Adaptive dating and fast proposals: revisiting the phylogenetic relaxed clock model

Jordan Douglas[1,2*], Rong Zhang[1,2], Alexei J. Drummond[1,2,3], Remco Bouckaert[1,2]

1 Centre for Computational Evolution, University of Auckland, Auckland, New Zealand
2 School of Computer Science, University of Auckland, Auckland, New Zealand
3 School of Biological Sciences, University of Auckland, Auckland, New Zealand

* jordan.douglas@auckland.ac.nz

Abstract

Author summary

Introduction

The molecular clock hypothesis states that the evolutionary rates of biological sequences are approximately constant through time [1]. This assumption forms the basis of phylogenetics, under which the evolutionary trees and divergence dates of life forms are inferred from biological sequences, such as nucleic and amino acids [2,3]. In Bayesian phylogenetics, these trees and their associated parameters are estimated as probabilitiy distributions [4–6]. Statistical inference can be performed by the Markov chain Monte Carlo (MCMC) algorithm [7,8] using platforms such as BEAST [9], BEAST2 [10], MrBayes [11], and RevBayes [12].

The simplest phylogenetic clock model – the strick clock – makes the mathematically convenient assumption that the evolutionary rate is constant across all lineages [4,5,13]. However, molecular substitution rates are known to vary over time, over population sizes, over evolutionary pressures, and over nucleic acid replicative machineries [14–16]. Moreover, any given dataset may be clock-like (where the substitution rates have a small variance across lineages) or non clock-like (a large variance). In the latter case, a strict clock is probably not suitable.

This led to the development of relaxed (uncorrelated) clock models, under which each branch in the phylogenetic tree has its own molecular substitution rate [3]. Branch rates can be drawn from a range of probability distributions including Log-Normal, Exponential, Gamma, and Inverse-Gamma distributions [3,17,18]. This class of models is widely used, and has aided insight into many recent biological problems, including the 2016 Zika virus outbreak [19] and the COVID-19 pandemic [20].

Finally, although not the focus of this article, the class of correlated clock models assumes some form of auto-correlation between rates over time. The correlation itself can invoke a range of stochastic models, including compound Poisson [21] and CIR processes [17], or it can exist as a series of local relaxed clocks [22]. However, due to the correlated and discrete nature of such models, the time until MCMC convergence may be cumbersome, particularly for larger datasets [22].

With the overwhelming availabilty of biological sequence data, the development of efficient Bayesian phylogenetic methods is more important than ever. The performance of MCMC is dependent not only on computational runtime but also the efficacy of an

MCMC setup to achieve its convergence. A critical task therein lies the further advancement of MCMC operators. Recent developments in this area include the advancement of guided tree proposals [23–25], coupled MCMC [26, 27], adaptive multivariate transition kernels [28], and other explorative proposal kernels such as the Bactrian and mirror kernels [29, 30]. In the case of clock models, informed tree proposals can account for correlations between substitution rates and divergence times [31]. The rate parameterisation itself can also affect the ability to "mix" during MCMC [3, 18, 31].

While a range of advanced operators and other MCMC optimisation methods have arisen over the years, there has yet to be a widescale performance benchmarking of such methods as applied to the relaxed clock model. In this article, we systematically evaluate how the relaxed clock model can benefit from i) adaptive operator weighting, ii) different substitution rate parameterisations, iii) the use of Bactrian proposal kernels [29], iv) tree operators which account for correlations between substitution rates and times, and v) adaptive multivariate operators [28]. The discussed methods are implemented in and compared using BEAST2 [10].

Models and Methods

Preliminaries

Let $\mathcal{T}$ be a binary rooted time tree with N taxa. Let L be the number of sites within the multiple sequence alignment D. The posterior density of a phylogenetic model is described by

$$p(\mathcal{T}, \vec{\mathcal{R}}, \sigma, \mu_C, \theta | D) \propto p(D|\mathcal{T}, r(\vec{\mathcal{R}}), \mu_C)\, p(\mathcal{T}|\theta)\, p(\vec{\mathcal{R}}|\sigma)\, p(\sigma)\, p(\mu_C)\, p(\theta). \quad (1)$$

σ represents clock model related parameters, and $p(\mathcal{T}|\theta)$ is the tree prior where θ describes further unspecified parameters. The tree likelihood $p(D|\mathcal{T}, r(\vec{\mathcal{R}}), \mu_C)$ is computed using the tree-peeling algorithm [32], where μ_C is the overall clock rate and $\vec{\mathcal{R}}$ is a vector of abstracted branch rates which is transformed into real rates by function $r(\vec{\mathcal{R}})$. Branch rates have a mean of 1 under the prior to avoid non-identifiability with node heights and the clock rate μ_C. Three methods of representing rates as $\vec{\mathcal{R}}$ are presented in **Substitution rate parameterisations**.

Let t_i be the height (time) of node i. Each node i in $\mathcal{T}$, except for the root, is associated with a parental branch length τ_i (the height difference between i and its parent) and a parental branch substitution rate $r_i = r(\mathcal{R}_i)$. In a relaxed clock model, each of the $2N - 2$ elements in $\vec{\mathcal{R}}$ are independently distributed under the prior $p(\vec{\mathcal{R}}|\sigma)$.

The posterior distribution is sampled by the Metropolis-Hastings-Green MCMC algorithm [7, 8, 33], under which the probability of accepting proposed state x' from state x is equal to:

$$\alpha(x'|x) = \min\left(1, \frac{p(x'|D)}{p(x|D)} \frac{q(x|x')}{q(x'|x)} |J|\right). \quad (2)$$

$q(a|b)$ is the transition kernel: the probability of proposing state b from state a. The ratio between the two $\frac{q(x|x')}{q(x'|x)}$ is also known as the Hastings ratio [8]. The determinant of the Jacobian matrix $|J|$ solves the dimension-matching problem for proposals which operate on multiple terms across one or more spaces [33, 34]. This term is known as the Green ratio.

Substitution rate parameterisations

In Bayesian inference, the way parameters are represented in the model can affect the mixing ability of the model and the meaning of the model itself [35]. Three methods for parameterising substitution rates are described below. Each parameterisation is associated with i) an abstraction of the branch rate vector $\vec{\mathcal{R}}$, ii) some function for transforming this parameter into unabstracted branch rates $r(\vec{\mathcal{R}})$, and iii) a prior density function of the abstraction $p(\vec{\mathcal{R}}\,|\sigma)$. The three methods are summarised in **Fig 2**.

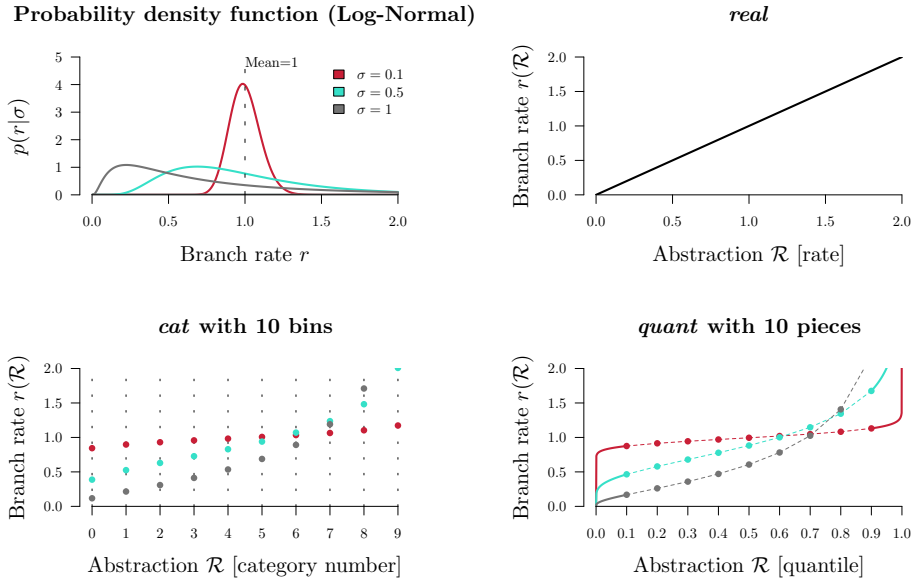

Fig 1. Branch rate parameterisations. Top left: the prior density of a branch rate r under a Log-Normal$(-0.5\sigma^2, \sigma)$ distribution (with its mean fixed at 1). The function for transforming $\mathcal{R}$ into branch rates $r(\mathcal{R})$ is depicted for *real* (top right), *cat* (bottom left), and *quant* (bottom right). For visualisation purposes, there are only 10 bins/pieces displayed, however in practice there are $2N - 2$ bins for *cat* and 100 pieces for *quant*. The first and final *quant* pieces are equal to the underlying function however the pieces in between use linear approximations of this function.

1. Real rates

The natural (and unabstracted) parameterisation of a substitution rate is a real number $\mathcal{R}_i \in \mathbb{R}, \mathcal{R}_i > 0$ which is equal to the rate itself. Thus, under the *real* parameterisation:

$$r(\vec{\mathcal{R}}\,) = \vec{\mathcal{R}}\,. \tag{3}$$

Under the Log-Normal clock prior $p(\vec{\mathcal{R}}\,|\sigma)$, rates are distributed with a mean of 1:

$$p(\mathcal{R}_i|\sigma) = \frac{1}{\mathcal{R}_i\sigma\sqrt{2\pi}} \exp\big(-\frac{(\ln\mathcal{R}_i - \mu)^2}{2\sigma^2}\big) \tag{4}$$

where $\mu = -0.5\sigma^2$ is set such that the expected value of the Log-Normal distribution is 1. In this article we only consider Log-Normal clock priors, however the methods discussed are general.

Zhang and Drummond 2020 introduced a series of tree operators which propose node heights and branch rates, such that the resulting genetic distances ($r_i \times \tau_i$) remain constant [31]. These operators account for correlations between branch rates and branch times. By keeping the genetic distance of each branch constant, the likelihood is unaltered by the proposal.

2. Categories

The category parameterisation (*cat*) is an abstraction of the *real* parameterisation. Each of the $2N-2$ branches are assigned an integer from 0 to $n-1$:

$$\vec{\mathcal{R}} \in \{0, 1, \ldots, n-1\}^{2N-2}. \tag{5}$$

These integers correspond to n rate categories (**Fig. 2**). The domain of $\vec{\mathcal{R}}$ is uniformly distributed under the prior:

$$p(\mathcal{R}_i|\sigma) = p(\mathcal{R}_i) = \frac{1}{n}. \tag{6}$$

Let $f(x|\sigma)$ be the probability density function (PDF) and let $F(x|\sigma) = \int_0^x f(t|\sigma)\,dt$ be the cumulative distribution function (CDF) of the prior distribution used by the underlying *real* clock model (a Log-Normal distribution in this project). Then, in the *cat* parameterisation, $f(x|\sigma)$ is discretised into n bins and each element within $\vec{\mathcal{R}}$ points to one such bin, where each bin has uniform prior density. The rate of each bin is equal to the median value within the bin

$$r(\mathcal{R}_i) = F^{-1}\big(\frac{\mathcal{R}_i + 0.5}{n}\big), \tag{7}$$

where F^{-1} is the inverse cumulative distribution function (i-CDF).

The key advantage of the *cat* parameterisation is the removal of a term from the posterior density (Equation 1), or more accurately the replacement of a non-trivial $p(\vec{\mathcal{R}}\,|\sigma)$ term with that of a uniform prior. This may facilitate efficient traversal of the parameter space by MCMC.

This parameterisation has been widely used in BEAST and BEAST2 analyses [3]. However, the recently developed constant distance operators – which are incompatible with the *cat* parameterisation – can yield an increase in mixing rate under *real* by up to an order of magnitude over that of *cat*, depending on the dataset [31].

