Supplementary information for Probabilistic programming: a powerful new approach to statistical phylogenetics

Fredrik Ronquist[1†*], Jan Kudlicka[2†], Viktor Senderov[1†], Johannes Borgström[2], Nicolas Lartillot[3], Daniel Lundén[4], Lawrence Murray[5], Thomas B. Schön[2], David Broman[4]

[1]Department of Bioinformatics and Genetics, Swedish Museum of Natural History, Box 50007, SE-104 05 Stockholm, Sweden
[2]Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden
[3]Laboratoire de Biométrie et Biologie Evolutive, UMR CNRS 5558, Université Claude Bernard Lyon 1, FR-69622 Villeurbanne Cedex, France
[4]Department of Computer Science, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
[5]Uber AI, San Francisco CA 94105, United States

1 Probabilistic programming: an introduction

In this section, we give a brief introduction to (universal) probabilistic programming languages (PPLs), focusing on the key constructs that are available in most PPLs. First, we give a short overview of different PPLs, followed by an introduction to three essential concepts in probabilistic programming: (i) sampling, (ii) conditioning, and (iii) inference. These concepts are illustrated here in the WebPPL language[a], which is based on a functional subset[b] of the JavaScript language.

1.1 Overview

A central objective of probabilistic programming is to separate the *model* from the *inference* algorithm, such that a user can construct and use a probabilistic model without the need to implement the inference algorithm explicitly. Instead, it is the task of the runtime system of the PPL to automatically perform the inference, potentially based on some method preferences specified by the user.

Programming and modeling languages that separate the model specification from the inference algorithm have been around for several decades. One of the first of these languages is BUGS (Bayesian inference Using Gibbs Sampling)[1]. BUGS allows users to describe probabilistic graphical models[2]—in particular Bayesian networks—in a declarative way. The model parameters of interest are then estimated by automatically applying Bayesian inference using Gibbs sampling (and some other methods). More recent languages that separate modeling and inference of graphical models include Infer.NET[3].

Although the above-mentioned languages and environments have shown great success in their application areas, they have certain model restrictions. In particular, they are limited to models where the dependencies between random variables can be expressed as a Bayesian network, that is, a finite directed acyclic graph, potentially with if-then-else conditions over variables. In some domains, this is not sufficient to describe the models of interest. Rather recently, the concept of *probabilistic programming languages*[4] has gained significant attention as a promising solution, in particular within the machine learning and programming language communities. The key idea of this new paradigm is to extend Turing-complete programming languages with probabilistic operations that include, for example, the drawing of (random) samples from a given probability distribution, the conditioning of random variables on observed outcomes, and the marginalization of random variables[5,6].

Such languages are sometimes referred to as *universal probabilistic programming languages* to clearly differentiate them from languages based on Bayesian networks, which have sometimes in recent years also been included in the probabilistic programming family. Here, we will use the terms "probabilistic programming" and "probabilistic programming language (PPL)" exclusively for universal languages.

Turing-completeness is an important concept in computer science, describing how expressive a programming language is. The famous Church-Turing thesis conjectures that any function, whose value can be computed by an algorithm, can be computed by a Turing-complete programming language. For instance, PPLs make it possible to use recursion (or loops) dependent on a stochastic expression when defining probabilistic models. This means that the graphical network describing the model is *dynamic* and can change during inference due to random sampling and observed data. In a PPL, the probabilistic model *is a program*, where the inference algorithm is not part of that program (the model). Hence, an alternative and potentially more intuitive name for a probabilistic program may be a *programmatic model*: a model that is implemented as a program.

*E-mail: fredrik.ronquist@nrm.se
†F.R., J.K., and V.S. contributed equally to this work.
[a]`http://webppl.org`
[b]Functional programming (FP) is a programming paradigm, in which code is structured in units called *functions* that have no side effects; i.e. they only operate on a given input and produce an output but do not manipulate external objects.

One of the earliest PPLs is Church[7], which extends a functional subset of the Scheme programming language. Other PPLs (both universal and non-universal) include Figaro[8] (a PPL embedded in Scala), WebPPL[9] (a recent PPL embedded into JavaScript), Anglican[10] (a general-purpose PPL embedded into Clojure that runs on the Java virtual machine), Venture[11] (a PPL with syntax similar to JavaScript), Edward[12] (a Python library for probabilistic modelling), Pyro[13] (a PPL built on top of PyTorch), Birch[14] (a PPL that compiles into C++), and Stan[15] (a platform for statistical modelling and computation). Note that this list is far from complete, and there exist many more experimental PPLs.

In this paper, WebPPL and Birch have been used for the reason of simplicity and efficiency, respectively. Some of the authors of this paper are currently developing a new domain-specific probabilistic programming language on top of the Miking[16] platform. This language, called TreePPL, is designed specifically for the domain of statistical phylogenetics.

In the rest of this section, we describe the key concepts of probabilistic programming. The examples are given in WebPPL, but could easily be translated into any of the other universal PPLs. The WebPPL code can be run in the web browser using the following web page: `http://webppl.org`. For a more comprehensive introduction to probabilistic programming, see for example the introductory text by van de Meent et al.[17].

1.2 Sampling

The first key construct in probabilistic programs is *sample*, meaning that a value is drawn from a given probability distribution. Consider the following WebPPL code:

```
sample(Bernoulli({p: 0.5}))
```

The program models a simple coin flip scenario, where we sample from the Bernoulli distribution with probability 0.5, that is, a fair coin. When executed, the program returns either `true` or `false`, with probabilities corresponding to the sampled distribution.

Suppose we instead introduce another random variable x that models the probability of getting heads on the toss of the coin. Mathematically, such a model can be defined as follows:

$$x \sim \text{Beta}(\alpha, \beta)$$
$$y \sim \text{Bernoulli}(x)$$

where the beta distribution is used as a prior probability distribution for the value of x. The same model can be written as a PPL program (assuming we set $\alpha = \beta = 2$)

```
var x = sample(Beta({a: 2, b: 2}))
var y = sample(Bernoulli({p: x}))
```

Note that the `sample` construct is conceptually used to denote random variables, in this case the two variables `x` and `y`. If we run the program many times and plot the values of `x`, we get an approximation of the probability density function (PDF) for x in our mathematical model, that is, an approximation of the Beta(2, 2) distribution. Because the expected value of x is 0.5, `y` in the program still models an unbiased coin.

1.3 Conditioning and observations

Probabilistic programs are based on Bayesian statistics, and typically are intended to compute the posterior distribution, given a prior distribution and some observations. In the coin flip example, suppose we observe heads (encoded as `true`) after flipping the coin once. We want to infer $p(x|y)$, the posterior distribution of x, conditioned on the new observation $y =$ `true`. As in the previous example, we assume that the prior distribution of x is Beta(2, 2). This model can be defined as follows.

```
var coinFlip = function() {
    var x = sample(Beta({a: 2, b: 2}))
    observe(Bernoulli({p: x}), true)
    return x
}
```

Note how the second `sample` construct is replaced with an `observe` construct. The program returns `x`, which is a (weighted) sample from the posterior distribution of x, computed by updating the prior distribution of x (defined explicitly in the model) with the observation that the coin flip resulted in `true` (conditioning on the observation). Fig 1(a) shows the prior distribution of the bias x, which corresponds to the Beta(2, 2) distribution. Note how the posterior distribution in Fig 1(b) has moved closer to 1 (bias towards heads), compared to the prior distribution.

The `observe` statement is a way of weighting a sample according to some distribution. It is basically equivalent to sampling from a distribution, followed by conditioning, using the `condition` statement covered in the main text[c]. In WebPPL, there is also a construct called `factor`, which performs explicit weighting (also called scoring) of samples. A

[c] According to the WebPPL documentation, for efficiency, the `observe` statement should be used instead of the combination of sampling and conditioning, especially for continuous distributions.

score or weight for a specific value can be computed using a distribution's PDF. In WebPPL, there is a `score` method that gives back the score of a value for a specific distribution. Thus, the `observe` statement in the previous example could be replaced with the following equivalent encoding using `factor`:

```
factor(Bernoulli({p: x}).score(true))
```

The `factor` construct is often used explicitly in the phylogenetic models described in this paper, especially in WebPPL.

1.4 Recursive models and stochastic branching

In the previous examples, the models were very simple. However, the power of universal probabilistic programming is that a model can be *any* Turing-complete program. Consider the following program that includes an `if` statement:

```
var mixture = function(){
  if (sample(Bernoulli({p: 0.7}))) {
    return sample(Gaussian({mu: -2, sigma: 1}))
  } else {
    return sample(Gaussian({mu: 3, sigma: 1}))
  }
}
```

The model illustrates the use of *stochastic branching*, meaning that the paths taken in a program depend on the outcome of sampling. In the example, the guard of the `if` statement samples from the Bernoulli distribution. Depending on whether the `true` or `false` branch is taken, sampling of the resulting value is done with different Gaussian distributions (different μ values). The plot of the model is shown in Fig 1(c). As can be seen in the figure, the `true` branch has larger weight, because of the probability of 0.7 of it being chosen.

Stochastic branching can be combined with recursion: this is a key building block for phylogenetic models. Consider the following model, which describes a model of the geometric distribution:

```
var geometric = function(p) {
    if (sample(Bernoulli({p: p}))) {
        return 1
    } else {
        return geometric(p) + 1
    }
}
```

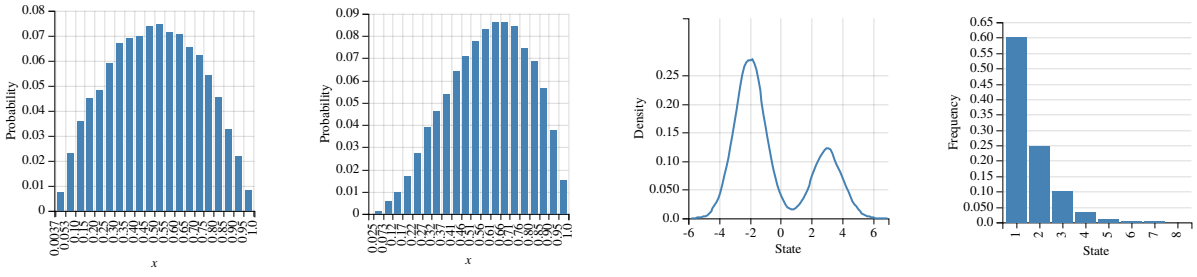
Note that there is no requirement of a deterministic termination of the recursion: the termination of the recursion depends on the stochastic branch. Fig 1(d) shows the plot of `geometric(0.6)`. The simple recursion above generates a linear sequence of random length. We use similar recursions in our scripts to model the processes that generate bifurcating phylogenetic trees. We do that by including two recursive calls within the same function, one for each descendant of a speciation event.

1.5 Inference

The focus of this tutorial text has so far been on the model (the probabilistic program), and not on the inference algorithms. As discussed previously, in probabilistic programming, the choice of inference algorithm is intentionally separated from the model. For instance, using the `Infer` method of WebPPL, a user can apply the Sequential Monte Carlo (SMC) method to perform the inference of the coin example

```
Infer({model: coinFlip, method: 'SMC', particles: 20000})
```

or, alternatively, a Markov chain Monte Carlo (MCMC) method can be used:



(a) (b) (c) (d)

Figure 1: (a) Prior distribution of the bias for the coin flip example. (b) Posterior distribution of the bias after observing heads once. (c) Mixture model of two Gaussian models mixed using stochastic branching. (d) Resulting geometric distribution with $p = 0.6$. All plots are generated using the WebPPL environment.

```
Infer({model: coinFlip, method: 'MCMC', samples: 20000, burn: 5000})
```

The user also needs to specify the granularity of the approximation, using the number of particles or samples for SMC or MCMC, respectively.

In general, a key strength of the probabilistic programming paradigm is its expressive power, which is clearly shown in this paper within the domain of phylogenetics. One of the main research challenges within the PPL community is how to develop inference algorithms and compilers that scale to very large and complex models. Although this is an active area of research, our study hopefully demonstrates that already state-of-the-art PPL systems make it possible to perform effective inference on non-trivial phylogenetic models.

2 Tools for phylogenetic probabilistic programming

The paper is accompanied by a code repository containing all the sources, tools and data used for the study, including documentation. Specifically, the resources in the repository are designed to facilitate the use of two existing probabilistic programming languages—WebPPL and Birch—for phylogenetic inference. The code repository is available at:

`https://github.com/phyppl/probabilistic-programming`

The reader is referred to the `README.md` file in the repository, in which we describe how to install the tools and how to use them to rerun our analyses or to experiment with probabilistic programming for phylogenetic problems.

2.1 WebPPL for statistical phylogenetics

WebPPL is a universal probabilistic programming language based on JavaScript. We have written two packages, `phywppl` and `phyjs` that enable the reader to run phylogenetic simulations in WebPPL. The verification of the WebPPL programs relies on an auxiliary R package, `rppl`. Please refer to the aforementioned online documentation for further explanation.

2.2 Birch for statistical phylogenetics

Birch is a universal probabilistic programming language compiling into C++. The models presented in this paper are run like regular Birch packages. Refer to the aforementioned online documentation for further explanation.

2.3 Reading in phylogenetic trees

The WebPPL and Birch scripts we provide either simulate the diversification process along an observed reconstructed tree or computes the likelihood using analytical equations for such a tree. To facilitate the import of the observed tree data, we use a new JSON format for phylogenetic trees named PhyJSON[18]. Supported by the resources we provide in the repository, both WebPPL and Birch have mechanisms for reading in phylogenetic trees stored in the PhyJSON format. We also provide a stand-alone tool, `nexus2phyjson`, which can be used to convert trees in Nexus tree files to PhyJSON format[18].

For convenience, we include several phylogenetic trees in the `phyjs` package for purposes such as testing and verification (Table 1). An up-to-date account of the included test trees is provided in the `webppl/phyjs/README.md` file.

Tree	Leaves	Age (Ma)	Description	Reference
`phyjs.bisse_32`	32	13.0	Example tree from Mesquite software	[19]
`phyjs.cetacean_87`	87	35.9	Cetacean tree from BAMMtools package	[20]
`phyjs.primates_233`	233	65.1	Primate tree from Diversitree package	[21]

Table 1: Example trees provided in the `phyjs` package.

3 Diversification models

3.1 Basic notation and terminology

All of the diversification models considered in this study can be generically described as follows (see Table 2 for a summary of the notation). The process starts at some time $t_0 > 0$ in the past, where $t = 0$ represents the present time. Evolutionary lineages split (speciate) at a per-lineage rate λ and go extinct at per-lineage rate μ. The rates λ and μ are either constant, time-dependent or lineage-dependent, depending on the specific model. Each speciation event produces

two lineages that further evolve independently of each other. The process is stopped when reaching the present ($t = 0$), at which point lineages still surviving are sampled (included in the observed tree) with probability $\rho \leq 1$.

The diversification process generates both trees with lineages that survive until the present, and trees that go completely extinct. Many surviving trees include side branches or whole subtrees that went extinct along the way. If the extinct parts and the branches leading to unsampled taxa are pruned away, we get what is called the "reconstructed tree". In other words, the reconstructed tree is the subtree spanned by only those surviving lineages that have been sampled.

In diversification analyses, the focus is typically on reconstructed trees. For simplicity, we assume here that the reconstructed tree is known without error, but we note that it is straightforward to extend our probabilistic programming approach to accommodate uncertainty about the tree by drawing the tree from an appropriate tree sample. Learning the parameters of the diversification process involves computing the likelihood of one or more reconstructed trees given different parameter values.

We denote a reconstructed tree $\psi = (V, t)$, where V is a set of nodes (vertices) and t is a corresponding vector of speciation ages. The tree has n tips (terminal nodes or leaves) of degree one, $n - 1$ interior nodes of degree three, and the origin node of degree one. We index the nodes and their ages as follows:

- the origin has index 0;
- internal nodes have indices $\{1, 2, \ldots, n - 1\}$, ordered in decreasing age;
- tips have indices $\{n, n + 1, \ldots, 2n - 1\}$ (in any order)

The node V_1 corresponds to the first split between extant (surviving) lineages; it is referred to as *the most recent common ancestor* (MRCA) or *the root* of the reconstructed tree. The age of a node i is t_i; leaves have age 0. A subtree with root at node i and origin at time $t \geq t_i$ is denoted $\psi_i(t)$.

We will often find it convenient to distinguish between the two descendants of a node; without loss of generality, refer to them as the left and right descendant, respectively. A tree without leaf labels where nodes have been oriented in this way is an "oriented tree"[22].

We define three mapping functions for indices in an oriented tree:

- $a(i)$ is the index of the immediate ancestor of node i
- $l(i)$ is the index of the left descendant of node i

Table 2: Summary of notation.

Symbol	Interpretation
λ	speciation (birth) rate
μ	extinction (death) rate
ϵ	turnover, μ/λ
ρ	probability of sampling a leaf
$\lambda(t)$	function characterizing time dependence of λ in some models
λ_0	initial λ, when λ varies over time
z	rate of exponential increase or decrease in λ
λ^i	speciation rate of process or branch i
$\lambda^i(t)$	time-dependent speciation rate of process i
μ^i	extinction rate of process i
z^i	exponential rate of increase or decrease in λ for process i
α	long-term trend in λ inheritance at speciation in ClaDS models
σ	standard deviation (log scale) in λ inheritance at speciation in ClaDS models
η	rate of switching of diversification process in the BAMM and LSBDS models
t_{MRCA}	age of most recent common ancestor
ψ	reconstructed tree (extinct and unsampled side branches pruned away)
n	number of leaves in the reconstructed tree
V	the set of nodes (vertices) in the reconstructed tree
$a(i)$	index of immediate ancestor of node i
$l(i)$	index of left descendant of node i in oriented tree
$r(i)$	index of right descendant of node i in oriented tree
c	number of cherries (terminal bifurcations) in a tree
$P(\cdot)$	probability (density)
$L(\cdot)$	likelihood
$S(t, \theta)$	probability of process with parameters θ surviving from t until the present
Z	normalization constant of Bayes's theorem

- $r(i)$ is the index of the right descendant of node i

3.2 Conversions between tree spaces

In phylogenetics, we are interested in computing the likelihood of a labelled reconstructed tree, that is, a tree with leaf labels but with no distinction between the two descendants of a given ancestor. However, it is often convenient to derive the probability density of oriented trees without leaf labels first, and then convert it to a density on labelled trees without orientation[22]. The conversion factor is easy to find if we consider what happens if we start with a density on an oriented tree, then label it and finally remove the orientation. There are $n!$ unique ways of labelling an oriented tree, each with probability $1/n!$. When we remove the orientation, there are 2^{n-1} labelled oriented trees that produce the same labelled tree without orientation, where $n-1$ is the number of interior nodes in the tree. Thus, the conversion factor is $2^{n-1}/n!$.

For completeness, we derive the conversion factor with the operations in the reverse order, first dropping the orientation and then applying the labels. When labels are missing, there are 2^{n-1-c} unique oriented trees for each tree without orientation, where c is the number of "cherries". A cherry is a pair of leaves that are each other's closest relatives[23]; without labels the descendants of a cherry are identical and there is only one unique way in which they can be oriented. Labelling a tree without orientation is similarly affected by cherries, so that there are $n!\,2^{-c}$ unique label assignments. The conversion factor is thus $2^{n-1-c}/(n!\,2^{-c}) = 2^{n-1}/n!$.

In the literature on advanced diversification models, it is common practice to derive the density on unlabelled oriented trees and ignore the conversion to a density on labelled unoriented trees; in fact, the omission of this factor is rarely acknowledged. This contrasts with the derivation of the analytical likelihood for simple models, such as CRBD, where the conversion factor is almost always accommodated. Previous work on diversification models has focused on a single model and a single tree; in such cases, ignoring the conversion factor is not a problem. However, here we compare diversification models using Bayes factors, so the normalization constant needs to be computed consistently for all models, that is, based on the same outcome space.

For convenience, our simulations of diversification processes assume unlabeled and oriented trees. This makes the scripts simpler, and it facilitates comparison to previous descriptions of these models. Our simulations are weighted with the appropriate conversion factor to generate the density for labelled and unoriented trees. Thus, the normalization constants we compute are directly comparable to the likelihoods computed using the standard analytical equations established for the simple diversification models, such as CRBD[22].

3.3 Conditioning on the age of the MRCA

A process that starts at some time t_0 in the remote past will produce a reconstructed tree that has a *stalk*, i.e., a branch leading to the MRCA. However, we usually do not have any information about the length of this stalk. For this reason, and others, it is often more convenient in practice to condition the process on the first split in the reconstructed tree, t_1[22]. This can easily be done by noting that t_1 can be considered the time of origin for both the left and the right subtrees originating from the first split, and that both of these lineages survived until the present by the very definition of the concept of MRCA. Thus, the probability of the reconstructed tree, conditioned on the age of the MRCA, is obtained by multiplying together the probabilities of the left and the right subtrees and by conditioning on their joint survival. Given an oriented tree ψ, now without the "stalk" from the origin to the most recent common ancestor, the likelihood is thus given by

$$L(\psi \mid \theta, t_1) = \frac{P(\psi_{l(1)}(t_1)|\theta, t_1)P(\psi_{r(1)}(t_1)|\theta, t_1)}{(S(t_1, \theta))^2},$$

where θ is the vector of parameters of the model, and $S(t, \theta)$ is the probability of the process surviving (producing at least one sampled descendant) after time t.

3.4 Diversification models

In the paper, we consider nine different diversification models (Table 3).

The CRB(D) and TDB(D) models are simple diversification models, which assume that the process is the same for the entire tree, even though it can change over time. The other models (the advanced models) accommodate lineage-specific variation in diversification rates. The BAMM and LSBDS assume that the diversification process changes in a major way at certain points in time. In fact, the process is completely reset. Thus, LSBDS and BAMM can be described as models of punctuated change in diversification. The ClaDS models instead assume gradual, heritable changes in speciation and extinction rates. Specifically, this is modeled as small stepwise changes associated with speciation events (also called cladogenetic events).

We provide a tabular summary of the parameters of each model (Table 4) to facilitate comparison across them. We describe each model in detail below.

3.4.1 Constant rate birth-death models (CRBD and CRB)

The constant rate birth-death (CRBD) model is the simplest model considered here. Evolutionary lineages split at a constant per-lineage birth rate λ and go extinct at a constant per-lineage death rate μ. The parameter vector for this model is thus $\theta = (\lambda, \mu, \rho)$. As a special case, we consider the constant rate birth (CRB) model, also known as the Yule model, with $\mu = 0$[25].

