How to validate a Bayesian evolutionary model

Fábio K. Mendes[1†*], Remco Bouckaert[2†],
Luiz M. Carvalho[3†], Alexei J. Drummond[4]

[1]Department of Biological Sciences, Louisiana State University, Baton Rouge, LA 70803, United States

[2]School of Computer Science, The University of Auckland, Auckland 1010, New Zealand

[3]Escola de Matemática Aplicada, Fundação Getulio Vargas, Rio de Janeiro, RJ 22250-900, Brazil

[4]School of Biological Sciences, The University of Auckland, Auckland 1010, New Zealand

[*]Corresponding author: fmendes@lsu.edu

[†]Authors contributed equally to this work

October 25, 2024

Supplementary Material

1 Examples of software for evolutionary model simulation

Supplementary Table 1: A non-exhaustive list of simulation software used for various models in evolutionary biology.

Software package	Model type	Platform	Reference
Seq-Gen	Molecular sequence evolution models	Standalone	?
ms	Coalescent model	Standalone	?
msprime	Coalescent model	Python	?
SLiM	Population genetic models	Standalone	?
mvMORPH	Continuous trait evolution models	R	?
TreeSim	Birth-death models	R	?
diversitree	Several birth-death models	R	?
castor	Several birth-death models	R	?
PhyloJunction	Several birth-death models	Python	?
ReMASTER	Several birth-death models	BEAST 2	?
RPANDA	Several evolutionary models	R	?
LPhy	Several evolutionary models	Standalone	?
RevBayes	Several evolutionary models	RevBayes	?

2 Validating a phylogenetic Brownian motion simulator

One commonly used model in macroevolution for the study of continuous traits is the phylogenetic Brownian motion model ("PhyloBM" in Fig. 1 of the main text; ?). The pdf characterizing this model's sampling distribution is in fact the pdf of the multivariate normal (MVN) probability distribution:

$$\begin{aligned}\log f(\boldsymbol{y} \mid \boldsymbol{y_0}, r, \boldsymbol{T}) = &-\frac{1}{2}\Big[n\log(2\pi) + \log|r\boldsymbol{T}|\Big] \\ &-\frac{1}{2}\Big[(\mathbf{y}-\boldsymbol{y_0})^T(r\boldsymbol{T})^{-1}(\mathbf{y}-\boldsymbol{y_0})\Big].\end{aligned} \tag{1}$$

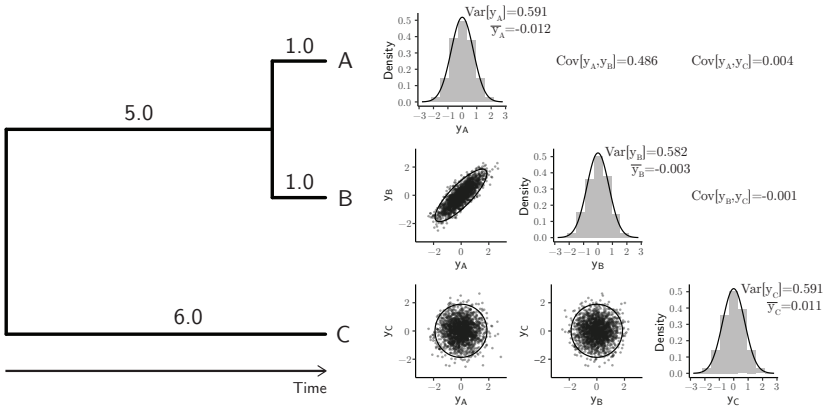
This probability density function describes the distribution obtained from an infinite number of BM "experiments" (each experiment being non-mean-reverting, and representing an independent evolutionary trajectory). Under this model, one has data $d = \{\boldsymbol{y}\}$ and parameters $\boldsymbol{\theta} = \{\boldsymbol{y_0}, r, \boldsymbol{T}\}$ (sometimes researchers treat ϕ and consequently $\boldsymbol{T}$ as data instead of as a parameter).

The data, $\boldsymbol{y}$, corresponds to the observed values of a trait scored for n species. Among the parameters, $\boldsymbol{y_0}$ is the

trait value at the root of the tree, r is the instantaneous rate of change (i.e., the evolutionary rate, and sometimes represented by σ^2), and $\boldsymbol{T}$ is a matrix whose elements are deterministically defined by tree ϕ's topology and branch lengths. Matrix $r\boldsymbol{T}$ is also known as the variance-covariance matrix; as an example, such matrix in supplementary figure 1 would evaluate to:

$$r\boldsymbol{T} = 0.1 \begin{bmatrix} 6 & 5 & 0 \\ 5 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix} \tag{2}$$

Together with $\boldsymbol{y_0} = [0.0, 0.0, 0.0]$, this matrix characterizes a population of phylogenetically related species trait values whose means are 0.0, variances are 6.0, and co-variances are 5.0 (between species "A" and "B") and 0.0 (between species "C"and either "A" or "B"). Supplementary figure 1 shows the distributions of trait values and their variances and covariances for one sample of one thousand independent realizations of phylogenetic BM processes. One can see that the sample's average trait value and the variances and covariances approach their expectations.



Supplementary Figure 1: A sample of 1000 draws from a MVN distribution, each representing the evolutionary trajectory of one continuous trait along the species tree on the left. The root trait value, $\boldsymbol{y_0}$, and the evolutionary rate of the process, r, were set to 0.0 and 0.1, respectively. The panel on the right shows histograms of 1000 trait values sampled from the MVN for each species, as well as their covariation.

