.

~~*Structured coalescent*~~

#### Structured coalescent model

The structured coalescent [29, 30] generalizes the constant-population size coalescent [22] by allowing multiple demes, each of which are characterized by a distinct population size~~(in~~. In the simplest case this population size does not change through time~~)~~. Demes exchange individuals according to migration rates $m$ specified in the off-diagonal elements of a migration matrix $M$, where the diagonal elements store the population sizes, $\theta$, of each deme. For $K$ demes, the population size parameter ~~"theta"~~ 'theta' is a $K$-tuple, and $m$ is a $(K^2 - K)$-tuple.

```
M = migrationMatrix(theta=[0.1, 0.1], m=[1.0, 1.0]);
g ~ StructuredCoalescent(M=M, n=[15, 15]);
```

In the example above, `migrationMatrix` is a deterministic function and `StructuredCoalescent` is a generative distribution~~; both are generators~~. A stochastic node ~~"g"~~ 'g' stores a gene tree sampled from a two-deme structured coalescent process.

~~*Skyline coalescent model*~~

#### Skyline coalescent model

The skyline coalescent model [31] is a coalescent process that models changes in population sizes. This model is characterized having a constant population size for each coalescent interval, with instantaneous changes in population size at some coalescent events.

The following script specifies a Skyline coalescent model with 10 coalescent intervals (~~hence~~ 11 taxa), ~~governed by~~ with four distinct population sizes.

```
g ~ SkylineCoalescent(theta=[0.1, 0.2, 0.3, 0.4], groupSizes=[4,3,2,1]);
```

Here, ~~"g"~~ 'g' is a stochastic node in the PGM, with its value sampled from the `SkylineCoalescent` generative distribution. Ten coalescent intervals are defined through the ~~"groupSizes"~~ 'groupSizes' argument: the first four coalescent intervals will be drawn assuming a ~~"theta"~~ 'theta' of 0.1, the next three intervals with ~~"theta"~~ 'theta' equal to 0.2, and so on.

**Birth-death models**

Birth-death models are commonly used in macroevolution as sampling distributions for species trees. Models that parameterize the fossilization process can be especially useful, as they allow users to leverage fossil ages as data~~; when~~. When fossil morphological characters have also been scored, total-evidence dating can be carried out [11]. One such tree model is the serially sampled birth-death process [32], whose parameters ~~"psi" and "rho" (see below)~~ 'psi' and 'rho' below represent the rate of sampling extinct and extant lineages, respectively:

```
ages = [0.0, 1.0, 2.0, 3.0, 4.0];
tree ~ BirthDeathSerialSampling(lambda=1, mu=0.5, rho=0.1,
psi=1, rootAge=5, ages=ages);
```

Other tree models include the birth-death [33] and fossilized birth-death processes [34], as well as the Yule process [35]. ~~See the Supplementary Material for more examples.~~

### 2.1.9 Substitution models

Substitution models consist of continuous-time Markov chains (CTMC) used to model the evolution of discrete characters, such as nucleotides and amino acid residues. LPhy implements a general formulation of a phylogenetic CTMC, known as the GTR model [17], under which several nested models can be specified. The first line below constructs the instantaneous rate matrix ~~(Q)~~ $Q$ for an HKY model [36], which is then used to in `PhyloCTMC`, the generative distribution ~~over~~ for the sequence alignment data ~~(D)~~$D$:

```
taxa = taxa(names=1:10);
Q = hky(kappa=2.0, freq=[0.2, 0.25, 0.3, 0.25]);
Θ ~ LogNormal(meanlog=3.0, sdlog=1.0);
ψ ~ Coalescent(theta=Θ, taxa=taxa);
D ~ PhyloCTMC(L=200, Q=Q, tree=ψ);
```

Other substitution models can be easily specified by ~~assigning~~ passing different instantaneous transition rate ~~(Q) matrices~~ matrices $Q$ to `PhyloCTMC`, e.g., the matrix of the Jukes-Cantor model [21]:

```
D ~ PhyloCTMC(L=200, Q=jukesCantor(), tree=ψ);
```

For forward simulation `PhyloCTMC` is used as a generative distribution for a multiple sequence alignment, which is here represented by stochastic node ~~"D"~~ 'D'. When the model is employed for statistical inference, and data ~~D~~ $D$ is known, the `PhyloCTMC` represents the phylogenetic likelihood~~(see Data clamping below)~~. More details are discussed in the Data clamping section.

