Bayesian Analysis (0000) **00**, Number 0, pp. 1

Bayesian analysis of continuous time Markov chains with application to phylogenetic modelling

Tingting Zhao[*], Ziyu Wang[†], Alexander Cumberworth[‡], Joerg Gsponer[‡], Nando de Freitas[†], and Alexandre Bouchard-Côté[*]

Abstract. Bayesian analysis of continuous time, discrete state space time series is an important and challenging problem, where incomplete observation and large parameter sets call for user-defined priors based on known properties of the process. Generalized linear models have a largely unexplored potential to construct such prior distributions. We show that an important challenge with Bayesian generalized linear modelling of continuous time Markov chains is that classical Markov chain Monte Carlo techniques are too ineffective to be practical in that setup. We address this issue using an auxiliary variable construction combined with an adaptive Hamiltonian Monte Carlo algorithm. This sampling algorithm and model make it efficient both in terms of computation and analyst's time to construct stochastic processes informed by prior knowledge, such as known properties of the states of the process. We demonstrate the flexibility and scalability of our framework using synthetic and real phylogenetic protein data, where a prior based on amino acid physicochemical properties is constructed to obtain accurate rate matrix estimates.

MSC 2010 subject classifications: Primary 60K35, 60K35; secondary 60K35.

Keywords: CTMCs, Bayesian GLMs rate matrix, MCMC, AHMC, Uniformization, Phylogenetic tree.

1 Introduction

Continuous Time Markov Chains (CTMCs) are used to model a variety of data types, including time series, survival, and phylogenetic data (Jukes and Cantor (1969); Kay (1977); Neuts (1981); Kalbfleisch and Lawless (1985), *inter alia*). A CTMC specifies a random piecewise constant path via memoryless holding times on a continuous time axis and jump conditional probabilities over a discrete state space. This description can be conveniently organized into a matrix of parameters called the rate matrix. As the number of discrete states increases, the number of parameters therefore grows quadratically in a naive model.

The practical challenge of reliably estimating a large number of parameters from partially observed time series has motivated the construction of lower-dimensional pa-

[*]Department of Statistics, University of British Columbia, CA, tingting.zhao@stat.ubc.ca; bouchard@stat.ubc.ca

[†]Department of Computer Science, University of Oxford, UK, ziyuw@cs.ox.ac.uk; nando@cs.ox.ac.uk

[‡]Department of Biochemistry and Molecular Biology, Centre for High-throughput Biology, University of British Columbia, CA, alexandercumberworth@gmail.com; gsponer@chibi.ubc.ca

DOI: 0000

rameterizations of rate matrices; one example of this is parameter tying, a practice that is widespread in CTMC modelling (Kimura (1980); Hasegawa et al. (1985), *inter alia*). However, constructing lower-dimensional parameterizations of rate matrices creates another difficulty: the question of selecting one lower dimensional model from an intractable class of possible such models. Prior knowledge helps making these decisions. In DNA evolutionary modelling, for example, biochemical considerations suggest tying *transversion* parameters together (a transversion is a mutation changing purines (adenine or guanine) into pyrimidines (cytosine or thymine), or vice-versa; these mutations are less frequent than the mutations within each class).

This still leaves open many possible ways of tying parameters. First, there are various levels of granularity to be considered, the trade-off being that reducing the number of parameters (going from a fine-grained parameterization to a coarser one) generally achieves better performance on small datasets, but can create a ceiling on the performance on larger datasets. Hierarchical models provide an approach to this issue. However, using a single hierarchy is not always natural. For example, in the protein evolution problem motivating the work in this paper, one could tie parameters based on mass, charge, and a wide range of other largely orthogonal factors.

From a Bayesian perspective, we can view the problem of constructing tied rate matrix parameterizations as a special case of the problem of constructing structured priors on rate matrices. We take this route, applying Generalized Linear Models (GLM) ideas to Bayesian rate matrix modelling. Parameter tying emerges as a special case of this Generalized Linear Model for Continuous Time Markov Chains (GLM-CTMC) structured prior construction. The structured priors we describe also allow the combination of several granularities concurrently, and importantly, non-hierarchical combinations. In fact, we show an example of a structured prior where naive counting of the number of parameters would suggest poor generalization (the number of parameters being larger than the number of upper-triangular entries in the rate matrix of a certain reversible CTMC rate matrix), but where performance is in fact superior for all dataset sizes considered to that of a model with an unstructured prior on a full rank parameterization.