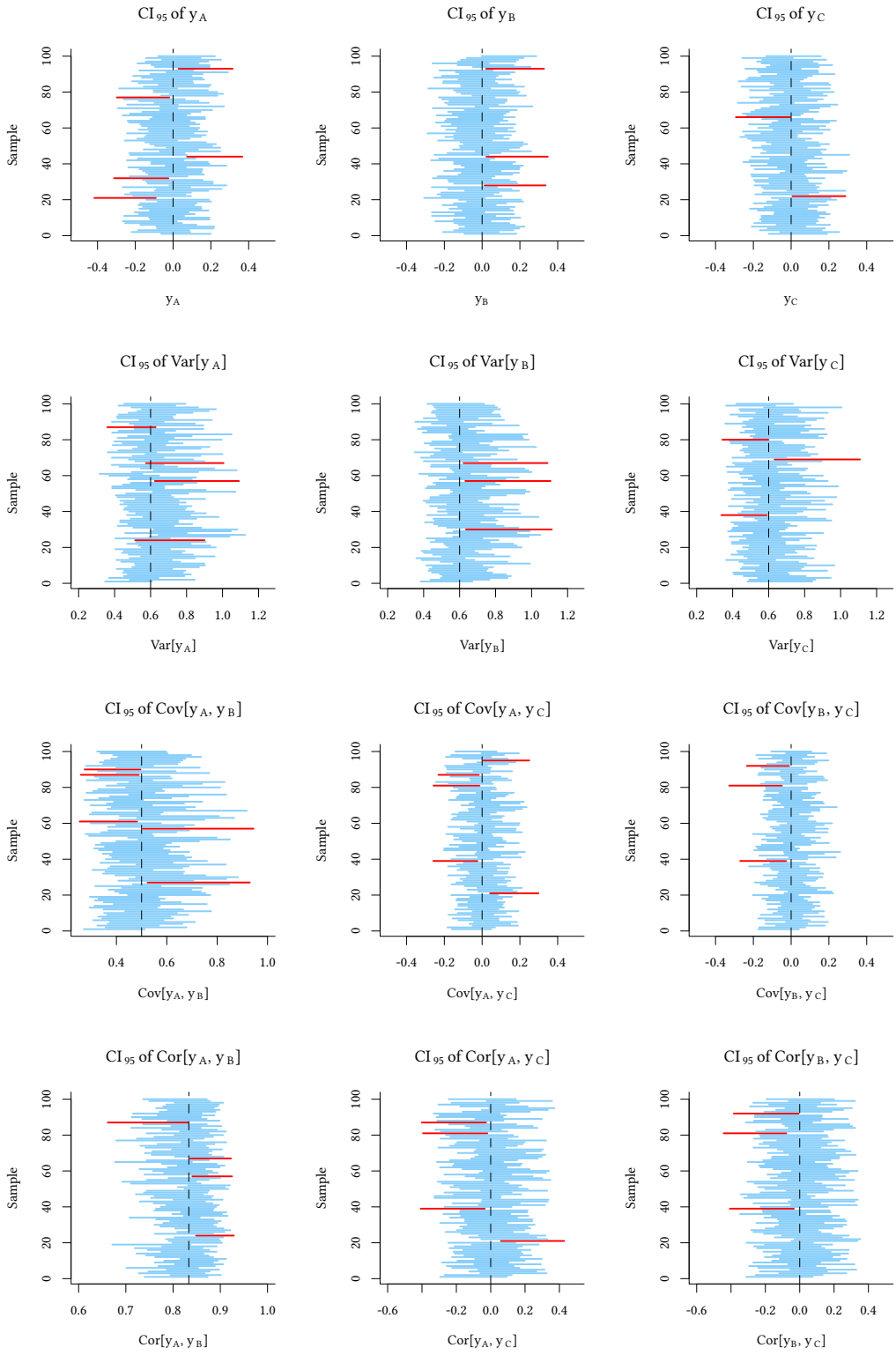
More rigorously, one can follow the method described in the main text – as we do in what follows – and verify that those expectations fall within their 95%-confidence intervals 95% of the time, as calculated from a large number of samples (Supplementary Fig. 2 and Supplementary Table 2). Here, we will verify that the expected value of certain summary statistics (given a specific combination of parameter values) falls within its α-confidence intervals approximately α% of the time. We will build confidence intervals about statistics calculated from several PhyloBM samples of size 10,000, and then ask if the "population" value of a statistic – given by the parameters of the multivariate normal sampling distribution – is contained within its confidence interval frequently enough.

For summary statistics, we pay attention to the trait value's (i) species mean, (ii) species variance, (iii) among-

species covariance, and (iv) among-species correlation coefficient. Supplementary figure 2 shows one hundred confidence intervals for each of these statistics, under multivariate normal MVN($\boldsymbol{y_0}, r\boldsymbol{T}$), where $\boldsymbol{y_0} = \{0, 0, 0\}$, $r = 0.1$ and $\boldsymbol{T}$ is given by the tree in Fig. 2 in the main text. Supplementary table 2 summarizes how often each statistic fell within its 95%-confidence interval. These results indicate the PhyloBM simulator produces appropriate confidence intervals and behaves as expected.

Supplementary Table 2: The number of times k that a summary statistic was contained within its corresponding 95%-confidence interval. Each statistic was calculated from 100 datasets of size 10,000 simulated under the PhyloBM model described in the text.

Statistic	Species s (and v)	k
$\mathrm{E}[y_s]$	A	95
	B	97
	C	98
$\mathrm{Var}[y_s]$	A	93
	B	97
	C	97
$\mathrm{Cov}[y_s, y_v]$	A and B	95
	A and C	95
	B and C	97
$\mathrm{Cor}[y_s, y_v]$	A and B	96
	A and C	96
	B and C	97



Supplementary Figure 2: One hundred 95%-confidence intervals (blue and red lines) built for four different summary statistics, when validating a phylogenetic Brownian motion model simulator. Red lines represent intervals that do not contain the value expected under the MVN sampling distribution defined by a bifurcating three-taxon phylogenetic tree. Summary statistics include each leaf's character-value mean (top row) and variance (second row from the top), as well as pairwise (leaf) character-value co-variances (third row from the top) and correlations (bottom row).

3 Model validation with rejection sampling: a simple example

In this section, we experiment with a simple hierarchical Gaussian (toy) model to further examine the effect of rejection sampling in coverage validation and rank-uniformity validation (RUV). This experiment is motivated by results described in the main text, namely, the model in scenario 3 (Fig. 1, with extreme rejection sampling) passing coverage validation but not RUV.

Let us devise the following data-generating process for obtaining different levels of model misspecification:

$$\begin{aligned} \mu &\sim \text{Normal}\,(0,1)\;T[t,], \\ y_1,\ldots,y_K &\overset{\text{iid}}{\sim} \text{Normal}\,(\mu,1)\,, \end{aligned}$$

where K is the sample size, and notation $T[,t]$ indicates the distribution is truncated **below** at $t \in \mathbb{R}$, i.e.,

$$\pi_t(\mu) = \frac{\phi(\mu)}{1-\Phi(t)}\mathbb{I}(\mu > t).$$

and ϕ and Φ are the pdf and CDF of a standard normal, respectively. Here, μ and $y_1,\ldots,y_K$ correspond to θ_i and d_i in Fig. 4 (main text), respectively.

Inference is then done under the misspecified model

$$\begin{aligned} \mu &\sim \text{Normal}\,(0,1)\,, \\ y_1,\ldots,y_K &\overset{\text{i.i.d.}}{\sim} \text{Normal}\,(\mu,1)\,. \end{aligned}$$

It is well-known that:

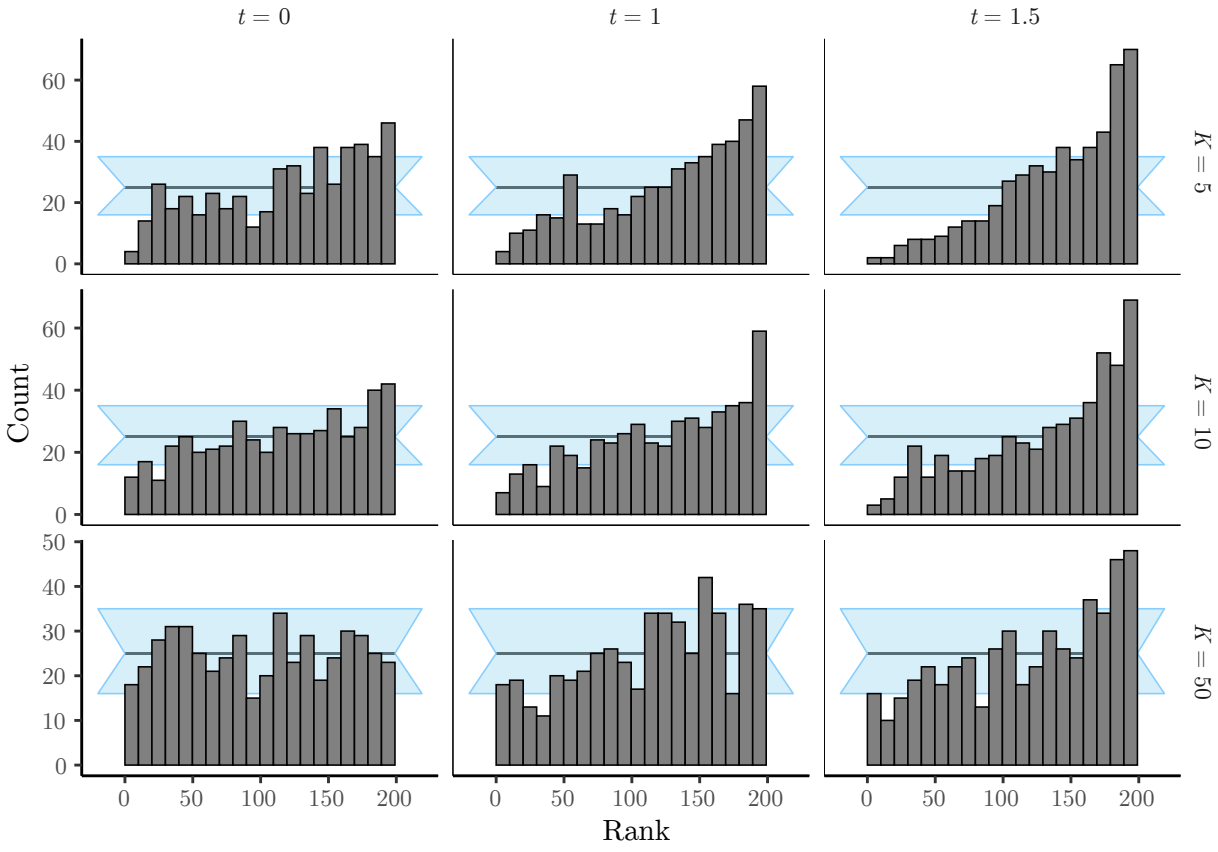
$$\mu \mid y_1,\ldots,y_K \sim \text{Normal}\left(\frac{\sum_{k=1}^{K} y_k}{K+1}, \frac{1}{K+1}\right), \tag{3}$$