The probability that a CRBD process starting at time t survives until the present and is sampled is known analytically[22]; it is

$$S(t, \lambda, \mu) = \frac{r}{\lambda - (\lambda - r/\rho)\, e^{-rt}},$$

where $r = \lambda - \mu$ is known as the "net diversification rate". The likelihood of a reconstructed tree conditioned on the time of the MRCA is also known analytically; it is given by:

$$L(\psi|\theta, t_1) = \frac{2^{n-1}}{n!}\, \lambda^{n-2}\, \rho^n\, \frac{\widehat{g}(t_1)^2 \prod_{i=2}^{n-1} \widehat{g}(t_i)}{\widehat{g}(0)^n\, S(t_1)^2},$$

where

$$\widehat{g}(t) = \frac{e^{-rt}}{(\lambda - (\lambda - r/\rho)\, e^{-rt})^2}.$$

Even though the likelihood is known analytically, there are no conjugate priors for λ and μ that would yield an analytical posterior. Thus, we end up with an intractable integral if we want to learn these parameters from one or more observed trees.

3.4.2 Time-dependent birth-death models (TDBD and TDB)

In the time-dependent birth-death model (TDBD), the speciation rate is assumed to change continuously through time. More specifically, we consider the following time-dependence:

$$\lambda(t) = \lambda_0\, e^{z(t_1 - t)}.$$

Thus, λ_0 is the speciation rate prevailing at the time of the MRCA. Furthermore, if $z < 0$ (resp. $z > 0$), the speciation rate decreases (resp. increases) exponentially when going toward the present. An exponentially decreasing speciation rate can be seen as an approximate model for diversity dependence. The parameter vector for this model is $\theta = (\lambda_0, \mu_0, x, \rho)$.

Table 3: Overview of phylogenetic diversification models considered in the paper.

Model	Full name	Reference
CRB	Constant rate birth model	Yule[24], Nee[25]
CRBD	Constant rate birth-death model	Feller[26]
TDB	Time-dependent birth model	Kendall[27]
TDBD	Time-dependent birth-death model	Kendall[27]
BAMM	Bayesian analysis of macro-evolutionary mixtures	Rabosky[28]
LSBDS	Lineage-specific birth-death shift model	Höhna et al.[29]
ClaDS[0-2]	Cladogenetic diversification rate shift models	Maliet et al.[30]

Table 4: Summary of diversification model parameters.

Model	Parameters	Notes
CRB	λ	λ is speciation rate
CRBD	λ, ϵ	$\epsilon = \mu/\lambda$ is turnover rate
TDB	$\lambda(t), z$	z is exponential time-dependence parameter
TDBD	$\lambda(t), \epsilon, z$	
LSBDS	$\eta, \{(\lambda^i, \mu^i)\}$	η is change rate, i is index of process
BAMM	$\eta, \{(\lambda^i, \mu^i, z^i)\}$	z^i is time-dependence parameter of process i
ClaDS0	$\alpha, \sigma, \{\lambda^i\}$	α is trend parameter, σ is noise parameter in λ inheritance at speciation; i is branch index in complete tree
ClaDS1	$\alpha, \sigma, \mu, \{\lambda^i\}$	μ is extinction rate
ClaDS2	$\alpha, \sigma, \epsilon, \{\lambda^i\}$	ϵ is turnover rate

The likelihood appears to be intractable for the present model with exponentially varying speciation rate and constant extinction rate. On the other hand, a simple solution is available for the slightly different model examined here, in which λ and μ are both exponentially decreasing or increasing at the same rate z (and thus the turnover rate λ/μ is constant):

$$\lambda(t) = \lambda_0 \, e^{z(t_1 - t)},$$
$$\mu(t) = \mu_0 \, e^{z(t_1 - t)}.$$

Under this model, the probability that a lineage starting at time t survives until the present and is sampled is now (for a general method of deriving the likelihood for time-dependent birth-death models, see Yang [31]):

$$S(t, \lambda_0, \mu_0, z) = \frac{r_0}{\lambda_0 - (\lambda_0 - r_0/\rho) \, e^{-(r_0/z)(1 - e^{-zt})}},$$

where $r_0 = \lambda_0 - \mu_0$. The likelihood of a reconstructed tree conditioned on the time of the MRCA has the same general form as for the CRBD:

$$L(\psi|\theta, t_1) = \frac{2^{n-1}}{n!} \, \rho^n \, \frac{\widehat{g}(t_1)^2 \prod_{i=2}^{n-1} \widehat{g}(t_i) \, \lambda(t_i)}{\widehat{g}(0)^n \, S(t_1)^2}$$

with S such as just given and:

$$\widehat{g}(t) = \frac{e^{-(r_0/z)\left(1 - e^{-zt}\right)}}{\left(\lambda_0 - (\lambda_0 - r_0/\rho) \, e^{-(r_0/z)(1 - e^{-zt})}\right)^2}.$$

As a special case, we consider the time-dependent birth (TDB) model, which is equivalent to TDBD except that there is no extinction, that is, $\mu = 0$. Note that the TDBD model collapses to CRBD when $z = 0$. Similarly, TDB becomes equivalent to CRB when $z = 0$.

3.4.3 Bayesian analysis of macroevolutionary mixtures (BAMM)

The BAMM model was proposed by Rabosky [28]. The original formulation of the change process is statistically incoherent [32] but it is straightforward to fix this, and we follow the slight reinterpretation of the model suggested by Moore et al. [32]. In this version, BAMM is an episodic, Poisson-modulated, birth-death process with exponentially decaying speciation rate. To describe the process, consider a generic lineage e at time t. At this time point, the lineage is associated with rate parameters with index $e(t)$. Specifically, the lineage carries with it a triplet of rates $(\lambda^e(t), \mu^e(t), z^e(t))$. Then:

- at rate $\mu^e(t)$, the lineage goes extinct;
- at rate $\lambda^e(t)$, the lineage splits into two lineages (say f and g), in which case the two daughter lineages inherit the current rates, i.e.

$$(\lambda^f(t), \mu^f(t), z^f(t)) = (\lambda^e(t), \mu^e(t), z^e(t)),$$
$$(\lambda^g(t), \mu^g(t), z^g(t)) = (\lambda^e(t), \mu^e(t), z^e(t));$$

- $\lambda^e(t)$ increases or decays exponentially at rate z;
- at rate η, the triplet of rate parameters is redrawn from a pre-specified trivariate distribution Φ, i.e.

$$(\lambda^e(t_-), \mu^e(t_-), z^e(t_-)) \sim \Phi$$

The process starts with a single lineage a at some time t_0, with rate parameters $(\lambda^a(t_0), \mu^a(t_0), z^a(t_0)) \sim \Phi$. Specifically, we start the process at the time immediately before the first split in the tree (at the MRCA), and we assume that the process index at this point is $a(t_{\text{MRCA}}) = o$ (o for origin). The prior distributions used in this paper for Φ were chosen to harmonize with the priors used for other models, as specified in the section on priors below. The process stops when reaching the present ($t = 0$) and the surviving lineages are sampled with probability ρ.

The likelihood under the BAMM model does not have an analytical solution, nor does it seem to be amenable to any known numerical techniques for solving the complex differential equations involved [32]. However, describing the model using a PPL is straightforward, and effective inference can be performed using more generic techniques available for PPLs, such as SMC or PMCMC (particle Markov chain Monte Carlo), as we show in the current paper.

3.4.4 LSBDS

The recently proposed LSBDS model[29] can be seen as a specialized version of the BAMM model, in which $z = 0$ at all times and for all lineages. In other words, there is no exponential decay of speciation rates; the speciation rate remains constant between rate shift events. As a result, Φ is now a bivariate distribution.

Under these conditions, it becomes possible to compute the likelihood of a reconstructed tree by approximating Φ as a product of two discrete distributions, with a finite (and small) number of possible values for λ and μ, and then relying on standard numerical techniques for solving the differential equations involved[33,29]. This is similar to the discretization approach frequently used in phylogenetics in order to efficiently approximate the likelihood when rates vary across sites according to a continuous gamma distribution[34].

Using this approach, the backward-in-time recursion for the extinction probability and for the conditional likelihood, which are both conditioned on the current value of λ and μ for the lineage under consideration, entails a set of LM coupled master equations, where L and M are the number of bins used for the λ and μ distributions, respectively. In practice, this imposes a rather strict constraint on the number of discretization bins that can be used, as the computational complexity otherwise becomes unmanageable. We note that the empirical examples discussed in the LSBDS paper all use a fixed value for μ across the tree, thus effectively setting $L = 1$. The SMC techniques we use in the current paper do not suffer from such limitations, as they rely on sampling values from the λ and μ distributions.

3.4.5 The cladogenetic diversification rate shift models (ClaDS)

The ClaDS models[30] assume that the speciation rate changes by a small random amount at each speciation event. The extinction rate is assumed to be either equal to 0 (ClaDS0), constant but positive (ClaDS1), or proportional to the speciation rate, such that the turnover rate ($\epsilon = \mu/\lambda$) is constant (ClaDS2). Thus, in all cases, $\mu = \mu(\lambda)$ can be seen as a (possibly constant, for ClaDS0 and ClaDS1) deterministic function of λ.

Consider a generic lineage e at time t. This lineage carries with it a rate λ^e. Then:

- at rate $\mu(\lambda^e)$, the lineage goes extinct;
- at rate λ^e, the lineage splits into two lineages (say f and g), in which case the two daughter lineages draw their respective speciation rates, λ^f and λ^g as follows:

$$\log \lambda^f \sim \mathcal{N}(\log(\alpha\lambda^e), \sigma^2).$$
$$\log \lambda^g \sim \mathcal{N}(\log(\alpha\lambda^e), \sigma^2).$$

The process starts with a single lineage o at some time t_0, with speciation rate λ^o.

The α parameter introduces a deterministic long-term trend in the otherwise random variation of λ through time, across the many speciation events typically occurring over the complete phylogeny. When $\alpha < 1$ (resp. $\alpha > 1$), the speciation rate decreases (resp. increases) exponentially on average, thus corresponding to $z < 0$ (resp. $z > 0$) in the case of the TDBD, TDB and BAMM models.

The likelihood under the ClaDS1 and ClaDS2 models is not analytically available, but it can be numerically evaluated (Maliet et al, 2019). The evaluation involves various numerical approximation techniques, including discretization of time and rate space, and expands over thousands of lines of code in the RPANDA R package.

4 Prior probability distributions

To facilitate the interpretation of the Bayes factor tests, we standardized prior probability distributions across diversification models as much as possible in our analyses. Before going into details, it may be helpful to explicitly declare the parameterizations we assume for the statistical distributions used. Thus, for the exponential distribution, we assume the rate parameterization, for the inverse gamma distribution we use the shape-scale parameterization, and finally, for the normal (Gaussian) distribution, the parameters are the mean and the variance of the distribution.

Across all models, we used an Exponential(1) prior for the speciation rate, and a Uniform(0, 1) prior for the turnover rate, both common priors in the diversification model literature. The specific implementations are listed for each model below. For the σ parameter of the ClaDS models, Maliet et al.[30] used a prior with most probability mass close to 0 ($\sigma \sim \text{InvGamma}(1, \log(1.1))$. Upon examination of the empirical results published in the same paper, we concluded that this choice is overly conservative. We also note that it is more natural to consider an inverse gamma prior for the variance rather than the standard deviation of the normal distribution, since this is a conjugate prior for the normal distribution. Therefore, we used a $\sigma^2 \sim \text{InvGamma}(1, 0.2)$ prior in our analyses.

The original ClaDS paper[30] used an improper prior for the α parameter. This is not suitable for our purposes, as we need to simulate from the prior in SMC. We instead assumed $\log \alpha \sim \mathcal{N}(0, \sigma^2)$. By making the variance of the $\log \alpha$ prior dependent on σ^2, we establish a conjugate normal-inverse-gamma prior. This results in a joint prior on $(\log \alpha, \sigma^2)$ that has its mode for α at 0, at which point there is neither acceleration nor deceleration of speciation rates. The posterior

distribution of α values reported earlier for the bird trees under the Clads2 model[30] is also well covered by this joint prior. For z, we used the prior proposed in the original BAMM paper[28], namely $z \sim \mathcal{N}(0, 0.05^2)$.

Finally, for the LSBDS and BAMM models, we wanted a prior on η that was scaled to time. In the LSBDS and BAMM papers[29,28,35], it has been common to instead specify a prior scaled to the total length of the tree. This allows one, for instance, to specify a prior with an expectation of one change in the diversification process over the reconstructed tree. However, if the changes we observe in diversification rates are the result of some evolutionary process, then it would seem more reasonable to assume that the expected number of changes is a function of evolutionary time rather than of an arbitrarily circumscribed reconstructed tree. To obtain this effect, while still maintaining some scaling to tree size, we chose a prior with one expected change in diversification rates over the time period from the most recent common ancestor to the present. For a small reconstructed tree, this would correspond to an expectation of slightly more than one change over the tree, while the expectation could be more than a few changes in a big tree. Specifically, we assumed $\eta \sim \text{Exponential}(t_{\text{MRCA}})$, where t_{MRCA} is the age of the first split in the tree.

For completeness, all prior probability distributions are listed below for each of the examined models (see also Fig. 2).

4.1 CRB

The CRB model has only one parameter, λ, for which we use the standard prior:

$$\lambda \sim \text{Exponential}(1).$$

4.2 CRBD

The CRBD model has two parameters, λ and μ. For λ we use the standard prior, and for μ the indirect prior induced by assuming a uniform prior on the turnover rate $\epsilon = \mu/\lambda$.

$$\begin{aligned}\lambda &\sim \text{Exponential}(1),\\ \epsilon &\sim \text{Uniform}(0, 1).\end{aligned}$$

4.3 TDB

For the TDB model, we applied the standard priors as follows:

$$\begin{aligned}\lambda_0 &\sim \text{Exponential}(1),\\ z &\sim \mathcal{N}(0, 0.05^2),\end{aligned}$$

where λ_0 is the initial speciation rate, and z is the time dependence parameter in $\lambda(t) = \lambda_0 e^{zt}$. In other words, the standard λ prior applies to the initial speciation rate in this model.

4.4 TDBD

The TDB priors are extended to the TDBD case as follows:

$$\begin{aligned}\lambda_0 &\sim \text{Exponential}(1),\\ z &\sim \mathcal{N}(0, 0.05^2),\\ \epsilon &\sim \text{Uniform}(0, 1),\end{aligned}$$

where λ_0 is the initial speciation rate, z is the time dependence parameter in $\lambda(t) = \lambda_0 e^{zt}$, and ϵ is the turnover rate. Note that in our implementation we have kept the turnover rate constant (rather than the extinction rate), i.e., $\mu(t) = \epsilon\lambda(t)$.

4.5 ClaDS0

For the ClaDS0 model, we applied the standard λ prior to the initial speciation rate, in line with the TDB(D) models. The α and σ priors are justified above. Specifically, the ClaDS0 priors we used are:

$$\begin{aligned}\lambda_0 &\sim \text{Exponential}(1),\\ \sigma^2 &\sim \text{InvGamma}(1, 0.2),\\ \log\alpha &\sim \mathcal{N}(0, \sigma^2),\end{aligned}$$

where λ_0 is the initial speciation rate, σ^2 represents the variance in the inherited speciation rate and α is the *speciation trend parameter*.

4.6 ClaDS1

The prior distributions related to the speciation rates are the same as for ClaDS0. In addition, we assume

$$\epsilon \sim \text{Uniform}(0, 1),$$

where ϵ is the initial turnover rate. The extinction rate, $\mu = \epsilon\lambda_0$, remains constant in the whole tree.

4.7 ClaDS2

The prior distributions related to the speciation rates are the same as for ClaDS0. In addition, we assume

$$\epsilon \sim \text{Uniform}(0, 1),$$

where ϵ is the turnover rate. Unlike ClaDS1, the extinction rate changes at each speciation such that the turnover remains constant over the whole tree.

4.8 LSBDS

For the LSBDS model, we define the joint prior Φ, such that the all λ^i values, including the speciation rate for the MRCA (λ^o), are drawn from the standard λ prior used for other models, and such that the μ^i values, including the value of the MRCA, are drawn independently from the distribution induced by drawing ϵ from the standard uniform distribution used for other models. Specifically,

$$\begin{aligned}\eta &\sim \text{Exponential}(t_{\text{MRCA}}),\\ \lambda^i &\sim \text{Exponential}(1),\\ \epsilon^i &\sim \text{Uniform}(0, 1),\end{aligned}$$

where η is the rate of shifts in diversification processes, t_{MRCA} is the time (age) of the most recent common ancestor, and λ^i and ϵ^i are the speciation and the turnover rates of the i-th diversification process.

BAMM

The prior distributions for BAMM are the same as for LSBDS, with addition of

$$z^i \sim \mathcal{N}(0, 0.05^2),$$

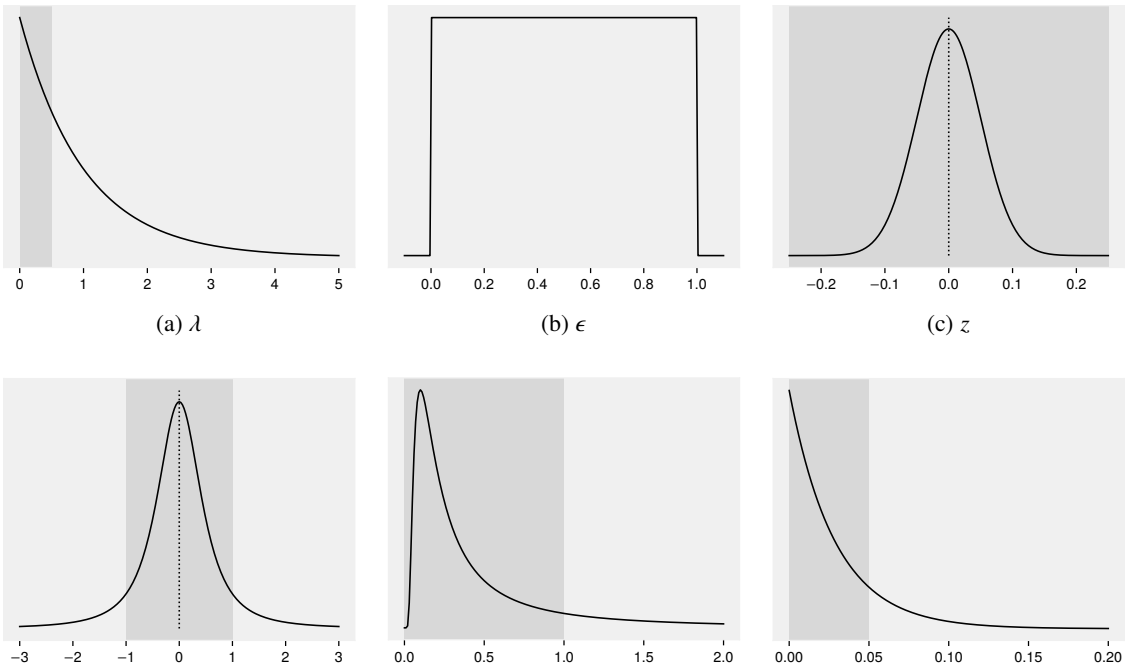
where z^i is the time dependence parameter for the speciation rate of the i-th diversification process. This is the same prior distribution used for the z parameter of the TDB and TDBD models.

5 PPL model descriptions

In this section, we describe the PPL model scripts we used in the paper. We focus on WebPPL, as we think these model scripts are the most accessible to biologists. We start by describing model scripts that make use of the analytical likelihood equations. We then present the complete description of the explicit simulation script for the CRBD model that is partly covered in the main paper. Finally, we provide a brief overview of the simulation scripts for the remaining models. We end the section with a brief discussion of how the Birch scripts are similar to and how they differ from the WebPPL scripts. For full details, we refer the interested reader to the code repository accompanying the paper.

5.1 Scripts based on analytical likelihoods

As mentioned in Section 3, the likelihood of a reconstructed tree conditioned on the age of the MRCA and the parameters of the diversification process is known analytically for the simple diversification models (CRB, CRBD, TDB, TDBD). We can take advantage of this in probabilistic programs, facilitating efficient inference of model parameters, by simply scoring simulations according to the analytical likelihood. To simplify the implementation of such scripts, we provide the analytical likelihoods as deterministic functions in the `phyjs` library. Four functions are available. The function `exactCRBDLikelihoodComplete(tree, lambda, mu)` computes the likelihood of a reconstructed tree under the CRBD model for specific values of λ and μ, assuming complete sampling of the leaves (tips) in the tree, $\rho = 1$. The function `exactCRBDLikelihoodRandom(tree, lambda, mu, rho)` computes the same likelihood when the leaves are randomly sampled with probability $\rho < 1$. Finally, the functions `exactTDBDLikelihoodComplete`



(a) λ (b) ϵ (c) z

(d) $\log \alpha$ (e) σ^2 (f) η

Figure 2: Prior distributions of model parameters. The shaded regions correspond to the region of parameter space illustrated in the posterior plots for the empirical analyses (Figs. 12–21). See also Fig. 22.

`(tree, lambda, mu, z)` and `exactTDBDLikelihoodRandom (tree, lambda, mu, z, rho)` compute the corresponding probabilities for the TDBD model. By setting `mu = 0`, the functions can be used to compute the likelihoods for the CRB and TDB models.

The following listing shows how to infer the posterior distribution of λ and ϵ for the CRBD model using the analytical likelihood and the MCMC inference method:

Listing 1: CRBD model with analytical likelihood.

```
var tree = phyjs.bisse_32

var model = function() {
    var lambda = exponential({ a: 1 })
    var epsilon = uniform({ a:0.0, b: 1.0 })
    var mu = epsilon*lambda

    factor( exactCRBDLikelihoodComplete(tree, lambda, mu) )

    return [lambda, epsilon]
}

var dist = Infer({method: 'MCMC', samples: 100, lag: 10, burn: 1000})

dist
```

In the script, we first select one of the provided trees in the `phyjs` package. The model is then set up by specifying the priors on the model parameters, and computing the value of the extinction rate μ, encoded as the variable `mu`. The simulation then simply scores the simulation according to the analytical likelihood of the sampled parameter values using the `factor` construct. In the final line of the `model` function, the values of the model parameters are returned.

For inferring the posterior distribution induced by the model function, the MCMC method is a good choice. For explanation of the inference settings, see the WebPPL documentation of the MCMC method[d]. The last line ensures that the estimated joint posterior distribution, encoded as `dist`, is printed.

The script can be run using the commands we provide in the code repository accompanying this paper[e], as explained in the documentation provided there. In the directory `webppl/phywppl/examples/` in the repository, we provide analytical scripts of this kind for the CRB, CRBD, TDB and TDBD models.

[d] `http://docs.webppl.org/en/master/inference/methods.html#mcmc`
[e] `https://github.com/phyppl/probabilistic-programming`

5.2 Basic script for CRBD

Here, we give a complete WebPPL implementation of the CRBD model. The program describing the model is divided into two files to facilitate reuse of the code. The simulation part is specified in one file, and the analysis part in another. The simulation file contains code that simulates the CRBD process along a given tree for specified values of the model parameters. This file can be reused unaltered regardless of the particular analysis one wants to perform. The analysis file contains the specification of the priors, the data, and the inference method. This file needs to change from one analysis to another.