3. Quantiles

Finally, rates can be parameterised as real numbers $0 < \mathcal{R}_i < 1$ which describe the rate's quantile with respect to some underlying clock model distribution. Under the *quant* parameterisation, each of the $2N-2$ elements in $\vec{\mathcal{R}}$ are uniformly distributed.

$$\vec{\mathcal{R}} \in \mathbb{R}^{2N-2}, 0 < \mathcal{R}_i < 1 \tag{8}$$

$$p(\mathcal{R}_i|\sigma) = p(\mathcal{R}_i) = 1 \tag{9}$$

Transforming these quantiles into rates invokes the i-CDF of the underlying *real* clock model distribution. However, this function is computationally demanding to evaluate and therefore an approximation of the i-CDF is used:

$$r(\mathcal{R}_i) = \hat{F}^{-1}(\mathcal{R}_i) \tag{10}$$

where $\hat{F}^{-1}$ is a linear piecewise approximation with 100 pieces. While this approach has clear similarities with *cat*, the domain of rates here is continuous instead of discrete. In this project we extended the family of constant distance operators [31] so that they are compatible with *quant*. Further details on the *quant* piecewise approximation and constant distance operators can be found in **S1 Appendix**.

Clock model operators

The weight of an operator determines the probability of the operator being selected and is typically fixed throughout MCMC. In BEAST2, operators can have its own tunable parameter s which determines the step size of the operator, and this term is learned over the course of the MCMC [10]. Here, we define clock model operators as those which generate proposals for either $\vec{\mathcal{R}}$ or σ. Pre-existing BEAST2 clock model operators are summarised in **Table 1**, and further operators are introduced throughout the paper.

Operator	Description	Parameters	Parameterisations
`RandomWalk`	Moves a single element by a tunable amount.	$\vec{\mathcal{R}}, \sigma$	*cat*, *real*, *quant*
`Scale`	Applies `RandomWalk` on the log-transformation (suitable for parameters with positive domains).	$\vec{\mathcal{R}}, \sigma$	*real*, *quant*
`Interval`	Applies `RandomWalk` on the logit-transformation (suitable for parameters with upper and lower limits).	$\vec{\mathcal{R}}$	*quant*
`Swap`	Swaps two random elements in the vector [3].	$\vec{\mathcal{R}}$	*cat*, *real*, *quant*
`Uniform`	Resamples one element in the vector from a uniform distribution.	$\vec{\mathcal{R}}$	*cat*, *quant*
`ConstantDistance`	Adjusts an internal node height and recalculates all incident branch rates such that the genetic distances remain constant [31].	$\vec{\mathcal{R}}, \mathcal{T}$	*real*, *quant*
`SimpleDistance`	Applies `ConstantDistance` to the root node [31].	$\vec{\mathcal{R}}, \mathcal{T}$	*real*, *quant*
`SmallPulley`	Proposes new branch rates incident to the root such that their combined genetic distance is constant [31].	$\vec{\mathcal{R}}$	*real*, *quant*
`CisScale`	Applies `Scale` to σ. Then recomputes all rates such that their quantiles are constant (for *real* [31]) or recomputes all quantiles such that their rates are constant (*quant*).	$\vec{\mathcal{R}}, \sigma$	*real*, *quant*

Table 1. Summary of pre-existing BEAST2 operators, which apply to either branch rates $\vec{\mathcal{R}}$ or the clock standard deviation σ, and the substitution rate parameterisation they apply to. `ConstantDistance` and `SimpleDistance` also adjust node heights in the tree $\mathcal{T}$.

The family of constant distance operators (`ConstantDistance`, `SimpleDistance`, and `SmallPulley` [31]) are best suited for larger datasets (or datasets with strong

signal) where the likelihood distribution is peaked (**Fig. 2**). While simple one dimensional operators such as `RandomWalk` or `Scale` must make small steps in order to stay "on the ridge" of the likelihood function, the constant distance operators "wander along the ridge" by ensuring that genetic distances are constant after the proposal.

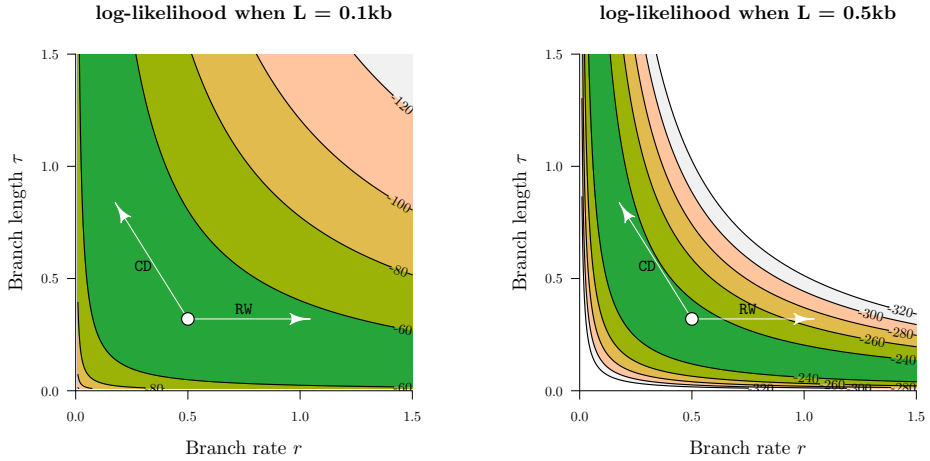

Fig 2. Traversing likelihood space. The z-axes above are the log-likelihoods of the genetic distance $r \times \tau$ between two simulated nucleic acid sequences of length L, under the Jukes-Cantor substitution model [36]. Two possible proposals from the current state (white circle) are depicted. These proposals are generated by the `RandomWalk` (`RW`) and `ConstantDistance` (`CD`) operators. In the low signal dataset ($L = 0.1$kb), both operators can traverse the likelihood space effectively. However, the exact same proposal by `RandomWalk` incurs a much larger likelihood penalty in the $L = 0.5$kb dataset by "falling off the ridge", in contrast to `ConstantDistance` which "walks along the ridge". This discrepancy is even stronger for larger datasets and thus necessitates the use of operators such as `ConstantDistance` which account for correlations between branch lengths and rates.

`Scale` and `CisScale` both operate on the clock model standard deviation σ however they behave differently in the *real* and *quant* parameterisations (**Fig. 3**). In *real*, large proposals of $\sigma \to \sigma'$ made by `Scale` are likely to be rejected due to large penalties in clock model prior $p(\vec{\mathcal{R}}\,|\sigma')$. This led to the development of the fast clock scaler [31] (herein referred to as `CisScale`), which recomputes all branch rates $\vec{\mathcal{R}} \to \vec{\mathcal{R}}\,'$ such that their quantiles under the new clock model prior remain constant $p(\vec{\mathcal{R}}\,|\sigma) = p(\vec{\mathcal{R}}\,'|\sigma')$. In contrast, a proposal made by `Scale` $\sigma \to \sigma'$ under the *quant* parameterisation implicitly alters all branch rates $r(\vec{\mathcal{R}}\,)$ while leaving the quantiles $\vec{\mathcal{R}}$ themselves constant. But by applying `CisScale` under *quant*, all quantiles are recomputed $\vec{\mathcal{R}} \to \vec{\mathcal{R}}'$ such that the respective branch rates are constant i.e. $r(\vec{\mathcal{R}}\,) = r(\vec{\mathcal{R}}\,')$. Overall, `Scale` and `CisScale` propose rates/quantiles in the opposite (trans) or same (cis) space that the clock model is parameterised under (**Fig. 3**).

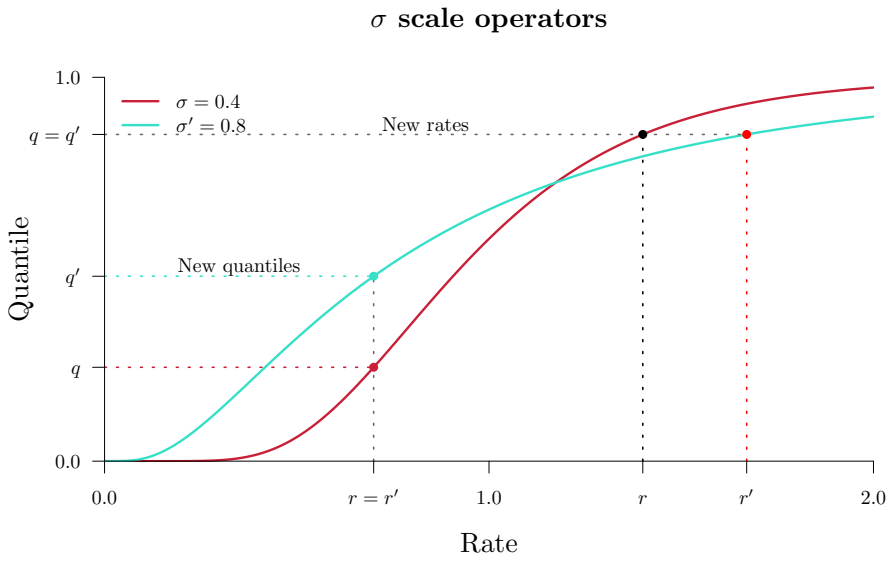

Fig 3. Clock standard deviation scale operators. The two operators above propose a clock standard deviation $\sigma' \leftarrow \sigma \times e^{s\Sigma}$ for some step size $s\Sigma$. Then, either the new quantiles are such that the rates remain constant ("New quantiles", above) or the new rates are such that the quantiles remain constant ("New rates"). In the *real* parameterisation, these two operators are known as `Scale` and `CisScale`, respectively. Whereas, in *quant*, they are known as `CisScale` and `Scale`.

Adaptive operator weighting

It is not always clear which mixture of operators is best for a given dataset. In this article we introduce `AdaptiveOperatorSampler` – a meta-operator which learns operator weights during MCMC and samples operators according to these weights. The meta-operator undergoes three phases. In the first phase (burn-in), `AdaptiveOperatorSampler` samples from its set of sub-operators uniformly at random. In the second phase (learn-in), the meta-operator continues to sample operators uniformly at random however it begins learning several terms detailed below. In its final phase, `AdaptiveOperatorSampler` samples operators (denoted by ω) using the following distribution:

$$p(\omega_i) \propto \begin{cases} 1 & \text{with probability } \Omega \\ \frac{1}{\mathbb{T}(\omega_i)} \sum_{p \in \text{POI}} \sum_{x \in \text{accepts}(\omega_i)} \mathbb{D}(x_p, x'_p) & \text{with probability } 1-\Omega \end{cases} \tag{11}$$

where $\Omega = 0.01$ allows any sub-operator to be sampled regardless of its performance. The parameters of interest (POI) may be either a set of numerical parameters (such as branch rates or node heights), or it may be the tree itself, but it cannot be both. The distance between states x_p and its (accepted) proposal x'_p with respect to parameter p is determined by

$$\mathbb{D}(x_p, x'_p) = \begin{cases} \texttt{RF}(x_p, x'_p)^2 & \text{if } p \text{ is a tree} \\ \frac{1}{|p|}\left[\frac{\|x_p - x'_p\|}{\sigma_p}\right]^2 & \text{if } p \text{ is numerical} \end{cases} \tag{12}$$

where `RF` is the Robinson-Foulds tree distance [37], and $|p|$ is the number of dimensions of numerical parameter p (1 for σ, $2N-2$ for $\vec{\mathcal{R}}$, and $2N-1$ for node heights t). The remaining terms are trained during the second and third phases: the cumulative computational runtime spent on each operator $\mathbb{T}(\omega_i)$, the sample standard deviation σ_p of each numerical parameter p, and the sum distances $\sum_x \mathbb{D}(x_p, x'_p)$.

Under **Equations 11 and 12**, operators which effect larger changes on the parameters of interest, in shorter runtime, are sampled with greater probabilities. Division of the squared distance by a parameters variance σ_p^2 enables comparison between different numerical parameters.