i.e., the posterior is known in closed-form so MCMC is not required and one can sample directly and independently from it. One can thus control how extreme the model misspecification will be during inference by increasing t. Here we explore $t = \{0, 1, 1.5\}$ and $K = \{5, 10, 50\}$ in order to show how the method behaves in various misspecification scenarios. We ran $n = 1000$ replications of the experiment, draw 200 i.i.d. samples directly from the posterior in (3).

The resulting rank histograms (Supplementary Fig. 3) show how the posterior consistently underestimates the true mean, as indicated by a pattern of ranks bunching up towards the right-hand side of the histogram. The pattern is clearer in situations where the evidence in the data is weaker (e.g., when K is small), and the effect of the prior on the posterior is stronger. Moreover, the misspecification becomes more apparent the larger t is, that is, the more

extreme the misspecification becomes.

It is worth noting a couple of things. First, unlike what we observed and reported for the model in the main text, coverage validation does suggest an incorrectly specified model for certain combinations of t and K (Supplementary Table 2). Second, when the truncation is less extreme ($t = 0$) and there is a substantial amount of data ($K = 50$), RUV fails to detect any problems. This is an important insight about validation protocols: their sensitivity is context-dependent, and responds to the combination of model and data regime.



Supplementary Figure 3: Rank-uniformity validation (RUV) of the hierarchical Gaussian toy model. We show the rank histograms in three misspecification scenarios (t is 0.0, 1.0 or 1.5) and three data regimes (K is 5, 10 or 50). Results are based on $n = 1000$ replicates of $L = 200$ i.i.d. draws each. Horizontal black lines show the expected count for each rank (the same for all ranks due to uniformity), and the light-blue bands represent the 95%-confidence interval for the counts.

Supplementary Table 3: Coverage results for the 95%-HPD under the hierarchical Gaussian toy model. For each combination of truncation (t) and data set size (K), we show the estimated coverage over $n = 1000$ replicates, and whether the coverage procedure passes. A pass is determined according to table 2 in the main text.

Truncation (t)	Number of observations (K)	Estimated coverage	Pass
0	5	0.93	No
1	5	0.92	No
1.5	5	0.90	No
0	10	0.95	Yes
1	10	0.91	No
1.5	10	0.91	No
0	50	0.96	Yes
1	50	0.93	No
1.5	50	0.93	No

4 Validating a phylogenetic model with respect to a phylogenetic tree parameter, Φ

Our goal is to evaluate if an inferential implementation of model $\mathcal{M}$, $\mathrm{I}[\mathcal{M}]$, is well-calibrated and correct. In this section, we will focus on one critical parameter of $\mathcal{M}$: the phylogenetic tree Φ. We will pay special attention to Φ while carrying out coverage validation and the RUV procedure. In practice, this amounts to computing the coverage of Φ and the rank distribution of this parameter in comparison to its posterior samples.

What feature of Φ should one use when calculating coverage and ranks? Because of the nature of tree space, summarizing and comparing phylogenetic trees with univariate measures is not a trivial task. The key to an effective validation effort is to choose a functional that reflects relevant estimators of the quantity of interest, in our case, Φ. Fortunately, it is possible to exploit the metric nature of tree space and compute quantities both from a sampled phylogenetic tree ϕ – i.e., the "true" tree simulated during the validation process – as well as distances with respect to a reference phylogeny ϕ_0. We summarize some of these distances in supplementary table 4.

Supplementary Table 4: Metrics (functionals) for investigating model correctness in phylogenetic tree space. ϕ is a (phylogenetic tree) sample of Φ, sampled from a tree model (e.g., a Yule process). ϕ_0 is a reference tree to which ϕ and its posterior samples (see Algorithm 1 and the RUV section in the main text) are being compared, with respect to any of the metrics in the table. For the KC metric, the parameter (ω) controls the balance between topological and branch length information, and was set at $\omega = 0.5$.

Metric	Notation	Ref.
The largest branch length in ϕ	$\mathrm{LB}(\phi)$	N/A
The length of ϕ (the sum of all branch lengths)	$\mathrm{LEN}(\phi)$	N/A
The difference between the largest and smallest branch length of ϕ	$\mathrm{R}(\phi)$	N/A
The Robinson-Foulds distance between ϕ and ϕ_0	$\mathrm{RF}_0(\phi)$	(?)
The Kendall-Coljin distance between ϕ and ϕ_0	$\mathrm{KC}_0(\phi;\omega)$	(?)
The Billera-Holmes-Vogtman distance between ϕ and ϕ_0	$\mathrm{BHV}_0(\phi)$	(?)

In the case of tree-space distance metrics, we must slightly modify our RUV procedure (Algorithm 1). The key difference is the sampling of a reference phylogenetic tree, ϕ_0, prior to the simulation of the n i.i.d. Φ samples,