The analysis file (Listing 2) is structured in the same way as the script using the analytical likelihood for CRBD. However, instead of calling a function to compute the analytical likelihood, we call the simulation function for the CRBD model. This function weights the simulation appropriately for the given parameter values, conditioned on the observed reconstructed tree. The model function returns the model parameters, as before. However, instead of inferring the posterior distribution on those parameters, we now use SMC and focus on the normalization constant (the model evidence). The normalization constant estimate is available in the `normalizationConstant` property of the distribution object returned by the `Infer` function when the method is SMC. Similar example scripts are available in the `webppl/phywppl/examples/` directory for all models studied in the paper.

Listing 2: Analysis script for CRBD simulation.

```
var tree = phyjs.read_phyjson("bisse_32.phyjson")

var model = function() {
    var lambda = exponential({ a: 1 })
    var epsilon = uniform({ a:0.0, b:1.0 })
    var mu = epsilon*lambda

    simCRBDNaive( tree, lambda, mu)

    return [lambda, epsilon]
}

var dist = Infer({method: 'SMC', particles: 10000})

dist.normalizationConstant
```

Let us now turn to the simulation script (Listing 3). The script presented here is a naive PPL implementation of the CRBD model in that it does not use the analytical likelihood. Instead, it explicitly simulates the speciation and extinction process conditioned on the reconstructed tree. The script is also naive in the sense that it does not include any modifications to support aligned SMC inference, which is important for improving inference efficiency. The advanced inference techniques we used in the paper, including alignment, are discussed in Section 6. The script forms a basic template that can be used to express all diversification models analyzed in our paper. It should also be straightforward to extend the script to a range of new diversification models that have not been explored previously.

Listing 3: A complete WebPPL script for simulating CRBD.

```
var goesExtinct = function( startTime, lambda, mu )
{
    var t = exponential( {a: lambda + mu} );

    var currentTime = startTime - t;

    if ( currentTime < 0 ) {
           return false
    }

    var speciation = flip( lambda/(lambda+mu) )
    if ( !speciation )
       return true;

    return( crbdGoesExtinct( currentTime, lambda, mu )
           && crbdGoesExtinct( currentTime, lambda, mu ) );
}

var simBranch = function( startTime, stopTime, lambda, mu )
{
    var t = exponential ( {a: lambda} );

    var currentTime = startTime - t;

    if ( currentTime <= stopTime )
```

```
        return 0.0;

    factor( Math.log( 2.0 ) );
    condition ( crbdGoesExtinct( currentTime, lambda, mu ) )

    return simBranch( currentTime, stopTime, lambda, mu )
}

var simTree = function( tree, parent, lambda, mu )
{
    factor( - mu * ( parent.age - tree.age ) );

    simBranch( parent.age, tree.age, lambda, mu );

    if ( tree.type == 'node')
    {
        factor( Math.log( lambda ) );

        simTree( tree.left, tree, lambda, mu )
        simTree( tree.right, tree, lambda, mu )

    }
}

var simCRBDNaive = function( tree, lambda, mu )
{
    var numLeaves = phyjs.countLeaves( tree )
    var corrFactor = ( numLeaves - 1 ) * Math.log( 2.0 ) - phyjs.lnFactorial( numLeaves )
    factor( corrFactor )

    simTree( tree.left, tree, lambda, mu )
    simTree( tree.right, tree, lambda, mu )
}
```

The main function in the script is `simCRBDNaive`, defined at the end of the script. It takes three parameters: the model parameters `lambda` and `mu`, and the `tree` on which to condition the simulation. For simplicity, the process is simulated along an oriented and unlabelled tree (see Section 3.2). This allows us to ignore the probability factor associated with rotation and labeling of the reconstructed tree during the main part of the simulation. To ensure that the simulation nevertheless carries the right weight, it is first endowed with the appropriate rotation and labeling probability (see Section 3.2) using two utility functions in the `phyjs` library and the `factor` construct in WebPPL. This is important for computing the correct normalizing constant, but does not affect inference otherwise, since this probability factor is the same for all simulations. Note that, for numerical stability, the particle weights in WebPPL are stored as logarithms.

Next, the function `simTree` is called on both children of the root node (the MRCA), initiating the recursion over the observed tree. Note that `simCRBDNaive` does not return anything. It is called only for the side-effect of weighting the sampled `lambda` and `mu` values by conditioning the simulation on the observed tree.

The function `simTree` is similar in structure to `simCRBDNaive`: it computes various weights and, if we have not reached a leaf, continues the recursion. The first probability factor or weight, `lnProb1`, accounts for the probability of no extinction along the current branch. The second weight, `lnProb2`, accounts for the speciation event at the end of the branch, if there is one. The third weight, `lnProb3`, is based on a call to the simulation function `simBranch`, defined just above `simTree`. The `simBranch` function recursively simulates speciation events along the branch. If there is a speciation event, the side branch it generates must go extinct, as it is not present in the observed reconstructed tree. We call such a speciation event a "hidden" speciation because it is not visible in the observed tree. To condition the simulation on the extinction of the side branch resulting from a hidden speciation, we require the call to the recursive simulation function `goesExtinct` to return true. The `goesExtinct` function is described in the main paper; it is defined at the top of the script presented here. It simply simulates an outcome of the birth-death process for given `lambda` and `mu` values, starting at a given time in the past and counting downwards until the present (time 0). If all lineages go extinct before reaching the present, the function returns `true`, otherwise it returns `false`. In connection with the call to `goesExtinct`, the `simBranch` function also needs to take a rotational factor into account. This arises because there are two indistinguishable simulations that correctly account for the tree we condition on: one in which the right descendant of the hidden speciation event goes extinct and the left descendant gives rise to the observed continuation of the lineage, and one in which the left descendant goes extinct and the right descendant gives rise to the observed continuation of the lineage. Thus, the correct probability score for the simulation is twice what would have resulted from a single call to `goesExtint`, and we therefore need to add log 2 to the weight (recall that probability factors are represented on the log scale in WebPPL) before continuing the recursion.

The analysis and simulation scripts described above are simplified versions of the example script `crbd-naive.wppl` in the `webppl/phywppl/examples/` directory, and the similarly named model script in the `webppl/phywppl/models/` directory. The simulation script presented here differs in four details from the model script in the repository. First, the script in the repository accommodates the possibility of incomplete sampling of the leaves in the tree. Thus, there is an

additional parameter ρ in the model, encoded as the variable `rho` in the script. This variable appears as an argument to all simulation functions. The `goesExtinct` function needs to take the sampling probability into account, and is aptly renamed to `goesUndetected`.

Second, the definitions of the `simTree`, `simBranch` and `goesUndetected` functions are hidden inside the `simCRBDNaive` function. This allows us to use the same generic names for these functions in all diversification models; only the simulation function needs to have a unique name. Hopefully, this facilitates for readers to recognize how we extended the basic template to accommodate the other diversification models.

Third, the script in the repository employs guards against extreme values of the `lambda` variable, which can otherwise cause problems with numerical exceptions or stalled simulations. We solve these problems by simply assigning zero weight to the simulation if the `lambda` value is above or below certain threshold values. We verified that the discarded simulations have negligible impact on the inference for all the examined models using the chosen guard values.

Finally, unlike the simple script described here, the script in the repository corrects for survivorship bias as explained in the next section. Before moving on to this, we want to point out that the naive CRBD simulation is suitable mainly for exploratory analyses of small trees. For efficient inference in WebPPL on phylogenetic diversification models for larger trees, it is important to manually modify the scripts so that they support aligned SMC inference (see Section 6.1). The CRBD model is the only model for which we provide an unaligned ("naive") model script.

5.3 Correcting for survivorship bias

As discussed above (Section 3.3), if we condition the simulation on the age of the MRCA, we implicitly condition on the survival of the two subtrees originating at this point in time. To do this in a probabilistic program, we need to divide the probability of a simulation by $S(t_{\text{MRCA}}, \theta)^2$, that is, the square of the probability that the process survives (and is sampled) if it starts at t_{MRCA}, and the model parameter values are θ. If $S(t, \theta)$ is not available in closed form, this is potentially cumbersome because it involves a sum and integral over an infinite number of realizations of the process for each simulation. However, we can solve this by observing that the division by $S(t_{\text{MRCA}}, \theta)^2$, which we cannot evaluate in general, can be rewritten as follows:

$$p(\theta|\psi, \text{survival}) \propto \frac{p(\theta)p(\psi|\theta)}{S(t_{\text{MRCA}}, \theta)^2} = p(\theta)p(\psi|\theta) \sum_{M=1}^{\infty} M \, (1 - S(t_{\text{MRCA}}, \theta)^2)^{M-1} \, S(t_{\text{MRCA}}, \theta)^2.$$

This shows that we can correct for the survivorship bias by using the generative model encoded in the function `goesExtinct` (or `goesUndetected`) to simulate two evolutionary processes starting at t_{MRCA}. We repeat this until both simulations survive to the present time, and multiply the weight of the rest of the simulated diversification process along the observed tree by the number of repetitions required to achieve this.

In WebPPL, we use the following recursive function to compute the number of simulations required until both trees survive:

```
var M_goesExtinct = function( t, lambda, mu )
{
   if ( !goesExtinct( t, lambda, mu ) && !goesExtinct( t, lambda, mu ) )
      return 1
   else
      return 1 + M_goesExtinct( t, lambda, mu )
}
```

The following lines are then inserted at the end of the simulation function `simCRBDNaive` to correctly condition on the survival of the two subtrees defining the MRCA:

```
var M = M_goesExtinct( t, lambda, mu )
factor( Math.log( M ) )
```

The script in the repository is slightly more complex because we take incomplete sampling into account, and also implement a guard against an excessive number of repetitions.

5.4 Scripts for other diversification models

Example analysis scripts for all models are provided in the directory `webppl/phywppl/examples/`, and generic model simulation scripts in the directory `webppl/phywppl/models/`. All simulation scripts we provide in the latter directory are set up to trigger aligned SMC inference in WebPPL. As mentioned above, the only exception is the CRBD model, where we provide both a naive, unaligned version (`phywppl/models/crbd-naive.wppl`) and an aligned version (`phywppl/models/crbd.wppl`) for instructional and testing purposes. We provide both scripts using analytical likelihoods and scripts using explicit simulation for all simple diversification models (CRB, CRBD, TDB, TDBD). The scripts for the CRB, TDB and TDBD models involve simple and straightforward modifications of the corresponding scripts for the CRBD model, described above.

The model scripts for all lineage-specific diversification models (ClaDS0, ClaDS1, ClaDS2, LSBDS, BAMM) follow the template described above for the CRBD model, including the modifications needed to trigger aligned SMC inference. Analogous component functions are used in the simulation scripts; they are even named the same except for the main simulation function, which is named after the corresponding diversification model.

In probabilistic programming, you have to be explicit about the model variables that you want to estimate. These are the variables that are returned from the model function. The focus in our study was on the normalization constant and the main model parameters. Therefore, our model simulation scripts do not have to return anything, as all relevant parameters are defined already in the analysis scripts in the `webppl/phywppl/examples/` folder. However, readers may well be interested in sampling the outcome of a diversification process along the tree. For instance, it may be desirable to analyze parameters such as the number and location of change events on different lineages, or the mean speciation rate for individual branches in the tree. To facilitate such analyses, we give an example model script for the ClaDS2 model returning the entire reconstructed tree, with descriptions of the outcome of the simulation process for each branch and node in the tree in extended Newick format. This script is found in the `webppl/phywpppl/models/` folder.

5.5 Birch model scripts

Birch is an object oriented probabilistic programming language. It uses more concise syntax than WebPPL for the probabilistic constructs. For example, the assume statement in the form

```
x ~ Exponential(1);
```

is used to express that a random variable (`x` in the example above) is distributed according to a given probability distribution (an exponential distribution with rate 1). Execution of such a statement depends on whether the variable has a value or not. If it has, its behavior is equivalent with an observe statement; if not, the variable is associated with the given distribution. Birch uses delayed sampling, so a concrete value might not be sampled until needed. Birch also supports explicit sample and observe statements. To draw a value from an exponential distribution with rate λ, we would write

```
t <~ Exponential(λ);
```

To state that an outcome of a random variable distributed according to a Poisson distribution with rate λ is 0, we would write

```
0 ~> Poisson(λ);
```

The factor statement in WebPPL corresponds to `yield FactorEvent(log_factor)`. To simplify the diversification model definitions, we have defined two helper functions for commonly used `yield` statements.

```
yield duple();
```

corresponds to

```
yield FactorEvent(log(2));
```

and is used to account for the rotational factor at hidden speciation events. Similarly,

```
yield impossible();
```

is the same as

```
yield FactorEvent(-inf);
```

and it is used when simulated side branches resulting from hidden speciation do not go extinct, that is, when they are incompatible with the observed tree. Note that `yield impossible()` statement also ceases the execution of the particle.

As we have mentioned above, Birch is an object-oriented language and the models take advantage of this. For instance, the CRBD model script defines a `CRBDModel` class, which is derived from a base class called `PhyModel`. Let us examine a somewhat simplified version of the `CRBDModel` class definition (Listing 4), to see how it compares to the WebPPL script.

Listing 4: CRBDModel class definition in Birch (somewhat simplified)

```
class CRBDModel < PhyModel<PhyNode, PhyParameter> {
  λ_k:Real;
  λ_θ:Real;
  ε_min:Real;
  ε_max:Real;
  ρ:Real;

  fiber initial() -> Event {
    super.initial();
    θ.λ ~ Gamma(λ_k, λ_θ);
    θ.ε ~ Uniform(ε_min, ε_max);
  }

```

```
  fiber step() -> Event {
    count:Random<Integer>; // number of (hidden) speciation events
    count ~ Poisson(θ.λ * (node.t_beg - node.t_end));
    for i in 1..Integer(count) {
      t:Random<Real>;
      t ~ Uniform(node.t_end, node.t_beg);
      simulateUnobserved(t);
      yield duple();
    }

    0 ~> Poisson(θ.λ * θ.ε * (node.t_beg - node.t_end));

    if node.isSpeciation() {
      0.0 ~> Exponential(θ.λ);
    }
  }

  fiber simulateUnobserved(t_beg:Real) -> Event {
    Δ_d:Random<Real>; // waiting time until an extinction event
    Δ_d ~ Exponential(θ.λ * θ.ε);
    t_d:Real <- t_beg - Δ_d;
    if t_d < 0 {
      // Species survived to the present time
      yield impossible();
    }

    count:Random<Integer>; // number of speciation events
    count ~ Poisson(θ.λ * (t_beg - t_d));
    for i in 1..Integer(count) {
      t:Random<Real>;
      t ~ Uniform(t_d, t_beg);
      simulateUnobserved(t);
    }
  }
}
```

The `CRBDModel` class is derived from the class `PhyModel`, which is a templated class. The base class takes care of tasks that are common to all diversification models, such as walking over the tree. This is analogous to the recursive calls in the `simTree` function in the WebPPL script (Listing 3), which also walk over the branches in the tree. At the top of the class definition, the member variables and their types are declared. These are the parameters of the prior distributions for the model variables λ and ϵ. The parameters are assigned specific values when the class is instantiated in connection with running the program. The inference settings and input values for the analyses are in the `config/crbd.json` file and in each of the `input/<name of tree>.json` input files.

Instead of member functions, the class defines several member fibers. A `fiber` (also known as a coroutine) is similar to a function, but the execution might be paused (e.g., to resample the particles) and resumed. The `initial` fiber simply initializes the simulation by assuming `lambda` and `epsilon` to be distributed according to the appropriate priors. Note that these model variables are packaged inside an object called θ.

The `step` fiber corresponds to the `simBranch` function in the WebPPL script. In Birch, we use a different method for simulating the speciation and extinction events than in WebPPL. Rather than drawing the waiting times between hidden speciation events, we use the fact that the number of hidden events is described by a Poisson distribution, and the event positions are uniformly distributed over the branch length. This simulation method is faster than drawing each of the waiting times. In the line

```
0 ~> Poisson(θ.λ * θ.ε * node.branch_length);
```

we condition on the fact that there are 0 extinction events on the branch (recall that $\mu = \lambda\epsilon$). In WebPPL, we used a `factor` statement with the appropriate probability instead, which is an alternative way of accomplishing the same thing. Finally, in the line

```
0.0 ~> Exponential(θ.λ);
```

we condition on there being a speciation at the end of the branch (if it ends in an interior node). Equivalently, we could have factored in $\log \lambda$, as we did in WebPPL, with a `yield` statement.

The `simulateUnobserved` fiber corresponds to the `goesExtinct` function in WebPPL. However, here we first simulate the time until the branch goes extinct. If the branch does not go extinct, we set the weight to zero, effectively killing off the simulation. If it does go extinct, we simulate the hidden speciation events along the branch, and call `simulateUnobserved` recursively for each of those events.

The code described above is subject to change, as Birch is developing rapidly. However, this section illustrates the basic Birch features, and how they can be used to code diversification models efficiently. Hopefully, it also sheds additional light on general PPL concepts, as it gives alternative but equivalent ways of coding some model elements compared to the WebPPL scripts we have seen previously.

6 Inference

In this section, we provide additional details on the non-standard algorithms we used to allow efficient PPL inference on phylogenetic diversification models.

6.1 Alignment

The encoding of the CRBD model given in Section 5.2 is rather natural—it is simply a description of the birth-death process, with a few calls to `factor` to correct for some probability effects that we do not model explicitly. Unfortunately, the default SMC algorithm implemented in WebPPL is quite inefficient for this naive implementation of the birth-death process. The algorithm *always* resamples particles (simulations are called *particles* in the SMC algorithm) at calls to `factor` and `condition`. Since, for every execution of the program, there is a different number of hidden speciation events on each branch in the observed tree, this will cause the SMC particles to get out of sync at resampling points, so that we will be comparing particles that can be at very different points in the simulation.

Intuitively, one might expect that it would be better to compare the particles only when they reach the same points in the probabilistic program. We call this *alignment* of the SMC resampling points. In the diversification models, we could, for instance, make sure that the resampling occurs only at the branching points in the observed tree. To explore this idea, we simply "tricked" the SMC algorithm in WebPPL to align the resampling points by introducing a few modifications to the birth-death simulation in the `simBranch` and `simTree` functions, as illustrated in the code below (compare to the naive CRBD simulation presented above):

```
var simBranch = function( startTime, stopTime, lambda, mu )
{
    var t = sample{ Exponential ( {a: lambda} ) };

    var currentTime = startTime - t;

    if ( currentTime <= stopTime )
        return 0.0;

    var sideExtinction = goesExtinct( currentTime, lambda, mu )
    if ( sideExtinction == false )
        return ( -Infinity );

    return simBranch( currentTime, stopTime, lambda, mu ) + Math.log( 2.0 );
}

var simTree = function( tree, parent, lambda, mu )
{
    var lnProb1 = - mu * ( parent.age - tree.age );

    var lnProb2 = ( tree.type == 'node' ? Math.log( lambda ) : 0 );

    var lnProb3 = simBranch( parent.age, tree.age, lambda, mu );

    factor( lnProb1 + lnProb2 + lnProb3 )

    if ( tree.type == 'node')
    {
        simTree( tree.left, tree, lambda, mu )
        simTree( tree.right, tree, lambda, mu )
    }
}
```

Specifically, we need the WebPPL SMC implementation to skip the resampling induced at the calls to `factor` and `condition` within `simBranch` in the naive model script. We achieve this by replacing the `factor` and `condition` statements in the `simBranch` function by code that accumulates the weight and returns it to `simTree`. The accumulated weight is then passed as an argument to `factor` in `simTree`, after the entire branch has been processed, triggering resampling at this point. The `factor` statement is also passed the probability of no extinction on the branch (`lnProb1`), and the likelihood of a speciation at the end of the branch, if it is an interior branch in the observed tree (`lnProb2`). Note that, to improve efficiency, we immediately return `-Infinity` in `simBranch` if a call to `goesExtinct` returns false, since there is no need to continue the recursion if this occurs. By modifying the simulation script in this way, the SMC particles stay in sync. There are no triggers of resampling in the `simBranch` recursion, so resampling is always performed in `simTree`, in between processing branches of the observed tree.

Simulations on a few example trees of varying sizes confirm that this indeed improves SMC efficiency on diversification models considerably (Fig. 3). The larger the tree, the more important it is for SMC performance to align the resampling points in this way. Ideally, one should not have to manipulate model scripts in the way described above; alignment should be applied automatically when it improves SMC efficiency. This is an idea that we are exploring within the TreePPL project. The goal is to analyze the potential performance gains induced by resampling, and then apply it intelligently

either in the compiler and/or the language runtime. We separately present a static analysis for automatic alignment of programs[36]. Note that alignment is not *guaranteed* to improve accuracy—in certain cases, it might actually degrade performance. However, for all models considered here, alignment is beneficial.

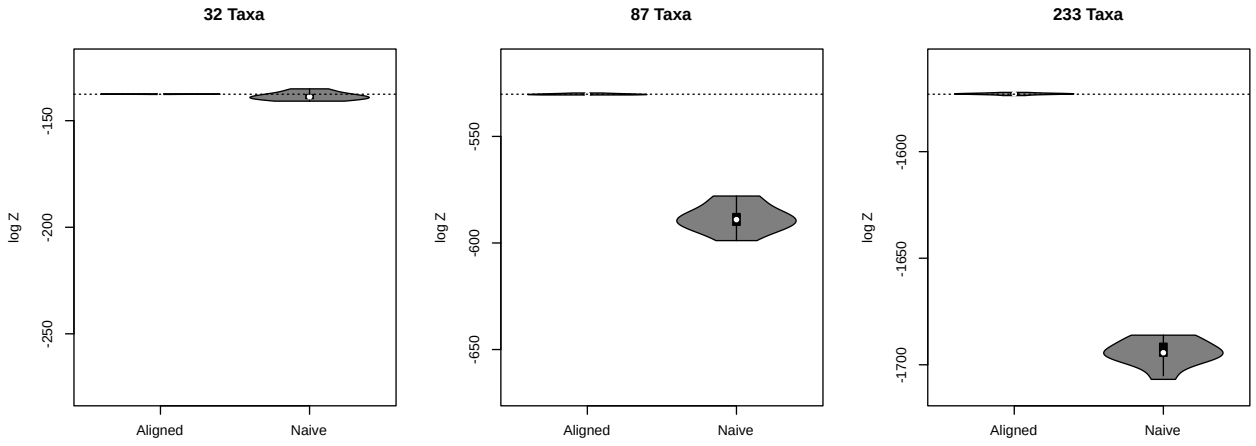


Figure 3: A comparison of the precision in the estimated normalization constant between naive and aligned CRBD. Left: 32-taxon tree (Bisse_32). Center: 87-taxon tree (Cetaceans_87). Right: 233-taxon tree (Primates_233). SMC inference with 10,000 particles in WebPPL. Dotted line: exact analytical solution. Parameters: $\lambda = 0.2$, $\epsilon = 0.5$, complete sampling of leaves assumed ($\rho = 1$).