### 2.1.10 Evolutionary clock models

~~Evolutionary (molecular) clock models~~ Molecular clocks are used to model the rate of evolutionary change and ~~whether/~~how it varies over time. The LPhy language supports strict clock [37,38], local clock [39] and relaxed clock [40] models. Specifying a clock model is done by generating evolutionary rate values, one per phylogenetic tree branch, and then multiplying those rates by the length of the corresponding branch~~(measured in the chosen~~. The branches of the tree are measured in units of time~~)~~, effectively scaling the tree to ~~units~~ the number of expected substitutions per site.

The simplest clock model is the strict clock, ~~under which~~ where the evolutionary rate remains constant over the entire tree. Specifying a strict molecular clock can be done by specifying the ~~"mu"~~ 'mu' parameter in the `PhyloCTMC` distribution~~(default~~. The default value for the clock rate 'mu' is 1.0~~):~~.

```
λ ~ LogNormal(meanlog=3.0, sdlog=1.0);
ψ ~ Yule(lambda=λ, n=16);
D ~ PhyloCTMC(L=200, Q=jukesCantor(), tree=ψ, mu=0.5);
```

More realistic clock models like the uncorrelated relaxed clock model [40] assume the rate for each branch is drawn according to a parametric distribution. For example, a relaxed clock with rates drawn from a log-normal distribution can be constructed as follows:

```
λ ~ LogNormal(meanlog=3.0, sdlog=1.0);
ψ ~ Yule(lambda=λ, n=16);
branchRates ~ LogNormal(sdlog=0.5, meanlog=-0.25, replicates=ψ.branchCount());
D ~ PhyloCTMC(L=200, Q=jukesCantor(), branchRates=branchRates, tree=ψ);
```

Here, 30 rates are drawn independently from a log-normal distribution, and then each is assigned to one of the 30 branches of tree $\psi$.

### 2.1.11 Inference and data clamping

In addition to simulation, LPhy allows users to use a specified model for inference~~(at the moment, LPhy interfaces only with BEAST 2, see the "LPhy and BEAST 2" section below). The key step when setting~~. We have developed LPhyBEAST which provides support for inference with the BEAST2 engine.

To set up an inferential analysis with LPhy ~~, after specifying the model, is to carry out "data clamping".~~

~~Data clamping should be familiar to users of~~ 'data clamping' is performed similar to the Rev language [8]~~, and consists of assigning~~. Data clamping involves associating an observed value ~~to~~ with a random variable in the ~~probabilistic model(i.e., to~~ model, which is represented as a stochastic node in the probabilistic graphical model (PGM). ~~Effectively, by~~ By clamping data to a node, ~~a user tells the inference machinery~~ the user is informing the inference engine that the value of ~~a random~~ that particular variable is known and will be conditioned on for ~~purposes~~ the purpose of inference.

In LPhy, data clamping can be accomplished using the 'data block'. This block allows the user to specify the observed values of certain variables in the model, effectively clamping these variables to their observed values during inference. This is useful when working with real data, as it allows the user to incorporate the observed data into the analysis and improve the accuracy of the results.

In LPhy, data clamping can be achieved using the ~~"data block"~~ 'data block', for example:

In ~~the example above~~Example 2, we used a Respiratory syncytial virus subgroup A (RSVA) dataset [41,42] containing 129 molecular sequences coding for the G protein

```
data {
   options = {ageDirection="forward", ageRegex="s(\d+)"};
   nexusFilePath = "tutorials/data/RSV2.nex";
   D = readNexus(file=nexusFilePath, options=options);
   codon = D.charset(["3-629\3", "1-629\3", "2-629\3"]);
   n = 3;
   L = [209, 210, 210];
   taxa = D.taxa();
}
model {
   π ~ Dirichlet(replicates=n, conc=[2.0, 2.0, 2.0, 2.0]);
   κ ~ LogNormal(sdlog=0.5, meanlog=1.0, replicates=n);
   r ~ WeightedDirichlet(conc=rep(element=1.0, times=n), weights=L);
   μ ~ LogNormal(meanlog=-5.0, sdlog=1.25);
   Θ ~ LogNormal(meanlog=3.0, sdlog=2.0);
   ψ ~ Coalescent(taxa=taxa, theta=Θ);
   Q = hky(kappa=κ, freq=π, meanRate=r);
   codon ~ PhyloCTMC(L=L, Q=Q, mu=μ, tree=ψ);
}
```