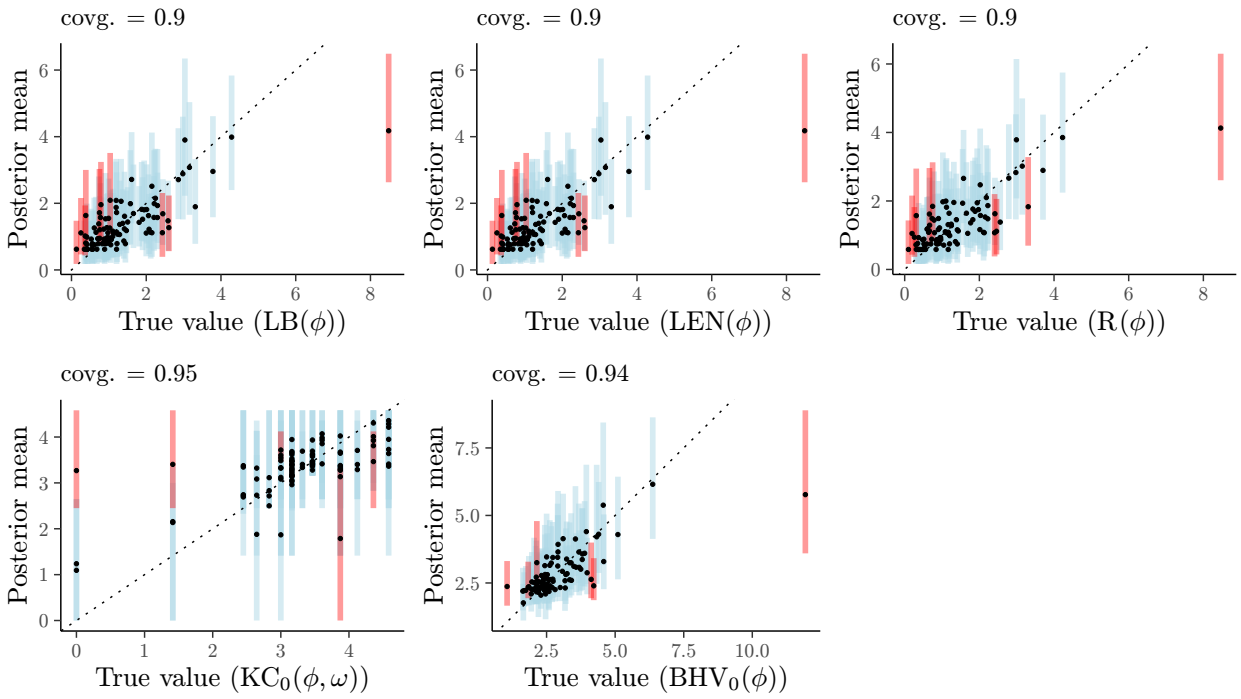
$\boldsymbol{\phi} = \{\phi_i : 1 \leq i \leq n\}$. Given a reference tree ϕ_0, each sampled ϕ_i and all of its L posterior MCMC samples are then compared to ϕ_0 with respect to a chosen distance metric. It is the evaluated distance metric that will underlie the ranking of ϕ_i relative to its posterior MCMC samples. Once ranks are computed, RUV proceeds as usual.

Algorithm 1: Algorithm for carrying out a rank-uniformity validation procedure with respect to the phylogenetic tree parameter Φ. Parameters $\boldsymbol{\theta}$ include both the tree parameter, θ_Φ, and non-tree (scalar) parameters, $\boldsymbol{\theta}_s$. Data $\boldsymbol{d}$ represents the output of an evolutionary process taking place along the phylogenetic tree (e.g., a continuous-time Markov chain modeling DNA substitutions).

$n \leftarrow 100$
/* Number of data sets to simulate */
$\theta_{\mathcal{H}} \leftarrow$ `InitializeHyperparameters()`
/* Hyperparameters initialized to constant values */
$\boldsymbol{\theta}_0 \leftarrow$ `SampleNonTreeParameters`$(\theta_{\mathcal{H}})$
/* $\boldsymbol{\theta}_0 \sim f_\Theta(\cdot)$, with $\boldsymbol{\theta}_0 = \{\theta_{\Phi,0}, \boldsymbol{\theta}_{s,0}\}$ */
$\phi_0 \leftarrow$ `SampleTree`$(\theta_{\Phi,0})$
/* $\phi_0 \sim f_{\Phi|\Theta_\Phi}(\cdot|\Theta_\Phi = \theta_{\Phi,0})$ */
for $i \leftarrow 1$ **to** n **do**
  $\boldsymbol{\theta}_i \leftarrow$ `SampleNonTreeParameters`$(\theta_{\mathcal{H}})$
  /* $\boldsymbol{\theta}_i \sim f_\Theta(\cdot)$, with $\boldsymbol{\theta}_i = \{\theta_{\Phi,i}, \boldsymbol{\theta}_{s,i}\}$ */
  $\phi_i \leftarrow$ `SampleTree`$(\theta_{\Phi,i})$
  /* $\phi_i \sim f_{\Phi|\Theta_\Phi}(\cdot|\Theta_\Phi = \theta_{\Phi,i})$ */
  $d_i \leftarrow$ `SampleData`$(\phi_i, \boldsymbol{\theta}_{s,i})$
  /* $d_i \sim f_{D|\Phi,\boldsymbol{\Theta}_s}(\cdot|\Phi = \phi_i, \boldsymbol{\Theta}_s = \boldsymbol{\theta}_{s,i})$ */
  $\boldsymbol{\theta}'_i \leftarrow \text{MCMC}(f_{D|\boldsymbol{\Theta}}(d_i|\boldsymbol{\Theta} = \boldsymbol{\theta}_i) f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_i))$
  /* $\boldsymbol{\theta}'_i = \{\boldsymbol{\theta}_i^j : 1 \leq j \leq L\}$, where L is the number of MCMC samples */
  $\bar{\delta}_i \leftarrow$ `CalculateDistance`(ϕ_0, ϕ_i)
  /* According to a distance metric of choice */
  $\boldsymbol{\delta}_i^j \leftarrow$ `CalculateDistance`(ϕ_0, ϕ_i^j)
  /* $\boldsymbol{\delta}_i = \{\delta_i^j : 1 \leq j \leq L\}$ */
  $r_i \leftarrow$ `GetRank` $(\bar{\delta}_i, \boldsymbol{\delta}_i^j)$
  /* $r_i = \sum_{j=1}^{L} \mathbb{I}(\delta_i^j < \bar{\delta}_i)$ */
end
if `IsRankDistributionUniform`$(\boldsymbol{r})$ **then**
  return true
end
else
  return false
end

Supplementary figures 4 and 5 summarize the coverage and RUV results, respectively, for the metrics listed in supplementary table 4, when the model is the simple Kingman's coalescent. The model consists of a single five-taxon phylogenetic tree parameter, Φ, assuming a known effective population size of 1.0.

We can first verify that our model implementation is well-calibrated with respect to the phylogenetic tree parameter, as shown by appropriate 95%-HPD-coverage statistics over the different tree metrics (Supplementary Fig. 4). The evidence for implementation correctness is enhanced by observing that the ranks for the different metrics