6.2 Delayed sampling

Probabilistic computations involve not only simulation and observation, as represented by the `sample` and `observe` statements in a PPL, but also such computations as marginalization, enumeration, and conjugate updating.

Consider the following joint distribution between two variables x and λ:

$$p(x, \lambda) = p(x \mid \lambda)p(\lambda),$$

where the two factors on the right are encoded in the probabilistic program as, for example:

$$\lambda \sim \text{Gamma}(1, 1),$$
$$x \sim \text{Poisson}(\lambda).$$

We may wish to compute the marginal distribution of x:

$$p(x) = \int p(x \mid \lambda)p(\lambda)\,\mathrm{d}\lambda,$$

or, given a value of x, compute the posterior distribution over λ:

$$p(\lambda \mid x) = \frac{p(x \mid \lambda)p(\lambda)}{p(x)}.$$

Evaluations such as these can be performed analytically for random variables with a conjugate relationship (such as the gamma-Poisson relationship in the example above), or for discrete random variables where all possible outcomes can be enumerated. This can improve the performance of inference by, for example, reducing the variance in statistical estimators, such as that for the marginal likelihood.

Delayed sampling[6] is a particular heuristic that may be employed by a PPL to identify and leverage such situations to improve inference outcomes. It does so in a manner that produces correct results, even for programs with stochastic branches and unbounded recursion as may be encountered in Turing-complete programming languages. It is not necessary for the programmer to painstakingly code such computations by hand.

We have used delayed sampling extensively in this work, significantly reducing the variance in marginal likelihood estimates for the models of interest. In particular, for Poisson processes on trees, gamma prior distributions over rates are conjugate either to the Poisson-distributed number of events in a given time interval, or the exponentially-distributed time between events. These rate parameters are then automatically marginalized out by delayed sampling, significantly reducing variance in the marginal likelihood estimate for these models. The same approach to handling parameters is used in Kudlicka et al.[37] and Wigren et al.[38]. Delayed sampling is only available in Birch at this point.

6.3 Alive particle filter

The resampling step in SMC amounts to drawing N samples (with replacement) from the current set of N particles with probabilities proportional to their weights. While simulating the evolution of unobserved side branches, if any species survives to the present day (and is sampled), the weight of the particle must be set to 0. This leads to sample impoverishment—there are simply fewer particles to choose from during resampling. In extreme cases, where all particles have zero weight, there are no particles to choose from at all, and the algorithm fails. This can be a serious problem for SMC inference on diversification models when the likelihood of extinction of side branches is low. For instance, this can occur if the net diversification rate ($\lambda - \mu$) is high.

The extended alive particle filter[37]—the development of which from the initial version of this algorithm[39] was inspired by phylogenetic diversification models—solves these two problems by replacing the particles with zero weights with new samples drawn from the particle set at the previous time step (again with probabilities proportional to the particle weights at that time) and repeating the propagation step (the simulation from the previous resampling point until the current resampling point). This replacing procedure is repeated until the weights of all particles are positive. Note that in order to estimate the marginal likelihood without bias, one needs to repeat this procedure for one additional particle. However, with a reasonable number of particles, this extra computational cost is negligible, and we therefore applied the alive particle filter to all analyses. The alive particle filter is only available in Birch at this point.

6.4 Tree orientation

During the course of the study, we discovered that the orientation of the nodes in the observed tree can have a significant influence on the efficiency of SMC inference for some trees. The effect appears to be associated with highly imbalanced trees, which may be oriented such that left and right subtrees systematically have different properties. A depth-first SMC algorithm can apparently become misled by the imbalance between left and right descendants in such trees, so that early resampling events can select particles that do not do well towards the end of the simulation, decreasing the quality of the final estimate. We found that orienting all nodes such that the descendant branch with the shortest subtree length was always processed first solved this problem. Thus, all trees were reoriented in this way before final analyses in Birch.

7 Verification

We performed a wide range of experiments to verify that the model scripts are correct. For all tests involving WebPPL or third-party software, the full set of experiments—including the source code, data, graphs and reports—can be found in the directory `verification` of the `phywppl` package. The verification experiments involving Birch were performed by changing the input files and/or models to fix the values of selected parameters. The results from these experiments are included in the above-mentioned directory, together with the results from the experiments involving WebPPL.

Here, we only present a summary of the experiments. They all use the 32-taxon example tree, which we provide as one of the builtin trees in the `phyjs` package. The tree has been previously used as an example in diversification model papers; it is originally from the Mesquite software[19] but does not appear to have been published separately. Only scrips adapted for aligned SMC inference were used in the verification experiments.

The experiments are based on several lines of attack. In the first round of tests, we used the fact that there are analytical solutions for the likelihood of the simple diversification models (CRB, CRBD, TDB and TDBD) under specific parameter values. Thus, we could verify that the normalization constant computed by SMC from our explicit simulation scripts for the same models (the scripts that simulate the process along the tree instead of calling the likelihood function) matched the corresponding analytical likelihoods for a wide range of specific parameter values. These tests are important because the explicit simulation scripts for the simple models served as templates for the scripts describing the more complex models.

The second round of tests were based on the observation that the more complex diversification models (ClaDS0, ClaDS1, ClaDS2, LSBDS and BAMM) all collapse to simpler models with analytically known likelihoods under specific parameter settings. This allows us to verify that the normalization constant computed from the scripts for the complex models matched the corresponding analytical likelihoods for select points in parameter space.

For other points in parameter space, we cannot verify the scripts for the more complex models against analytical likelihoods, but we can use other approaches to test their correctness. For instance, the WebPPL and Birch scripts for the complex models were implemented independently by different developers, and the inference algorithms in WebPPL and Birch were also different and based on independent implementations. In the third round of tests, we verified that the WebPPL and Birch scripts for the complex models gave the same normalization constant for a grid of parameter values despite these differences.

The fourth round of tests took advantage of the independent implementations available in third-party software for the ClaDS models[40] and for LSBDS[29]. We verified that our scripts for these models resulted in the same estimates of the likelihood as these implementations for a select set of parameter values, despite being based on entirely different code bases and computational strategies.

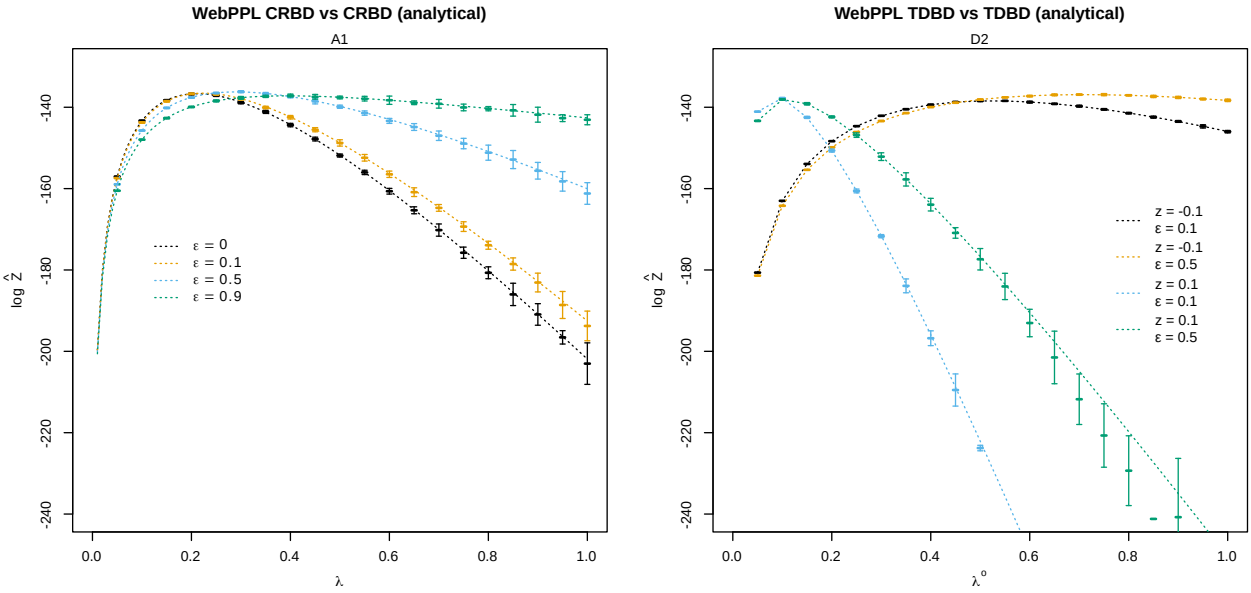


Figure 4: Verification of explicit simulation scripts (WebPPL) for simple diversification models: normalization constants match analytical likelihoods for select parameter values. Error bounds: ±2 standard deviations. Experiment codes in the GitHub repository indicated under the main title.

Third-party software also exists for BAMM[35] but it does not compute correct likelihoods for the model[32], so it cannot be used to verify our scripts. However, the BAMM model collapses to the LSBDS model when all $z^i = 0$. We therefore verified that our BAMM model script results in the same likelihood estimates as the LSBDS model script under select parameter values matching this constraint but lacking analytical solution.

7.1 Simple models against analytical likelihoods

All simulation scripts for simple models (CRB, CRBD, TDB and TDBD) generated normalization constant estimates that matched the corresponding analytical likelihoods very closely. We observed some variance in the estimates for high λ values, but these parameter values have low likelihood and are thus less important for inference (Figure 4).

7.2 Complex models against analytical likelihoods

All model scripts for advanced diversification models (ClaDS0, ClaDS1, ClaDS2, LSBDS and BAMM) generated normalization constant estimates that matched analytical likelihoods under parameter settings for which closed solutions exist (Figure 5).

7.3 Birch and WebPPL cross-verification

Under parameter and prior settings for which closed solutions do not exist, the independently developed Birch and WebPPL scripts for advanced diversification models resulted in matching normalization constant estimates. Birch estimates were slightly more precise than WebPPL estimates (Figure 6) but this is expected given the more powerful inference algorithms used by Birch.

7.4 Verification of ClaDS models against RPANDA

Verification of the probabilistic programs and PPL inference algorithms described in this paper against the reference RPANDA implementation of the ClaDS models is quite involved, and would not have been possible without extensive help from the author of the ClaDS code in RPANDA (Odile Maliet), as computing the likelihoods with RPANDA is not part of its public API. RPANDA computes the likelihood for points in parameter space where all the initial λ^i values for the branches in the reconstructed tree are known, as well as the λ^o value, pertaining to the MRCA of the tree. The likelihood in RPANDA is also conditioned on specific values of the model parameters α and σ, as well as on μ (for ClaDS1) or ϵ (for ClaDS2). The ClaDS0 likelihood function in RPANDA is based on analytical equations, while the ClaDS1 and ClaDS2 functions are based on numerical approximations using a variety of techniques. The functions only give the density up to a proportionality constant, complicating direct comparisons with our scripts.

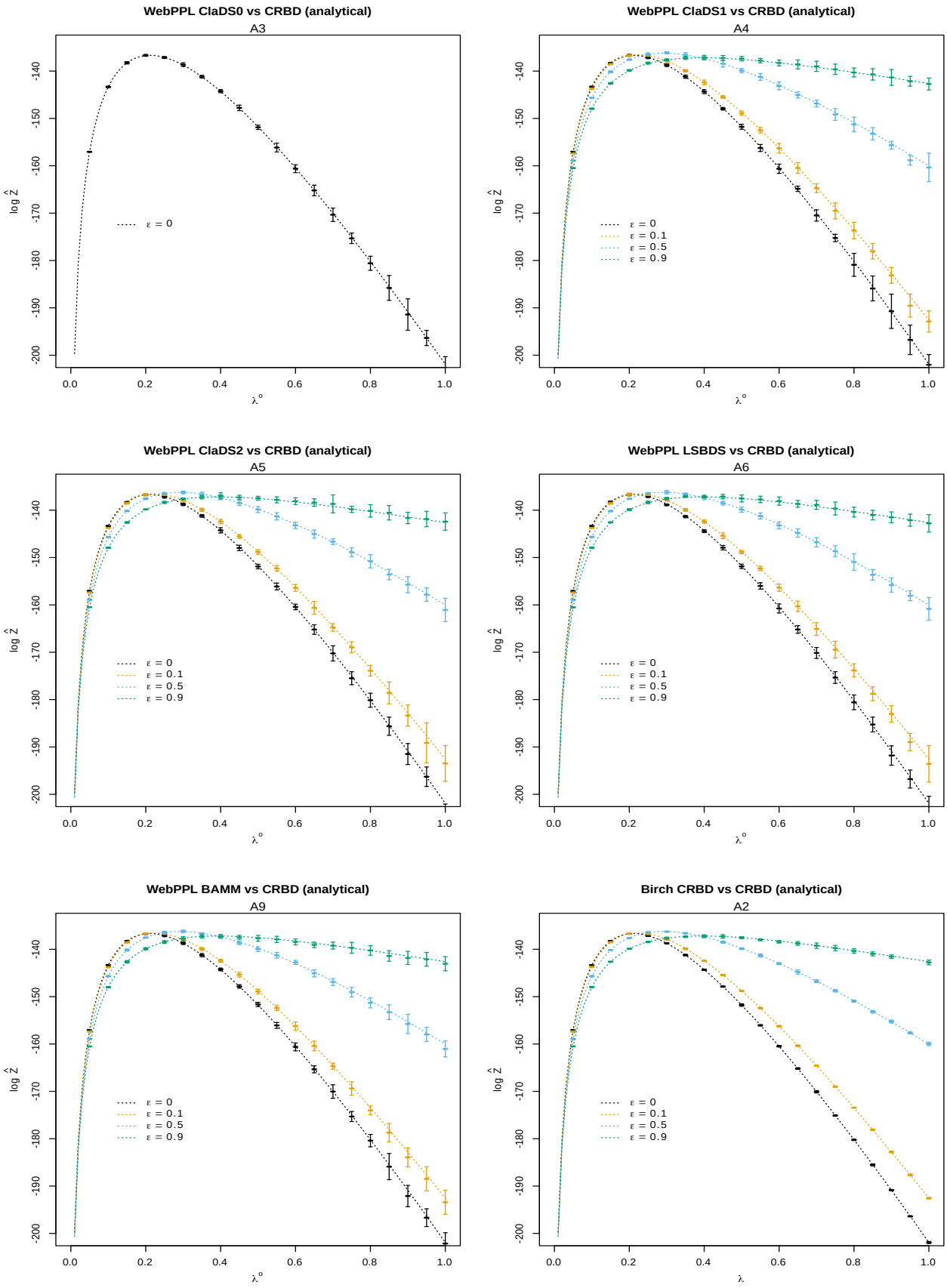


Figure 5: Verification of WebPPL simulation scripts for lineage-specific diversification models (complex models): normalization constants match analytical likelihoods for select parameter values. Error bounds: ±2 standard deviations. Experiment codes in the GitHub repository indicated under the main title. For completeness, we also give Birch CRBD (simulation) vs the analytical solution (last).

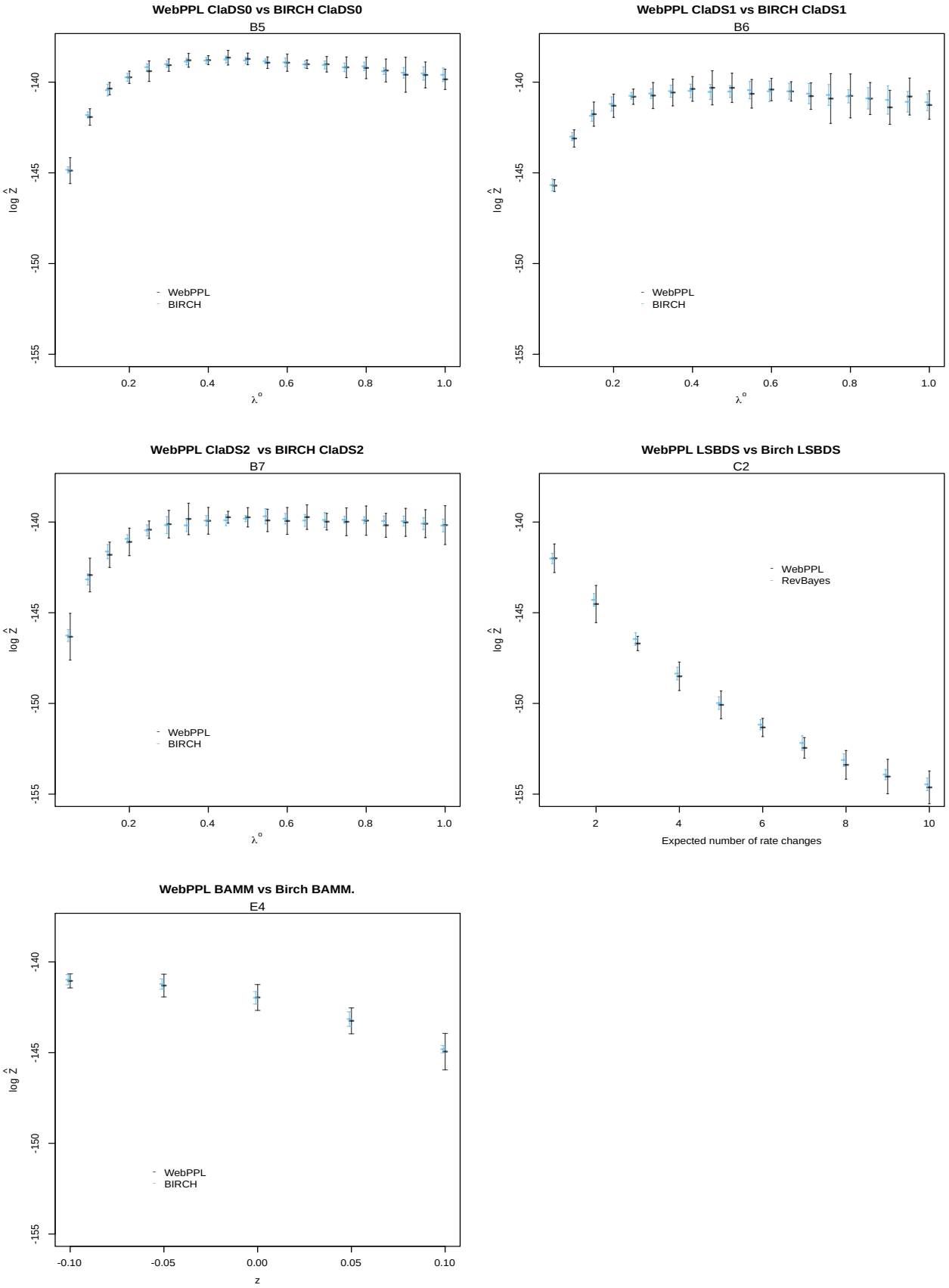


Figure 6: Verification that the WebPPL and Birch scripts for lineage-specific diversification models generate normalization constants that match each other for select parameter values.

The RPANDA setup means that the PPL scripts have to condition on specific values for all of the model parameters, including the initial λ values for all branches, to emulate the RPANDA likelihood computations. To be able to conduct the verification experiments, we decided to use a fixed value λ_f for λ^o and all λ^i parameters of the model in our WebPPL scripts; we did not attempt to perform these verification experiments in Birch. We then chose a range of λ_f values, and explored these points in parameter space under some specific values of α, σ and μ (for ClaDS1) and ϵ (for ClaDS2). Likelihoods for the same points in parameter space were then computed in RPANDA with the analytical likelihood function (for ClaDS0) and the numerical solvers (for ClaDS1 and ClaDS2). In the git repository accompanying the paper, we provide both the WebPPL scripts emulating the RPANDA computations and the R scripts we used to compute likelihoods for the corresponding points in parameter space with RPANDA.

For ClaDS0, the initial experiments showed that the likelihood function in RPANDA computes densities that very closely match the densities expected for oriented and unlabeled trees. Thus, we concluded that the proportionality constant for the ClaDS0 likelihood function in RPANDA is the same as the conversion factor from densities on oriented and unlabeled trees to densities on labeled, unoriented trees. This factor is $L_p = \log(2^{(n-1)}/n!)$, where n is the number of leaves in the tree (see Section 3.2).

When controlling for this, the likelihoods estimated by WebPPL for ClaDS0 are consistent with those computed by RPANDA (Fig. 7). For points in parameter space where ClaDS1 and ClaDS2 collapse to ClaDS0, that is, for points where $\mu = \epsilon = 0$, likelihoods estimated by WebPPL and RPANDA are also very similar. The same is true for small values of λ and μ) in ClaDS1, and for small values of λ and ϵ in ClaDS2. For larger values, RPANDA apparently overestimates the likelihood for both models, and there are also some apparent discretization effects at very high values of λ. We tried to examine the effects of these inaccuracies in RPANDA on the posterior estimates of the ClaDS1 and ClaDS2 model parameters for the test tree, but were unable to get sufficiently good MCMC convergence in RPANDA to allow meaningful analysis of the results.

7.5 Verification of LSBDS against RevBayes

In the current implementation of LSBDS in RevBayes (the SCM algorithm), the likelihoods are computed by discretizing the λ and μ priors. Transitions happen by "jumping" from one pair of discrete values of λ and μ to a different pair. We discovered that λ and μ are coupled when these jumps are made: i.e., the discrete vectors representing the prior distributions f_λ and f_μ have to be of the same length and, when a jump happens, a single new array index is chosen for both the λ and the μ vector. Thus, usually, the RevBayes LSBDS examples[f] fix λ but discretize μ (or vice-versa). However, it is possible to discretize both λ and μ and then expand the two arrays so that all possible combinations of λ and μ values appear by sweeping both vectors simultaneously with a single array index. This has to be done manually.

We verified our LSBDS scripts against RevBayes for specific values of η and integrated out $\lambda \sim \text{Exponential}(1)$ and $\mu = \epsilon\lambda$, where $\epsilon \sim \text{Uniform}(0, 1)$ using $k = 10$ rate categories for both λ and μ. We implemented the appropriate vector of parameter values manually, as described above. The RevBayes scripts used in the verification experiments are provided in the git repository accompanying the paper.

Under these settings, both the WebPPL and Birch scripts for the LSBDS model generate normalization constant estimates that match the likelihoods computed by RevBayes (Figure 8). As observed previously in several experiments, Birch provides slightly more precise estimates of the normalization constant than WebPPL.

7.6 Verification of BAMM against LSBDS

There is no third-party software implementing BAMM that we can verify the WebPPL and Birch scripts against. However, we can use the fact that BAMM collapses to LSBDS when all z^i values approach 0. Under these conditions, and when integrating out the other model parameters, both the WebPPL and Birch simulations scripts for BAMM produce the same normalization constants as the corresponding LSBDS scripts (Figure 9),

8 Empirical data

For the empirical analyses illustrating PPL inference for phylogenetic diversification models, we used the bird trees analyzed previously for the ClaDS2 model[30]. The trees originate from an earlier study inferring a global timed phylogeny of birds[41]. Specifically, clades with 50 or more leaves (excluding outgroups) from the earlier study were selected in the ClaDS2 study[30] and post-processed to remove outgroups and to rescale branch lengths to time units (myr). We downloaded these post-processed trees from the repository[g] accompanying the ClaDS2 paper.