While these GLM-CTMC-based structured priors are useful, their construction is not surprising from a statistical point of view. In fact, their cousins in the maximum likelihood framework are widely known and used (Kalbfleisch and Lawless, 1985). The surprise is rather that the Bayesian counterpart is not more broadly used in applications—with the important exception of a simple special case of our framework corresponding to Bayesian Cox models in survival analysis (Clayton, 1991; Ibrahim et al., 2005).

We show that an important challenge with Bayesian GLM modelling of CTMCs—and perhaps the reason for their lack of more widespread use—is that posterior simulation using classical Markov Chain Monte Carlo (MCMC) techniques can be highly ineffective. We then describe how recent advances in two disparate fields can be combined to address this issue. The first ingredient in our proposed solution is the use of an Adaptive Hamiltonian Monte Carlo (AHMC) algorithm, which uses the gradient of the log target density to efficiently sample from the posterior. The second ingredient comes from a set of techniques called *substitution mapping*, which have undergone intensive developments in the recent years. These techniques allow us to efficiently maintain a set of auxiliary variables that make gradient calculations fast and simple.

Our goal is to be efficient not only in a computational and statistical sense, but also in terms of the modeller's time. While some aspects of the inner working of our method may seem complex, this complexity is masked to the user. The user can specify the prior structure in a declarative fashion without having to worry about the details of the MCMC sampling strategy, parameter tuning, or about gradient calculations. In fact, much of the inner complexity arises from making the tool easy to use, in particular, the attention to automatic parameter tuning explained in Section 4.2. Our open-source implementation, publicly available on github `https://github.com/zhaottcrystal/conifer`, strives to be useable and customizable by non-programmers.

For pedagogical reasons we first discuss our notation and methodology in the context of a general Hidden Markov Model (HMM) (Sections 2, 3, and 4). We then show that our methodology readily extends to reversible processes (Section 5) and to phylogenetic trees (Section 6). We apply our method to amino acid evolutionary modelling, both on synthetic data (Section 7.2, Section 7.3 and Section 7.4) and on real data (Section 7.5). We conclude by discussing limitations and extensions of our work in Section 8.

2 Background and notation

We introduce the notation and background of CTMCs under a HMM framework. We discuss the priors on the rate matrix and existing MCMC algorithms in prior work, followed by a discussion on the limitations of previous approaches.

2.1 Conventions

We use bold letters for (random) vectors and sample paths, and normal fonts for (random) scalars, sets, and matrices. We denote random variable and vectors by capitalization of the symbol used for their realization, with the exception of greek letters, random sets, and random matrices, where we do not draw such distinction. Densities and conditional densities are denoted by f, and (conditional) probability mass functions, by p; subscripts are used in both cases to identify the random variables in question. Unless specified, the reference measures are (products) of Lebesgue measures and are omitted from the notation. The function $\mathbf{1}$ denotes the *indicator function*, i.e. for any set A, $\mathbf{1}_A(x)$ is equal to 1 if and only if $x \in A$, and equal to zero otherwise. To assist the reader, most symbols and abbreviations in the document are hyperlinked to their first occurrence in the text.

2.2 CTMCs

Recall that an (homogeneous) CTMC is a probability distribution over piecewise constant functions $\boldsymbol{X} := (X(t) : t \in \mathcal{T})$ where the abscissa $\mathcal{T} = [0, |\mathcal{T}|)$ is a segment of the real line representing time, and the ordinate is a discrete set $\mathcal{X}$ of states, which we assume to be finite. Without loss of generality, $\mathcal{X} = \{1, 2, \ldots, |\mathcal{X}|\}$. The distribution over the length of each piecewise constant function is exponential, with a rate that depends on the current state, and a transition matrix determines the conditional probability of jumping to a new state $x' \in \mathcal{X}$ given a position x just before the jump.

For computational reasons described in Section 2.3, it is convenient to organize the parameters for these exponential and categorical distributions into a *rate matrix* Q indexed by $\mathcal{X}$. The rate of the exponential departure distribution from state x is given by $-q_{x,x}$, and the probability of transitioning to x' given that a jump is initiated from x is given by $-q_{x,x'}/q_{x,x}$, $x \neq x'$, with the constraints that $q_{x,x} = -\sum_{x' \in \mathcal{X}: x \neq x'} q_{x,x'}$, $q_{x,x'} \geq 0$ for $x \neq x'$. We denote the induced probability model by $\mathbb{P}_Q$.