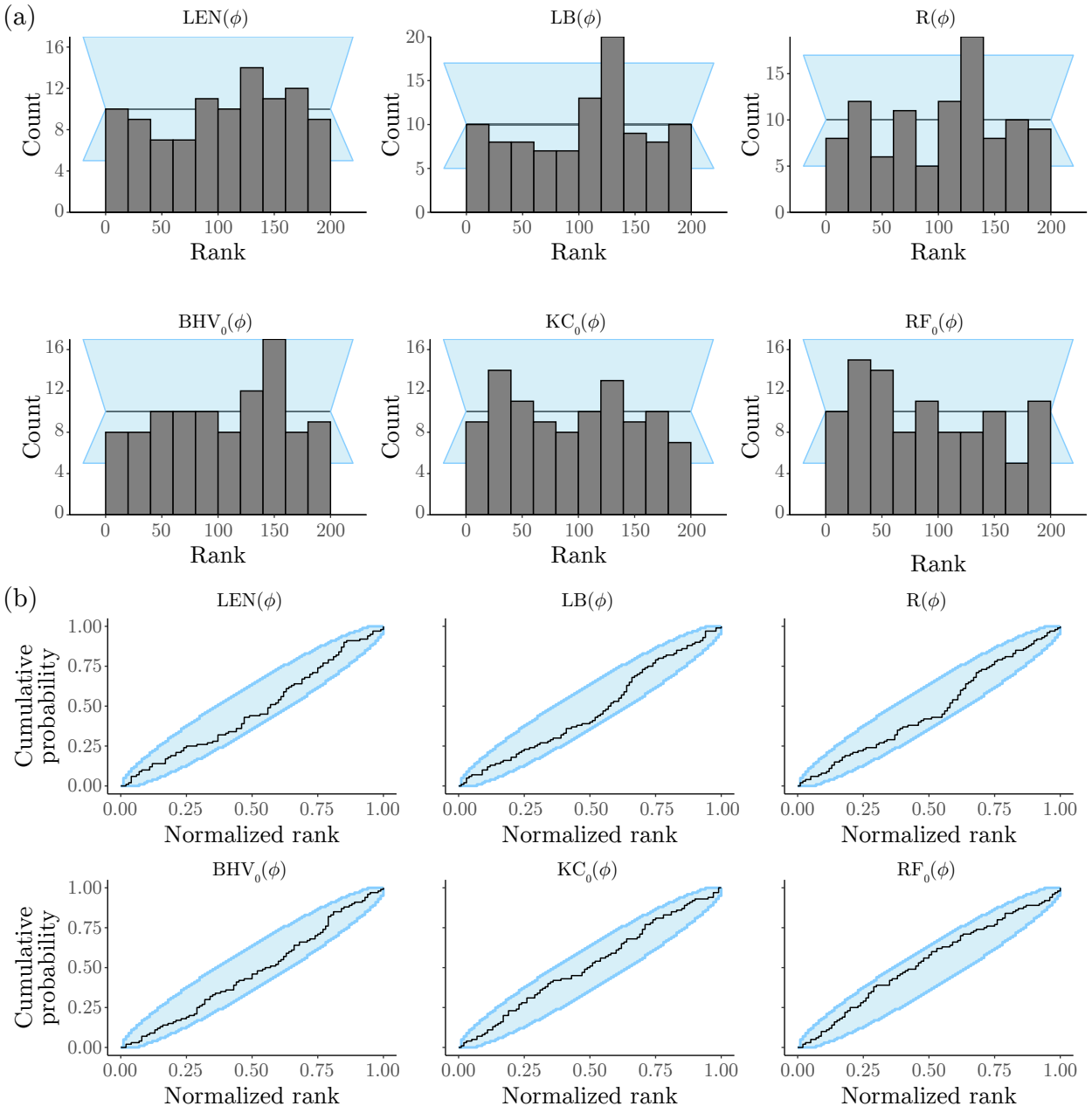
Supplementary Figure 4: Coverage validation of a phylogenetic model, focusing on the phylogenetic tree parameter (see Supplementary Table 4 for a description of the different distance metrics).

are (approximately) uniformly distributed, lying inside their confidence bands (Supplementary Fig. 5a), and their corresponding ECDFs also lie well inside their confidence ellipses (Supplementary Fig. 5b).

In the main text, we also mentioned the possibility of assessing the coverage of a phylogenetic tree's topology; specifically, by looking at the frequency of its clades as a function of their posterior support. An outcome of this procedure can be seen in supplementary figure 7, for a simple 50-sample Kingman coalescent model. We propose that the clade frequency diagnostic is further evaluated in two ways. One can first check that the resulting empirical cumulative distribution function (ECDF) fits that of an appropriately scaled uniform distribution using a Kolmogorov-Smirnov test (**?**). Additionally, one can fit a regression model to the attained frequencies against the bin midpoints, and test the null hypothesis that the intercept is zero and the slope is one, rejecting it at the usual confidence levels.

Importantly, there are two considerations we would like to bring to the reader's attention. First, depending on the starting tree's topology, its number of tips, and the length of the MCMC chain, true clades may never be sampled by chance more frequently than expected. This could be diagnosed by observing the left-most bar in supplementary figure 7 being taller than expected; here, one alternative is to ignore this first bar when carrying out the statistical tests suggested above.

Second, regardless of the simulated phylogenetic signal (with respect to the tree topology) – irrespective of, e.g., too few, an ideal amount, or too many molecular substitutions in simulated alignments – the ladder-like pattern in



Supplementary Figure 5: Rank-uniformity validation (RUV) of a phylogenetic model, focusing on the phylogenetic tree parameter (see Supplementary Table 4 for a description of the different distance metrics). (a) Rank distribution for each metric. (b) Empirical cumulative distribution function (ECDF) for each metric.

supplementary figure 7 should still be observed. This is because we suggest the count of clades with a given posterior clade support be normalized by the total number of clades with that support: if a very large number of true clades has support close to 0.0 or to 1.0, those counts will be divided by the total count of clades with corresponding support levels. Trees must have enough tips, however, or a large enough number of trees must be simulated so that all bins, including intermediate support ones, have large enough counts of sampled clades. Lastly, we emphasize that there is nothing intrinsically wrong with validating a model with a data set void of any signal. In coverage validation, the prior mean would be recovered for each parameter, and model correctness could still be in principle verified.

5 Proof for coverage validation

In this section we provide a mathematical argument that coverage-based validation is sound, i.e., that sampling from the prior, simulating data and then using the same prior for computing the posterior should give Bayesian credible intervals (BCIs) with nominal frequentist coverage.

For a number n of replicates, simulate parameter values θ_i, and then given those values, simulate data d_i:

$$\theta_i \sim f_\Theta(\cdot),$$
$$d_i \sim f_{D|\Theta}(\cdot|\Theta = \theta_i).$$

Now for notational convenience, define $a_i := a(d_i, \alpha)$ as the HPD lower bound and, similarly, $b_i := b(d_i, \alpha)$ as the HPD upper bound. Recall $I_\alpha\left(d_i\right)$ is such that:

$$Q_{d_i}\left(b_i\right) - Q_{d_i}\left(a_i\right) = p_1 - p_2 = \alpha,$$

where $Q_d(x)$ is the posterior CDF (conditional on data d) and $p_1, p_2 \in (0,1)$, with $p_1 < p_2$. A natural quantity to compute is:

$$S_n = n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(\theta_i \in I_\alpha\left(d_i\right)\right),$$

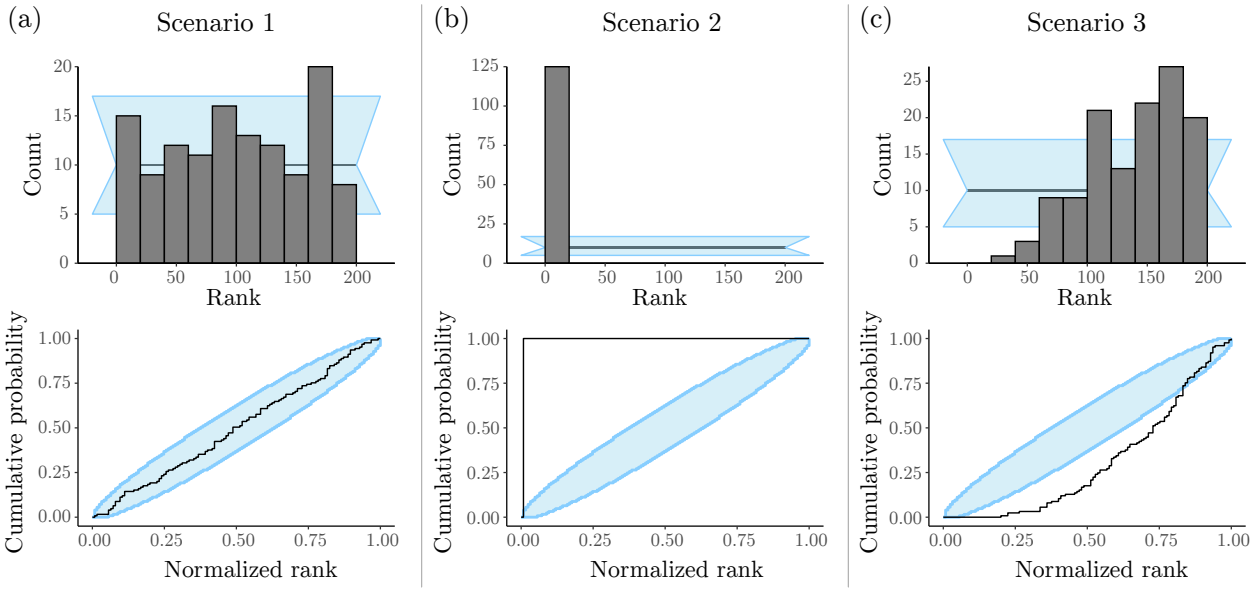
i.e., the attained coverage of the Bayesian intervals.