The trees were converted from binary R data (RData) to text format (Nexus) with the `ape` package. The Nexus files were then converted to PhyJSON with the `nexus2phyjson` tool that we provide[18]. Next, the PhyJSON trees were reoriented to avoid any systematic left-right imbalances in the original trees that could have a negative effect on inference (see Section 6.4). The resulting PhyJSON trees were then used as input data for the WebPPL and Birch analyses.

[f] `https://github.com/hoehna/birth-death-shift-analyses`
[g] `https://github.com/OdileMaliet/ClaDS/tree/master/birds_MCC_results`

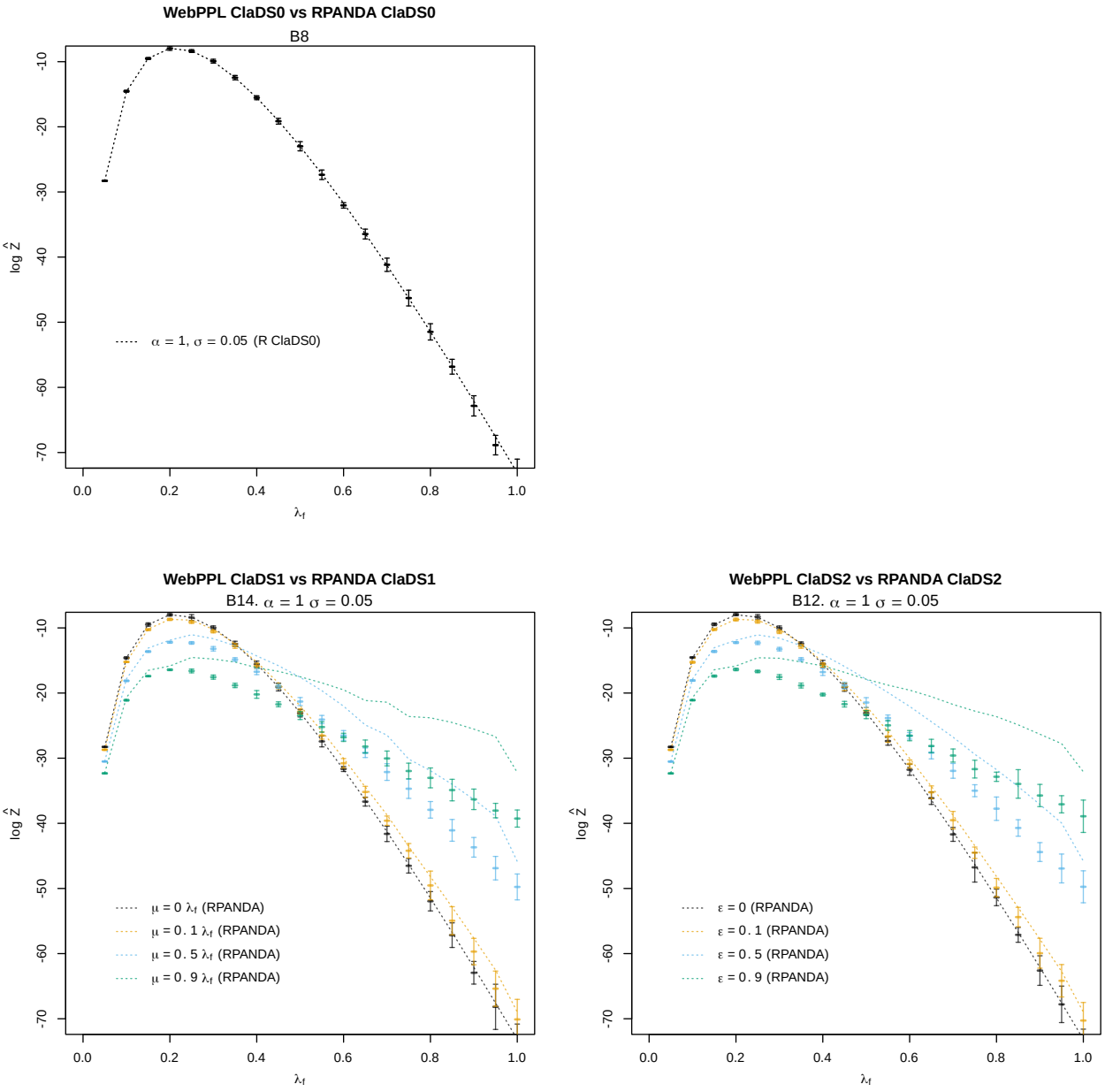


Figure 7: Verification of the likelihoods computed by WebPPL in programs emulating RPANDA against likelihoods computed by RPANDA for the ClaDS models.

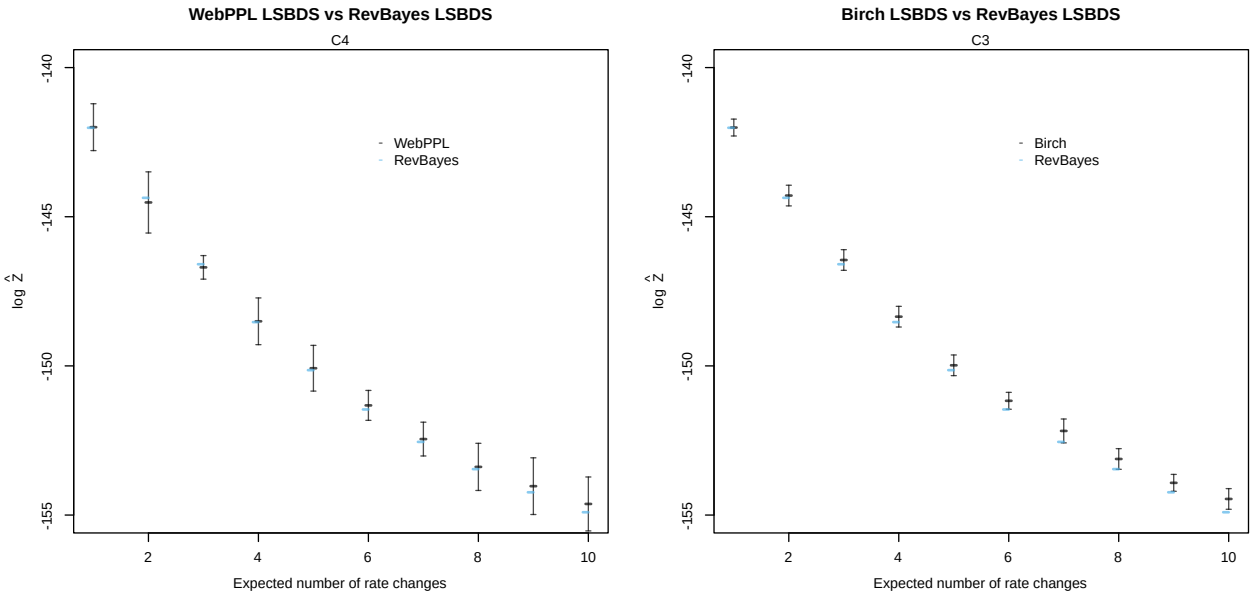


Figure 8: Verification that the WebPPL and Birch scripts for the LSBDS model generate normalization constant estimates that match the numerically estimated likelihood computed by RevBayes.

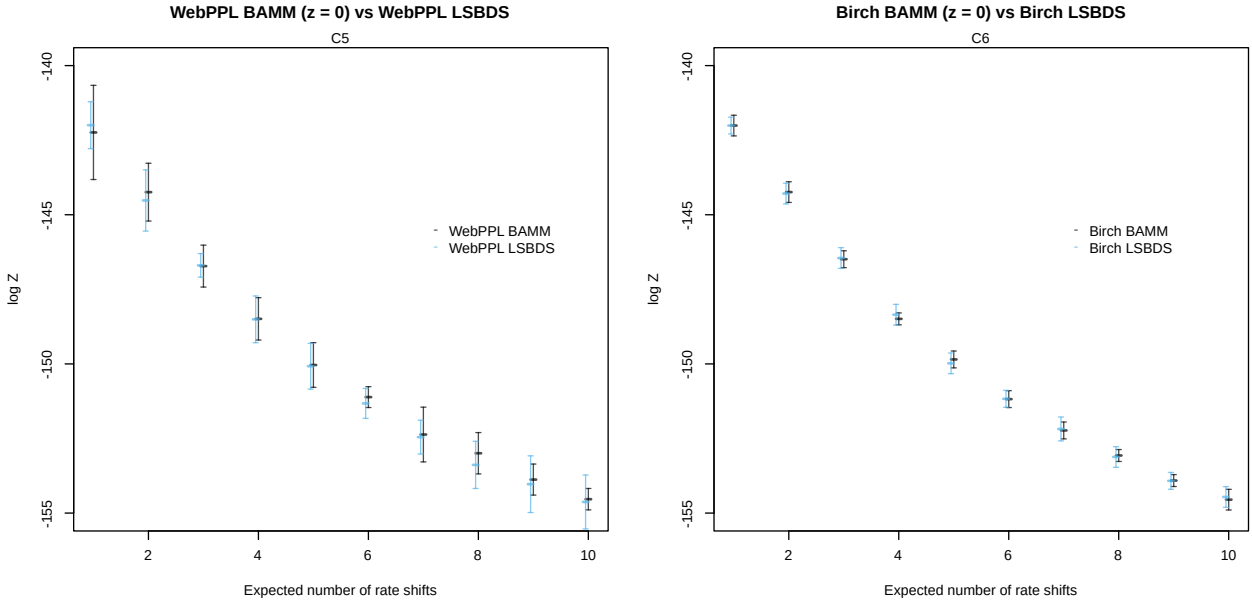


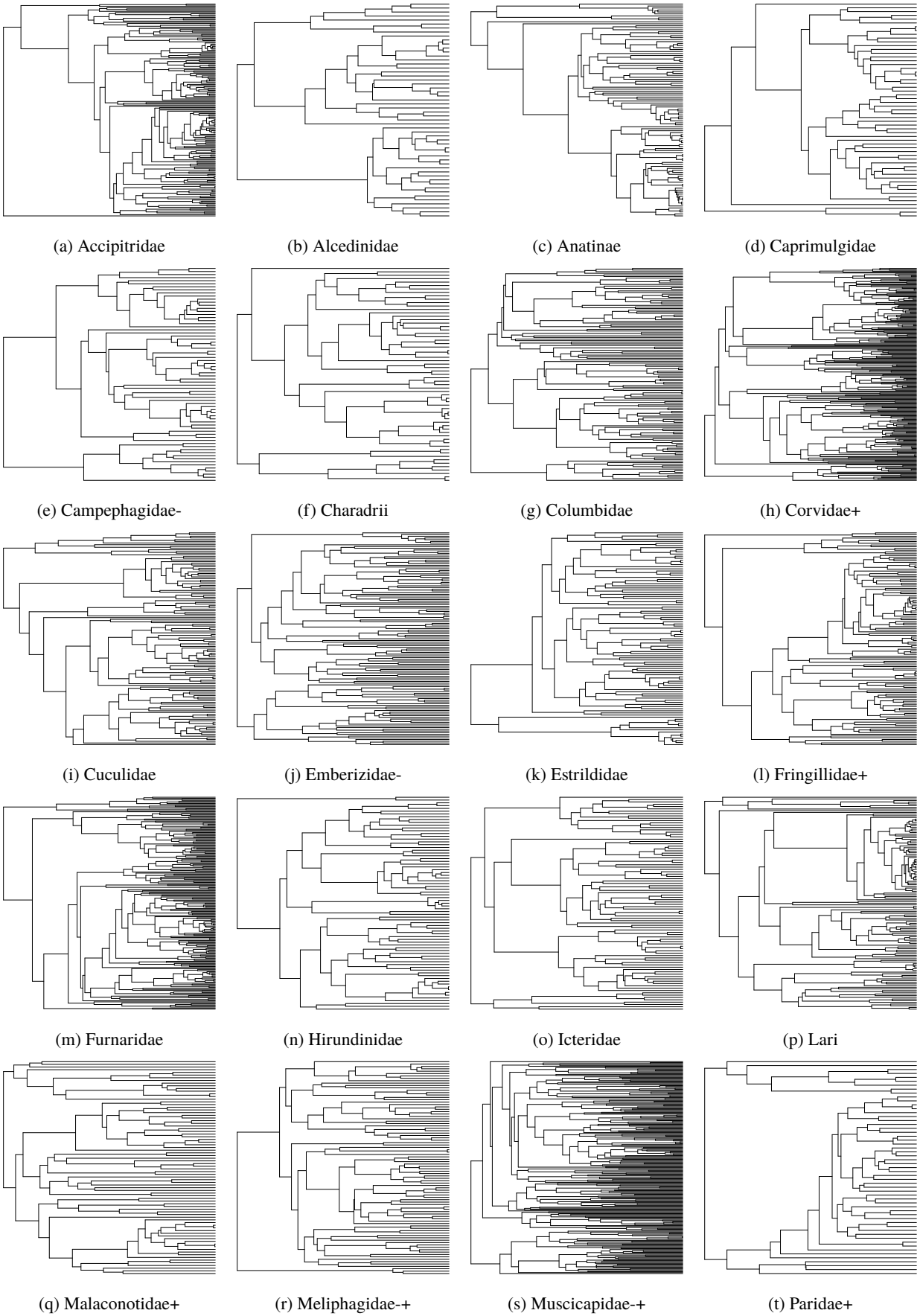
Figure 9: Verification that the WebPPL and Birch scripts for the BAMM model generate normalization constant estimates that match those of the corresponding scripts for the LSBDS model for some points in parameter space where the BAMM model collapses to LSBDS.

There are 42 bird clades in the Jetz et al.[41] study with more than 50 species excluding outgroups. However, we discovered that two of the trees, P2 and Scolopaci, have negative branch lengths. Rather than introducing arbitrary corrections for the negative branch lengths, we excluded these trees from further analysis. The remaining 40 bird trees are summarized in Table 5. The original names of the bird clades[41] are rather cryptic. Here, we named the clades after the family (or other higher taxon) to which most members belong according to the taxonomic classification used in the original bird study[41]. If a family is split between two clades, the clades are numbered 1 and 2. A '-' sign after the family name indicates that some members of the family are not included in the clade; a '+' sign indicates that the clade includes some members of other families. Four of the trees in the repository accompanying the ClaDS paper[30] are mislabeled there: Caprimulgidae is incorrectly labeled CC7, CC4 is labeled Cathartidae, CC7 is labeled CC5CC6B, and CC8 is labeled CC5CC6C.

Tree	Clade (Jetz et al)	Leaves	Age (Ma)	Notes
Accipitridae	Accipitridae	175	59.6	Hawks, eagles, kites and allies
Alcedinidae	Alcedinidae	54	34.9	Kingfishers
Anatinae	Anatinae	108	20.3	Dabbling ducks
Caprimulgidae	Caprimulgidae	57	57.3	Nightjars
Campephagidae-	CC4	70	30.1	Cuckooshrikes and allies
Charadrii	Charadrii	63	59.6	Waders
Columbidae	Columbidae	133	35.9	Pigeons and doves
Corvidae+	CC8	234	30.0	Crows, magpies, monarchs and allies
Cuculidae	Cuculidae	126	66.6	Cuckoos
Emberizidae-	P20b	125	14.8	Buntings
Estrildidae	P7	101	19.5	Estrildid finches
Fringillidae+	P10	123	25.4	True finches
Furnaridae	Furnaridae	205	19.9	Ovenbirds
Hirundinidae	S6	77	23.1	Swallows, martins and allies
Icteridae	P21	92	14.0	New World blackbirds, New World orioles and allies
Lari	Lari	127	24.6	Gulls
Malaconotidae+	CC7	80	31.4	Bushshrikes
Meliphagidae-+	BC7	90	37.1	Honeyeaters
Muscicapidae-+	M6	231	20.2	Old world flycatchers
Paridae+	S2	55	40.9	Tits
Parulidae+	P20a	111	17.2	New World warblers
Phasianidae	Phasianidae	131	27.2	Pheasants, partridges and allies
Picidae	Picidae	137	27.1	Woodpeckers
Procellariidae	Procellariidae	105	59.6	Shearwaters, fulmarine petrels and allies
Psittacidae1	Psittacidae1	111	33.2	True parrots (part)
Psittacidae2	Psittacidae2	118	34.9	True parrots (part)
Pycnonotidae-+	S9	95	29.4	Bulbuls
Ramphastidae	Ramphastidae	81	32.2	Toucans
Strigidae	Strigidae	101	45.7	True owls
Sturnidae+	M4	130	24.9	Starlings, mockingbirds and allies
Syvliidae1+	S11	79	28.1	Warblers, parrotbills and allies (part)
Sylviidae2+	S7S8	93	24.3	Warblers, parrotbills and allies (part)
Thamnophilidae	Thamnophilidae	165	22.4	Antbirds
Thraupidae1+	P13P14P16	158	12.5	Tanagers (part)
Thraupidae2+	P17P18	139	13.7	Tanagers (part)
Timaliidae-+	S13	180	21.0	Old World babblers
Trochilidae	Trochilidae	233	28.1	Hummingbirds
Troglodytidae+	M1	91	32.7	Wrens
Turdidae-+	M5	134	21.7	Thrushes
Tyrannidae+	Tittyranrest	316	33.6	Tyrant flycatchers

Table 5: Overview of the bird trees used for diversification analyses.

The size of the trees vary from 54 (Alcedinidae) to 316 leaves (Tyrannidae+), and the ages from 12.5 Ma (Thraupidae1+) to 66.6 Ma (Cuculidae). The tree shapes are depicted in Figures 10, 11.



(a) Accipitridae (b) Alcedinidae (c) Anatinae (d) Caprimulgidae

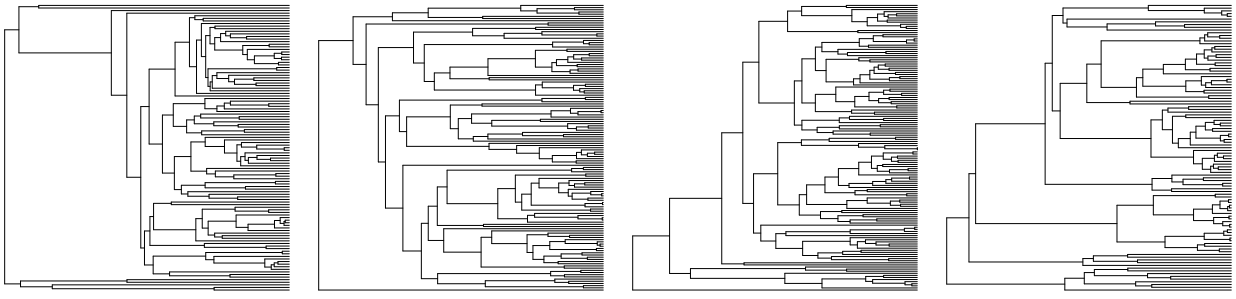
(e) Campephagidae- (f) Charadrii (g) Columbidae (h) Corvidae+

(i) Cuculidae (j) Emberizidae- (k) Estrildidae (l) Fringillidae+

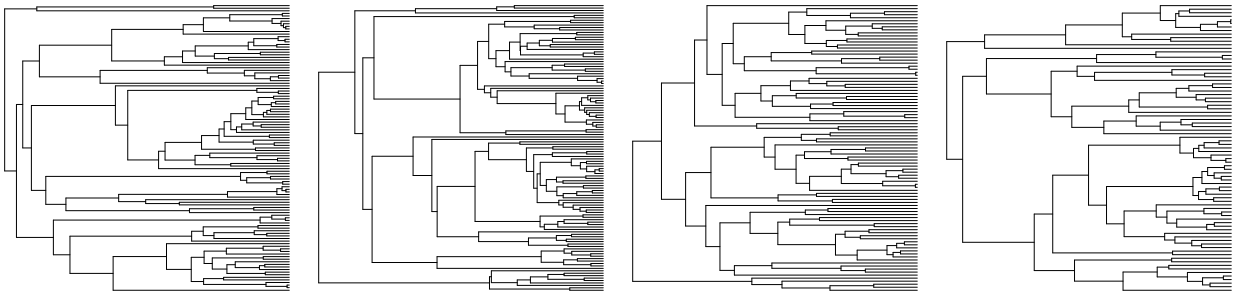
(m) Furnaridae (n) Hirundinidae (o) Icteridae (p) Lari

(q) Malaconotidae+ (r) Meliphagidae-+ (s) Muscicapidae-+ (t) Paridae+

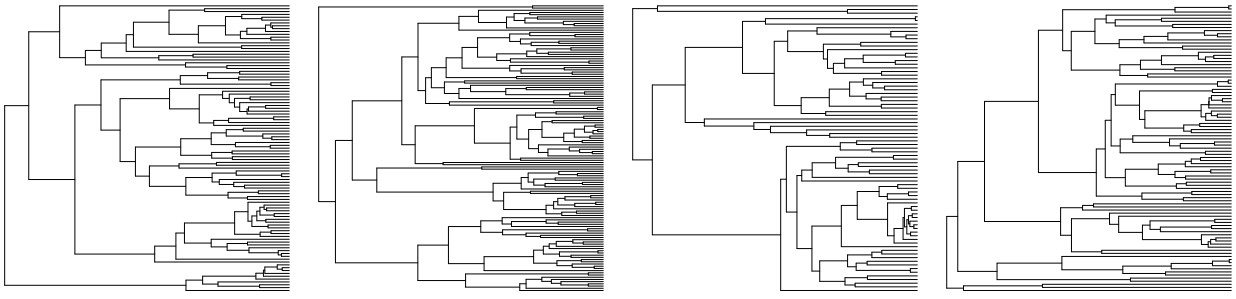
Figure 10: Shape of the bird trees, part 1



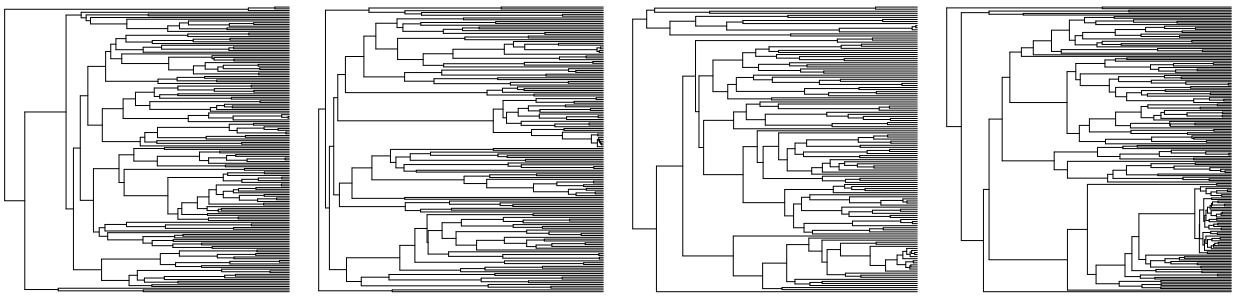
(a) Parulidae+ (b) Phasianidae (c) Picidae (d) Procellariidae



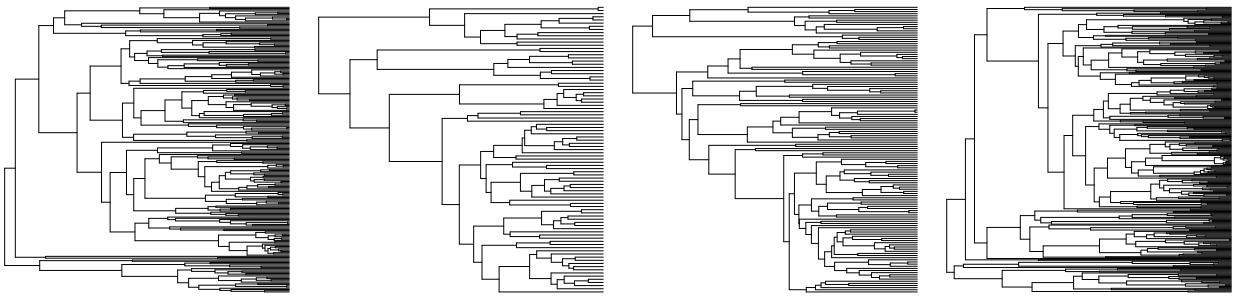
(e) Psittacidae1 (f) Psittacidae2 (g) Pycnonotidae-+ (h) Ramphastidae



(i) Strigidae (j) Sturnidae+ (k) Syvliidae1+ (l) Sylviidae2+



(m) Thamnophilidae (n) Thraupidae1+ (o) Thraupidae2+ (p) Timaliidae-+



(q) Trochilidae (r) Troglodytidae+ (s) Turdidae-+ (t) Tyrannidae+

Figure 11: Shape of the bird trees, part 2

9 Extended results

The main purpose of the empirical analyses is to demonstrate the power of probabilistic programming in addressing inference problems in phylogenetics, not to advance the field of diversification studies. Nevertheless, there are several interesting patterns in the results that deserve attention and that may inspire further study. In this section, we present model likelihoods and posterior estimates of model parameters for all bird clades and diversification models (Figs. 12–21). The plots of posterior distributions should be interpreted in relation to the prior distributions for the corresponding regions of parameter space (Fig. 22). We structure the discussion of the results around several cross-cutting themes.