**Example 2.** An LPhy script for phylodynamic analysis of a virus dataset containing Respiratory syncytial virus subgroup A (RSVA) genomic samples [41, 42].

collected between years 1956 and 2002. We use three partitions corresponding to the codon position, an HKY substitution model [36], coalescent tree prior [22] and a strict molecular clock with a ~~Lognormal~~ log-normal prior on the mean clock rate. Within the ~~data~~ *data* block we clamp the value of ~~"codon"~~ 'codon', a stochastic node that appears below inside the ~~model~~ *model* block. This is achieved by specifying a data node of the same name (codon) in the data block. In this example the data is vectorized into three codon positions to allow different site models for the different codon positions.

## 2.2 LPhyStudio

Along with the language definition, we introduce LPhy Studio, a ~~GUI~~ Graphical User Interface, GUI, intended for (i) model specification, (ii) PGM graphical and textual display, and (iii) ~~simulated data visualization~~visualization of simulated data. Figure 2 shows a screenshot of LPhyStudio after a simple phylogenetic model was specified. LPhyStudio's ~~additional features include the option to specify models via loading LPhy scripts (rather than building the model line-by-line), and to export PGMs and their descriptions~~ features include a scripting console, syntax highlighting, generation of PGMs and natural language text narratives of models with citations, and optionally exporting these PGM and narratives as LaTeX documents. LPhy scripts can be imported using the toolbar or file menu, or created using the scripting console.

## 2.3 LPhy and BEAST2

To facilitate the application of specified models for evolutionary inference, the companion program ~~"LPhyBEAST"~~ 'LPhyBEAST' was developed as an interface between LPhy and BEAST2. LPhyBEAST is a command-line tool that takes as input an LPhy script file specifying a model ~~and clamping~~ with simulated or observed data, and produces a BEAST2 XML file as output.

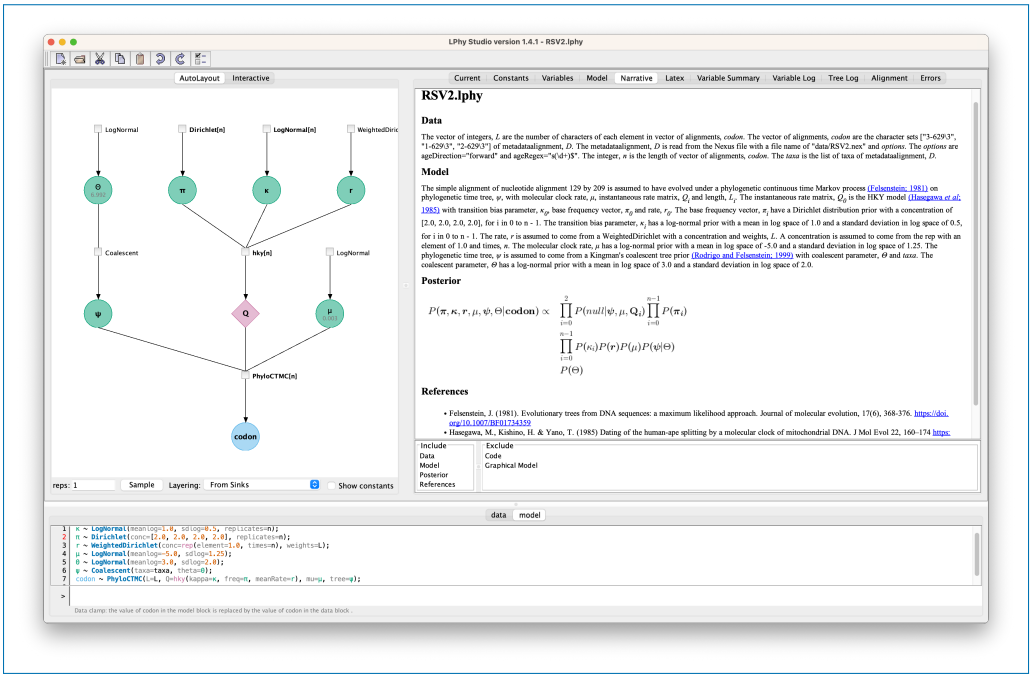


**Fig 2.** A screenshot of LPhy Studio showing the probabilistic graphical model on the left panel (constants hidden), and the auto-generated text description of the data and phylogenetic model on the right panel.