Let $F_U(x) = x$ be the CDF of a $\text{Uniform}(0,1)$ random variable. Now we can consider what happens when the number of simulations grows, i.e., the limit $\lim_{n\to\infty} S_n$. We may re-write the limit as:

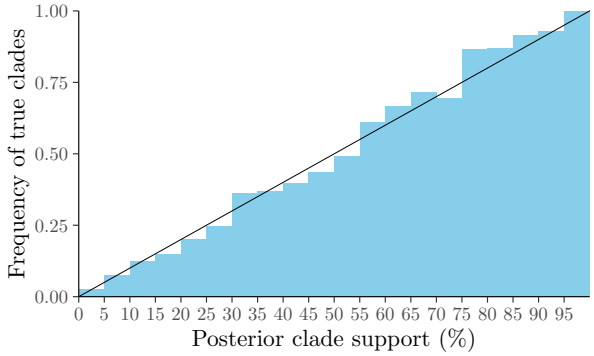
$$
\begin{aligned}
\lim_{n\to\infty} S_n &= \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(\theta_i \in I_\alpha\left(d_i\right)\right), \\
&= \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \left\{\mathbb{I}\left(\theta_i \leq b_i\right) - \mathbb{I}\left(\theta_i \leq a_i\right)\right\}, \\
&= \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(\theta_i \leq b_i\right) - n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(\theta_i \leq a_i\right), \\
&= \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(Q_{d_i}^{-1}\left(\theta_i\right) \leq p_1\right) - \lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathbb{I}\left(Q_{d_i}^{-1}\left(\theta_i\right) \leq p_2\right), \\
&= F_U(p_1) - F_U(p_2) = \alpha,
\end{aligned}
$$

where the last line follows from the fact that the CDF of θ_i is uniformly distributed on $(0, 1)$ (Theorem 1 in **?**) and almost sure convergence of the ECDF to the true CDF due to the Glivenko-Cantelli theorem (**?**, page 275).

6 Other supplementary figures



Supplementary Figure 6: Rank-uniformity validation (RUV) of the Bayesian hierarchical model in Fig. 1 in the main text. Panels in the top row show the histograms of $n = 100$ ranks, for parameter Λ in each scenario, obtained after 10% burnin and thinning of posterior samples down to 200 out of 10,000. Panels in the bottom row show the corresponding ECDF plots, for parameter Λ in each scenario. (a) In "Scenario 1", the model was correctly specified, and we simulated trees with 3 to 300 taxa using rejection sampling (approximately one in ten trees were rejected). (b) In "Scenario 2", the model was incorrectly specified in inference (see main text), and we used the same data sets simulated in "Scenario 1". (c) In "Scenario 3", the model was correctly specified, but rejection sampling was more intense (we rejected a large number of trees, approximately 90%, keeping those having between 100 to 200 tips).



Supplementary Figure 7: Coverage validation in phylogenetic tree (topological) space. Bars represent the counts of true clades (in $n = 100$ true, simulated trees) with their corresponding bin's posterior clade support, normalized by the total count of clades with that same posterior support value.

7 Coverage validation using the Experimenter package for BEAST 2: a worked example

In this section we present a worked example of validation study using BEAST 2's Experimenter package. We will start from working installations of BEAST 2 and the Experimenter package, which can be found on `https://github.com/christiaanjs/beast-validation/`. Readers following along will also profit from having installed the `Tracer` (?) and `LogCombiner` applications.

The hierarchical model whose implementation we will validate in this section is slightly more complex (and thus less pedagogical) than the one covered in the main manuscript, but more closely resembles a typical model employed by empirical phylogeneticists. Much of the model specification will be familiar to this journal's readership. For the sake of brevity, we leave the details on hyperprior specifications (parametric distributions) to the XML code snippets (those were commented so that they are human readable).

7.1 Simulation

We will use BEAST 2's direct simulators to generate a set of phylogenetic trees under a Yule prior with normally distributed birth rates, $\lambda \sim \mathrm{N}(1.0, 0.1)$. Such a narrow normal prior constrains tree heights such that substitution rates are neither too low (i.e., we observe some phylogenetic signal) nor too high (i.e., we do not saturate our molecular alignments). We shall assume a strict clock model with global substitution rate of 1.0.

Molecular alignments are simulated under an HKY model (?) with the transition to transversion rate, κ, being drawn from a log-normal distribution, and nucletotide frequencies, π, sampled from a Dirichlet prior. Among-site substitution-rate variation is modeled with a discrete Gamma model (?), where the shape parameter is drawn from an exponential distribution.

```
<beast version="2.0" namespace="beast.pkgmgmt:beast.base.core:beast.base.inference:beast.base.evolution.
    alignment:beast.base.evolution.tree :beast.base.evolution.speciation:beast.base.inference.distribution:
    beast.base.inference.util :beast.base.inference.parameter">

    <run spec="DirectSimulator" nSamples="101">
        <distribution spec="CompoundDistribution" id="fullModel">
            <!-- Sampling the tree -->
            <distribution spec="YuleModel" id="yuleModel">
                <tree spec="Tree" id="Tree">
                    <taxonset spec="TaxonSet">
                        <plate var="n" range="t0,t1,t2,t3,t4,t5,t6,t7,t8,t9">
                            <taxon spec="Taxon" id="$(n)"/>
                        </plate>
                    </taxonset>
```

```
            </tree>
            <birthDiffRate spec="RealParameter" id="birthRate" value="1.0"/>
        </distribution>

        <!-- Sampling the birth rate -->
        <distribution spec="beast.base.inference.distribution.Prior" id="birthDiffRatePrior">
            <x idref="birthRate"/>
            <distr spec="Normal" mean="4" sigma="0.1"/>
        </distribution>

        <!-- Sampling kappa for the HKY model -->
        <distribution spec="beast.base.inference.distribution.Prior" id="kappaPrior">
            <x spec="RealParameter" id="kappa" value="1.0"/>
            <distr spec="LogNormalDistributionModel" M="1.0" S="1.25"/>
        </distribution>

        <!-- Sampling the nucleotide frequencies for the HKY model -->
        <distribution id="FrequenciesPrior.s:dna" spec="beast.base.inference.distribution.Prior" >
           <x spec="RealParameter" id="freqParameter" value="0.25 0.25 0.25 0.25"/>
           <distr spec="Dirichlet" alpha="4.0 4.0 4.0 4.0"/>
        </distribution>

        <!-- Sampling the shape parameter of the discrete gamma model for rate heterogeneity -->
        <distribution spec="beast.base.inference.distribution.Prior" id="GammaShapePrior.s:dna">
            <x spec="RealParameter" id="gammaShape" lower="0.1" value="1.0"/>
            <distr spec="Exponential" mean="1.0"/>
        </distribution>
    </distribution>

    <!-- Logging all sampled scalar values into file truth.log-->
    <logger logEvery="1" fileName="truth.log">
        <log id="TreeHeight" spec="beast.base.evolution.tree.TreeStatLogger" tree="@Tree"/>
        <log idref="birthRate"/>
        <log idref="kappa"/>
        <log idref="gammaShape"/>
        <log idref="freqParameter"/>
    </logger>

    <!-- Logging all sampled tree values into truth.trees-->
    <logger logEvery="1" fileName="truth.trees">
        <log idref="Tree"/>
    </logger>
  </run>
</beast>
```

