BEAST2 Analysis of Algonquian Language Data

Benjamin Teo

June 15, 2020

1 Problem setup

1.1 Dataset

The dataset comprises cognate data from 52 words across 9 languages. It is our understanding that for a given word, (1) each language either has or does not have a form for that word, and (2) the languages that have forms for that word can be partitioned into cognate classes. As stated in *Lexical Relationships in Central Algonquian*,

> The 52 words were compared in the seven languages plus our two controls, but not every language had a form for all 52 words. Reconstructed forms are available for these items ... When there was doubt as to the cognacy of forms, we were conservative and treated them as non-cognate. (p. 10)

The dataset is represented by a 52-by-9 matrix. Each row corresponds to a different word, and uses non-negative integers to indicate the cognate classes for that word. We interpret the occasional X's to indicate cases where cognacy was undeterminable. Rows 14-15 from the dataset are shown in Table 1.

Pl Cree	SW Oj	Chey	Menom	Pot.	Meskw.	Shaw	Miami	Pass-M
0	0	2	1	0	0	0	0	0
2	0	X	0	0	3	0	1	4

Table 1: Rows 14-15 from the dataset. Row 14 indicates 3 cognate classes, while row 15 indicates 5 cognate classes and one language (Chey) for which cognacy was undeterminable.

1.2 Reformatting the data

For the purposes of phylogenetic analysis, we reformatted the original dataset into a 9-by-152 matrix of mostly 0's and 1's, with the exception of some ?'s. Each row corresponds to a different language, and each column corresponds to a cognate class that was identified for

some word. A 0 indicates cognate absence, a 1 cognate presence, and a ? indicates a case of undeterminable cognacy. The reformatted information from rows 14-15 of the original dataset is presented in Table 2.

Pl Cree	1	0	0	0	0	1	0	0
SW Oj	1	0	0	1	0	0	0	0
Chey	0	0	1	?	?	?	?	?
Menom	0	1	0	1	0	0	0	0
Pot.	1	0	0	1	0	0	0	0
Meskw.	1	0	0	0	0	0	1	0
Shaw	1	0	0	1	0	0	0	0
Miami	1	0	0	0	1	0	0	0
Pass-M	1	0	0	0	0	0	0	1

Table 2: Reformatted information from rows 14-15 of the original dataset. The light-gray columns correspond to row 14, and the gray columns correspond to row 15.

1.3 Objectives

1. Compare how well 6 different models (Pick 1 of 3 site-state models: Binary CTMC, Binary Covarion, Stochastic Dollo. Pick 1 of 2 clock models: Strict, Log-normal relaxed.) of cognate evolution explain the dataset.

2. Infer the phylogeny of these 9 languages, assuming this is a tree.

2 Data-generating process

To gain a better intuition for the models of cognate evolution mentioned in 1.3, it is useful to consider how we might use these models to simulate cognate data for a set of genetically-related, extant languages. This involves 2 main steps, the first and second of which are affected by our choices of clock model and site-state model respectively:

1. Simulate a tree (topology and branch lengths) that describes the evolutionary relationships of the set of languages. We refer to this as the "tree-generating process". We make 2 assumptions at this point:

   - No extant language is directly descended from another extant language.
   - Each birth event produces 2 offspring (i.e. each internal node has 2 child nodes).

   These assumptions imply that the tree is a full binary tree with the leaf nodes that represent the extant languages.

2. Apply a chosen site-state model on the tree to sequentially generate cognate states from the root, through the internal nodes, and to the leaves. We refer to this as the "site-state-generating process".

- Leaf nodes correspond to extant languages, while internal nodes correspond to extinct ancestral languages.
- In the parlance of molecular phylogenetics, cognates correspond to "sites" (hence "site-state model" and "site-state-generating process"). As illustrated in 1.2, a cognate can either be present (1) or absent (0) in a language. Hence, every language in the tree is associated with a binary vector indicating which cognates are present or absent in that language.

2.1 Tree-generating process

1. Generate a full binary tree (every node has either 0 or 2 children) with N leaves/taxa using a Yule process with birth rate, λ_y.

   i Start from the trivial binary tree, a single root node, which is also a leaf node.

   ii Select the leaf node in the tree that is nearest (in terms of branch length traversed) to the root. Give the selected node 2 child nodes, with the corresponding edge lengths independently sampled from the $\text{Exp}(\lambda_y)$ distribution. The number of leaf nodes in the tree has now increased by 1.

   iii Repeat ii until the tree has N leaf nodes, then truncate the pendant edge lengths as needed so that all leaf nodes are equidistant (in terms of branch length traversed) from the root.

   iv Denote the resultant set of branch lengths by $\{l_1, ..., l_{2(N-1)}\}$.

   By iii we assume that data for the extant languages are simultaneously observed at a time point just before the Nth birth event. For reference, the 1st birth event occurs at the root.

2. Generate clock rates, $\{c_1, ..., c_{2(n-1)}\}$ for each branch.

   - Under the **strict clock model**, $c_i = 1$ for all i.
   - Under the **log-normal relaxed clock model**, $c_i \overset{\text{iid}}{\sim} \text{Lognormal}(-\sigma^2_{\text{clock}}/2, \sigma^2_{\text{clock}})$ for all i. Note that this lognormal distribution is constrained to have a mean of 1.
   - Scale the edge lengths of the tree generated in the previous step, by replacing l_i with $l^*_i = l_i \cdot c_i$ for all i.