### 2.4 A community resource

LPhy, LPhySTudio and LPhyBEAST were developed to support both end-users and model developers. As such, this suite of programs is accompanied by extensive documentation available on the homepage `https://linguaphylo.github.io/`, and a growing list of tutorials ~~(available on~~ available at `https://linguaphylo.github.io/tutorials/` ~~)~~ ~~covering~~ which cover common use cases and extension mechanisms. LPhy is designed in a modular fashion~~:~~, researchers interested in implementing new models within the LPhy language can do so by releasing extension modules that can extend the LPhy application post-deployment.

## 3 Results

We present key features of the LPhy software using the GT16 substitution and error models [43]. Starting from an LPhy script, our software generates a text description of the model, a graphical representation of the model (i.e., a PGM), and multiple displays of the simulated data. Additionally, we showcase how LPhy can be used to validate the correctness of the BEAST 2 implementation of GT16 [44].

#### LPhy script

We start from an LPhy script ~~below~~ shown in Example 3 which specifies a GT16 substitution and error model [43] for simulating single-cell diploid nucleotides with sequencing error (epsilon) and allelic dropout error ~~.~~(delta). The script defines that 16 sequences 'A' are generated from a GT16 substitution model with rates and genotype frequencies drawn from dirichlet distributions, and a coalescent tree prior with a log-normally distributed theta. The observed noisy sequences 'E' are generated by

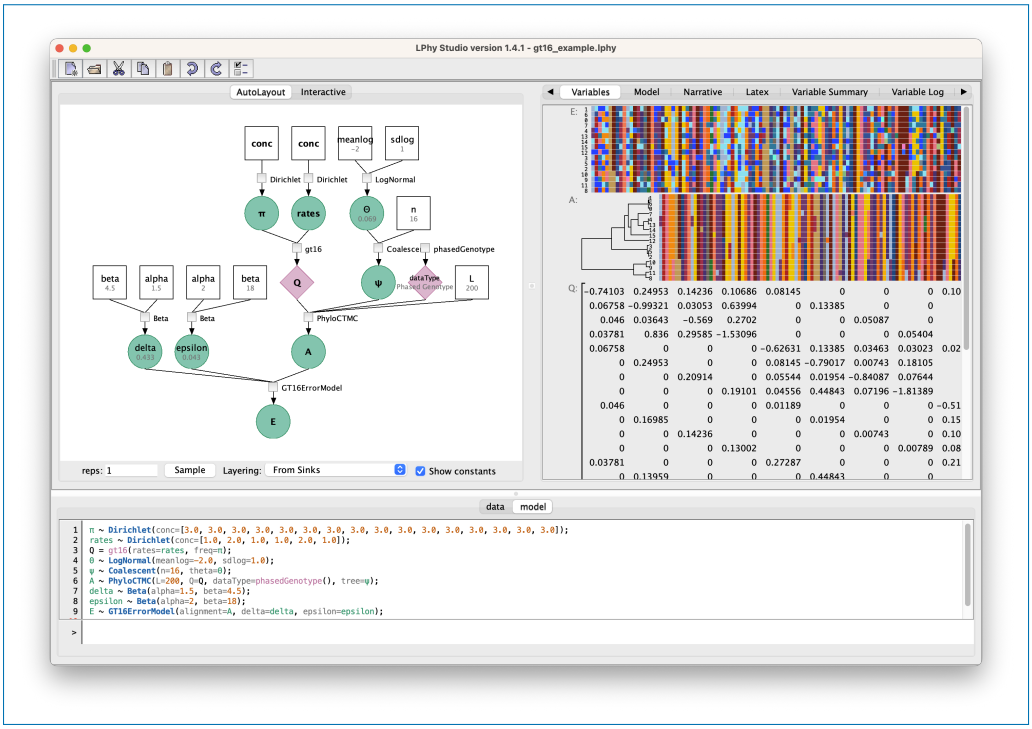


**Fig 3.** A screenshot of LPhyStudio showing the GT16 substitution and error model [43, 44]. The left panel shows the graphical model representation, the right panel shows the simulated tree and diploid nucleotide genotypes, and the bottom panel shows the LPhy script.