Datasets with strong signal (or large L) are likely to achieve faster convergence when the more precise and meticulous kinds of operators are employed, such as those informed by correlations within the posterior distribution (**Fig. 2**). Whereas, datasets which contain very poor signal (or small L) are likely to mix faster when more weight is placed on bold operators. We therefore introduce the `SampleFromPrior(`$\vec{x}$`)` operator. This operator resamples ψ randomly selected elements within vector $\vec{x}$ from their prior distributions, where $\psi \sim \text{Binomial}(n = |\vec{x}\,|, p = \frac{s}{|\vec{x}\,|})$ for tunable term s. `SampleFromPrior` is always included among the mixture of operators under `AdaptiveOperatorSampler` and serves to make the boldest proposals for datasets with poor signal. Ideally, an operator to the likes of `AdaptiveOperatorSampler` would learn the combination of weights behind these classes of operators best suited for any given dataset.

In this article we apply three instances of the `AdaptiveOperatorSampler` meta-operator to the *real*, *cat*, and *quant* parameterisations. These are summarised in **Table 2**.

Meta-operator	POI	Operators
`AdaptiveOperatorSampler(`σ`)`	σ	`CisScale(`$\sigma, \vec{\mathcal{R}}$`)`
		`RandomWalk(`σ`)`
		`Scale(`σ`)`
		`SampleFromPrior(`σ`)`
`AdaptiveOperatorSampler(`$\vec{\mathcal{R}}$`)`	$\vec{\mathcal{R}}, t$	`ConstantDistance(`$\vec{\mathcal{R}}, \mathcal{T}$`)`
		`RandomWalk(`$\vec{\mathcal{R}}$`)`
		`Scale(`$\vec{\mathcal{R}}$`)`
		`Interval(`$\vec{\mathcal{R}}$`)`
		`Swap(`$\vec{\mathcal{R}}$`)`
		`SampleFromPrior(`$\vec{\mathcal{R}}$`)`
`AdaptiveOperatorSampler(root)`	$\vec{\mathcal{R}}, t$	`SimpleDistance(`$\vec{\mathcal{R}}, \mathcal{T}$`)`
		`SmallPulley(`$\vec{\mathcal{R}}, t$`)`

Table 2. Summary of `AdaptiveOperatorSampler` operators and their parameters of interest (POI). Different operators are applicable to different substitution rate parameterisations (**Table 1**). `AdaptiveOperatorSampler(root)` applies the root-targeting constant distance operators only [31] while `AdaptiveOperatorSampler(`$\vec{\mathcal{R}}$`)` targets all rates and all nodes heights t. These two operators are weighted proportionally to the contribution of the root node to the total node count.

Bactrian proposal kernel

The step size of a proposal kernal $q(x'|x)$ should be such that the proposed state x' is sufficiently far from the current state x to explore vast areas of parameter space, but not so large that the proposal is rejected too often [38]. Operators which attain an acceptance probability of 0.234 are often considered to have arrived at a suitable midpoint between these two extremes [10, 38]. The standard uniform distribution kernel has recently been challenged by the Bactrian kernel [29, 30]. The Bactrian(m) distribution is defined as the sum of two Normal distributions:

$$\Sigma \sim \text{Bactrian}(m) \equiv \frac{1}{2}\text{Normal}(-m, 1-m^2) + \frac{1}{2}\text{Normal}(m, 1-m^2) \qquad (13)$$

where $0 \leq m < 1$ describes the modality of the Bactrian distribution. When $m = 0$, the Bactrian distribution is equivalent to a Normal$(0, 1)$ distribution. As $m \to 1$, the distribution becomes increasingly bimodal (**Fig. 4**). Yang et al. 2013 [29] suggest that Bactrian$(m = 0.95)$ yields a proposal kernel which traverses the posterior distribution more efficiently that the standard uniform kernel, by placing minimal probability on steps which are too small or too large. In this case, a target acceptance probability of around 0.3 is optimal.

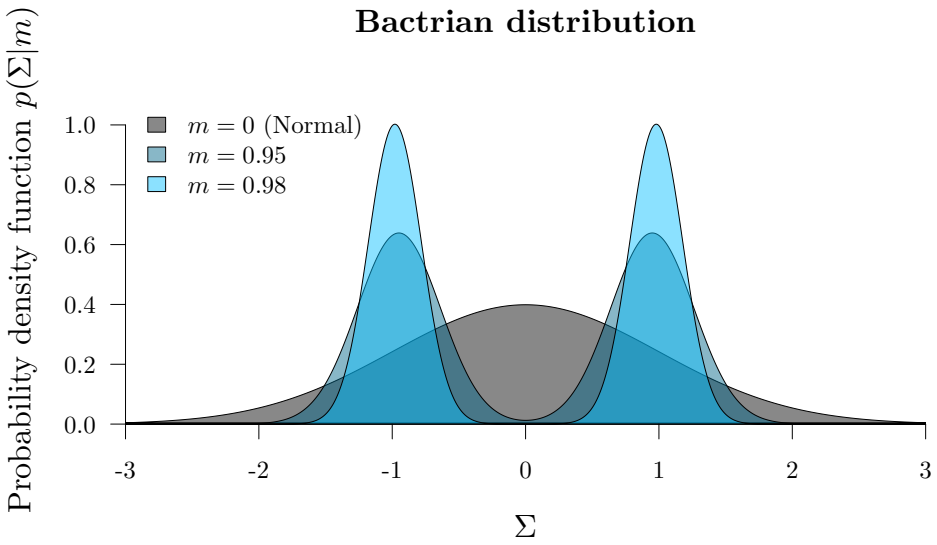

Fig 4. The Bactrian proposal kernel. The step size made under a Bactrian proposal kernel is equal to $s\Sigma$ where Σ is drawn from the above distribution and s is tunable.

In this article we compare the performance of uniform and Bactrian(0.95) proposal kernels with respect to estimating clock model parameters $\vec{\mathcal{R}}$ and σ. The clock model operators which these proposal kernels apply to are described in **Table 3**.

	Operator(s)	Proposal	Parameter x
1	`RandomWalk`	$x' \leftarrow x + s\Sigma$	$\vec{\mathcal{R}}\,, \sigma$
2	`Scale`	$x' \leftarrow x \times e^{s\Sigma}$	$\vec{\mathcal{R}}\,, \sigma$
3	`Interval`	$y \leftarrow \frac{1-x}{x} \times e^{s\Sigma}$ $x' \leftarrow \frac{y}{y+1}$	$\vec{\mathcal{R}}$
4	`ConstantDistance` `SimpleDistance`	$x' \leftarrow x + s\Sigma$	t
5	`SmallPulley`	$x' \leftarrow x + s\Sigma$	$\vec{\mathcal{R}}$
6	`CisScale`	$x' \leftarrow x \times e^{s\Sigma}$	σ

Table 3. Proposal kernels $q(x'|x)$ of clock model operators. In each operator, Σ is drawn from either a Bactrian(m) or Uniform distribution. The scale size s is tunable. `ConstantDistance` and `SimpleDistance` propose tree heights t. The `Interval` operator applies to rate quantiles and respects its domain i.e. $0 < x, x' < 1$.

Narrow Exchange Rate

The `NarrowExchange` operator [39], used widely in BEAST [9, 40] and BEAST2 [10], is similar to nearest-neighbour-interchange [41], and works as follows (**Fig. 5**):

Step 1. Sample an internal/root node E from tree $\mathcal{T}$, where E has grandchildren.

Step 2. Identify the child of E with the greater height. Denote this child as D and its sibling as C (i.e. $t_D > t_C$).

Step 3. Randomly identify the two children of D as A and B.

Step 4. Relocate the $B - D$ branch onto the $C - E$ branch, so that B and C become siblings and their parent is D. All node heights are unchanged.

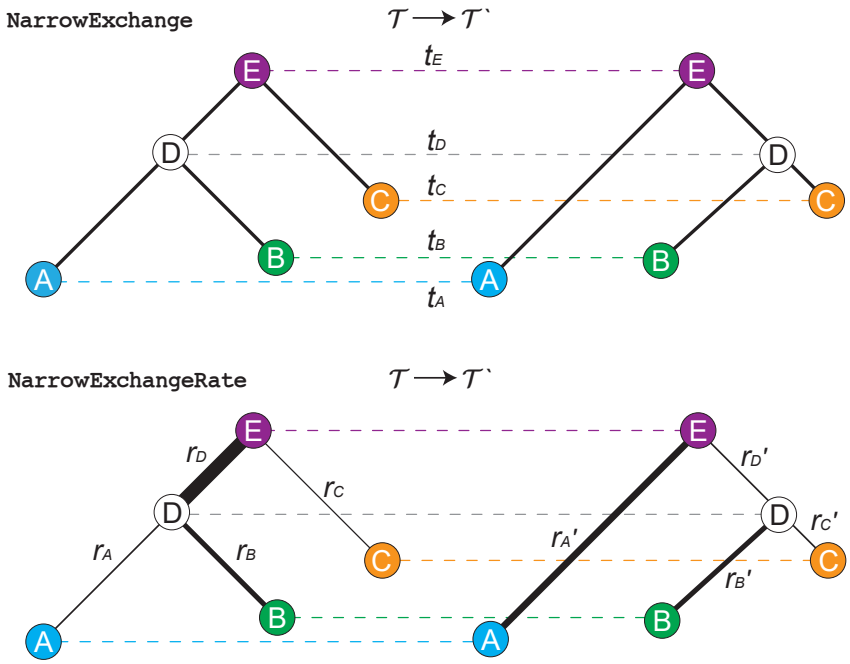

Fig 5. Depiction of `NarrowExchange` and `NarrowExchangeRate` operators. Proposals are denoted by $\mathcal{T} \rightarrow \mathcal{T}'$. The vertical axis corresponds to node height t. In the bottom figure, branch rates r are indicated by line thickness. In this example, the $\mathcal{D}_{AE}$ and $\mathcal{D}_{CE}$ constraints are satisfied.

Lakner et al. 2008 [42] found that tree operators which perterb topology (such as nearest-neighbour-interchange and subtree-prune-and-regraft [41]) consistently perform better than those which also change branch lengths (such as LOCAL [43] and Continuous Change [44]). If `NarrowExchange` was adapted the relaxed clock model by ensuring that genetic distances remain constant after the proposal (akin to the constant distance operators [31]), then its ability to traverse the state space may improve. This may in turn permit proposing a new node height t_D and therefore changing branch lengths.

Here, we present the `NarrowExchangeRate` (NER) operator. Let r_A, r_B, r_C, and r_D be the substitution rates of nodes A, B, C, and D, respectively. In addition to the modest topological change applied by `NarrowExchange`, NER also proposes new branch rates r_A', r_B', r_C', and r_D'. While NER does not alter t_D (i.e. $t_D' \leftarrow t_D$), we also consider NERw - a special case of the NER operator which embarks t_D on a random walk:

$$t_D{}' \leftarrow t_D + s\Sigma \tag{14}$$

for random walk step size $s\Sigma$ where s is a tunable scaler parameter and Σ is drawn from a uniform or **Bactrian proposal kernel**. NER (and NERw) are compatible with both the *real* and *quant* parameterisations. Analagous to the `ConstantDistance` operator, the proposed rates ensure that the genetic distances between nodes A, B, C, and E are constant. There are six pairwise distance between these four nodes and therefore six constraints:

$$\begin{aligned}
\mathcal{D}_{AB}: \quad & r_A(t_D - t_A) + r_B(t_D - t_B) = \\
& r_A{}'(t_E - t_A) + r_D{}'(t_E - t_D{}') + r_B{}'(t_D{}' - t_B) && (15)\\
\mathcal{D}_{AC}: \quad & r_A(t_D - t_A) + r_D(t_E - t_D) + r_C(t_E - t_C) = \\
& r_A{}'(t_E - t_A) + r_D{}'(t_E - t_D{}') + r_C{}'(t_D{}' - t_C) && (16)\\
\mathcal{D}_{AE}: \quad & r_A(t_D - t_A) + r_D(t_E - t_D) = \\
& r_A{}'(t_E - t_A) && (17)\\
\mathcal{D}_{BC}: \quad & r_B(t_D - t_B) + r_D(t_E - t_D) + r_C(t_E - t_D) = \\
& r_B{}'(t_D{}' - t_B) + r_C{}'(t_D{}' - t_C) && (18)\\
\mathcal{D}_{BE}: \quad & r_B(t_D - t_B) + r_D(t_E - t_D) = \\
& r_B{}'(t_D{}' - t_B) + r_D{}'(t_E - t_D{}') && (19)\\
\mathcal{D}_{CE}: \quad & r_C(t_E - t_C) = \\
& r_C{}'(t_D{}' - t_C) + r_D{}'(t_E - t_D{}') && (20)
\end{aligned}$$

Further constraints are imposed by the model itself:

$$r_i > 0 \text{ and } r_i{}' > 0 \text{ for } i \in \{A, B, C, D\} \tag{21}$$

$$\max\{t_B, t_C\} < t_D{}' < t_E. \tag{22}$$

Unfortunately, there is no solution to all six $\mathcal{D}_{ij}$ constraints unless non-positive rates or illegal trees are permitted. Therefore rather than conserving all six pairwise distances, NER conserves a *subset* of distances. It is not immediately clear which subset should be conserved.