9.1 Conservative nature of Bayesian model tests

One of the most striking patterns across the bird trees, especially given the recent debate about the importance of accommodating lineage-specific diversification rates, is that simple birth-death models do so well in a Bayesian model comparison. For 16 of the 40 bird trees, there is no strong Bayes factor against the simple CRB and CRBD models. In fact, in most of these cases, the simple models (either CRB(D) or TDB(D)) have the best normalizing constants. There are also some cases where the TDB(D) models do clearly better than the other models, significantly so in a couple of cases (Emberizidae- and Muscicapidae-+, Figs. 14 and 16, respectively).

There is a clear correlation between the size of the tree and the outcome of the model comparison. Of the trees with less than 100 leaves, the CRB(D) models adequately describe the diversification process in a majority of cases, as indicated by Bayes factors. The largest trees lacking strong evidence of lineage-specific or slowing diversification have around 130 leaves (Cuculidae, Phasianidae and Sturnidae+; Figs. 14, 17 and 19, respectively). Above that size, all trees bear a clear mark of lineage-specific or, at least, slowing (TDB(D)) diversification. The age of the tree appears to be inversely related to the adequacy of simple diversification models. Of the ten youngest trees, only two (Estrildidae+ and Icteridae; Figs. 14 and 15, respectively) lack strong support for lineage-specific or slowing diversification, while this is fairly common among the oldest trees.

These patterns appear to be best explained by the density of branching events in the reconstructed tree. The more branching events there are per unit time, the more likely it is that the evolutionary process has left signs of density-dependent or lineage-specific diversification. These fluctuations in diversification rates may tend to even out over longer time scales, as old and species-poor trees are often adequately explained by simple models. However, there are clear exceptions. For instance, the Paridae+ (Fig. 16) shows clear evidence of lineage-specific diversification, despite being an old group (40.9 Ma) with relatively few species (55).

Overall, our results clearly illustrate that Bayes factors are inherently conservative, preferring simpler models unless the signal in the data is sufficiently strong to decisively reject them. While this could be considered a reasonable feature, some caution is nevertheless needed when interpreting the outcome of the model comparison experiment. In particular, the fact that very simple models seem adequate for so many of the bird clades appears to largely reflect the lack of (sufficiently strong) evidence and should not be interpreted as evidence of absence. Many of the trees analysed here (and elsewhere) are simply too small or not informative enough to allow for a non-trivial outcome. An alternative to be considered here is the data consistency criterion[42], which is a general criterion to evaluate the consistency of a model with respect to the observed data.

The degree to which Bayes factors are conservative is dependent on the prior distributions used for the additional parameters of the more complex models. More diffuse priors automatically result in higher penalties in the model comparison—and this even if the posterior distribution itself is not impacted or only marginally impacted. Often, this is not a major problem, for instance when comparing models of sequence evolution, where the signal contributed by the sequence data easily overwhelms the penalty induced by diffuse priors. Here, in contrast, the empirical signal contributed by phylogenetic trees of surviving lineages about the underlying diversification process is somewhat weaker, making the relative impact of the prior on the outcome of Bayesian model tests more substantial.

Whether our priors strike a reasonable balance between simple and complex models is, of course, open to discussion. We note, however, that our priors for the ClaDS models are less conservative than the ones proposed originally for these models[30]. Thus, our priors penalize the ClaDS models less than would otherwise have been the case. We also want to re-emphasize that, to allow fair model comparisons, we chose priors on analogous model parameters that were similar, if not identical, across models.

9.2 Robustness of complex models

An important result that emerges from our analyses, and that we want to emphasize, is the robustness of complex diversification models. Even when Bayes factors indicate that simple models are adequate, the more sophisticated models often give consistent estimates for the additional model parameters. Good examples are provided by the posterior estimates for σ^2, describing the rate of gradual, lineage-specific change in diversification rates in the ClaDS models, and η, denoting the rate of punctuated change in the LSBDS and BAMM models; both of these parameters are usually estimated to be close to 0 when simple models appear adequate. Similarly, the parameters related to potential density-dependent effects (z for TDB(D) and BAMM, and $\log \alpha$ for ClaDS) are often close to 0 when the CRB(D) models have the best marginal

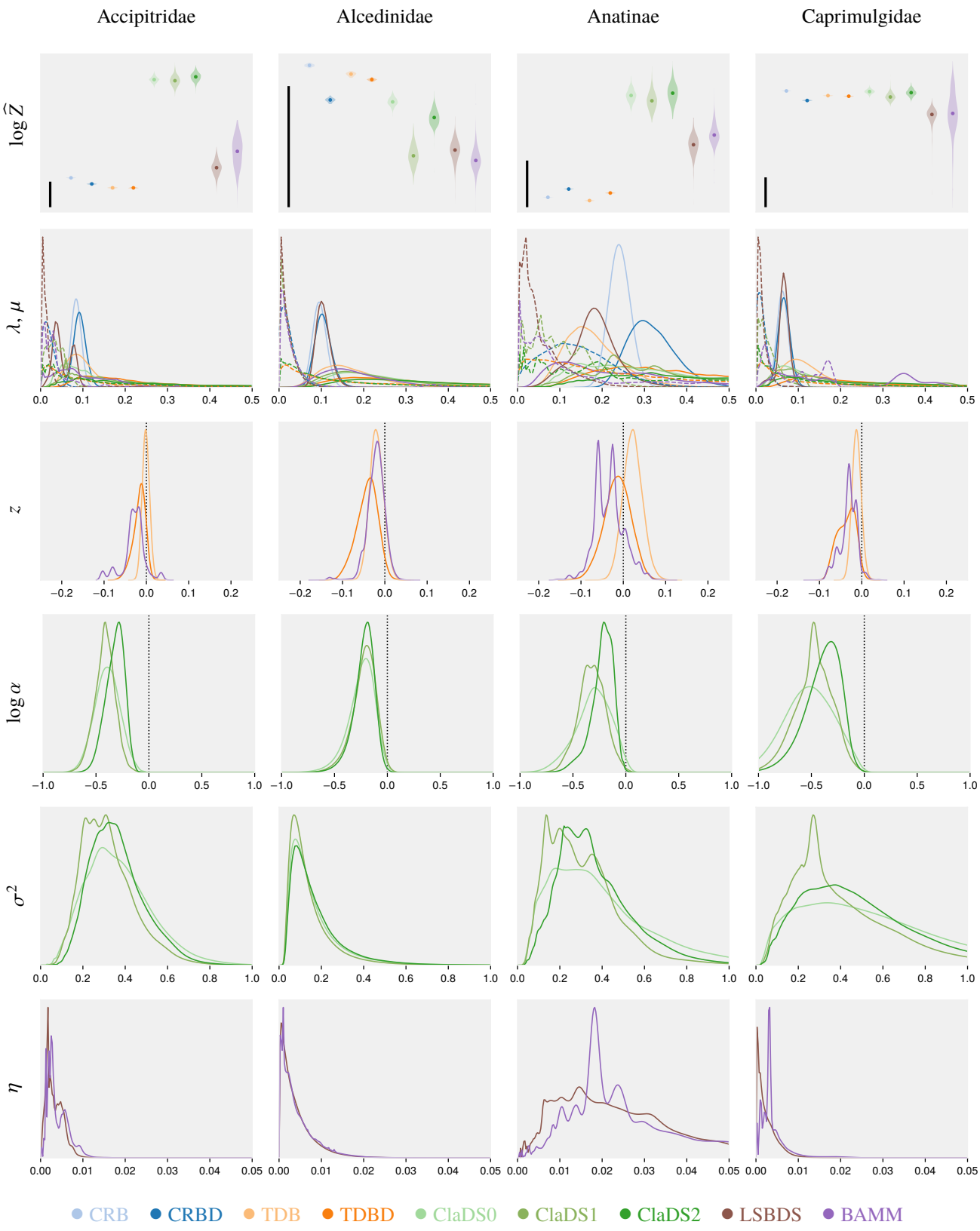


Figure 12: Normalization constants and parameter estimates for Accipitridae, Alcedinidae, Anatinae, Caprimulgidae.

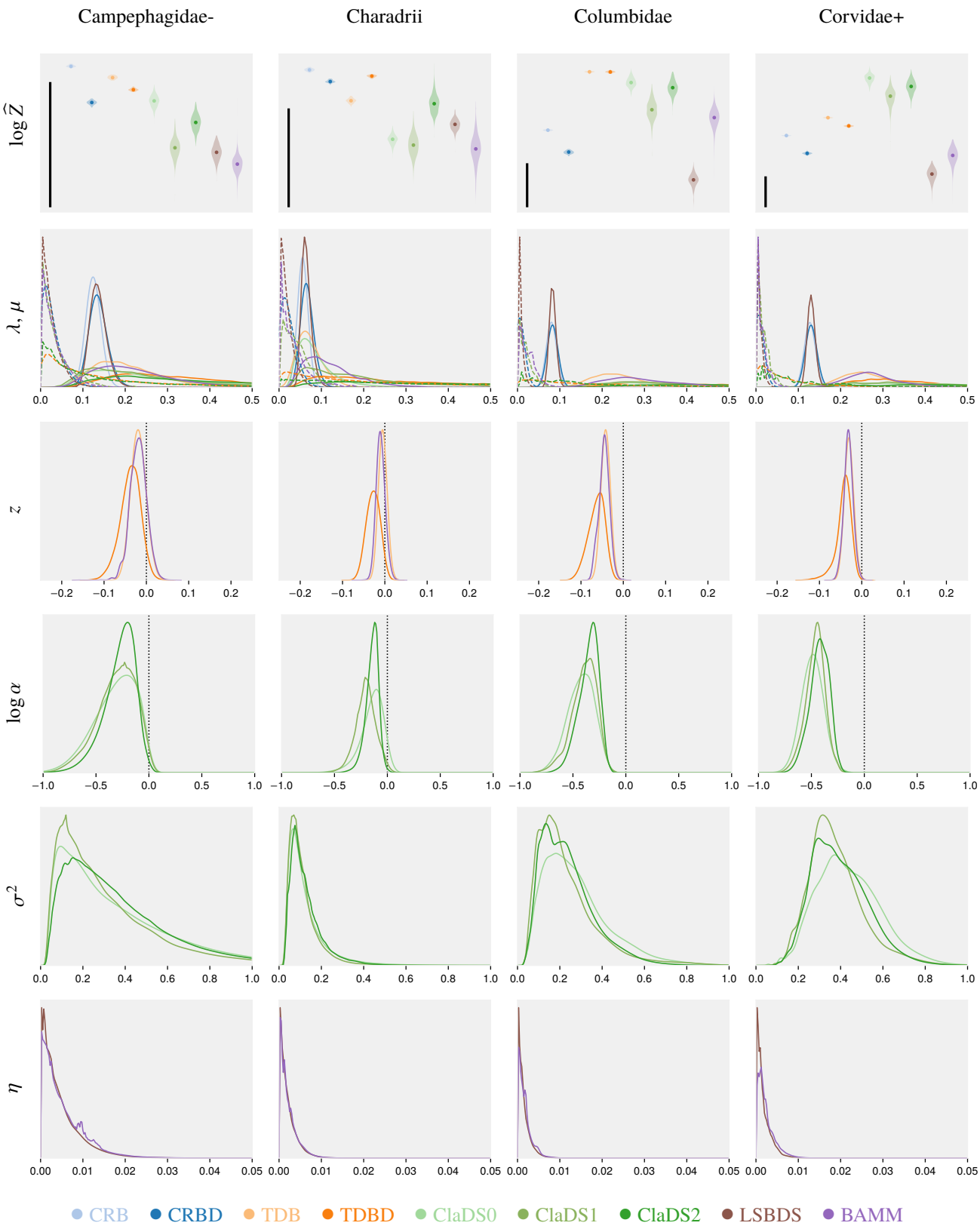


Figure 13: Normalization constants and parameter estimates for Campephagidae-, Charadrii, Columbidae, Corvidae+.

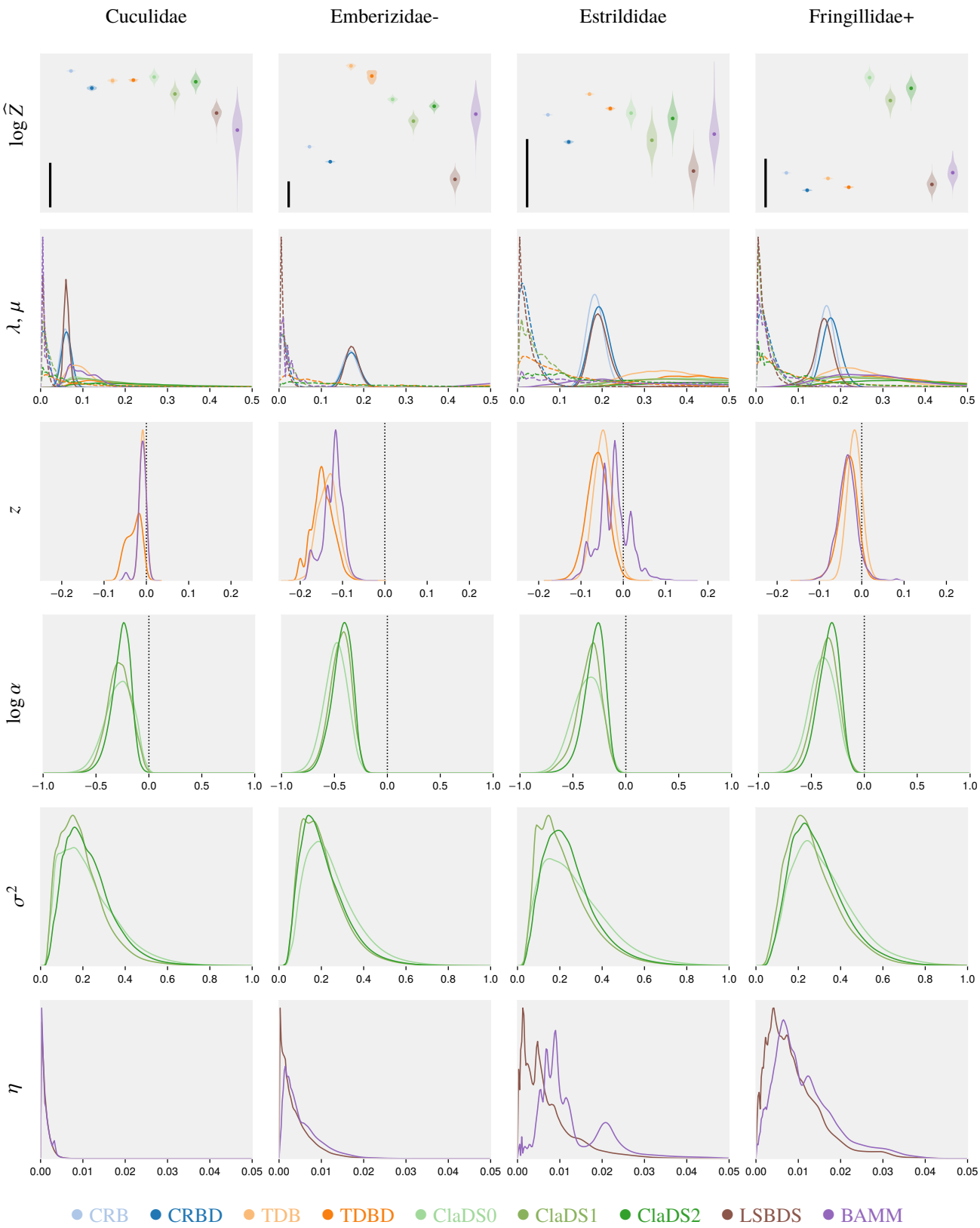


Figure 14: Normalization constants and parameter estimates for Cuculidae, Emberizidae-, Estrildidae, Fringillidae+.

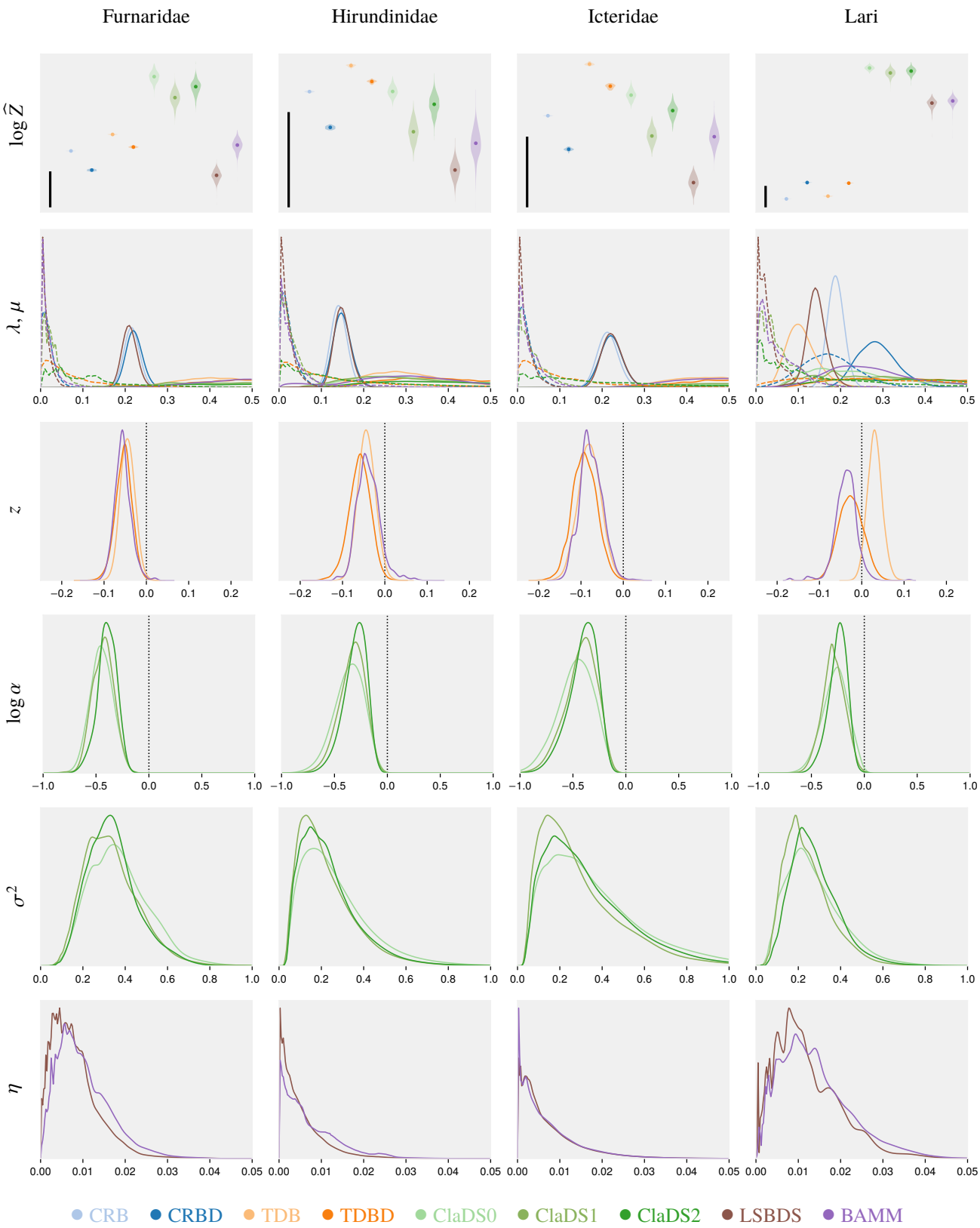


Figure 15: Normalization constants and parameter estimates for Furnaridae, Hirundinidae, Icteridae, Lari.