applying sequencing error (epsilon), and allelic dropout error (delta) which are drawn from beta distributions. In the dirichlet generator, the argument 'conc' represents the concentration parameter.

```
model {
  π  ~ Dirichlet(conc=[3.0, 3.0, 3.0, 3.0,
                       3.0, 3.0, 3.0, 3.0,
                       3.0, 3.0, 3.0, 3.0,
                       3.0, 3.0, 3.0, 3.0]);
  rates ~ Dirichlet(conc=[1.0, 2.0, 1.0, 1.0, 2.0, 1.0]);
  Q = gt16(rates=rates, freq=π);
  Θ ~ LogNormal(meanlog=-2.0, sdlog=1.0);
  ψ ~ Coalescent(n=16, theta=Θ);
  A ~ PhyloCTMC(L=200, Q=Q, dataType=phasedGenotype(), tree=ψ);
  delta ~ Beta(alpha=1.5, beta=4.5);
  epsilon ~ Beta(alpha=2, beta=18);
  E ~ GT16ErrorModel(alignment=A, delta=delta, epsilon=epsilon);
}
```

**Example 3.** An Lphy script defining a GT16 substitution and error model for diploid single-cell nucleotide data.

### Natural ~~text~~ language description

LPhyStudio ~~automatically generates~~ can automatically generate a text description of the model ~~above:~~ as a narrative. The implementation of automated natural language descriptions for phylogenetic models is not new, and similar efforts (albeit for a different scope of phylogenetic methods) can be found in SplitsTree [45] and MEGA4 [46]. The natural language narrative tool we have developed provides a precise starting point for the model description section in a research article that uses Bayesian phylogenetic inference. The LPhy script in Example 3 generates the following narrative.

> The alignment, $\boldsymbol{E}$ ~~is assumed to come from a GT16ErrorModel. The~~ has an error model [43] with sequencing and amplification error probability, epsilon, allelic dropout probability, delta and genotype alignment, $\boldsymbol{A}$. The genotype alignment, $\boldsymbol{A}$ is assumed to have evolved under a phylogenetic continuous time Markov process [47] on phylogenetic time tree, $\boldsymbol{\psi}$, with instantaneous rate matrix, $\boldsymbol{Q}$, ~~a~~ an alignment length of 200 and ~~a dataType~~the data type used for simulations. The instantaneous rate matrix, $\boldsymbol{Q}$ is the general time-reversible rate matrix on phased genotypes [43] with relative rates, **rates** and base frequencies, $\boldsymbol{\pi}$. The base frequencies, $\boldsymbol{\pi}$ have a Dirichlet distribution prior with a concentration of [3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0]. The relative rates, **rates** have a Dirichlet distribution prior with a concentration of [1.0, 2.0, 1.0, 1.0, 2.0, 1.0]. The ~~dataType~~ data type used for simulations is the phased genotype data type. The phylogenetic time tree, $\boldsymbol{\psi}$ is assumed to come from a Kingman's coalescent tree prior ~~[22]~~ [28] with coalescent parameter, $\Theta$ and an n of 16. The coalescent parameter, $\Theta$ has a log-normal prior with a mean in log space of -2.0 and a standard deviation in log space of 1.0. The allelic dropout probability, delta has a Beta distribution prior with an alpha of 1.5 and a beta of 4.5. The sequencing and amplification error probability, epsilon has a Beta distribution prior with an alpha of 2 and a beta of 18.

~~This natural text narrative can provide a precise starting point for the model description section in a research article.~~

### Model validation

The LPhy framework can be used to verify implementation correctness of new models, in which case they are said to be well-calibrated. Bayesian model validation consists of a series of steps, the first of which is simulation of synthetic data (for recent examples within the BEAST 2 platform, see [44, 48]). By making it possible to simulate under complex models, LPhy greatly simplifies the validation procedure. Figure 4 presents the validation results for the model described above, when model specification and simulation were performed using LPhy and LPhyBEAST [44].