Automated generation of operators and constraint satisfaction

The total space of NER operators is comprised of all possible subsets of distance constraints (i.e. $\{\}, \{\mathcal{D}_{AB}\}, \{\mathcal{D}_{AC}\}, \ldots, \{\mathcal{D}_{AB}, \mathcal{D}_{AC}, \mathcal{D}_{AE}, \mathcal{D}_{BC}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$) which are solvable. The simplest NER – the null operator denoted by NER$\{\}$ – does not satisfy any distance constraints. This is equivalent to `NarrowExchange`. To determine which NER variants have the best performance, we developed an automated pipeline for generating and testing these operators.

1. Solution finding. Using standard analytical linear-system solving libraries in MATLAB [45], the $2^6 = 64$ subsets of distance constraints were solved. 54 of the 64 subsets were found to be solvable, and the unsolvables were discarded.

2. Solving Jacobian determinants. The determinant of the Jacobian matrix J is required for computng the Green ratio of the proposal. J is defined as

$$J = \begin{bmatrix} \frac{\partial {r_A}'}{\partial r_A} & \frac{\partial {r_A}'}{\partial r_B} & \frac{\partial {r_A}'}{\partial r_C} & \frac{\partial {r_A}'}{\partial r_D} \\ \frac{\partial {r_B}'}{\partial r_A} & \frac{\partial {r_B}'}{\partial r_B} & \frac{\partial {r_B}'}{\partial r_C} & \frac{\partial {r_B}'}{\partial r_D} \\ \frac{\partial {r_C}'}{\partial r_A} & \frac{\partial {r_C}'}{\partial r_B} & \frac{\partial {r_C}'}{\partial r_C} & \frac{\partial {r_C}'}{\partial r_D} \\ \frac{\partial {r_D}'}{\partial r_A} & \frac{\partial {r_D}'}{\partial r_B} & \frac{\partial {r_D}'}{\partial r_C} & \frac{\partial {r_D}'}{\partial r_D} \end{bmatrix}. \tag{23}$$

Computing the determinant $|J|$ invokes standard analytical differentiation and linear algebra libraries of MATLAB. 6 of the 54 solvable operators were found to have $|J| = 0$, corresponding to irreversible proposals, and were discarded.

3. Automated generation of BEAST2 operators. Java class files are generated using string processing. Each class corresponds to a single operator, extends the class of a meta-NER-operator, and is comprised of the solutions found in **1** and the Jacobian determinant found in **2**. $|J|$ is further augmented if the *quant* parameterisation is employed (**S1 Appendix**). Two such operators are expressed in **Algorithms 1 and 2**.

Algorithm 1 The NER$\{\mathcal{D}_{BC}, \mathcal{D}_{CE}\}$ operator.

procedure PROPOSAL(t_A, t_B, t_C, t_D, t_E, r_A, r_B, r_C, r_D)

  $s\Sigma \leftarrow$ getRandomWalkSize() ▷ Random walk size is 0 unless this is NERw

  ${t'}_D \leftarrow t_D + s\Sigma$ ▷ Propose new node height for D

  ${r'}_A \leftarrow r_A$ ▷ Propose new rates

  ${r'}_B \leftarrow \frac{r_B(t_D - t_B) + r_D(t_E - t_D) + r_D(t_E - {t'}_D)}{{t'}_D - t_B}$

  ${r'}_C \leftarrow \frac{r_C(t_E - t_C) - r_D(t_E - {t'}_D)}{{t'}_D - t_C}$

  ${r'}_D \leftarrow r_D$

  $|J| \leftarrow \frac{(t_D - t_B)(t_E - t_C)}{({t'}_D - t_B)({t'}_D - t_C)}$ ▷ Calculate Jacobian determinant

  return $({r'}_A, {r'}_B, {r'}_C, {r'}_D, {t'}_D, |J|)$

4. Screening operators for acceptance rate using simulated data. The best NER operator variant to proceed to benchmarking on empirical data (**Results**) was selected by performing MCMC on simulated data, measuring the acceptance rates of each of the 96 NER/NERw variants, and comparing them with the null operator NER{} / `NarrowExchange`. In total, there were 300 simulated datasets each with $N = 30$ taxa and varying alignment lengths.

These experiments showed that NER variants which satisfy the genetic distances between nodes B and A (i.e. $\mathcal{D}_{AB}$) or between B and C (i.e. $\mathcal{D}_{BC}$) usually performed worse than the standard `NarrowExchange` operator (**Fig. 6**). This is an intuitive result. If there is high uncertainty in the positioning of B with respect to A and C, then there is no value in respecting either of these distance constraints, and the proposals made to the rates may often be too bold or the Green ratio $|J|$ too small for the proposal to be accepted.

Fig. 6 also reveals a cluster of NER variants which – under the conditions of the simulation – performed better than the null operator NER{} around 25% of the time

Algorithm 2 The NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ operator.

procedure PROPOSAL(t_A, t_B, t_C, t_D, t_E, r_A, r_B, r_C, r_D)

$s\Sigma \leftarrow$ getRandomWalkSize() ▷ Random walk size is 0 unless this is NERw

$t'_D \leftarrow t_D + s\Sigma$ ▷ Propose new node height for D

$r'_A \leftarrow \frac{r_A(t_D - t_A) + r_D(t_E - t_D)}{t_E - t_A}$ ▷ Propose new rates

$r'_B \leftarrow \frac{r_B(t_D - t_B) + r_D(t'_D - t_D)}{t'_D - t_B}$

$r'_C \leftarrow \frac{r_C(t_E - t_C) - r_D(t_E - t'_D)}{t'_D - t_C}$

$r'_D \leftarrow r_D$

$|J| \leftarrow \frac{(t_D - t_A)(t_D - t_B)(t_E - t_C)}{(t_E - t_A)(t'_D - t_B)(t'_D - t_C)}$ ▷ Calculate Jacobian determinant

return $(r'_A, r'_B, r'_C, r'_D, t'_D, |J|)$

and performed worse around 10% of the time. One such operator is NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ (**Algorithm 2**). This variant conserves the genetic distance between nodes A, B, C and their grandparent E. Moreover, this operator performs well when branch rates have a large variance ($\sigma > 0.5$), corresponding to data which is not clock-like. On the other hand, the null operator NER$\{\}$ performs better on shorter sequences ($L < 1$kb) and therefore weaker signal. Overall, NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ outperforms the standard `NarrowExchange` operator when the data is not clock-like and contains enough signal.

Finally, this initial screening showed that applying a (Bactrian) random walk to the node height t_D made the operator worse. This effect was most dominant for the NER variants which satisfy distance constraints (i.e. the operators which are not NER$\{\}$).

Although there were several operators which behaved equivalently during this initial screening process, we selected NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ to proceed to benchmarking (**Results**). Due to the apparent sensitivity of NER operators to the data, we introduce the adaptive operator `AdaptiveOperatorSampler(NER)` which allows the operator scheme to fall back on the standard `NarrowExchange` in the event of NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ performing poorly (**Table 4**).

Meta-operator	POI	Operators
`AdaptiveOperatorSampler(NER)`	$\mathcal{T}$	NER$\{\}$
		NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$

Table 4. The adaptive NER operator. The Robinson-Foulds distance between trees before and after every proposal accept is used to train the operator weights. In the special case of NER proposals, the `RF` distance is always equal to 1.

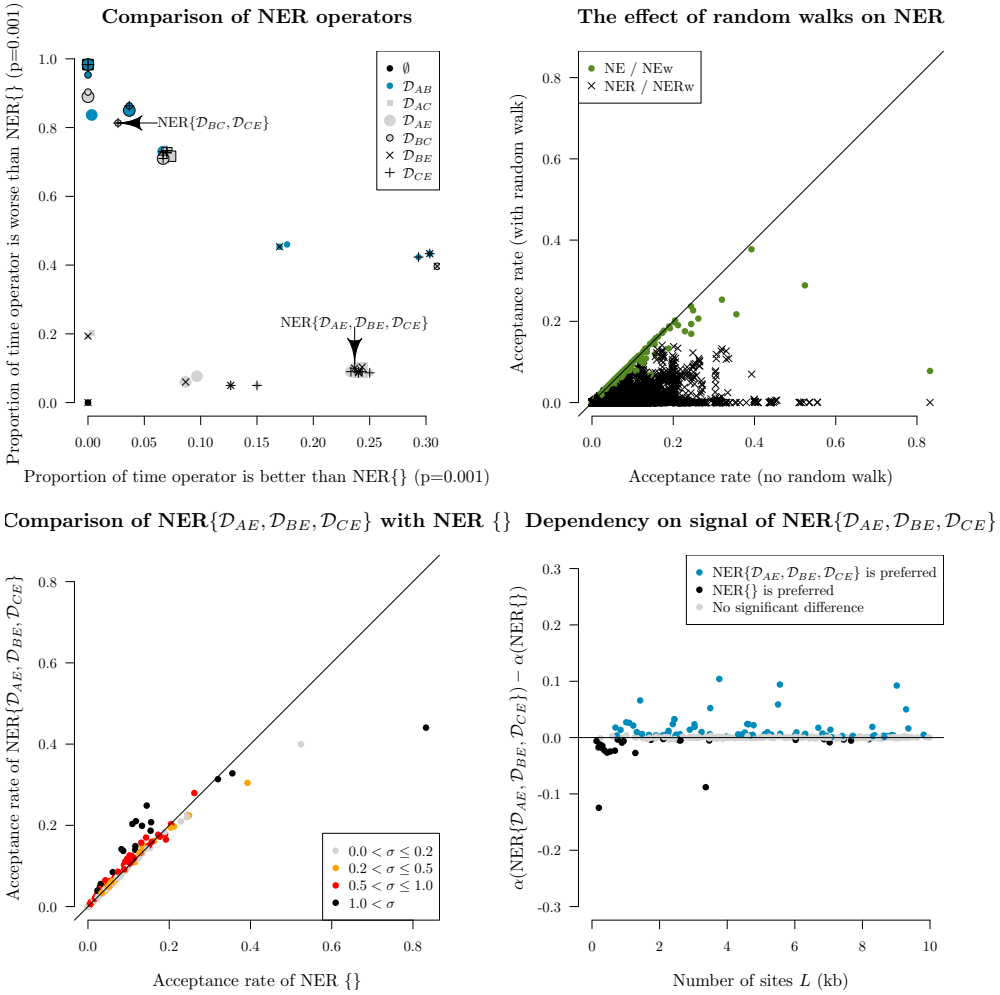

Fig 6. Screening of NER and NERw variants by acceptance rate. Top left: comparison of NER variants with the null operator NER{} / `NarrowExchange`. Each operator is represented by a single point, uniquely encoded by the point stylings. The number of times each operator is proposed and accepted is compared with that of NER{}, and one-sided z-tests are performed to assess the statistical significance between the two acceptance rates ($p = 0.001$). This process is repeated across 300 simulated datasets. The axes of each plot are the proportion of these 300 simulations for which there is evidence that the operator is significantly better than NER{} (x-axis) or worse than NER{} (y-axis). Top right: comparison of NER and NERw acceptance rates. Each point is one NER/NERw variant from a single simulation. Bottom: relationship between the acceptance rates α of NER{$\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}$} and NER{} with the clock model standard deviation σ and the number of sites L. Each point is a single simulation.