   The tree generated under the strict clock model is ultrametric (i.e. all leaves are equidistant from the root). This is not necessarily the case under the relaxed clock model.

2.2 Site-state-generating process

The mechanics of the Stochastic Dollo model are distinct enough from those of the Binary CTMC/Covarion models (which are quite similar in nature), to justify separate descriptions for Binary CTMC/Covarion vs Stochastic Dollo.

2.2.1 Binary CTMC/Covarion

1. Initialize the transition rate matrix, Q.

   - Binary CTMC:
     - This is equivalent to a General Time Reversible (GTR) model for 2 states. Let f_0 denote the equilibrium proportion of the 0/"absent" state.
     - The equilibrium proportions, π and normalized (i.e. expected mutation rate is 1) rate matrix, Q are respectively parametrized as follows:

$$\pi = [f_0, (1 - f_0))]$$

$$Q = \frac{1}{2f_0(1 - f_0)} \begin{bmatrix} -(1 - f_0) & (1 - f_0) \\ f_0 & -f_0 \end{bmatrix}$$

     The model has 1 free parameter, f_0.

   - Binary Covarion:
     - This is an extension of the Binary CTMC model, where we further characterize site-states by whether or not they were observed while the stochastic process was evolving in its slow mode or fast mode.
     - Let f_0 denote the equilibrium proportion of the "absent" state, s denote the switch rate between the slow and fast modes, and α_{slow} denote factor by which "absent"-"present" transition rates are scaled in the slow mode as compared to their fast mode counterparts. For the sake of preserving the time-reversibility property, we assume that the equilibrium proportions of being in the slow and fast modes are 1/2.
     - The equilibrium frequencies, π and normalized rate matrix, Q are respectively parametrized as follows:

$$\pi = [f_0/2, (1 - f_0)/2, f_0/2, (1 - f_0)/2]$$

$$Q = \frac{1}{\eta} \begin{bmatrix} -(\alpha_{\text{slow}}(1 - f_0) + s) & \alpha_{\text{slow}}(1 - f_0) & s & 0 \\ \alpha_{\text{slow}} f_0 & -(\alpha_{\text{slow}} f_0 + s) & 0 & s \\ s & 0 & -(1 - f_0 + s) & (1 - f_0) \\ 0 & s & f_0 & -(s + f_0) \end{bmatrix}$$

     where $\eta = f_0(1 - f_0)(1 + \alpha_{\text{slow}}) + s$. The model has 3 free parameters, f_0, s, and α_{slow}.

2. Carry out the following steps in order:

   i For each site at the root node, independently sample a state from π.

   ii Let S denote the number of sites, and $\{r_1, ..., r_S\}$ denote the set of site-specific rates. Initialize $\{r_1, ..., r_S\}$ by independently sampling S values from the Discretized (4 categories) version of Gamma$(\alpha_{\text{site}}, 1/\alpha_{\text{site}})$, where $1/\alpha_{\text{site}}$ is the scale parameter (implying a fixed mean of 1).

iii Let l^* denote the length of the edge between a child node and its parent. For site i, calculate the transition probability matrix $P = e^{Q(l^* r_i)}$. Find the row in P that corresponds to the state of site i in the parent. Use the probability mass function described by that row to sample the state of site i in the child.

iv Repeat iii down the levels of the tree to sample site-states for all the nodes.

2.2.2 Stochastic Dollo

All cognates are initially absent at the root. Each cognate becomes present at some point (this does not need to be at a node) along the tree. The total branch length traveled from the root to that point is exponentially distributed with rate $r_i\nu$, where r_i depends on the cognate. The birth events for different cognates are mutually independent. After a cognate becomes present on the tree, it undergoes a branching death process with rate $r_i\mu$. After its birth event, a cognate may die off at multiple points along the tree. The model has 2 free parameters ν and μ.

1. Let S denote the number of cognates/sites, and $\{r_1, ..., r_S\}$ denote the set of cognate/site-specific rates. Initialize $\{r_1, ..., r_S\}$ by independently sampling S values from the Discretized (4 categories) version of Gamma$(\alpha_{\text{site}}, 1/\alpha_{\text{site}})$, where $1/\alpha_{\text{site}}$ is the scale parameter.

2. For each cognate, sample its birth event as follows:

   i Start from the root node. For each out-edge, independently sample a value from the Exp$(r_i\nu)$ distribution. Denote these values as d_1, d_2 and the lengths of the corresponding out-edges as e_1, e_2.

   ii If $d_i \leq e_i$, mark a point p_i length d_i along that out-edge. If $d_i > e_i$ and the corresponding child node is **not further** (in terms of branch length traversed) from the root than any other point that has been marked, repeat i and ii but now starting from that child node.

   iii Of all the points marked on the tree, select the one that is nearest (in terms of branch length traversed) to the root. This point marks the birth event of the cognate on the tree.

   It is possible but unlikely that i and ii are repeated without producing a birth event on the tree.