## Discussion

Although there are many programming languages through which statistical models can be succintly described (e.g., Stan [24], JAGS [49], BUGS [26, 27]), these languages do not support the unique feature of phylogenetic models ~~:~~ the phylogenetic tree. Phylogenetic trees are complex high-dimensional objects, part discrete, part continuous. There is no bijection between tree space and Euclidean space, so these objects cannot

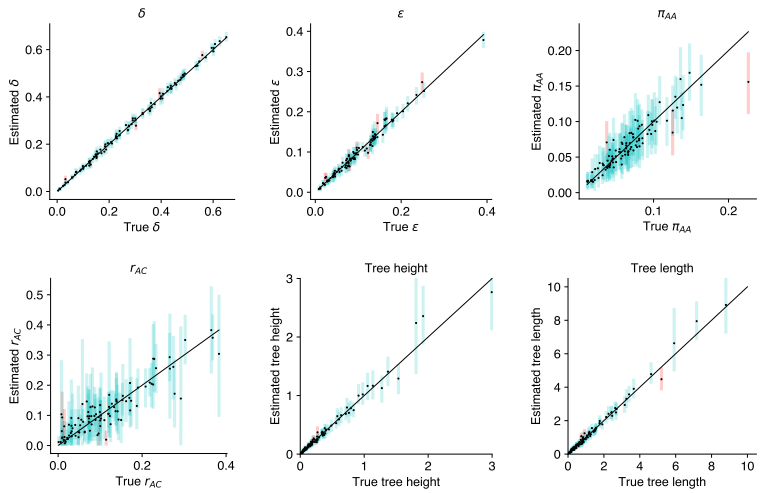


**Fig 4.** Model validation for the GT16 diploid nucleotide substitution model and GT16 error model. Each plot shows the 95% highest posterior density for model parameters: allelic dropout error $\delta$, sequencing error $\epsilon$, equilibrium frequency for $\pi_{AA}$, relative rate $r_{AC}$, tree height, and tree length.

be treated with standard statistical distributions [50]. Hence, specialist software is commonly employed to perform inference involving phylogenetic trees [51,52].

LinguaPhylo differs from existing specialist software in the way it handles model specification. By using vectorization, LinguaPhylo obviates the need for for-loop control flow to describe repetitive structural elements of a model. This feature lowers the risk of syntactic or programming logic mistakes when defining a model relative to a full programming language such as Rev [8]. In its declarative nature, LPhy's language resembles the XML specification adopted by BEAST 2 [53], but shares the central notion of probabilistic graphical models with the Rev language.

LinguaPhylo provides for a form of array programming (vectorization), so that any function or generative distribution can be called with its arguments in vectorized form. In such situations the function or generative distribution is ~~"broadcast"~~ 'broadcast' over each element of the array~~(or indeed pairs or tuples of elements across multiple parallel arrays)~~, which allows for very concise model descriptions.

Finally, future work on integration of LPhy with other popular Bayesian phylogenetic inference tools, such as RevBayes [51], BEAST [54], MrBayes [55] ~~, RevBayes [51]~~ will increase the flexibility of the framework, and enable easy validation of and comparison between different Bayesian phylogenetic inference engines.

## Acknowledgments

AJD was supported by a James Cook Fellowship from the Royal Society of New Zealand (JCF-UOA1901). FKM was supported by Marsden grant 16-UOA-277 and by National Science Foundation grant DEB-2040347. We thank the New Zealand eScience Infrastructure (NeSI) for access to high-performance computation resources.

## References

1. National Academies of Sciences, Engineering, and Medicine. Reproducibility and

replicability in science. Washington, DC: The National Academies Press; 2019.

2. Munafò MR, Nosek BA, Bishop DV, Button KS, Chambers CD, Percie du Sert N, et al. A manifesto for reproducible science. Nature human behaviour. 2017;1(1):1–9.

3. Baker M. Is there a reproducibility crisis? Nature. 2016;533:452–454.

4. Eren AM, Kiefl E, Shaiber A, Veseli I, Miller SE, Schechter MS, et al. Community-led, integrated, reproducible multi-omics with anvi'o. Nature microbiology. 2021;6(1):3–6.

5. Oakley TH, Alexandrou MA, Ngo R, Pankey MS, Churchill CK, Chen W, et al. Osiris: accessible and reproducible phylogenetic and phylogenomic analyses within the Galaxy workflow management system. BMC bioinformatics. 2014;15(1):1–9.

6. Drummond AJ, Rambaut A. BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evolutionary Biology. 2007;7:214.

7. Bouckaert R, Vaughan TG, Barido-Sottani J, Duchêne S, Fourment M, Gavryushkina A, et al. BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. PLoS computational biology. 2019;15:e1006650.