As the XML file above specifies, we have BEAST 2 write all the sampled scalar and tree ("true") values into log files `truth.xml` and `truth.trees`, respectively. Inspecting the `truth.log` with the Tracer app reveals that the simulator works as expected (the means and variances of parameters match those specified in the XML file). Note

that we generated 101 samples ($n = 100$ if we follow the main manuscript's notation; with an extra simulation for executing RUV in tree space), but nothing prevents users from generating more data sets.

At this point, the Experimenter package has the "true" simulated values it needs to populate an XML template 100 times. The outcome will be 100 independent BEAST 2 runnable XML files for statistical inference. The XML template looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><beast beautitemplate='Standard' beautistatus=''
     namespace="beast.core:beast.evolution.alignment:beast.evolution.tree.coalescent:beast.core.util:beast.
     evolution.nuc:beast.evolution.operators:beast.evolution.sitemodel:beast.evolution.substitutionmodel:beast.
     base.evolution.alignment:beast.pkgmgmt:beast.base.core:beast.base.inference:beast.base.evolution.tree.
     coalescent:beast.pkgmgmt:beast.base.core:beast.base.inference.util:beast.evolution.nuc:beast.base.evolution
     .operator:beast.base.inference.operator:beast.base.evolution.sitemodel:beast.base.evolution.
     substitutionmodel:beast.base.evolution.likelihood" required="" version="2.7">

<!-- Alignment specifying the data type and taxa -->
<data id="input_alignment" spec="Alignment" name="alignment"  dataType="nucleotide">
    <plate var="n" range="t0,t1,t2,t3,t4,t5,t6,t7,t8,t9">
        <sequence id="seq_$(n)" taxon="$(n)" totalcount="4" value="?"/>
    </plate>
</data>

<!-- The simulated sequence alignment -->
<data spec='beastfx.app.seqgen.SimulatedAlignment' id="dna" sequencelength="$(sl=100)">
    <!-- The true tree down which sequences are simulated. -->
    <tree id="trueTree" spec='beast.base.evolution.tree.TreeParser' newick="$(tree)" IsLabelledNewick="true"
         adjustTipHeights="false" taxa="@input_alignment"/>

    <!-- Alignment block specified above -->
    <data idref="input_alignment"/>

    <!-- Site model used to simulate sequence data -->
    <siteModel id="trueSiteModel.s:dna" spec="SiteModel" gammaCategoryCount="4" proportionInvariant="0.0"
         mutationRate="1.0">

        <!-- Discrete Gamma shape parameter value will be replaced by the true value -->
        <parameter id="truegammaShape.s:dna" spec="parameter.RealParameter"  name="shape">$(gammaShape)</
             parameter>

        <!-- Kappa and nucleotide frequency values will be replaced by the true values -->
        <substModel id="trueHKY.s:big2_barcs" spec="HKY" kappa="$(kappa)">
            <frequencies id="trueFreqs.s:dna" spec="Frequencies" frequencies="$(freqParameter.1) $(
                 freqParameter.2)  $(freqParameter.3) $(freqParameter.4)"/>
        </substModel>
    </siteModel>

    <!-- We use a strict clock with rate fixed to 1.0 -->
    <branchRateModel id="trueStrictClock.c:cytb" spec="beast.base.evolution.branchratemodel.StrictClockModel"
```

```
        clock.rate="1.0"/>
</data>

<!--
       The rest of the XML comes from a BEAUti generated XML that is too long to
       include here. The full file can be found here:
       https://github.com/rbouckaert/DeveloperManual/blob/master/examples/yule/input.xml
-->
```

Here, we can use the `CoverageTestXMLGenerator2` utility that comes with the Experimenter package to generate the 100 BEAST XML files. It can be run from the shell or Terminal application as follows:

```
applauncher CoverageTestXMLGenerator2 -workingDir . -out xml -log truth.log -tree truth.trees -xml input.xml
```

(Note that application `applauncher` above is part of BEAST 2, and that it must be in the PATH environmental variable of the Terminal to be run as specified.) This command generates a directory called `xml/` with files called `analysis-out0.xml` to `analysis99.xml`. It can be useful to make a subdirectory `run1/` inside the `xml` directory for the occasion of failed or alternative experiments (e.g., longer MCMC chains).

7.2 Statistical inference

Inside the `run1/` directory, we can run the 100 XML files from the Terminal:

```
for i in {0..99}; do beast -seed 127$i -overwrite ../analysis-out$i.xml > out$i 2>&1 ; done
```

Once the BEAST 2 runs have finished, we then use the `LogCombiner` application to summarize the parameter estimates in a single file:

```
loganalyser -oneline *log > summary
```

7.3 Calculating and visualizing coverage

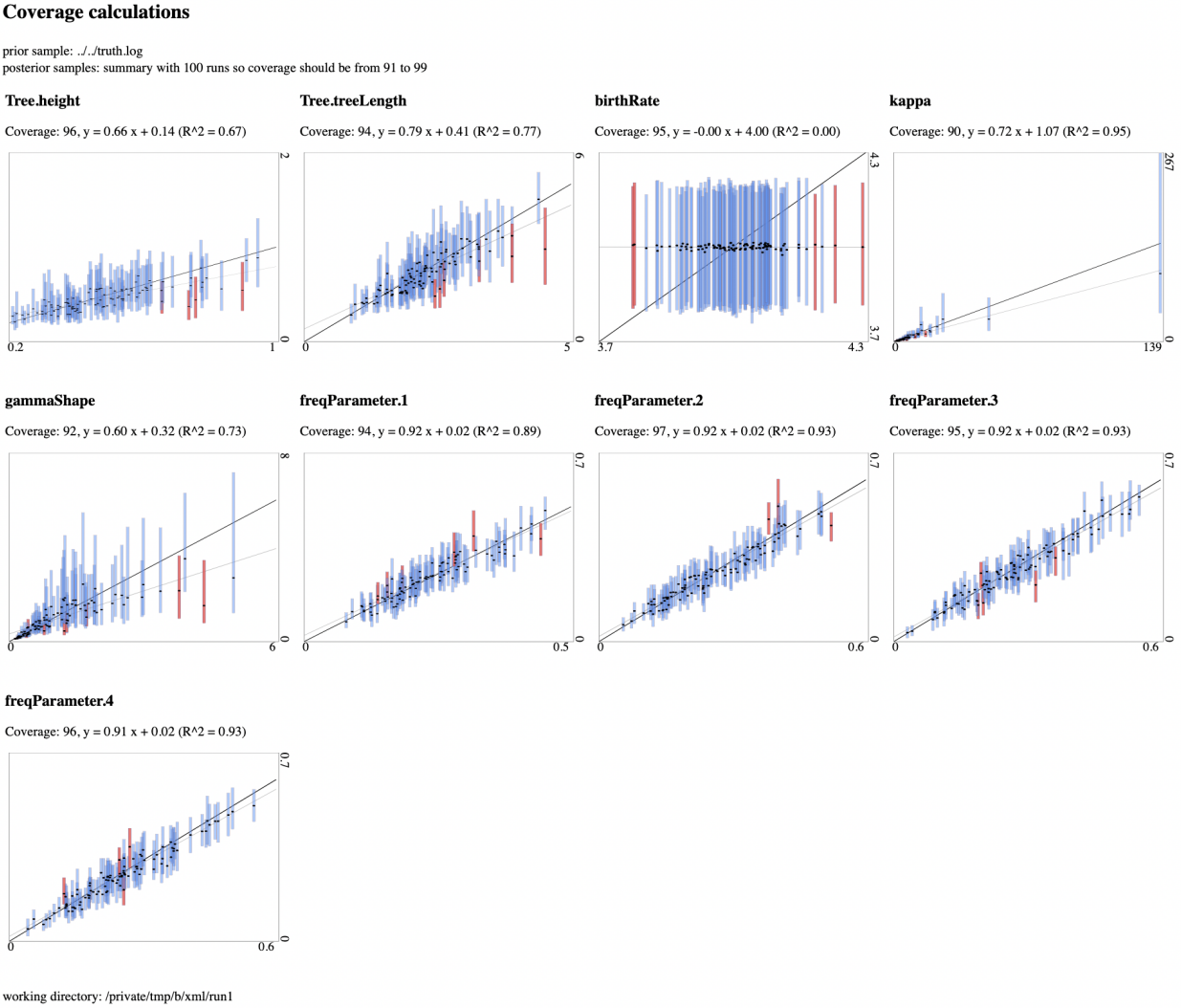
After statistical inference, we can run the `CoverageCalculator` tool (part of the Experimenter package) from the Terminal to produce coverage summary statistics:

```
applauncher beastvalidation.experimenter.CoverageCalculator -log ../../truth.log -logA summary -out /tmp -
    showRho true -showESS false -showMean false
```

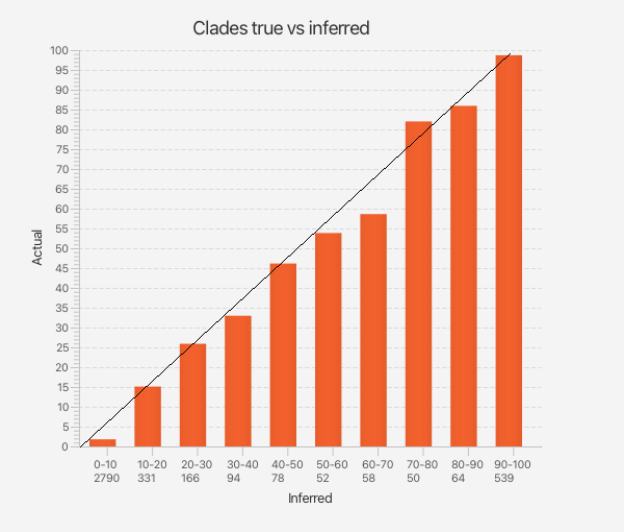
The Terminal output of coverage calculator in the Experimenter package for BEAST 2 is shown in supplementary figure 8. An asterisk is added next to the coverage column if it has a value outside the expected range. Here κ is marked, and judging from the low effective sample size (see `Min ESS` column) this can be fixed by running the MCMC chains longer. The Experimenter package should further output graphs for scalar parameters as shown in supplementary figure 9.

```
posterior samples: summary with 100 runs so coverage should be from 91 to 99
                                          coverage  Mean ESS      Min ESS missmatches
Guessing file order -- if this fails missarably, try setting guessFileOrder to false
Tree.height                                 96        409.64       37.04   ..............x.............x...........x.....................x.............................. 24 39 6 72
Tree.treeLength                             94        417.48       38.00   .x.............x.............x...........x.....................x.....x....................... 2 24 39 6 72 78
birthRate                                   95        841.01       560.97  ........x.....x.....................................x.............x..x........................... 18 23 54 70 72
kappa                                       90*       60.26        17.71   ............x.......x........x...x..x.................x...x.x....x.............x............ 21 30 39 42 45 61 65 67 72 88
gammaShape                                  92        68.34        5.76    ......x.....x.............x.x..........xx.........x.....................x................. 16 22 36 38 49 50 60 83
freqParameter.1                             94        50.37        14.98   ..........................x.....................x.........x...........x.................xx. 36 57 67 79 98 99
freqParameter.2                             97        51.39        13.89   ..........................x.....................x.....x.....x.....x.....x.................. 27 56 67
freqParameter.3                             95        50.16        15.98   ..........................x.....................x.....x.....x.....x.....x.....x...x.. 27 48 71 94 98
freqParameter.4                             96        49.81        14.73   ..............................................................x.............x.............xx..... 61 77 92 93
```

Supplementary Figure 8: Terminal output of the `CoverageCalculator` application/



Supplementary Figure 9: Graphs show the $x = y$ line in black and linear regression lines through the medians in gray. The regression lines and R^2 values indicate the influence of the prior vs. that of the likelihood over the parameter estimates. A low R^2 (e.g., birth rate) suggests the prior has more influence while a high R^2 (e.g., nucleotide frequencies) suggests the data has more influence on estimates.



Supplementary Figure 10: Clade coverages for the worked example using BEAST 2's Experimenter package.

Finally, we look at clade coverages, which can be calculated using the `CladeCoverageCalculator` application in the Experimenter package:

```
applauncher CladeCoverageCalculator -tru ../../truth.trees -pref analysis-out -png coverage.png -bins 10
```

This command executes the procedure described at the end of section 6, producing the graph illustrated by supplementary figure 7.

7.4 RUV validation

To do an RUV analysis, we must first combine the log files produced during inference into a single file:

```
logcombiner -b 10 -log analysis-out?.log analysis-out??.log -o combined.log
```

Next, we create summary statistics using the `SBCAnalyser` tool that is part of the Experimenter package:

```
applauncher SBCAnalyser -log ../../truth.log -logA combined.log -out /tmp -bins 10
```

The Terminal output is shown in supplementary figure 11; it displays the bin counts as well as ECDF plots (not all of them shown due to size of screen display). We can also graphically summarize the RUV procedure (Supplementary Fig. 12).

More details on the tools used above can be found on `https://github.com/rbouckaert/DeveloperManual/`.

	missed	bin0	bin1	bin2	bin3	bin4	bin5	bin6	bin7	bin8	bin9
Tree.height	0	3	12	8	7	9	13	14	11	13	10
Tree.treeLength	0	3	9	11	10	12	13	9	11	12	10
birthRate	0	8	8	9	10	10	11	12	17	6	9
kappa	0	14	8	11	9	9	7	14	11	4	13
gammaShape	0	5	10	16	9	7	8	8	12	16	9
freqParameter.1	0	13	16	7	17	9	7	8	7	9	7
freqParameter.2	1	7	9	12	16	7	9	9	4	21	6
freqParameter.3	0	9	13	8	6	7	9	13	12	13	10
freqParameter.4	0	8	9	6	7	9	15	15	11	11	9

Supplementary Figure 11: Frequency parameter 2 has one bin that is outside the expected range (indicated by the missed bin column).

Supplementary Figure 12: Rank histograms and ECDF plots for the worked example using BEAST 2's Experimenter package.