An adaptive leaf rate operator

The adaptable variance multivariate normal (AVMVN) kernel learns correlations between parameters during MCMC [28, 40]. Baele et al. 2017 observed a large increase ($\approx 5-10\times$) in sampling efficiency from using the AVMVN kernel substitution model parameters [28]. Here, we consider application of the AVMVN kernel to the branch rates of leaf nodes. This operator, referred to as `LeafAVMVN`, is not readily applicable to internal node branch rates due to their dependencies on tree topology.

Leaf rate AVMVN kernel

The AVMVN kernel assumes its parameters live in $x \in \mathbb{R}^N$ and that these parameters follow a Multivariate Normal distribution with covariance matrix Σ_N. Hence, the kernel operates on the logarithmic or logistic transformation of the N leaf branch rates, depending on the rate parameterisation:

$$x_i = \begin{cases} \log r_i \text{ for } \textit{real} \\ \log \frac{q_i}{1-q_i} \text{ for } \textit{quant} \end{cases} \tag{24}$$

where r_i is a real rate and q_i is a rate quantile. The AVMVN probability density is defined by

$$\mathcal{AVMVN}(x) = \mathcal{MVN}\big(x, (1-\beta)\frac{\Sigma_N}{N} + \beta\frac{\mathbb{I}_N}{N}\big), \tag{25}$$

where $\mathcal{MVN}$ is the Multivariate Normal probability density. β ($= 0.05$) is a constant which determines the fraction of the proposal determined by the identity matrix $\mathbb{I}_N$, as opposed to the covariance matrix Σ_D which is trained during MCMC. Our BEAST2 implementation of the AVMVN kernel is adapted from that of BEAST [40].

`LeafAVMVN` has the advantage of operating on all N leaf rates simultaneously (as well as learning their correlations), as opposed to `ConstantDistance` which operates on at most 2, or `Scale` which operates on at most 1 leaf rate at a time. As the size of the covariance matrix Σ_N grows with the number of taxa N, `LeafAVMVN` is likely to be less efficient with larger taxon sets. Therefore, we learn the weight behind this operator using `AdaptiveOperatorSampler`.

To avoid the learned weight behind `LeafAVMVN` dominating the `AdaptiveOperatorSampler` weighting scheme and therefore inhibiting the mixing of internal node rates, we introduce the `AdaptiveOperatorSampler(leaf)` and `AdaptiveOperatorSampler(internal)` meta-operators which operate exclusively on leaf node rates $\vec{\mathcal{R}}_{\text{leaf}}$ and internal node rates $\vec{\mathcal{R}}_{\text{int}}$ respectively (**Table 5**). The former employs the `LeafAVMVN` operator and learns its weight during MCMC (after providing it sufficient with time to learn Σ_N).

Meta-operator	POI	Operators
`AdaptiveOperatorSampler(leaf)`	$\vec{\mathcal{R}}_{\text{leaf}}, t$	`ConstantDistance(`$\vec{\mathcal{R}}_{\texttt{leaf}}, \mathcal{T}$`)`
		`LeafAVMVN(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
		`RandomWalk(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
		`Scale(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
		`Interval(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
		`Swap(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
		`SampleFromPrior(`$\vec{\mathcal{R}}_{\texttt{leaf}}$`)`
`AdaptiveOperatorSampler(internal)`	$\vec{\mathcal{R}}_{\text{int}}, t$	`ConstantDistance(`$\vec{\mathcal{R}}_{\texttt{int}}, \mathcal{T}$`)`
		`RandomWalk(`$\vec{\mathcal{R}}_{\texttt{int}}$`)`
		`Scale(`$\vec{\mathcal{R}}_{\texttt{int}}$`)`
		`Interval(`$\vec{\mathcal{R}}_{\texttt{int}}$`)`
		`Swap(`$\vec{\mathcal{R}}_{\texttt{int}}$`)`
		`SampleFromPrior(`$\vec{\mathcal{R}}_{\texttt{int}}$`)`

Table 5. Leaf rate $\vec{\mathcal{R}}_{\text{leaf}}$ and internal node rate $\vec{\mathcal{R}}_{\text{int}}$ operators. This division enables the two meta-operators to be weighted proportionally to the number of nodes (leaves or internal) which it applies to. This facilitates incorporation of the `LeafAVMVN` operator, which is only applicable to leaf nodes. In this setup, the `RandomWalk(`x`)`, `Scale(`x`)`, and `SampleFromPrior(`x`)` operators apply to the corresponding set of branch rates x, whereas `ConstantDistance(`$x, \mathcal{T}$`)` is only applicable to internal nodes which have at least one child of type $x \in \{\vec{\mathcal{R}}_{\text{leaf}}, \vec{\mathcal{R}}_{\text{int}}\}$.

Model specification and MCMC settings

In all phylogenetic analyses presented here, we use a Yule [46] tree prior $p(\mathcal{T}|\lambda)$ with birth rate $\lambda \sim \text{Log-Normal}(1, 1.25)$. The clock standard deviation has a $\sigma \sim \text{Gamma}(0.5396, 0.3819)$ prior. Datasets are partitioned into subsequences, where each partition is associated with a distinct HKY substitution model [47]. The transition-transversion ratio $\kappa \sim \text{Log-Normal}(1, 1.25)$, the four nucleotide frequencies $(f_A, f_C, f_G, f_T) \sim \text{Dirichlet}(10, 10, 10, 10)$, and the relative clock rate $\mu_C \sim \text{Log-Normal}(1, 0.6)$ are estimated independently for each partition. The operator scheme ensures that the clock rates μ_C have a mean of 1 across all partitions. To enable the rapid benchmarking of larger datasets we use BEAGLE for high-performance tree likelihood calculations [48] and coupled MCMC with four chains for efficient mixing [27]. The neighbour joining tree [49] is used as the initial state in each MCMC chain.

Operator configurations

Throughout the article, we have introduced four new operators. These are summarised in **Table 6**.

Operator	Description	Parameters
`AdaptiveOperatorSampler`	Samples sub-operators proportionally to their weights, which are learned (see Adaptive operator weighting).	$\vec{\mathcal{R}}, \sigma, \mathcal{T}$
`SampleFromPrior`	Resamples a random number of elements from their prior (see Adaptive operator weighting).	$\vec{\mathcal{R}}, \sigma$
`NarrowExchangeRate`	Moves a branch and recomputes branch rates so that genetic distances are constant (see Narrow Exchange Rate).	$\vec{\mathcal{R}}, \mathcal{T}$
`LeafAVMVN`	Proposals new rates for all leaves in one move (see An adaptive leaf rate operator) [28].	$\vec{\mathcal{R}}$

Table 6. Summary of clock model operators introduced throughout this article. Pre-existing clock model operators are summarised in **Table 1**

In **Table 7**, we define all operator configurations which are benchmarked throughout **Results**.

Configuration	Operator	Weight	*real*	*cat*	*quant*
nocons	`RandomWalk(`$\mathcal{R}$`)`	10	✓		
	`Scale(`$\vec{\mathcal{R}}$`)`	10	✓		
	`Uniform(`$\vec{\mathcal{R}}$`)`	10		✓	✓
	`Interval(`$\vec{\mathcal{R}}$`)`	10			✓
	`Swap(`$\vec{\mathcal{R}}$`)`	10	✓	✓	✓
	`Scale(`σ`)`	10	✓	✓	✓
cons	`ConstantDistance(`$\vec{\mathcal{R}}, \mathcal{T}$`)`	$20 \times \frac{2N-2}{2N-1}$	✓		✓
	`SimpleDistance(`$\vec{\mathcal{R}}, \mathcal{T}$`)`	$10 \times \frac{2N-2}{2N-1}$	✓		✓
	`SmallPulley(`$\vec{\mathcal{R}}$`)`	$10 \times \frac{2N-2}{2N-1}$	✓		✓
	`RandomWalk(`$\vec{\mathcal{R}}$`)`	5	✓		
	`Scale(`$\vec{\mathcal{R}}$`)`	2.5	✓		
	`Uniform(`$\vec{\mathcal{R}}$`)`	5			✓
	`Interval(`$\vec{\mathcal{R}}$`)`	2.5			✓
	`Swap(`$\vec{\mathcal{R}}$`)`	2.5	✓		✓
	`CisScale(`$\sigma, \vec{\mathcal{R}}$`)`	10	✓		✓
	`Scale(`σ`)`	10			✓
adapt	`AdaptiveOperatorSampler(`σ`)`	10	✓	✓	✓
	`AdaptiveOperatorSampler(`$\vec{\mathcal{R}}$`)`	$30 \times \frac{2N-2}{2N-1}$	✓		✓
	`AdaptiveOperatorSampler(`$\vec{\mathcal{R}}$`)`	30		✓	
	`AdaptiveOperatorSampler(root)`	$30 \times \frac{1}{2N-1}$	✓		✓
AVMVN	`AdaptiveOperatorSampler(`σ`)`	10	✓		✓
	`AdaptiveOperatorSampler(leaf)`	$30 \times \frac{N}{2N-1}$	✓		✓
	`AdaptiveOperatorSampler(internal)`	$30 \times \frac{N-2}{2N-1}$	✓		✓
	`AdaptiveOperatorSampler(root)`	$30 \times \frac{1}{2N-1}$	✓		✓
NER{}	`NarrowExchange`	15	✓	✓	✓
NER	`AdaptiveOperatorSampler(NER)`	15	✓		✓

Table 7. Operator configurations and the substitution rate parameterisations which each operator are applicable to. Within each configuration (and substitution rate parameterisation), the weight behind $\mathcal{R}$ sums to 30, the weight of σ is equal to 10, and the weight of NER is equal to 15. Operators which apply to either internal nodes or the root (but not both) are weighted according to leaf count N. The adaptive operators are further broken down in **Tables 2, 4, and 5**. All other operators (i.e. those which apply to which apply to other terms in the state such as the nucleotide substitution model) are held constant within each dataset.

Results

Assessment criteria and datasets

To avoid a cross-product explosion, the five targets for clock model improvement are evaluated sequentially in the following order: **Adaptive operator weighting**, **Substitution rate parameterisations**, **Bactrian proposal kernel**, **Narrow Exchange Rate**, and **An adaptive leaf rate operator**. The four operators introduced in these sections are summarised in **Table 6**. The setting which is considered to be the best in each step is then incorporated into the following step. This protocol and its outcomes are summarised in **Fig. 7**.

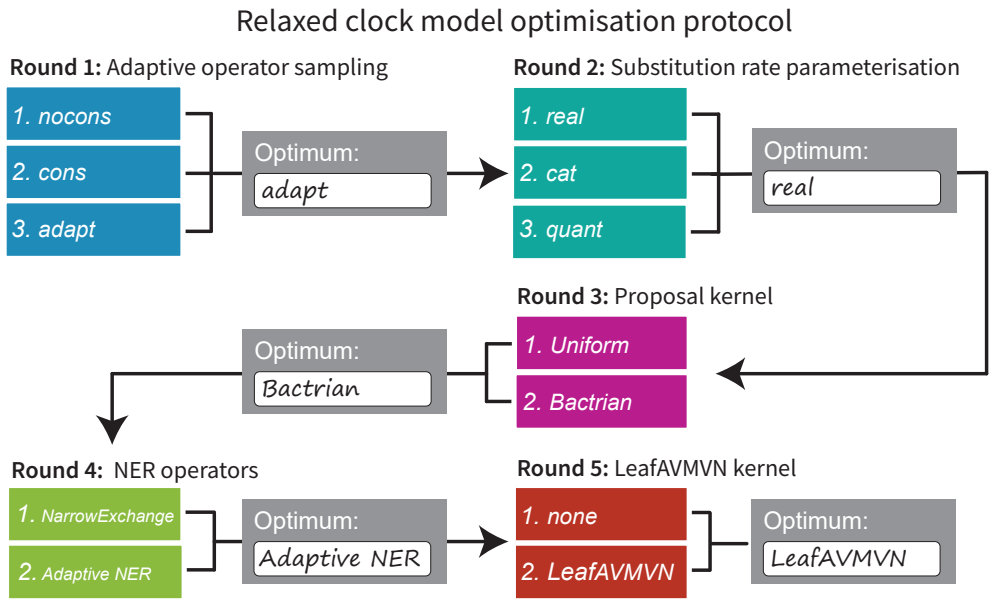

Fig 7. Protocol for optimising clock model methodologies. Each area (detailed in **Models and Methods**) is optimised sequentially, and the best setting from each step is used when optimising the following step.