8. Höhna S, Landis MJ, Heath TA, Boussau B, Lartillot N, Moore BR, et al. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification languages. Systematic biology. 2016;65:726–736.

9. Ronquist F, Teslenko M, van der Mark P, Ayres DL, Darling A, Höhna S, et al. MrBayes 3.2: efficient Bayesian phylogenetic inference and model selection across a large model space. Systematic biology. 2012;61:539–542.

10. Gavryushkina A, Heath TA, Ksepka DT, Stadler T, Welch D, Drummond AJ. Bayesian total-evidence dating reveals the recent crown radiation of penguins. Systematic biology. 2017;66.

11. Ogilvie HA, Mendes FK, Matzke NJ, Stadler T, Welch D, Drummond AJ. Novel integrative modeling of molecules and morphology across evolutionary timescales. Systematic Biology. 2022;71:208–220.

12. Zhang R, Drummond AJ, Mendes FK. Scalable Bayesian inference of phylogenies from molecular and continuous traits in a probabilistic total-evidence framework. bioRxiv. 2021;.

13. Lemey P, Rambaut A, Welch JJ, Suchard MA. Phylogeography takes a relaxed random walk in continuous space and time. Molecular biology and evolution. 2010;27:1877–1885.

14. Landis MJ, Freyman WA, Baldwin BG. Retracing the Hawaiian silversword radiation despite phylogenetic, biogeographic, and paleogeographic uncertainty. Evolution. 2018;72:2343–2359.

15. Faria NR, et al. Genomics and epidemiology of the P.1 SARS-CoV-2 lineage in Manaus, Brazil. Science. 2021;372:815–821.

16. Douglas J, Mendes FK, Bouckaert R, Xie D, Jiménez-Silva CL, Swanepoel C, et al. Phylodynamics reveals the role of human travel and contact tracing in controlling the first wave of COVID-19 in four island nations. Virus evolution. 2021;7(2):veab052.

17. Tavaré S, et al. Some probabilistic and statistical problems in the analysis of DNA sequences. Lectures on mathematics in the life sciences. 1986;17(2):57–86.

18. ISO Central Secretary. Information technology – Syntactic metalanguage – Extended BNF (Standard ISO/IEC 14977:1996); 1996. Available from: https://www.iso.org/standard/26153.html.

19. Parr TJ, Quong RW. ANTLR: A predicated-LL (k) parser generator. Software: Practice and Experience. 1995;25(7):789–810.

20. Parr T. The definitive ANTLR 4 reference. The Definitive ANTLR 4 Reference. 2013; p. 1–326.

21. Jukes TH, Cantor CR, et al. Evolution of protein molecules. Mammalian protein metabolism. 1969;3:21–132.

22. Kingman JFC. The coalescent. Stochastic processes and their applications. 1982;13(3):235–248.

23. Ihaka R, Gentleman R. R: a language for data analysis and graphics. Journal of computational and graphical statistics. 1996;5(3):299–314.

24. Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, et al. Stan: A probabilistic programming language. Journal of statistical software. 2017;76(1).

25. Plummer M. JAGS: Just another Gibbs sampler. 2004;.

26. Lunn D, Spiegelhalter D, Thomas A, Best N. The BUGS project: Evolution, critique and future directions. Statistics in medicine. 2009;28(25):3049–3067.

27. Gilks WR, Thomas A, Spiegelhalter DJ. A language and program for complex Bayesian modelling. Journal of the Royal Statistical Society: Series D (The Statistician). 1994;43(1):169–177.

28. Rodrigo AG, Felsenstein J. Coalescent Approaches to HIV Population Genetics. In: K C, editor. The Evolution of HIV. Baltimore: Johns Hopkins Univ. Press; 1999.

29. Hudson R. Oxford Surveys in Evolutionary Biology 7, chapter Gene genealogies and the coalescent process. Oxford. 1990;.

30. Notohara M. The coalescent and the genealogical process in geographically structured population. Journal of mathematical biology. 1990;29:59–75.

31. Drummond AJ, Rambaut A, Shapiro B, Pybus OG. Bayesian coalescent inference of past population dynamics from molecular sequences. Molecular biology and evolution. 2005;22(5):1185–1192.

32. Stadler T, Yang Z. Dating phylogenies with sequentially sampled tips. Syst Biol. 2013;62(5):674–88. doi:10.1093/sysbio/syt030.