3. For each cognate born on the tree, sample its death event as follows:

   i Start from its birth point, p_* on the tree. For each out-edge (we extend the scope of this term to include non-node points on the tree), independently sample a value from the Exp$(r_i\mu)$ distribution. Denote the value(s) as $d_1(, d_2)$ and the corresponding branch length(s) from p_* to the nearest child node(s) as $b_1(, b_2)$.

   ii If $d_i \leq b_i$, mark a point q_i length d_i from p_* along that out-edge. If $d_i > b_i$, repeat i and ii but now starting from the corresponding child node. The q_i's mark the death events of the cognate on the tree.

Within a clade, if a cognate was present at the root, it is possible for this cognate to be present in some leaves and absent in others.

3 The Bayesian approach

Recall the 2 objectives we listed in 1.3. We want to (1) compare models of cognate evolution, and (2) estimate the phylogeny of the languages of interest, assuming that this is a tree. The Bayesian paradigm offers methods for achieving both objectives.

3.1 General principle

In section 2, we implicitly characterized a phylogenetic model by 3 components: (1) the Yule model for generating the tree topology, (2) a clock model for adjusting the branch lengths, and (3) a site-state model for generating site-states throughout the tree. For a given set of parameter (see table 3) values, we have a procedure for generating a phylogenetic tree and site-state values for the extant languages. Conversely, **for a given tree and set of parameter values**, we can calculate the likelihood of observing particular site-state values for the extant languages.

Parameter	CTMC		Covarion		S. Dollo	
	strict	relaxed	strict	relaxed	strict	relaxed
Yule birth rate, λ_y	✓	✓	✓	✓	✓	✓
Scale parameter (relaxed clock), σ_{clock}		✓		✓		✓
Scale parameter (site-heterogeneity), α_{site}	✓	✓	✓	✓	✓	✓
Equilibrium proportion (absent state), f_0	✓	✓	✓	✓		
Switch rate (between slow/fast modes), s			✓	✓		
Scale parameter (slow mode), α_{slow}			✓	✓		
Cognate birth rate, ν					✓	✓
Cognate death rate, μ					✓	✓

Table 3: Model parameters

If further, we were to treat **the tree and the parameters of this phylogenetic model** as random elements and provide accompanying prior distributions, then we could carry out posterior estimation for these random elements in the Bayesian fashion. In contrast to various likelihood criteria that prioritize goodness-of-fit (i.e. what tree and parameter values maximize the probability of observing the data conditional on the model structure), the Bayesian approach (e.g. Highest Posterior Density (HPD) intervals) seeks to reflect how much the data support certain parameter values vis-à-vis other possible values. The tree, model parameters, and data (i.e. via sufficient statistics) are thought of as inhabiting a joint probability space in some dependent configuration that is determined by the form/structure of the likelihood (i.e. $P(\text{Data} \mid \text{Parameters, Tree, Model})$).

Bayes factors and maximum clade credibility (MCC) are Bayesian methods for performing model comparison and tree estimation respectively. Both involve calculating the marginal likelihood of the data, conditional only on the model structure (i.e. $P(\text{Data} \mid \text{Model})$). In general, this quantity is analytically intractable and computationally infeasible. Fortunately, various sampling schemes have been developed to either avoid the computation of this quantity or to estimate it in a computationally feasible manner. Bayes factors and MCC trees can be stochastically estimated through Stepping-stone sampling and MCMC sampling respectively.

4 MCMC sampling

4.1 General principle

MCMC sampling schemes generate a long sequence of parameter value combinations (a single parameter value combination assigns values to all the parameters characterizing the phylogenetic model) with the following two properties:

1. The empirical distribution of the sample approximates the desired posterior distribution.

2. Any two parameter value combinations along the sequence that are sufficiently far apart are essentially independent.

Thus, subsampling from the original sequence at a low enough frequency (so that subsampled values are far apart in the original sequence) approximates independent draws from the desired posterior distribution.

Such sampling schemes (to produce the original long sequence) generally proceed as follows:

1. Randomly initialize the first element (i.e. parameter value combination) in the sequence.

2. To generate the next element in the sequence, take the current element and propose some random change to it (e.g. take one/some of the parameter values and change them), then make a randomized decision (which depends on the "proposal" distribution) on whether or not to set either the current element or its modification as the next element.

3. Iteratively carry out 2. until a sequence of desired length is attained.

5 Stepping-stone sampling

5.1 General principle

Stepping-stone sampling uses a combination of importance sampling and MCMC sampling to compute the marginal likelihood. Strictly speaking, it should be viewed as a specific

application of MCMC sampling.

The marginal likelihood is given by the following integral:

$$\int_{\Theta} f(D \mid \theta, M)\pi(\theta \mid M)d\theta$$

The most straightforward way to estimate the above integral through importance sampling would be to view it as an Expectation with respect to the prior distribution (i.e. the prior distribution is taken to be the importance distribution), and to stochastically estimate that expectation as a sample mean. Under certain regularity conditions, the consistency of this estimator is guaranteed by the Law of Large Numbers.

$$\begin{aligned}\int_{\Theta} f(D \mid \theta, M) \cdot \pi(\theta \mid M)d\theta &= E_{\pi(\theta|M)}[f(D \mid \theta, M)] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} f(D \mid \theta_i, M),\ \theta_i \overset{\text{iid}}{\sim} \pi(\theta \mid M)\end{aligned}$$

The main disadvantage of this simple method is that the variance of the estimator can be quite large depending on the relative locations and shapes of the likelihood and prior.