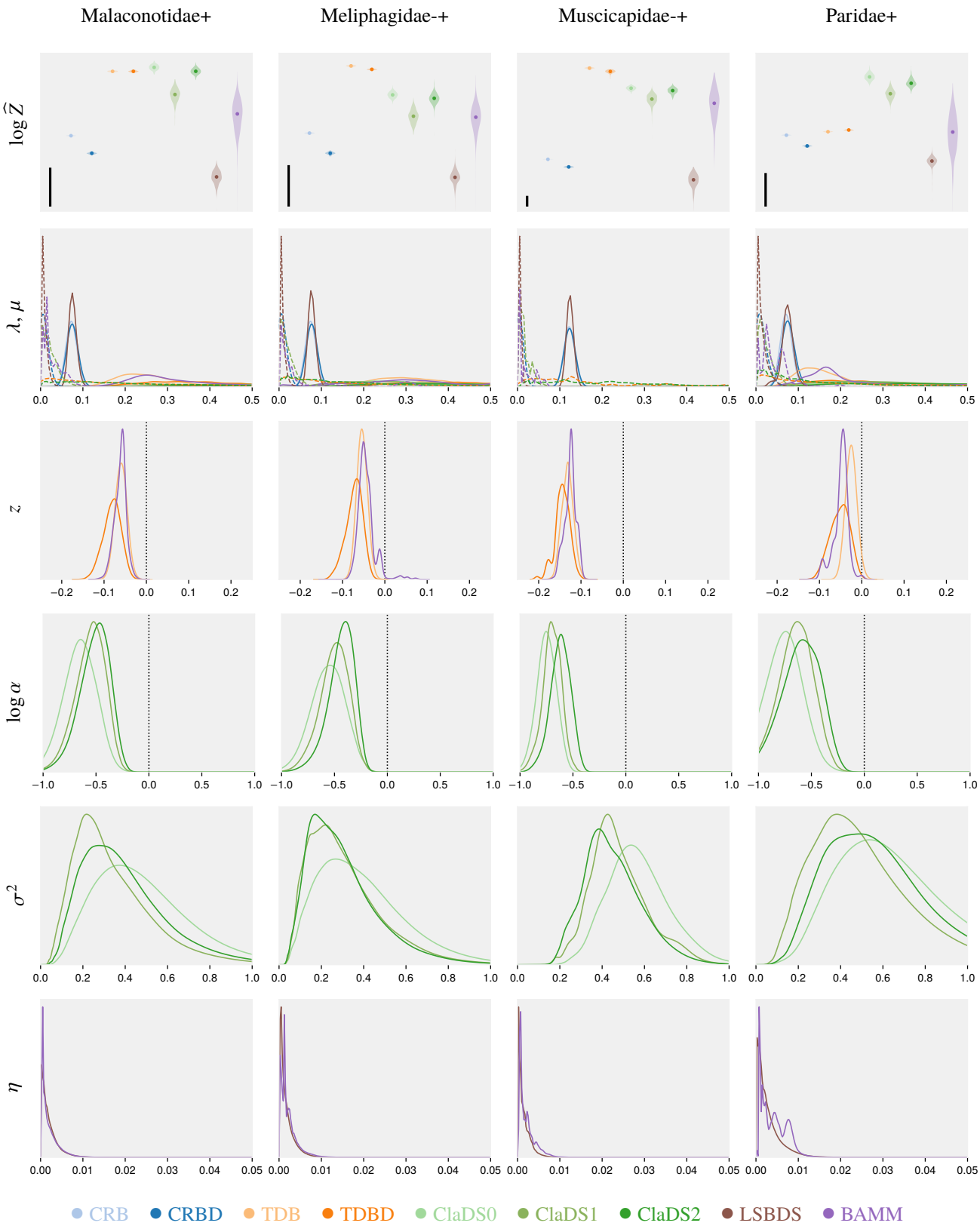


Figure 16: Normalization constants and parameter estimates for Malaconotidae+, Meliphagidae-+, Muscicapidae-+, Paridae+.

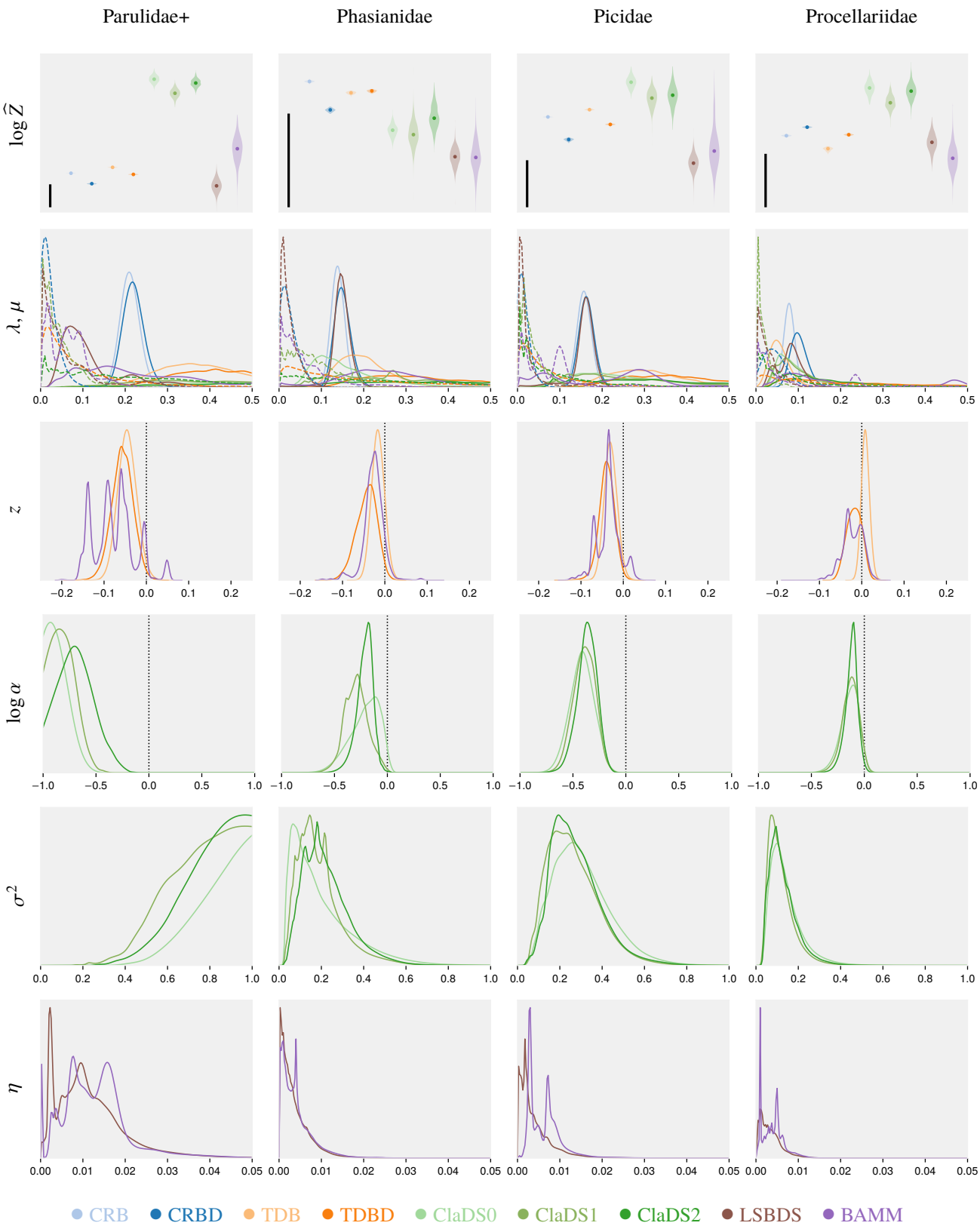


Figure 17: Normalization constants and parameter estimates for Parulidae+, Phasianidae, Picidae, Procellariidae.

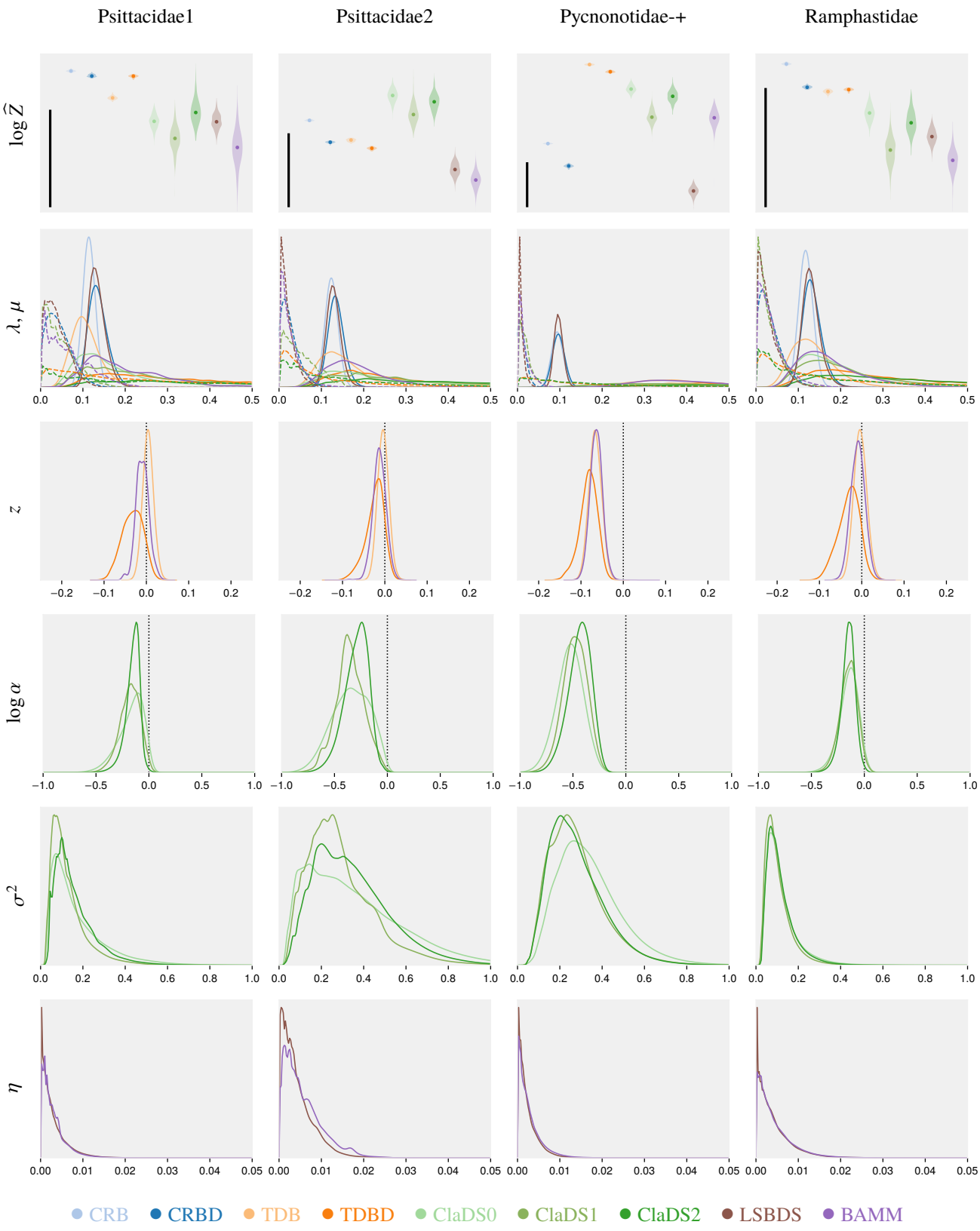


Figure 18: Normalization constants and parameter estimates for Psittacidae1, Psittacidae2, Pycnonotidae-+, Ramphastidae.

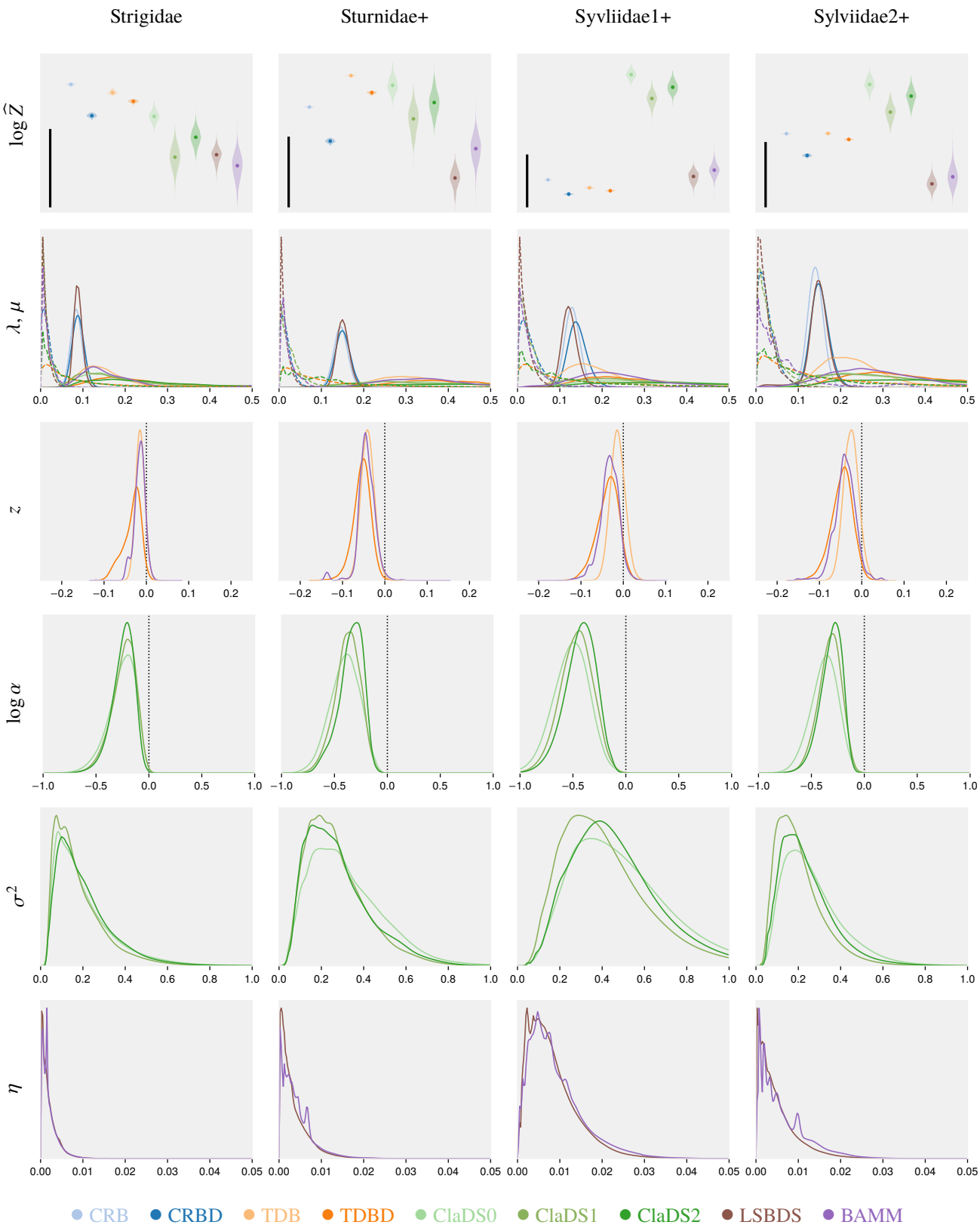


Figure 19: Normalization constants and parameter estimates for Strigidae, Sturnidae+, Syvliidae1+, Sylviidae2+.

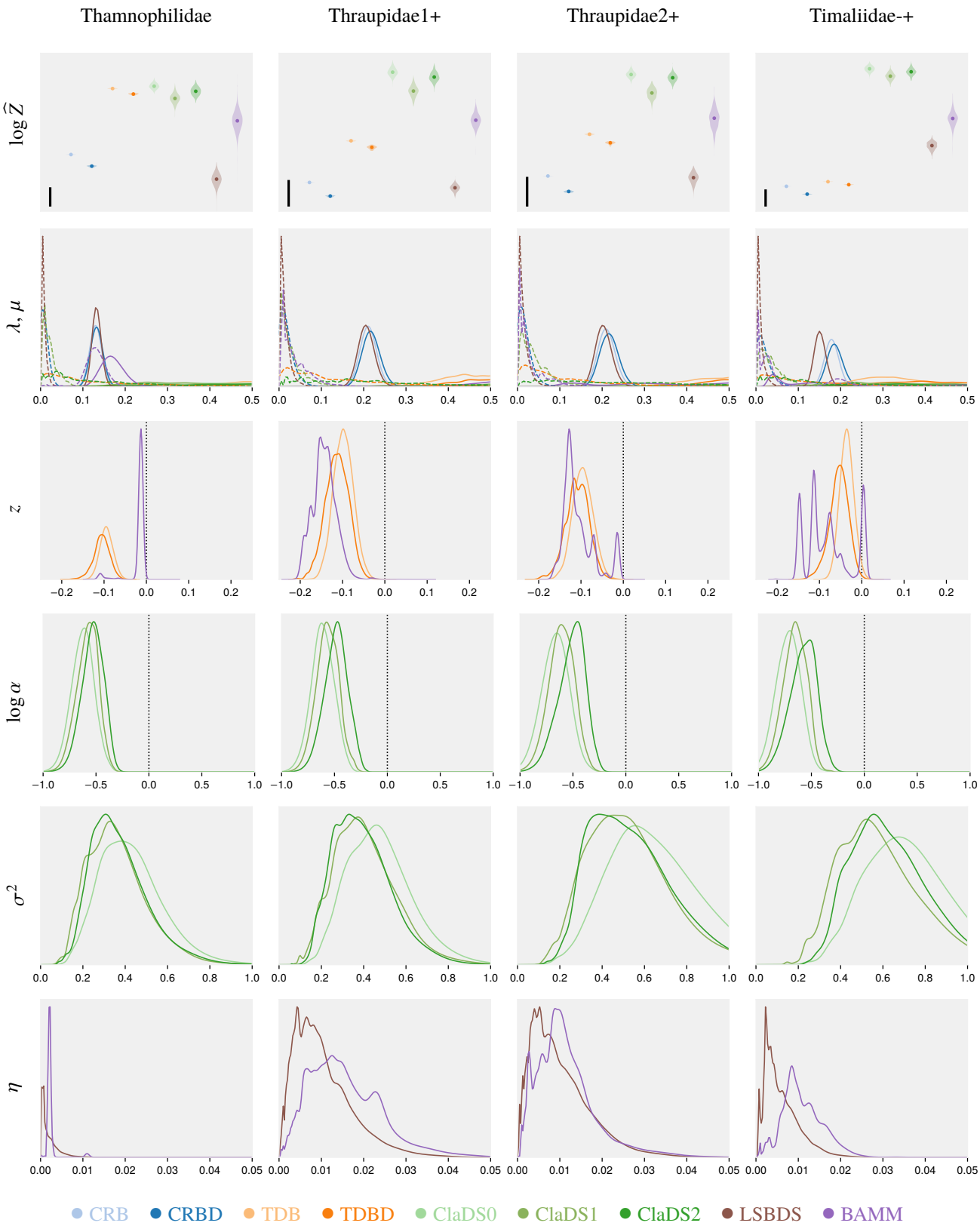


Figure 20: Normalization constants and parameter estimates for Thamnophilidae, Thraupidae1+, Thraupidae2+, Timaliidae-+.

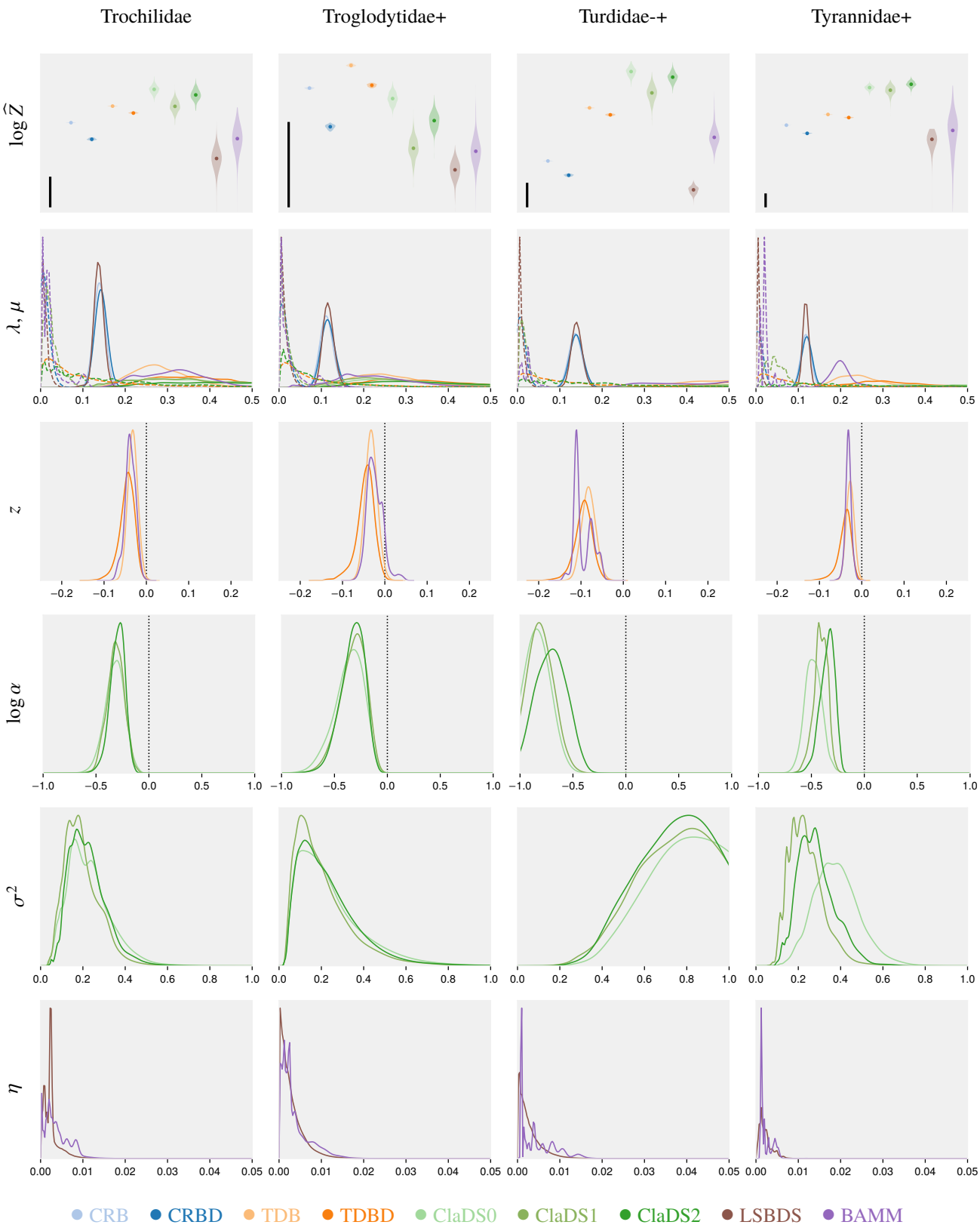
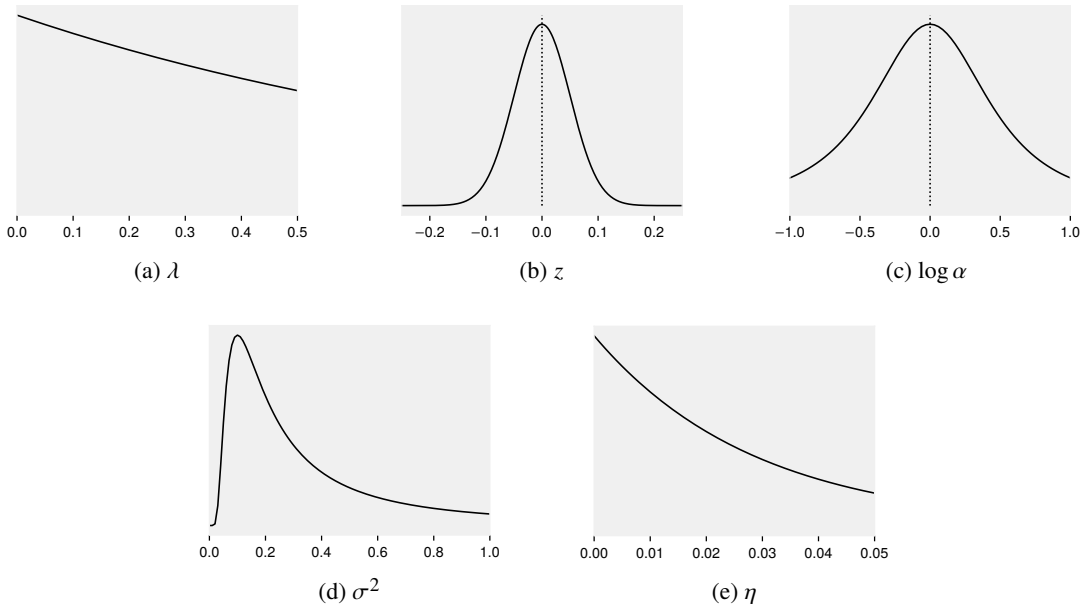


Figure 21: Normalization constants and parameter estimates for Trochilidae, Troglodytidae+, Turdidae-+, Tyrannidae+.