33. Kendall DG. On the generalized" birth-and-death" process. The annals of mathematical statistics. 1948;19(1):1–15.

34. Heath TA, Huelsenbeck JP, Stadler T. The fossilized birth-death process for coherent calibration of divergence-time estimates. Proceedings of the National Academy of Sciences. 2014;111(29):E2957–E2966.

35. Yule GU. II.—A mathematical theory of evolution, based on the conclusions of Dr. JC Willis, FR S. Philosophical transactions of the Royal Society of London Series B, containing papers of a biological character. 1925;213(402-410):21–87.

36. Hasegawa M, Kishino H, Yano Ta. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of molecular evolution. 1985;22(2):160–174.

37. Zuckerkandl E, Pauling L. Evolutionary divergence and convergence in proteins. In: Evolving genes and proteins. Elsevier; 1965. p. 97–166.

38. Zuckerkandl E, Pauling L. Molecules as documents of evolutionary history. Journal of theoretical biology. 1965;8(2):357–366.

39. Drummond AJ, Suchard MA. Bayesian random local clocks, or one rate to rule them all. BMC biology. 2010;8(1):1–12.

40. Drummond A, Ho S, Phillips M, Rambaut A. Relaxed phylogenetics and dating with confidence. PLOS Biology. 2006;4(e88):699–710. doi:10.1371/journal.pbio.0040088.

41. Zlateva KT, Lemey P, Vandamme AM, Van Ranst M. Molecular evolution and circulation patterns of human respiratory syncytial virus subgroup A: positively selected sites in the attachment G glycoprotein. Journal of virology. 2004;78(9):4675–4683.

42. Zlateva KT, Lemey P, Moës E, Vandamme AM, Van Ranst M. Genetic variability and molecular evolution of the human respiratory syncytial virus subgroup B attachment G protein. Journal of virology. 2005;79(14):9157–9167.

43. Kozlov A, Alves JM, Stamatakis A, Posada D. CellPhy: accurate and fast probabilistic inference of single-cell phylogenies from scDNA-seq data. Genome biology. 2022;23(1):1–30.

44. Chen K, Moravec JC, Gavryushkin A, Welch D, Drummond AJ. Accounting for errors in data improves divergence time estimates in single-cell cancer evolution. Molecular biology and evolution. 2022;in press. doi:10.1093/molbev/msac143.

45. Huson DH, Bryant D. Application of phylogenetic networks in evolutionary studies. Molecular biology and evolution. 2006;23(2):254–267.

46. Tamura K, Dudley J, Nei M, Kumar S. MEGA4: molecular evolutionary genetics analysis (MEGA) software version 4.0. Molecular biology and evolution. 2007;24(8):1596–1599.

47. Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol. 1981;17(6):368–76. doi:10.1007/BF01734359.

48. Gaboriau T, Mendes FK, Joly S, Silvestro D, Salamin N. A multi-platform package for the analysis of intra- and interspecific trait evolution. Methods in Ecology and Evolution. 2020;11:1439–1447.

49. Plummer M, et al. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In: Proceedings of the 3rd international workshop on distributed statistical computing. vol. 124. Vienna, Austria; 2003. p. 1–10.

50. Gavryushkin A, Drummond AJ. The space of ultrametric phylogenetic trees. Journal of theoretical biology. 2016;403:197–208.

51. Höhna S, Landis MJ, Heath TA, Boussau B, Lartillot N, Moore BR, et al. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. Systematic biology. 2016;65(4):726–736.

52. Bouckaert R, Vaughan TG, Barido-Sottani J, Duchêne S, Fourment M, Gavryushkina A, et al. BEAST 2.5: an advanced software platform for Bayesian evolutionary analysis. PLoS computational biology. 2019;15(4):e1006650. doi:10.1371/journal.pcbi.1006650.

53. Bouckaert R, Heled J, Kühnert D, Vaughan T, Wu CH, Xie D, et al. BEAST 2: a software platform for Bayesian evolutionary analysis. PLoS computational biology. 2014;10(4):e1003537.

54. Suchard MA, Lemey P, Baele G, Ayres DL, Drummond AJ, Rambaut A. Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10. Virus Evol. 2018;4(1):vey016. doi:10.1093/ve/vey016.

55. Ronquist F, Teslenko M, Van Der Mark P, Ayres DL, Darling A, Höhna S, et al. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. Systematic biology. 2012;61(3):539–542.