Stepping-stone sampling aims to reduce the variance of estimating the marginal likelihood by viewing it as a product of several integrals, each of which can be stochastically estimated by importance sampling. In this case however, the importance distribution for each of these integrals is a different power-posterior distribution. Let $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$, then we have:

$$\begin{aligned}\int_{\Theta} f(D \mid \theta, M)\pi(\theta \mid M)d\theta &= \prod_{k=1}^{K} \frac{\int_{\Theta} f(D \mid \theta, M)^{\beta_k}\pi(\theta \mid M)d\theta}{\int_{\Theta} f(D \mid \theta, M)^{\beta_{k-1}}\pi(\theta \mid M)d\theta}, \\ &= \prod_{k=1}^{K} \int_{\Theta} f(D \mid \theta, M)^{\beta_k-\beta_{k-1}} \cdot \left(\frac{f(D \mid \theta, M)^{\beta_{k-1}}\pi(\theta \mid M)}{\int_{\Theta} f(D \mid \theta, M)^{\beta_{k-1}}\pi(\theta \mid M)d\theta}\right) d\theta \\ &= \prod_{k=1}^{K} \int_{\Theta} f(D \mid \theta, M)^{\beta_k-\beta_{k-1}} \cdot p_{\beta_{k-1}}(\theta \mid D, M)d\theta \\ &= \prod_{k=1}^{K} E_{p_{\beta_{k-1}}(\theta|D,M)}[f(D \mid \theta, M)^{\beta_k-\beta_{k-1}}] \\ &\approx \prod_{k=1}^{K} \frac{1}{n_k}\sum_{i=1}^{n_k} f(D \mid \theta_{(i,k-1)}, M)^{\beta_k-\beta_{k-1}},\ \theta_{(i,k)} \overset{\text{iid}}{\sim} p_{\beta_k},\ i \in \{1, ..., n_k\}\end{aligned}$$

MCMC sampling then comes in as the method used to simulate parameter value combinations ($\theta_{(i,k)}$'s) from the various power-posteriors (p_{β_k}'s). K is referred to as the number of "stones" (hence the name of the sampling scheme).

TO-DO: Explain how this technique reduces variance when estimating the marginal likelihood.

6 BEAST2 analysis

The BEAST2 (Bayesian Evolutionary Analysis By Sampling Trees) software was used to simulate draws from the posterior distribution (i.e. P(Parameters, Tree | Data, Model)) for each of 6 different models (Pick 1 of 3 site-state models: Binary CTMC, Binary Covarion, Stochastic Dollo. Pick 1 of 2 clock models: Strict, Relaxed) via MCMC sampling. **For the sake of brevity, we will consider the tree as a parameter for the rest of this section.** Each run for a given model generated a sequence of 100,000,000 parameter value combinations (this includes trees!). The first 10% (10,000,000 observations) of this sequence was discarded as burn-in, and the remaining 90% was subsampled at a rate of 1 per 10,000 to produce a subsequence of length 9,000 (i.e. we end up with 9000 sets of parameters, trees included!).

The Tracer software was then used to analyze the set of runs for each model. Tracer diagnostics (see 7.1) suggested that the tree topology supported by a given model ought to be reliably estimated from a corresponding MCMC run with chain length (i.e. initial sequence length before burn-in removal and subsampling) 100,000,000. Likewise, Tracer diagnostics suggested that the marginal likelihood for a given model ought to be reliably estimated by Stepping-stone sampling with 100 stones and each stone corresponding to an MCMC run with chain length 100,000,000/(number of stones).

TO-DO: Justify the reliability of the Stepping-stone estimate for marginal likelihood by simulation.

6.1 BEAUti settings

For each model, the BEAUti program (packaged with BEAST2) was used to create an XML file containing specifications for BEAST2 to perform a single run. The settings used are listed below.

Note that if a parameter is assigned a prior, then it will be estimated (i.e. it will be part of the joint posterior that BEAST2 targets through MCMC sampling). Otherwise, it is held constant after initialization.

6.1.1 Tree parameters

- The Yule birth rate, λ_y is initialized to be 1. It is assigned the improper prior Unif$(0, \infty)$ **only for** the Binary CTMC/Covarion Strict Clock models. For all other models, it is held constant.

- The scale parameter for the relaxed clock, σ_{clock} is assigned the prior Gamma($\alpha_{\sigma_{\text{clock}}} = 0.5396, \beta_{\sigma_{\text{clock}}} = 0.3819$) and initialized to be 0.1. $\beta_{\sigma_{\text{clock}}}$ is the scale parameter.

6.1.2 Site-state parameters

- Binary CTMC

  - The equilibrium proportion for the absent state, f_0 is assigned the prior Unif$(0, 1)$ and initialized to be 0.5.

- Binary Covarion
  - The equilibrium proportion for the absent state, f_0 is assigned the prior Unif$(0, 1)$ and initialized to be 0.5.
  - The switch rate between the slow/fast modes, s is assigned the prior Gamma$(\alpha_s = 0.05, \beta_s = 10.0)$ and initialized to be 0.5. β_s is the scale parameter.
  - The scale parameter for the slow mode, α_{slow} is assigned the improper prior Unif$(0, \infty)$ and initialized to be 0.5.