Methodologies are assessed according to the following criteria.

1. Validation. This is assessed by measuring the coverage of all estimated parameters in well-calibrated simulation studies. These are presented in **S2 Appendix**.

2. Mixing of parameters. Key parameters are evaluated for the number of effective samples generated per hour (ESS/hr). These key parameters are the likelihood $\mathcal{L}$ and prior p densities, tree length l (i.e. the sum of all branch lengths), mean branch rate $\bar{r}$, branch rate of all leaf nodes r, and relaxed clock standard deviation σ. Ee also include the HKY substitution model term κ. The mixing of κ should not be strongly affected by any of the clock model operators, and thus it serves as a positive control in each experiment.

Methodologies are benchmarked using one simulated and eight empirical datasets. The latter were compiled [50] and partitioned [51] by Lanfear as "benchmark alignments" (**Table 8**). Each methodology is benchmarked for million-states-per-hour using the Intel Xeon Gold 6138 CPU (2.00 GHz). These terms are multipled by the ESS-per-state across 20 replicates on the New Zealand eScience Infrastructure (NeSI) cluster to compute the total ESS/hr of each dataset under each setting. All methodologies use identical models and operator configurations, except where a difference is specified.

	N	P	L (kb)	L_{unq} (kb)	**Description**
1	38	16	15.5	10.5	Seed plants (Ran 2018 [52])
2	44	7	5.9	1.8	Squirrel Fishes (Dornburg 2012 [53])
3	44	3	1.9	0.8	Bark beetles (Cognato 2001 [54])
4	51	6	5.4	1.8	Southern beeches (Sauquet 2011 [55])
5	61	8	6.9	4.3	Bony fishes (Broughton 2013 [56])
6	70	3	2.2	0.9	Caterpillars (Kawahara 2013 [57])
7	80	1	10.0	0.9	*Simulated data*
8	94	4	2.2	1	Bees (Rightmyer 2013 [58])
9	106	1	0.8	0.5	Songbirds (Moyle 2016 [59])

Table 8. Benchmark datasets, sorted in increasing order of taxa count N. Number of partitions P, total alignment length L, and number of unique site patterns L_{unq} in the alignment are also specified.

Round 1: A simple operator-weight learning algorithm can improve performance

We compared the nocons, cons, and adapt operator configurations (**Table 7**). nocons contained all of the standard BEAST2 operator configurations and weightings for *real*, *cat*, and *quant*. cons additionally contained (cons)tant distance operators and employed the same operator weighting scheme used previously [31] (*real* and *quant* only). Finally, the adapt configuration combined all of the above applicable operators, as well as the simple-but-bold `SampleFromPrior` operator, and learned the weights of each operator using the `AdaptiveOperatorSampler`.

This experiment revealed that nocons usually performed better than cons on smaller datasets (i.e. small L) while cons consistently performed better on larger datasets (**Fig. 8** and **S3 Appendix**). This result is unsurprising (**Fig. 2**). Furthermore, the adapt setup dramatically improved mixing for *real* by finding the right balance between cons and nocons. This yielded an ESS/hr (averaged across all 9 datasets) 80% faster than cons and 120% faster than nocons, with respect to leaf branch rates, and 250% and 1800% faster for σ. Similar results were observed with *quant*. However adapt neither helped nor harmed *cat*, suggesting that the default operator weighting scheme was sufficient.

This experiment also revealed that the standard `Scale` operator was preferred over `CisScale` for the *real* configuration. Averaged across all datasets, the learned weights behind these two operators were XXX and YYY. This is due to the computationally demanding nature of `CisScale` which invokes the i-CDF function. In contrast, `Scale` and `CisScale` performed equally well under the `quant` configuration and were weighted at XXX and YYY.

Overall, the `AdaptiveOperatorSampler` operator will be included in all subsequent rounds in the tournament.

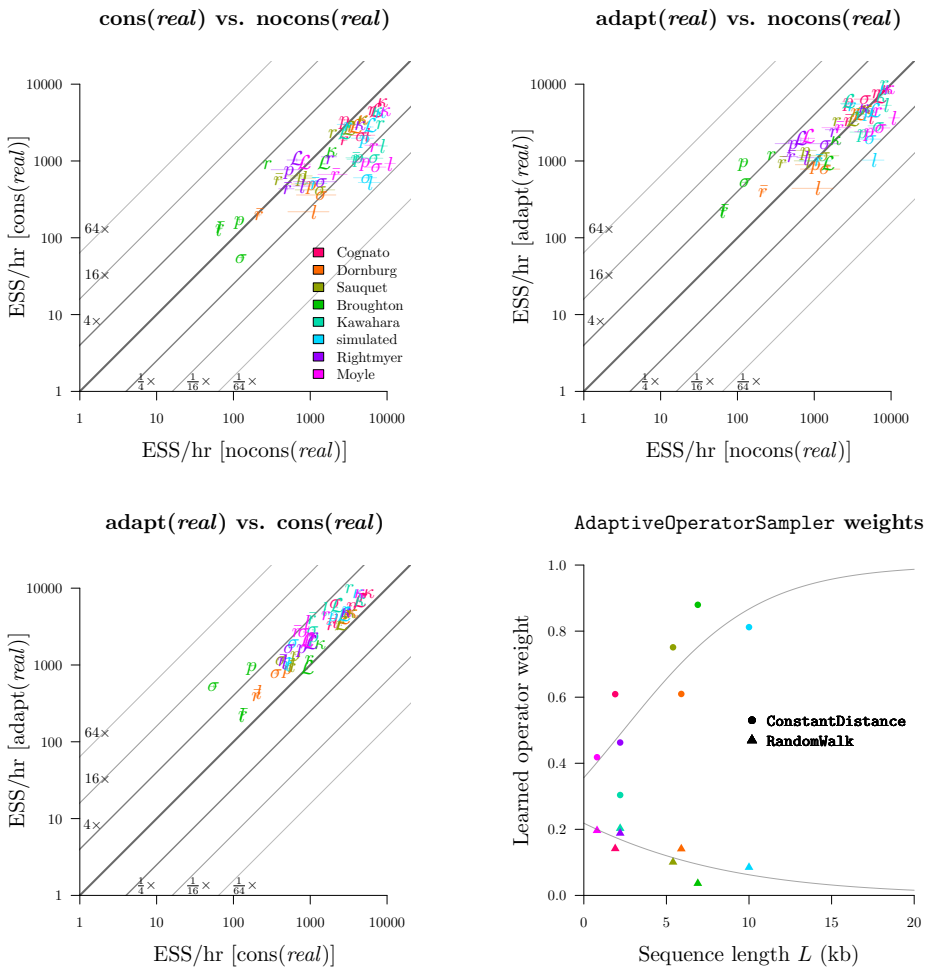

Fig 8. Round 1: benchmarking the `AdaptiveOperatorSampler` operator. Top left, top right, bottom left: each plot compares the ESS/hr (±1 standard error) across two operator configurations. Bottom right: the effect of sequence length L on operator weights learned by `AdaptiveOperatorSampler`. Both sets of results are fit by a logistic regression curve. The benchmark datasets are displayed in **Table 8**. The *cat* and *quant* settings are evaluated in **S3 Appendix**.

Round 2: The *real* parameterisation yielded the fastest mixing

We compared the three rate parameterisations described in **Substitution rate parameterisations**. adapt (*real*) and adapt (*quant*) both employed constant distance tree operators [31] and both used the `AdaptiveOperatorSampler` operator to learn clock model operator weights. Clock model operators weights were also learned in the adapt (*cat*) configuration.

This experiment showed that the *real* parameterisation greatly outperformed *cat* on most datasets (**Fig. 9**). This disparity is strongest for long alignments. In the most extreme case, leaf substitution rates r and clock standard deviation σ both mixed around XXX $\times$ faster on the 15.5 kb seed plant dataset [52] for *real* than they did for *cat*. The advantages in using constant distance operators would likely be even stronger for larger L. Furthermore, *real* outperformed *quant* on most datasets, but this is mostly due to the slow computational performance of *quant* compared with real, as opposed to differences in mixing prowess (**S3 Appendix**).

Overall, we have determined that *real*, and its associated operators, makes the best parameterisation covered here and will proceed to the next rounds of the benchmarking.

Round 3: Bactrian proposal kernels are around 15% more efficient than uniform kernels

We benchmarked the adapt (*real*) configuration with a) standard uniform proposal kernels, and b) Bactrian(0.95) kernels [29]. These kernels applied to all clock model related operators (**Table 3**). These results confirmed that the Bactrian kernel yields faster mixing than the standard uniform kernel (**Fig. 10**). All relevant continuous parameters considered had an ESS/hr, averaged across the 9 datasets, between XXX and YYY% faster compared to with the standard uniform kernel. Bactrian proposal kernels will proceed to round 4 of the relaxed clock model optimisation protocol.

Round 4: NER operators outperform on larger datasets

Our initial screening of the `NarrowExchangeRate` (NER) operators revealed that the NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ operator outperformed the standard NER{} / `NarrowExchange` operator about 25% of the time on simulated data, however it was also very sensitive to the dataset. Therefore we wrapped up the two operators (NER{} and NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$) within an `AdaptiveOperatorSampler` operator so that the appropriate weights can be learned, using the Robinson-Foulds metric. In this round we benchmarked the Bactrian + adapt (*real*) setting with a) NER{} only, and b) adaptive NER (**Table 7**).

Our experiments confirmed that NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ was indeed superior on larger datasets (where $L >$ 2kb; **Fig. 11**). While there was no significant difference in the ESS/hr of continuous parameters, NER$\{\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}\}$ did have an acceptance rate 40% higher than that of the standard `NarrowExchange` operator in the most extreme case (the 7kb alignment by Broughton et al. [56]). Every proposal acceptance changes the topology and thus facilitates traversal of tree space.

In contrast, the standard `NarrowExchange` operator outperformed on smaller datasets. Thus, the new operator is not always helpful and sometimes it even hinders performance. Using an adaptive operator (`AdaptiveOperatorSampler`) removes the burden from the user in making the decision of which operator to use. The `AdaptiveOperatorSampler(NER)` operator will proceed to the next round of the tournament.