(a) λ (b) z (c) $\log \alpha$

(d) σ^2 (e) η

Figure 22: Prior distributions plotted for the same region of parameter space used for the posterior distributions in Figs. 12–21.

likelihoods. Furthermore, no-extinction models , such as ClaDS0, often have higher likelihoods than their counterparts that accommodate extinction. However, in these cases, the more complex models almost always estimate extinction or turnover rates that are close to 0. This usually occurs with very little impact on the estimation of other parameters, as is well illustrated by the very similar posterior distributions obtained across the ClaDS model series, despite the fact that ClaDS0 often has better marginal likelihood than the more complex variants.

If the results are scrutinized, one discovers that the advanced diversification models actually appear to pick up weak but consistent signal for more complex patterns even when they are not favored by the model tests. For instance, when posterior estimates of $\log \alpha$ or z are significantly different from 0 in these cases, the estimates always suggest slowing diversification rates, and the models that accommodate such variation over time tend to be the ones with the best model likelihoods, even if they are only marginally better than the constant-rate models. Taken together, these observations suggest that the more complex models might in fact be generally more adequate than the simpler ones. The risk of obtaining erroneous or misleading inference under more complex models appears to be low, at least in comparisons among nested models with similar dimensionality.

9.3 Slowing diversification rates

The strongest signal across bird clades in our analyses is undoubtedly the support for slowing diversification rates. This is seen already in the model comparisons but perhaps more clearly in the posterior estimates of $\log \alpha$ in the ClaDS models, and z in the TDB(D) and BAMM models (Figs. 12–21). The estimates are almost universally below 0, indicating decelerating rates, and usually significantly so (more than 95% of the credible interval on negative values). Nowhere is the signal more evident than in the five bird clades where the models that only account for changing diversification rates over time—the TDB(D) models—come out distinctly ahead of all others in the model comparison (Columbidae, Emberizidae-, Meliphagidae-+, Muscicapidae-+ and Pycnonotidae-+). In two of those cases (Emberizidae- and Muscicapidae-+; Figs. 12 and 12, respectively), the Bayes factors even provide strong evidence in favor of TDB(D) over all other models.

Diversification rates that slow down over time are usually attributed to competition for limited resources or niches[43,44,45]. Alternative explanations that have been proposed include: (1) subdivision of geographic ranges at speciation; (2) speciation bursts driven by environmental or geological change; (3) failure to keep pace with environmental change; and (4) protracted speciation (related to the diversified sampling bias, see below)[46]. It might be possible to tease apart some of these factors by developing more sophisticated diversification models within the PPL framework, but this is outside the scope of the current paper. Regardless of the causes, it is clear that there is a strong signature of slowing diversification rates in the bird clades, and that it is important to account for this in diversification models.

9.4 Gradual change, punctuated change or both?

Unsurprisingly, there is also clear evidence of variation across lineages in diversification rates. Of the 40 bird trees, Bayes factors strongly favor models accommodating lineage-specific effects over simpler ones in 14 cases. Even in the remaining

cases, there is often some support for lineage-specific variation in diversification rates, as indicated by posterior estimates of model parameters.

The ClaDS models consistently explain this variation in diversification rates better than the LSBDS and BAMM models. In fact, there are only three groups for which the LSBDS and BAMM models are strongly favored over the corresponding simple models: Anatinae, Lari, and Timaliidae-+ (Figs. 12, 15 and 20, respectively), The BAMM model also does comparatively well on the Thraupidae1+ tree (Fig. 20). As expected, these trees are also associated with posterior estimates of η that differ substantially from 0. However, even for these trees, where BAMM and LSBDS detect major shifts in diversification rates, the ClaDS models provide a better fit to the data.

We may conclude that lineage-specific differences in diversification rates are better explained by slow, gradual changes, which accumulate over time, than by a few events that drastically alter the rates. One possible explanation for this is that the punctuated models (BAMM and LSBDS) draw the new λ and μ (and z for BAMM) values from diffuse priors at process switching events. This means that they carry heavy penalties in Bayes factor comparisons; the more switches there are, the heavier the penalty against these models. An interesting difference between the punctuated models and the gradual models is that the former allow both λ and μ to vary over the tree, while only λ is modulated over the tree in the latter. Could this be the explanation for the gradual models outperforming the punctuated models? We tested this by modifying LSBDS and BAMM such that they assumed a constant turnover rate ($\epsilon = \mu/\lambda$), as in ClaDS2, and only varied λ (and z for BAMM) at switching points. We then re-computed the normalization constants for Lari, one of the clades with the strongest evidence for major shifts in diversification rates. The normalization constants of the punctuated models did not improve significantly due to this modification (results not shown). This suggests that the strong evidence in favor of gradual over punctuated change is not due simply to the punctuated models postulating changes in extinction rates that are not supported by the data.

A fascinating question is whether there remains any evidence for occasional major shifts in diversification rates if one first adequately accounts for the strong underlying signal of slow and gradual change. This can now be examined by extending the PPL framework to diversification models that combine ClaDS-like and BAMM-like features. Given the general lack of support for radical shifts in diversification rates across the bird trees, it seems likely that such shifts are rare, if they occur at all. Therefore, identifying them would presumably require analyses of larger trees than the ones examined here. However, it also seems likely that the two processes interact, such that it becomes more difficult to detect major shifts when the gradual changes are not accounted for. Thus, it is possible that there are major shifts in the bird trees that our analyses failed to detect because of shortcomings in the models. We will have to await future analyses using more sophisticated models before we know whether this is the case.

9.5 Discretizing punctuated-change models

Computing likelihoods for punctuated models of diversification by integrating out the rate priors using discrete approximations is potentially a very powerful approach[29]. It allows for robust and computationally efficient MCMC inference, as long as a small number of rate categories yield sufficiently accurate likelihood estimates. This decidedly appears to be the case, especially if only changes in speciation rate are modeled; empirical analyses suggest that ten categories is quite sufficient in most cases[29].

Unfortunately, it is difficult to see how this approach can be extended to accommodate slowing (or increasing) diversification rates over times as in BAMM, because then it would be necessary to integrate out an infinite number of rate acceleration or deceleration processes with different starting points. This appears to be an important limitation from an empirical perspective, at least judging by the bird trees we analyzed. The LSBDS model simply does not fit many of the reconstructed trees well; this is undoubtedly linked to the substantial support for slowing diversification rates in most bird clades. If we restrict our attention to models of punctuated change, we find 12 trees with strong evidence favoring the BAMM model over the LSBDS model. In contrast, there are only a few trees for which LSBDS performs better than BAMM, Cuculidae being the best notable (Fig. 14), and in those cases there is never strong evidence in favor of LSBDS over BAMM.

9.6 Sampling biases

Some of the results that emerge from our analyses are probably due, at least in part, to sampling biases. The lack of evidence for significant extinction rates is an obvious case. Models without extinction (CRB, TDB, ClaDS0) almost always do better than models with extinction except in two cases: the related bird groups Charadrii (waders) and Lari (gulls) (Figs. 13 and 15, respectively). The outcome of the model comparison is generally consistent with posterior estimates of μ under the corresponding models that do accommodate extinction (CRBD and TDBD). Estimated extinction rates are usually low, except for Charadrii and especially Lari. Interestingly, Lari is also unusual in that there is evidence for accelerating speciation rates ($z > 0$). However, this occurs only in the TDB model, and is probably an artefact of not accounting for extinction, as significant extinction rates are expected to lead to an apparent acceleration of speciation rates close to the present in reconstructed trees[25].

Given the overwhelming evidence for frequent extinction in the fossil record, these results are not plausible, but they are consistent with results from previous diversification studies[45]. An important factor that is likely to contribute toward

underestimation of extinction rates is the *diversified sampling* bias[47]. This is the tendency of biologists to systematically select leaves of phylogenetic trees in such a way that the diversity represented in the sampled tree is maximized, instead of choosing leaves randomly as assumed by standard birth-death models. The diversified sampling bias will lead to pruning of the most recent splits from the complete reconstructed tree. The more incomplete the sampling is, the deeper the period devoid of splits will extend into the past. In the most extreme cases, the sampled tree will look like a bush: most of the splits will be close to the root of the tree, and all the leaves will sit on long terminal branches. If one analyzes a tree sampled to maximize diversity under a model assuming random sampling, extinction rates can be significantly underestimated[47]. Failing to account for diversified sampling bias can also result in severe biases in divergence time estimates[48,49].

The bird trees we examined here contain most or all of the described species in the corresponding clades[41], and our analyses therefore assumed that the sampling of leaves is complete ($\rho = 1$). Nevertheless, diversified sampling biases probably affect also our results, as biologists tend to sample morphologically distinct species, and omit incipient species, sibling species and subspecies. Such unacknowledged diversified sampling at the species level is linked to the phenomenon of *protracted speciation*[50]. Interestingly, diversified sampling bias that is not accounted for correctly could also, at least to some extent, explain the strong support for slowing diversification rates in diversification studies[46]. To understand why, consider that the extinction rate estimates are largely based on the apparent acceleration of speciation seen towards the recent seen in surviving trees because there has not been time enough for extinction of the side lineages that will eventually disappear[25]. Diversified sampling would systematically remove the evidence for this apparent acceleration in speciation rates.

As one might expect, there is also a link between the posterior estimates of extinction and the evidence for slowing diversification rates. Specifically, models that accommodate slowing diversification rates (TDBD, ClaDS models and BAMM) are also associated with distinctly higher estimates of extinction rates than models that do not (LSBDS, CRBD) (Figs. 12–21). How this link might be affected by diversified sampling biases is currently unclear. Pursuing this topic further would be outside of the scope of the current paper. However, we do note that it is relatively straightforward to modify our script to account for diversified sampling according to the model suggested by Höhna et al.[47], or potentially even more realistic models.

9.7 Statistical power

Reconstructed trees carry only a limited amount of information about absolute speciation and extinction rates. This is illustrated well by the underestimation of extinction rates discussed above. For really powerful analyses of diversification processes, we need trees that include data from the fossil record, that is, observations of both extinct and extant lineages[51]. In most cases, however, observation of extinct lineages is not possible, or the information about extinct lineages is bound to be very incomplete, so we need to extract as much information as possible from trees only (or mainly) comprising surviving lineages. There are several ways in which analyses of such trees can be improved. Addressing sampling biases appropriately would be an important step in the right direction. Making the model of the diversification process itself more realistic, for instance by combining gradual and punctuated change as suggested above, would be another. However, the most obvious way to improve the analyses would be to increase the amount of data.

A natural way of extending the present work in this direction would be to opt for a hierarchical model-averaging approach, in which all trees in a set, such as the bird clades, would be analyzed simultaneously. Specifically, each tree would randomly choose from a mixture of all available models, while the mixture proportions and the hyper-parameters tuning the priors over model-specific parameters would themselves be estimated across trees, using a hierarchical modeling design. Global estimation of hyper-priors and mixture weights based on the whole collection of trees is an efficient way to fit the priors to the true prevalence of alternative modes of diversification and the true variation in parameter values present in the data and therefore should result in well-calibrated model selection. Joint analysis of all trees would also make it possible to collect the weak signals disseminated across the many small trees of the analysis. Such developments are exactly what the probabilistic programming framework introduced here is meant to facilitate.

A completely different approach would be to analyze larger portions of the tree of life. For instance, our analyses of 40 bird clades could have been replaced with a single analysis of the entire bird tree, doubling the coverage of bird species (from 5,000 to 10,000). Such an analysis would not only include more of the variation seen across terminal clades, it would also add data on the deeper splits in the tree. These splits could potentially inform the model about long-term macroevolutionary patterns that could not be detected in analyses of only terminal clades, regardless of how sophisticated. For instance, it has been suggested that the bird radiation as a whole is characterized by rare but major boosts in diversification rates, presumably linked to key innovations opening up new ecological niches[41]. If these upward jumps in diversification rates are substantial enough, it could explain why there is overwhelming support for slowing diversification rates in individual bird clades, even though the rates appear to be accelerating over the bird tree as a whole[41]. Such a mega-analysis would have to be based on a model that is more sophisticated than the ones explored here. Minimally, it would have to account for both gradual and punctuated change in diversification rates. Ideally, it would also account for variation across lineages in the strength of the slowing forces on diversification, and in the rate of gradual change in speciation and extinction rates. Again, such developments are well supported by the probabilistic programming framework, although it is still an open question whether current inference strategies are efficient enough or whether further refinement is needed.

References

1. Gilks, W. R., Thomas, A. & Spiegelhalter, D. J. A language and program for complex Bayesian modelling. *The Statistician* **43**, 169–177 (1994).

2. Bishop, C. M. *Pattern Recognition and Machine Learning* (2006, New York, USA, Springer).

3. Minka, T. et al. /Infer.NET 0.3 (2018). Microsoft Research Cambridge. http://dotnet.github.io/infer.

4. Goodman, N. D. The principles and practice of probabilistic programming. *ACM SIGPLAN Notices* **48**, 399–402 (2013).

5. Gordon, A. D., Henzinger, T. A., Nori, A. V. & Rajamani, S. K. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, pages 167–181 (ACM, 2014).

6. Murray, L. M., Lundén, D., Kudlicka, J., Broman, D. & Schön, T. B. Delayed sampling and automatic Rao–Blackwellization of probabilistic programs. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 21, page 10 (Lanzarote, 2018).

7. Goodman, N. D., Mansinghka, V. K., Roy, D., Bonawitz, K. & Tenenbaum, J. B. Church: A language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, page 220–229 (AUAI Press, Arlington, Virginia, USA, 2008).

8. Pfeffer, A. Figaro: An object-oriented probabilistic programming language. *Charles River Analytics Technical Report* **137**, 96 (2009).

9. Goodman, N. D. & Stuhlmüller, A. The design and implementation of probabilistic programming languages. `http://dippl.org` (2014). Accessed: 2020-5-12.

10. Wood, F., Meent, J. W. & Mansinghka, V. A new approach to probabilistic programming inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 1024–1032 (Reykjavik, Iceland, 2014).

11. Mansinghka, V., Selsam, D. & Perov, Y. Venture: a higher-order probabilistic programming platform with programmable inference. Preprint at `https://arxiv.org/abs/1404.0099` (2014).

12. Tran, D. et al. Edward: A library for probabilistic modeling, inference, and criticism. Preprint at `https://arxiv.org/abs/1610.09787` (2016).

13. Bingham, E. et al. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research* **20**, 1–6 (2019).

14. Murray, L. M. & Schön, T. B. Automated learning with a probabilistic programming language: Birch. *Annual Reviews in Control* **46**, 29–43 (2018).

15. Carpenter, B. et al. Stan: A probabilistic programming language. *Journal of Statistical Software* **76** (2017).

16. Broman, D. A Vision of Miking: Interactive programmatic modeling, sound language composition, and self-learning compilation. In *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering (SLE)*, pages 55–60 (ACM, 2019).

17. van de Meent, J.-W., Paige, B., Yang, H. & Wood, F. An introduction to probabilistic programming. Preprint at `https://arxiv.org/abs/1809.10756` (2018).

18. PhyJSON—a simple JSON format for phylogenetic data. `https://github.com/kudlicka/nexus2phyjson/blob/master/doc/phyjson_format_description.md` (2018). Accessed: 2020-04-17.

19. BiSSE trees from MesquiteCore. `https://github.com/MesquiteProject/MesquiteCore/blob/master/Resources/examples/Diversification/06-BiSSEtrees.nex` (2009). Accessed: 2020-04-16.

20. Whale tree from BAMM. `https://github.com/macroevolution/bamm/blob/master/examples/diversification/whales/whaletree.tre` (2013). Accessed: 2020-04-16.

21. Diversitree raw content from GitHub. `https://raw.githubusercontent.com/richfitz/diversitree/master/pub/example/data/primates-10.nex` (2011). Accessed: 2020-04-16.

22. Gernhard, T. The conditioned reconstructed process. *Journal of Theoretical Biology* **253**, 769–778 (2008).

23. Semple, C. & Steel, M. *Phylogenetics* (Oxford University Press, Oxford, 2003).

24. Yule, G. U. A mathematical theory of evolution, based on the conclusions of Dr. JC Willis, FRS. *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character* **213**, 21–87 (1924).

25. Nee, S. Birth-death models in macroevolution. *Annual Review of Ecology, Evolution and Systematics* **37**, 1–17 (2006).

26. Feller, W. Die Grundlagen der Volterraschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung. *Acta Biotheoretica* **5**, 11–40 (1939).

27. Kendall, D. G. On the generalized "birth-and-death" process. *The Annals of Mathematical Statistics* **19**, 1–15 (1948).

28. Rabosky, D. L. Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. *PLoS ONE* **9**, e89543 (2014).

29. Höhna, S. et al. A Bayesian approach for estimating branch-specific speciation and extinction rates. Preprint at `https://www.biorxiv.org/content/10.1101/555805v1` (2019).

30. Maliet, O., Hartig, F. & Morlon, H. A model with many small shifts for estimating species-specific diversification rates. *Nature Ecology & Evolution* **3**, 1086–1092 (2019).

31. Yang, Z. Reconciling molecular phylogenies with the fossil record. *Journal of Molecular evolution* **108**, 16327–16332 (2011).

32. Moore, B. R., Höhna, S., May, M. R., Rannala, B. & Huelsenbeck, J. P. Critically evaluating the theory and performance of Bayesian analysis of macroevolutionary mixtures. *Proceedings of the National Academy of Sciences of the United States of America* **113**, 9569–9574 (2016).

33. Maddison, W. P., Midford, P. E. & Otto, S. P. Estimating a binary character's effect on speciation and extinction. *Systematic Biology* **56**, 701–710 (2007).

34. Yang, Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *Journal of Molecular evolution* **39**, 306–314 (1994).

35. Rabosky, D. L. et al. BAMMtools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. *Methods in Ecology and Evolution* **5**, 701–707 (2014).

36. Lundén, D., Broman, D., Ronquist, F. & Murray, L. M. Automatic alignment of sequential Monte Carlo inference in higher-order probabilistic programs. Preprint at `https://arxiv.org/abs/1812.07439` (2018).

37. Kudlicka, J., Murray, L. M., Ronquist, F. & Schön, T. B. Probabilistic programming for birth-death models of evolution using an alive particle filter with delayed sampling. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence 2019*, volume 2019, page 11 (Tel Aviv, Israel, 2019).

38. Wigren, A., Risuleo, R. S., Murray, L. & Lindsten, F. Parameter elimination in particle Gibbs sampling. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox & R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8918–8929 (Curran Associates, Inc., 2019).

39. Del Moral, P., Jasra, A., Lee, A., Yau, C. & Zhang, X. The alive particle filter and its use in particle Markov chain Monte Carlo. *Stochastic Analysis and Applications* **33**, 943–974 (2015).

40. Morlon, H. et al. RPANDA: an R package for macroevolutionary analyses on phylogenetic trees. *Methods in Ecology and Evolution* **7**, 589–597 (2016).

41. Jetz, W., Thomas, G. H., Joy, J. B., Hartmann, K. & Mooers, A. O. The global diversity of birds in space and time. *Nature* **491**, 444–448 (2012).

42. Lindholm, A., Zachariah, D., Stoica, P. & Schön, T. B. Data consistency approach to model validation. In *IEEE Access*, volume 7, pages 59788–59796 (2019).

43. McPeek, M. The ecological dynamics of clade diversification and community assembly. *The American Naturalist* **172**, E270–E284 (2008).

44. Weir, J. T. Divergent timing and patterns of species accumulation in lowland and highland neotropical birds. *Evolution* **60**, 842–855 (2006).

45. Pyron, R. A. & Burbrink, F. T. Phylogenetic estimates of speciation and extinction rates for testing ecological and evolutionary hypotheses. *Trends in Ecology & Evolution* **28**, 729–736 (2013).

46. Moen, D. & Morlon, H. Why does diversification slow down? *Trends in Ecology & Evolution* **29**, 190–197 (2014).

47. Höhna, S., Stadler, T., Ronquist, F. & Britton, T. Inferring speciation and extinction rates under different sampling schemes. *Molecular Biology and Evolution* **28**, 2577–2589 (2011).

48. Zhang, C., Stadler, T., Klopfstein, S., Heath, T. A. & Ronquist, F. Total-evidence dating under the fossilized birth-death process. *Systematic Biology* **65**, 228–249 (2016).

49. Ronquist, F., Lartillot, N. & Phillips, M. J. Closing the gap between rocks and clocks using total-evidence dating. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* **371**, 20150136 (2016).

50. Rosindell, J., Cornell, S. J., Hubbell, S. P. & Etienne, R. S. Protracted speciation revitalizes the neutral theory of biodiversity: Protracted speciation and neutral theory. *Ecology Letters* **13**, 716–727 (2010).

51. Quental, T. B. & Marshall, C. R. Diversity dynamics: molecular phylogenies need the fossil record. *Trends in Ecology & Evolution* **25**, 434–441 (2010).