- Stochastic Dollo
  - The cognate birth rate, ν is initialized to be 1.0. It is not estimated!
  - The cognate death rate, μ is assigned an Exponential prior with mean 10^{-4}, and is initialized to be 10^{-4}.

- Site-heterogeneity
  - The scale parameter for site-heterogeneity, α_{site} is assigned an Exponential prior with mean 1, and is initialized to be 1.

6.2 TreeAnnotator settings

An MCMC run initialized according to the BEAUti settings in 6.1 will produce a tree log file as one of its outputs. This log file will contain the information (topology and branch lengths) of a subsample of trees from the MCMC run. The TreeAnnotator program (packaged together with BEAST2) takes this log file as input and outputs a file describing the MCC tree constructed from the subsampled trees. The relevant TreeAnnotator settings are listed below:

- `-heights mean`
- `-burnin 10`

This process can be executed from the command-line through the following command (assuming that the shell knows where to look for the `treeannotator` executable):

- `treeannotator -heights mean -burnin 10 <input filepath> <output filepath>`

6.3 Stepping-stone modifications

To set up Stepping-stone sampling on BEAST2, the XML file described in 6.1 has to be directly modified. Essentially, a new element with the tag `run` will have to be defined. The key attributes for `run` and their corresponding values are listed below:

- `chainLength="1000000"`

- `alpha="0.3"`
- `burnInPercentage="50"`
- `preBurnin="100000"`
- `nrOfSteps="100"`

We briefly outline how to perform the required modifications in the steps provided below:

1. Rename the `run` element

   ```
   <run ...> ... </run>
   ```

   to `mcmc`

   ```
   <mcmc ...> ... </mcmc>
   ```

   Only change the tag. Do not modify any of the attribute name-value pairs!

2. Create a new `run` element as the direct parent of the `mcmc` element

   ```
   <run ...> <mcmc ...> ... </mcmc> </run>
   ```

3. Between `<run>` and `<mcmc>`, insert the following:

   ```
   cd $(dir)
   java -cp $(java.class.path) beast.app.beastapp.BeastMain
   ↪  $(resume/overwrite) -java -seed $(seed) beast.xml
   ```

4. Initialize the following attribute name-value pairs in the `run` element

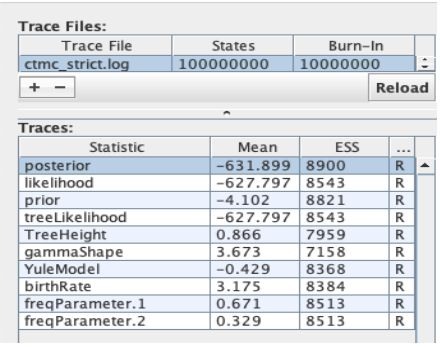
   ```
   <run spec="beast.inference.PathSampler" chainLength="1000000"
   ↪  alpha="0.3" rootdir="output_directory_path"
   ↪  burnInPercentage="50" preBurnin="100000" deleteOldLogs="true"
   ↪  nrOfSteps="100" doNotRun="true">
   ```

   Note that you will have to set the `rootdir` attribute equal to whichever directory you want the stepping-stone results to be saved to. If the `doNotRun` attribute is set to `"true"` (as shown above) the script to run the analysis will be saved to the directory specified by `rootdir`, and you will have to manually execute that script. If it is set to `"false"`, then that script will be automatically executed after being created.

To run Stepping-stone sampling, simply run BEAST2 with the modified XML file as input.

7 Results

7.1 Tracer statistics



Trace Files:

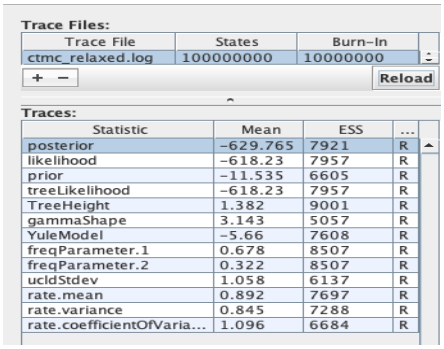
Trace File	States	Burn-In
ctmc_strict.log	100000000	10000000

Reload

Traces:

Statistic	Mean	ESS	...
posterior	−631.899	8900	R
likelihood	−627.797	8543	R
prior	−4.102	8821	R
treeLikelihood	−627.797	8543	R
TreeHeight	0.866	7959	R
gammaShape	3.673	7158	R
YuleModel	−0.429	8368	R
birthRate	3.175	8384	R
freqParameter.1	0.671	8513	R
freqParameter.2	0.329	8513	R

(a) Strict clock



Trace Files:

Trace File	States	Burn-In
ctmc_relaxed.log	100000000	10000000

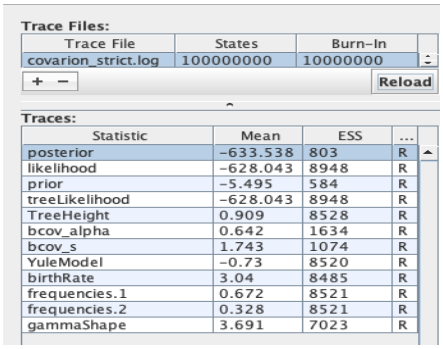
Reload

Traces:

Statistic	Mean	ESS	...
posterior	−629.765	7921	R
likelihood	−618.23	7957	R
prior	−11.535	6605	R
treeLikelihood	−618.23	7957	R
TreeHeight	1.382	9001	R
gammaShape	3.143	5057	R
YuleModel	−5.66	7608	R
freqParameter.1	0.678	8507	R
freqParameter.2	0.322	8507	R
ucldStdev	1.058	6137	R
rate.mean	0.892	7697	R
rate.variance	0.845	7288	R
rate.coefficientOfVaria...	1.096	6684	R

(b) Relaxed clock

Figure 1: CTMC model



Trace Files:

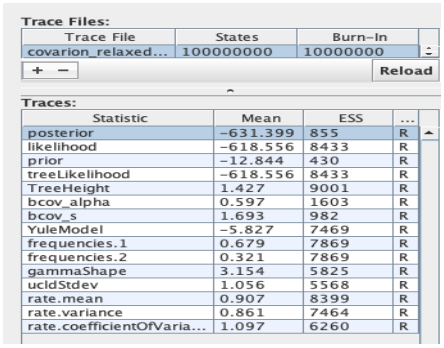
Trace File	States	Burn-In
covarion_strict.log	100000000	10000000

Reload

Traces:

Statistic	Mean	ESS	...
posterior	−633.538	803	R
likelihood	−628.043	8948	R
prior	−5.495	584	R
treeLikelihood	−628.043	8948	R
TreeHeight	0.909	8528	R
bcov_alpha	0.642	1634	R
bcov_s	1.743	1074	R
YuleModel	−0.73	8520	R
birthRate	3.04	8485	R
frequencies.1	0.672	8521	R
frequencies.2	0.328	8521	R
gammaShape	3.691	7023	R

(a) Strict clock



Trace Files:

Trace File	States	Burn-In
covarion_relaxed...	100000000	10000000

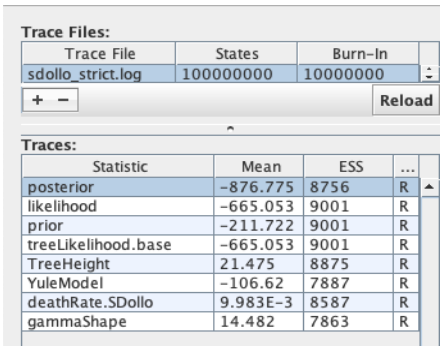
Reload

Traces:

Statistic	Mean	ESS	...
posterior	−631.399	855	R
likelihood	−618.556	8433	R
prior	−12.844	430	R
treeLikelihood	−618.556	8433	R
TreeHeight	1.427	9001	R
bcov_alpha	0.597	1603	R
bcov_s	1.693	982	R
YuleModel	−5.827	7469	R
frequencies.1	0.679	7869	R
frequencies.2	0.321	7869	R
gammaShape	3.154	5825	R
ucldStdev	1.056	5568	R
rate.mean	0.907	8399	R
rate.variance	0.861	7464	R
rate.coefficientOfVaria...	1.097	6260	R

(b) Relaxed clock

Figure 2: Covarion model



Trace Files:

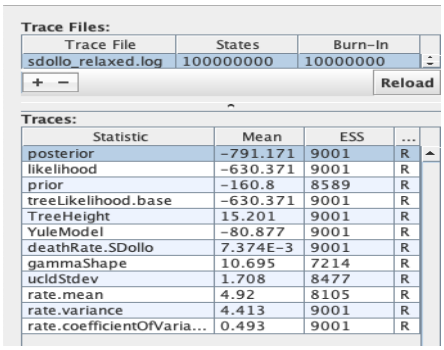
Trace File	States	Burn-In
sdollo_strict.log	100000000	10000000

Reload

Traces:

Statistic	Mean	ESS	...
posterior	−876.775	8756	R
likelihood	−665.053	9001	R
prior	−211.722	9001	R
treeLikelihood.base	−665.053	9001	R
TreeHeight	21.475	8875	R
YuleModel	−106.62	7887	R
deathRate.SDollo	9.983E−3	8587	R
gammaShape	14.482	7863	R

(a) Strict clock



Trace Files:

Trace File	States	Burn-In
sdollo_relaxed.log	100000000	10000000

Reload

Traces:

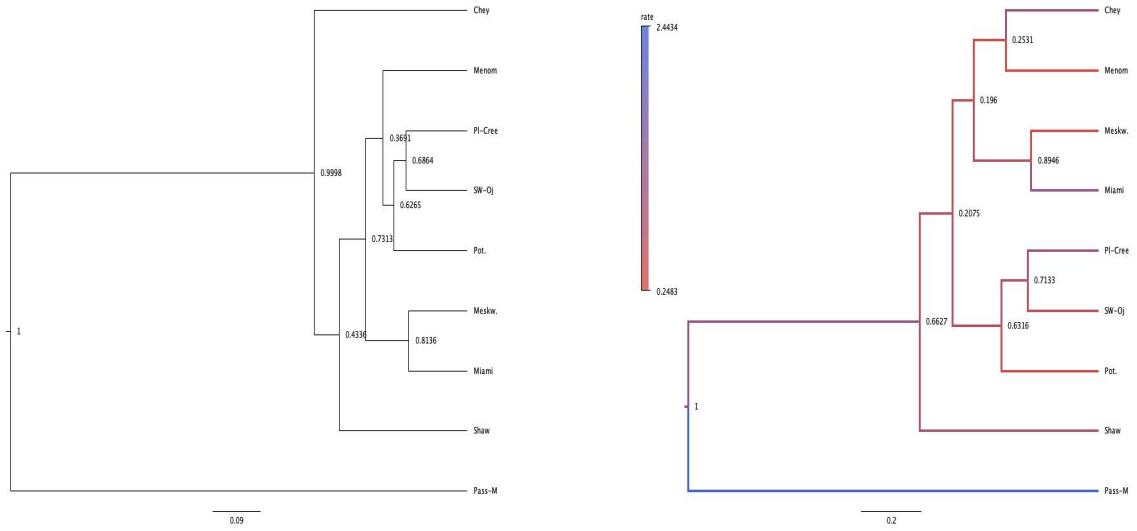
Statistic	Mean	ESS	...
posterior	−791.171	9001	R
likelihood	−630.371	9001	R
prior	−160.8	8589	R
treeLikelihood.base	−630.371	9001	R
TreeHeight	15.201	9001	R
YuleModel	−80.877	9001	R
deathRate.SDollo	7.374E−3	9001	R
gammaShape	10.695	7214	R
ucldStdev	1.708	8477	R
rate.mean	4.92	8105	R
rate.variance	4.413	9001	R
rate.coefficientOfVaria...	0.493	9001	R

(b) Relaxed clock

Figure 3: S. Dollo model

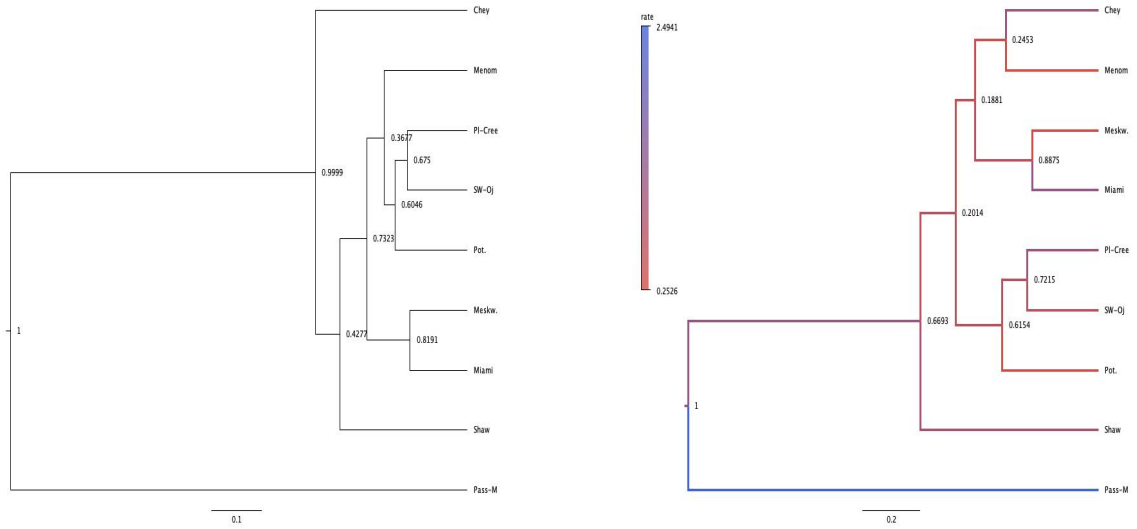
The high ESS values suggest good mixing of the subsampled MCMC chain.