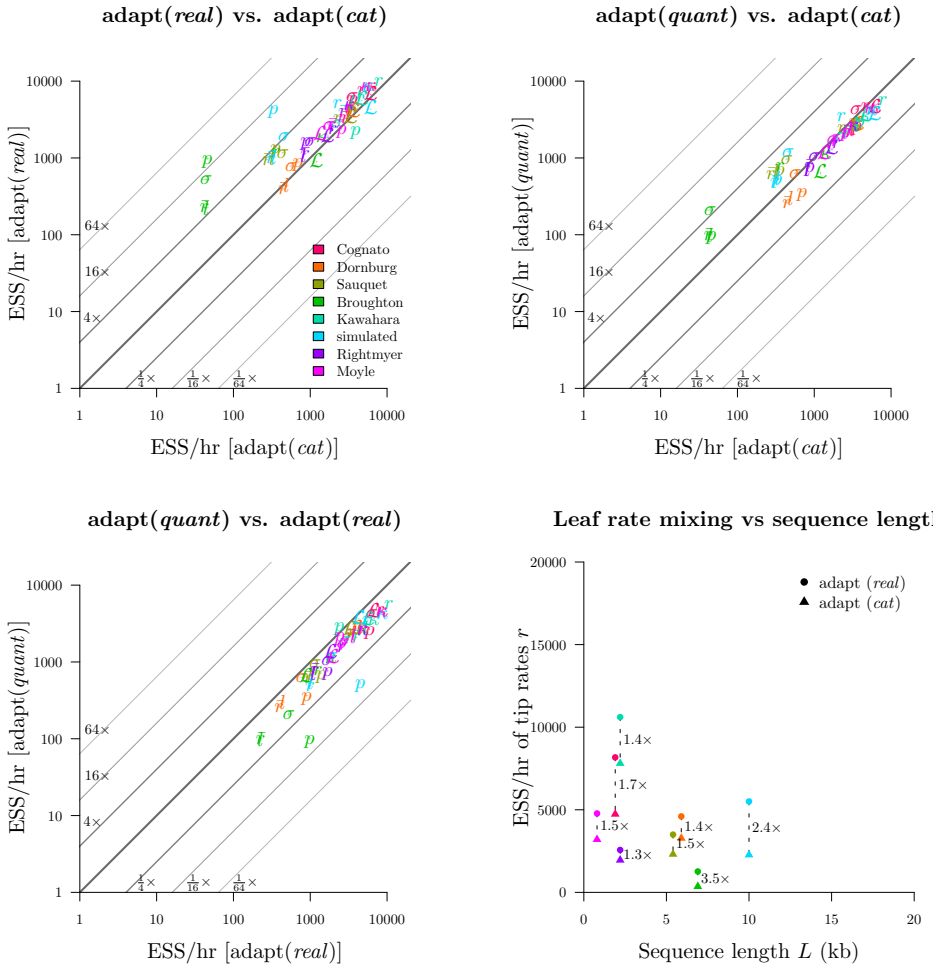

Fig 9. Round 2: benchmarking substitution rate parameterisations. Top left, top right, bottom left: the adapt (*real*), adapt (*cat*), and adapt (*quant*) configurations were compared. Bottom right: comparison of the mean tip substitution rate ESS/hr as a function of alignment length L.

Round 5: The AVMVN leaf rate operator is computationally demanding and improves mixing slightly

We sought to test the applicability of the adaptive AVMVN kernel to leaf rate proposals, by learning the correlation between their substitution rates. To do this, we wrapped the `LeafAVMVN` operator within an `AdaptiveOperatorSampler` (**Table 5**). The two configuations compared here were a) adapt + Bactrian + NER (*real*) and b) AVMVN + NER + Bactrian + adapt (*real*) (**Table 7**).

These results showed that the AVMVN operator yielded slightly better mixing for the tree likelihood, the tree length and the mean branch rate (**Table 12**). However, it also produced slightly slower mixing for κ, reflecting the high computational costs associated with the `LeafAVMVN` operator (**S3 Appendix**). The learned weight of the `LeafAVMVN` operator was quite small (ranging from XXX to YYY across all datasets), again reflecting its high computational costs, but also reinforcing the value in having an adaptive weight operator which penalises slow operators. Overall, the `LeafAVMVN`

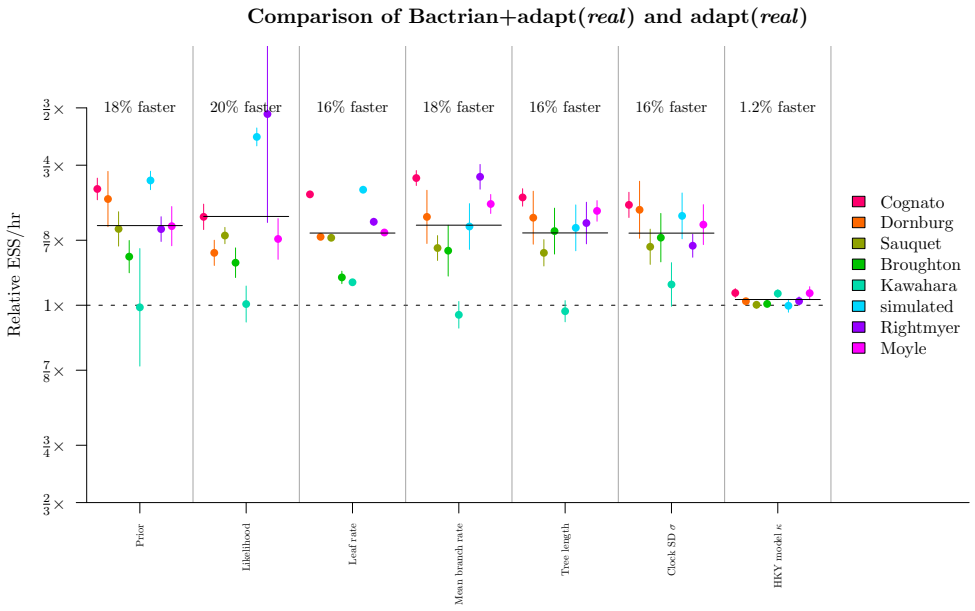

Fig 10. Round 3: benchmarking the Bactrian kernel. The ESS/hr (±1 s.e.) under the Bactrian congifuration, divided by that under the uniform kernel, is shown in the y-axis for each dataset and relevant parameter. Horizontal bars show the geometric mean under each parameter.

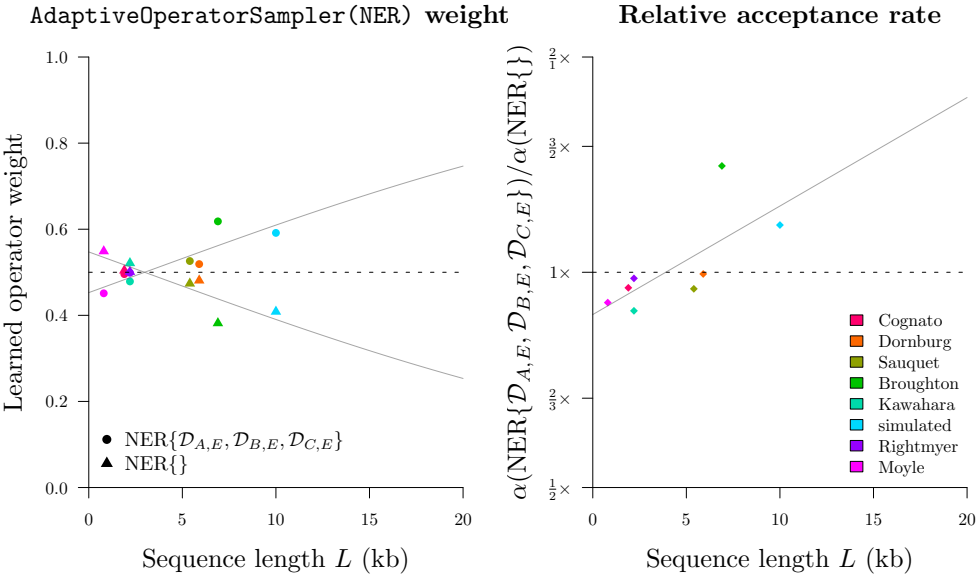

Fig 11. Round 4: benchmarking the NER operators. The learned weights (left) behind the two NER operators (NER{} and NER{$\mathcal{D}_{AE}, \mathcal{D}_{BE}, \mathcal{D}_{CE}$}), and the relative difference between their acceptance rates α (right), are presented as functions of sequence length. Logistic and logarithmic regression models are shown, respectively.

operator provided some, but not much, benefit in its current form.

Overall, this operator configuration is the final winner in the tournamant. In

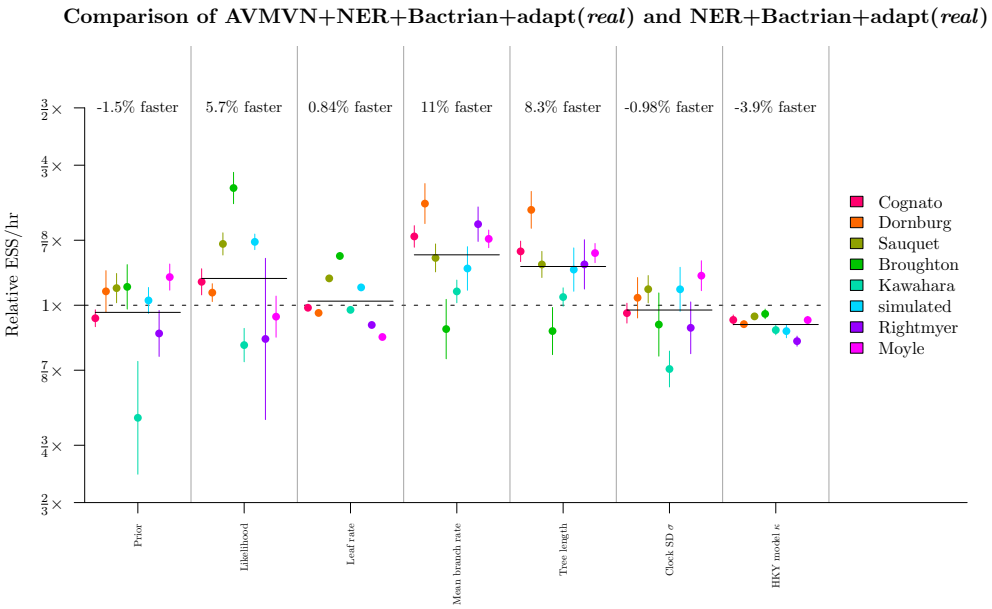

Fig 12. Round 5: benchmarking the `LeafAVMVN` operator. See **Fig. 10** caption for figure notation.

conjunction with all settings which came before it, this setup outperformed both the historic BEAST2 *cat* configuration as well as the recently developed cons (real) [31] scheme. Averaged across all datasets, this configuration yielded a mixing rate between XXX and YYY times as fast as *cat* and between XXX and YYY times as fast as *real*, depending on the parameter. In the most extreme case (Ran et al. 2018 [52]), the new settings were XXX and YYY times as fast respectively. This is likely to be even more extreme for larger alignments.

Discussion

In this article, we delved into the highly correlated structure of relaxed clock models, in order to develop MCMC operators which traverse its posterior space efficiently. We introduced a range of relaxed clock model operators and compared three molecular substitution rate parameterisations. These methodologies were compared by constructing phylogenetic models from several empirical datasets (**Fig. 13**) and comparing their abilties to converge in a tournament-like protocol (**Fig. 7**). This work has produced an operator configuration which explores relaxed clock model space between XXX and YYY times more efficiently than previous setups.