7.2 Maximum clade credibility (MCC) trees



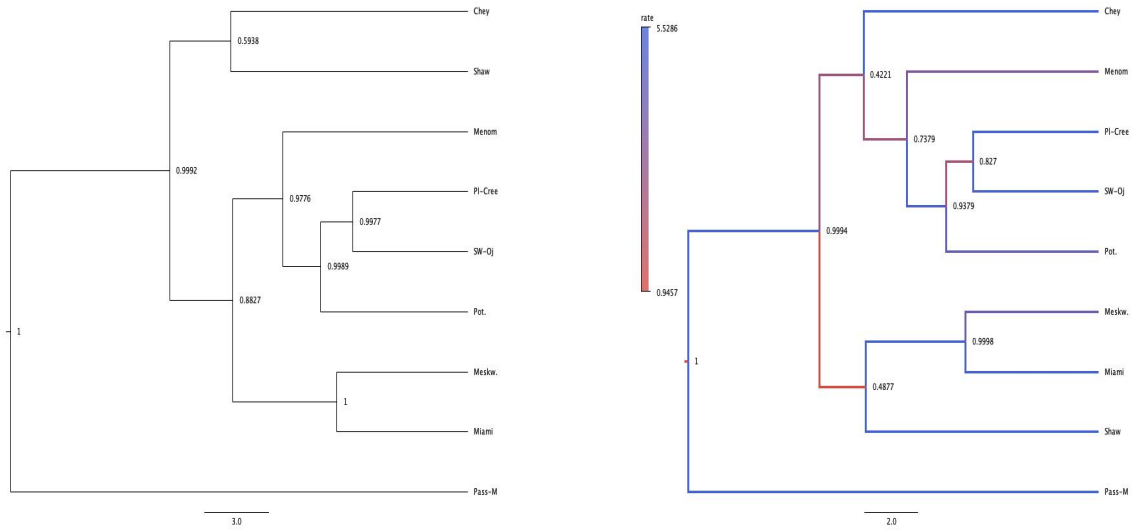
(a) Strict clock (b) Relaxed clock

Figure 4: CTMC model



(a) Strict clock (b) Relaxed clock

Figure 5: Covarion model



(a) Strict clock

(b) Relaxed clock

Figure 6: S. Dollo model

7.3 Bayes Factor Comparison

Model	Marginal L estimate	CTMC		Covarion		S. Dollo	
		strict	relaxed	strict	relaxed	strict	relaxed
CTMC strict	-691	—					
CTMC relaxed	-646	45	—				
Covarion strict	-693	-2	-47	—			
Covarion relaxed	-647	44	-1	46	—		
S. Dollo strict	-873	-182	-227	-180	-226	—	
S. Dollo relaxed	-818	-127	-172	-125	-171	55	—

Table 4: Table of marginal likelihoods and their differences, all on the log scale. The row corresponding to the best performing model is highlighted in light-gray.

7.4 Discussion

The Bayes Factor Comparison in 7.3 suggests the following:

- The Relaxed Clock assumption improves model fit.
- The Covarion assumptions do not improve upon the CTMC model (which is nested in the Covarion model).
- The Stochastic Dollo assumptions worsen model fit.

The CTMC relaxed and Covarion relaxed models perform noticeably better (i.e. have higher marginal likelihood estimates) than the other models. Since both models perform indistinguishably, we prefer the CTMC relaxed model since it makes fewer assumptions, and take the MCC tree it supports as an estimate of the phylogeny. Incidentally, a quick comparison in 7.2 shows that the topology of the MCC tree supported by the Covarion relaxed model is the same, while the posterior clade probabilities are essentially the same.

In figure 7, we revisit the MCC tree for the CTMC-relaxed model and additionally provide 95%-HPD intervals for the node ages. In table 5, we summarize the 8 clades (not disjoint!) of the MCC tree supported by the CTMC-relaxed model as well as their estimated posterior support.

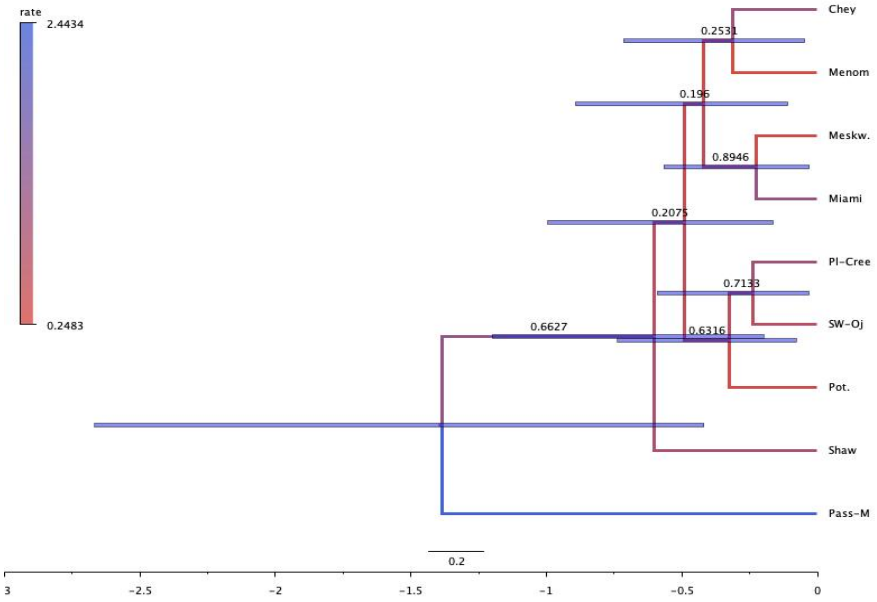


Figure 7: CTMC-relaxed model revisited. The 95% HPD intervals for node ages are shown as blue rectangles, whereas the internal edges are labeled by posterior support.

Clades	Posterior probability
(1) Pass-M, Shaw, Pot., SW-Oj, Pl-Cree, Miami, Meskw., Menom, Chey	1
(2) Shaw, Pot., SW-Oj, Pl-Cree, Miami, Meskw., Menom, Chey	0.6627
(3) Pot., SW-Oj, Pl-Cree, Miami, Meskw., Menom, Chey	0.2075
(4) Pot., SW-Oj, Pl-Cree	0.6316
(5) SW-Oj, Pl-Cree	0.7133
(6) Miami, Meskw., Menom, Chey	0.196
(7) Miami, Meskw.	0.8946
(8) Menom, Chey	0.2531

Table 5: MCC summary table for CTMC relaxed model

The MCC tree indicates the following:

- Clades 1 and 7 are strongly supported with posterior probabilities ≥ 0.8.
- Clades 3, 6, and 8 are poorly supported with posterior probabilities ≤ 0.5.
- Clades 2, 4, and 5 are moderately supported with posterior probabilities between $0.6 - 0.75$.

This suggests that many configurations of internal branching are plausible for explaining the data.