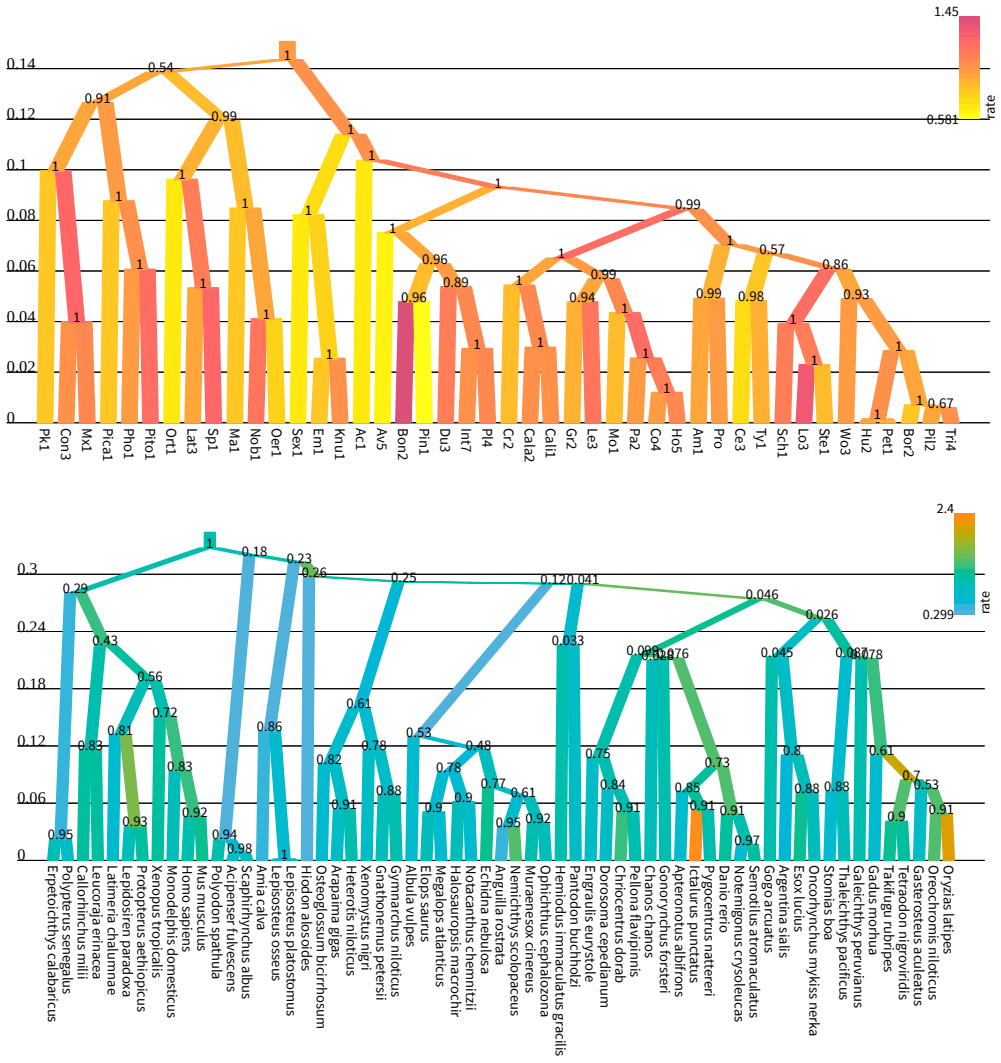

Fig 13. Relaxed clock phylogenies. Maximum clade credibility trees of bark beetles (Cognato et al. 2001 [54]; top) and bony fishes (Broughton et al. 2013 [56]; bottom) are presented. Branches are coloured by substitution rate (units: substitutions per site per unit of time) and the y-axis shows time, such that there is on average 1 substitution per unit of time. Internal nodes are labelled with posterior clade support. Figures generated by UglyTrees [60].

Supporting information

S1 Appendix. Rate quantiles. The linear piecewise approximation used in the *quant* parameterisation is described. `Constant distance` tree operators [31] are extended to the *quant* parameterisation.

S2 Appendix. Well-calibated simulation studies. Methodologies are validated using well-calibated simulation studies.

S3 Appendix. Supplementary results. Presentation and interpretation of further benchmarking results.

References

1. Zuckerkandl E. Molecular disease, evolution, and genetic heterogeneity. Horizons in biochemistry. 1962; p. 189–225.

2. Douzery EJ, Delsuc F, Stanhope MJ, Huchon D. Local molecular clocks in three nuclear genes: divergence times for rodents and other mammals and incompatibility among fossil calibrations. Journal of Molecular Evolution. 2003;57(1):S201–S213.

3. Drummond AJ, Ho SY, Phillips MJ, Rambaut A. Relaxed phylogenetics and dating with confidence. PLoS biology. 2006;4(5):e88.

4. Kuhner MK, Yamato J, Felsenstein J. Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. Genetics. 1995;140(4):1421–1430.

5. Larget B, Simon DL. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. Molecular biology and evolution. 1999;16(6):750–759.

6. Mau B, Newton MA, Larget B. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. Biometrics. 1999;55(1):1–12.

7. Metropolis N. Equation of state calculations by fast computing machines. J Chem Phys. 1953;21:1087–1092.

8. Hastings W. Monte-Carlo sampling methods using Markov chains and their applications. Biometrika. 1970;57:97–109.

9. Drummond AJ, Suchard MA, Xie D, Rambaut A. Bayesian phylogenetics with BEAUti and the BEAST 1.7. Molecular biology and evolution. 2012;29(8):1969–1973.

10. Bouckaert R, Vaughan TG, Barido-Sottani J, Duchêne S, Fourment M, Gavryushkina A, et al. BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. PLoS computational biology. 2019;15(4):e1006650.

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

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

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

14. Gillespie JH. The causes of molecular evolution. vol. 2. Oxford University Press On Demand; 1994.

15. Woolfit M. Effective population size and the rate and pattern of nucleotide substitutions. Biology letters. 2009;5(3):417–420.

16. Loh E, Salk JJ, Loeb LA. Optimization of DNA polymerase mutation rates during bacterial evolution. Proceedings of the National Academy of Sciences. 2010;107(3):1154–1159.

17. Lepage T, Bryant D, Philippe H, Lartillot N. A general comparison of relaxed molecular clock models. Molecular biology and evolution. 2007;24(12):2669–2680.

18. Li WLS, Drummond AJ. Model averaging and Bayes factor calculation of relaxed molecular clocks in Bayesian phylogenetics. Molecular biology and evolution. 2012;29(2):751–761.

19. Faria NR, Quick J, Claro I, Theze J, de Jesus JG, Giovanetti M, et al. Establishment and cryptic transmission of Zika virus in Brazil and the Americas. Nature. 2017;546(7658):406–410.

20. Giovanetti M, Benvenuto D, Angeletti S, Ciccozzi M. The first two cases of 2019-nCoV in Italy: Where they come from? Journal of medical virology. 2020;92(5):518–521.

21. Huelsenbeck JP, Larget B, Swofford D. A compound Poisson process for relaxing the molecular clock. Genetics. 2000;154(4):1879–1892.

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

23. Zhang C, Huelsenbeck JP, Ronquist F. Using parsimony-guided tree proposals to accelerate convergence in Bayesian phylogenetic inference. Systematic Biology. 2020;.

24. Meyer X. Adaptive Tree Proposals for Bayesian Phylogenetic Inference. BioRxiv. 2019; p. 783597.

25. Höhna S, Drummond AJ. Guided tree topology proposals for Bayesian phylogenetic inference. Systematic biology. 2012;61(1):1–11.

26. Altekar G, Dwarkadas S, Huelsenbeck JP, Ronquist F. Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics. 2004;20(3):407–415.

27. Müller NF, Bouckaert R. Coupled MCMC in Beast 2. bioRxiv. 2019;.

28. Baele G, Lemey P, Rambaut A, Suchard MA. Adaptive MCMC in Bayesian phylogenetics: an application to analyzing partitioned data in BEAST. Bioinformatics. 2017;33(12):1798–1805.

29. Yang Z, Rodríguez CE. Searching for efficient Markov chain Monte Carlo proposal kernels. Proceedings of the National Academy of Sciences. 2013;110(48):19307–19312.

30. Thawornwattana Y, Dalquen D, Yang Z, et al. Designing simple and efficient Markov chain Monte Carlo proposal kernels. Bayesian Analysis. 2018;13(4):1037–1063.

31. Zhang R, Drummond A. Improving the performance of Bayesian phylogenetic inference under relaxed clock models. BMC Evolutionary Biology. 2020;20:1–28.

32. Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of molecular evolution. 1981;17(6):368–376.

33. Green PJ. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika. 1995;82(4):711–732.

34. Geyer CJ. The metropolis-hastings-green algorithm; 2003.

35. Gelman A. Parameterization and Bayesian modeling. Journal of the American Statistical Association. 2004;99(466):537–545.

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

37. Robinson DF, Foulds LR. Comparison of phylogenetic trees. Mathematical biosciences. 1981;53(1-2):131–147.

38. Roberts GO, Gelman A, Gilks WR, et al. Weak convergence and optimal scaling of random walk Metropolis algorithms. The annals of applied probability. 1997;7(1):110–120.

39. Drummond AJ, Nicholls GK, Rodrigo AG, Solomon W. Estimating mutation parameters, population history and genealogy simultaneously from temporally spaced sequence data. Genetics. 2002;161(3):1307–1320.

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

41. Semple C, Steel M, et al. Phylogenetics. vol. 24. Oxford University Press on Demand; 2003.

42. Lakner C, Van Der Mark P, Huelsenbeck JP, Larget B, Ronquist F. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. Systematic biology. 2008;57(1):86–103.

43. Simon D, Larget B. Bayesian analysis in molecular biology and evolution (BAMBE) http://www.mathcs.duq.edu/larget/bambe.html. Pittsburgh, Pennsylvania. 1998;.

44. Jow H, Hudelot C, Rattray M, Higgs P. Bayesian phylogenetics using an RNA substitution model applied to early mammalian evolution. Molecular Biology and Evolution. 2002;19(9):1591–1601.

45. Higham DJ, Higham NJ. MATLAB guide. SIAM; 2016.

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

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

48. Ayres DL, Darling A, Zwickl DJ, Beerli P, Holder MT, Lewis PO, et al. BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. Systematic biology. 2012;61(1):170–173.

49. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular biology and evolution. 1987;4(4):406–425.

50. Lanfear R. BenchmarkAlignments https://github.com/roblanf/BenchmarkAlignments. GitHub. 2019;.

51. Lanfear R, Frandsen PB, Wright AM, Senfeld T, Calcott B. PartitionFinder 2: new methods for selecting partitioned models of evolution for molecular and morphological phylogenetic analyses. Molecular biology and evolution. 2016;34(3):772–773.

52. Ran JH, Shen TT, Wang MM, Wang XQ. Phylogenomics resolves the deep phylogeny of seed plants and indicates partial convergent or homoplastic evolution between Gnetales and angiosperms. Proceedings of the Royal Society B: Biological Sciences. 2018;285(1881):20181012. doi:10.1098/rspb.2018.1012.

53. Dornburg A, Moore JA, Webster R, Warren DL, Brandley MC, Iglesias TL, et al. Molecular phylogenetics of squirrelfishes and soldierfishes (Teleostei: Beryciformes: Holocentridae): Reconciling more than 100 years of taxonomic confusion. Molecular Phylogenetics and Evolution. 2012;65(2):727–738. doi:10.1016/j.ympev.2012.07.020.

54. Cognato AI, Vogler AP. Exploring Data Interaction and Nucleotide Alignment in a Multiple Gene Analysis of Ips (Coleoptera: Scolytinae). Systematic Biology. 2001;50(6):758–780. doi:10.1080/106351501753462803.

55. Sauquet H, Ho SYW, Gandolfo MA, Jordan GJ, Wilf P, Cantrill DJ, et al. Testing the Impact of Calibration on Molecular Divergence Times Using a Fossil-Rich Group: The Case of Nothofagus (Fagales). Systematic Biology. 2011;61(2):289–313. doi:10.1093/sysbio/syr116.

56. Broughton RE, Betancur-R R, Li C, Arratia G, Ortí G. Multi-locus phylogenetic analysis reveals the pattern and tempo of bony fish evolution. PLoS Currents. 2013;doi:10.1371/currents.tol.2ca8041495ffafd0c92756e75247483e.

57. Kawahara AY, Rubinoff D. Convergent evolution of morphology and habitat use in the explosive Hawaiian fancy case caterpillar radiation. Journal of Evolutionary Biology. 2013;26(8):1763–1773. doi:10.1111/jeb.12176.

58. Rightmyer MG, Griswold T, Brady SG. Phylogeny and systematics of the bee genus Osmia (Hymenoptera: Megachilidae) with emphasis on North American Melanosmia: subgenera, synonymies and nesting biology revisited. Systematic Entomology. 2013;38(3):561–576.

59. Moyle RG, Oliveros CH, Andersen MJ, Hosner PA, Benz BW, Manthey JD, et al. Tectonic collision and uplift of Wallacea triggered the global songbird radiation. Nature Communications. 2016;7(1). doi:10.1038/ncomms12709.

60. Douglas J. UglyTrees: a browser-based multispecies coalescent tree visualiser. Bioinformatics. 2020;doi:10.1093/bioinformatics/btaa679.