We start by reviewing the form of the density of fully observed paths in terms of known rate parameters. This density of a sample path $\boldsymbol{x}$ only depends on the following sufficient statistics:

Initial states: for all $x \in \mathcal{X}$, let $n_x(\boldsymbol{x}) \in \{0, 1, 2, \dots\}$ denote the number of paths initialized at state x. Let $\boldsymbol{n}(\boldsymbol{x}) \coloneqq (n_1(\boldsymbol{x}), n_2(\boldsymbol{x}), \dots, n_{|\mathcal{X}|}(\boldsymbol{x}))$.

Sojourn times: let $h_x(\boldsymbol{x}) \in [0, \infty)$ denote the total time spent in state x. In other words, it is the sum of all sojourn times across all jumps and paths. Let $\boldsymbol{h}(\boldsymbol{x}) \coloneqq (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \dots, h_{|\mathcal{X}|}(\boldsymbol{x}))$.

Transition counts: for all distinct $(x, x') \in \mathcal{X}^{\text{distinct}} \coloneqq \{(x, x') \in \mathcal{X}^2 : x \neq x'\}$, let $c_{x,x'}(\boldsymbol{x}) \in \{0, 1, 2, \dots\}$ denote the number of times that a jump ends at state x' right after jumping from x. Let $\boldsymbol{c}(\boldsymbol{x})$ denote the vector of all transition counts organized according to some arbitrary fixed order of $\mathcal{X}^{\text{distinct}}$.

Given a sample path $\boldsymbol{x}$, let $\boldsymbol{z}(\boldsymbol{x}) \coloneqq (\boldsymbol{n}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$ denote the corresponding sufficient statistics. In terms of these statistics $(\boldsymbol{n}, \boldsymbol{h}, \boldsymbol{c}) \coloneqq (\boldsymbol{n}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$, the density of a sample path $\boldsymbol{x}$ from a CTMC with rate matrix Q is given by:

$$\begin{aligned} f_{\mathrm{x}|\mathrm{Q},\Delta}(\boldsymbol{x}|Q, \Delta) \quad \coloneqq \quad & \left(\prod_{x \in \mathcal{X}} p_{\text{ini}}(x)^{n_x} \right) \left(\prod_{(x,x') \in \mathcal{X}^{\text{distinct}}} q_{x,x'}^{c_{x,x'}} \right) \\ & \times \left(\prod_{x \in \mathcal{X}} \exp\left(h_x q_{x,x} \right) \right) \mathbf{1}\left[\boldsymbol{x} \in S(\Delta) \right], \end{aligned} \tag{2.1}$$

where $p_{\text{ini}}(\cdot)$ is the probability mass function of the initial distribution, and $S(\Delta)$ denotes the set of CTMC paths of total length Δ. In Sections 2–4, we assume p_{ini} is fixed and known, reducing the sufficient statistics to $\boldsymbol{z} \coloneqq (\boldsymbol{h}, \boldsymbol{c})$. In Section 5.1, we show how to estimate p_{ini} when it is set to the stationary distribution. When $\Delta = |\mathcal{T}|$, we write $f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{x}|Q) \coloneqq f_{\mathrm{x}|\mathrm{Q},\Delta}(\boldsymbol{x}|Q, |\mathcal{T}|)$ for short. We say that a process is reversible if, for all $t, t' \in \mathcal{T}$, $\mathbb{P}_Q(X_t = x, X_{t'} = x') = \mathbb{P}_Q(X_t = x', X_{t'} = x)$.

2.3 Observations

The sample path is only partially observed in many scenarios of interest. Hence, it is typically not possible to work directly with the simple density given in Equation (2.1).

Let $V \coloneqq \{1, 2, \ldots, |V|\}$ denote a set indexing observation times, and E, the set of consecutive observation index pairs, $E \coloneqq \{(v, v+1) : v \in \{1, 2, \ldots, |V|-1\}\}$. For all $v \in V$, we denote the corresponding observation time by $t_v^{(\text{obs})}$. We denote the sorted times by $\boldsymbol{t}^{(\text{obs})} \coloneqq (t_1^{(\text{obs})}, t_2^{(\text{obs})}, \ldots, t_{|V|}^{(\text{obs})})$, $t_v^{(\text{obs})} < t_{v'}^{(\text{obs})}$ for all $(v, v') \in E$. From the known observed times $\boldsymbol{t}^{(\text{obs})}$, we construct a mapping $\boldsymbol{d}(\cdot)$ that takes a sample path $\boldsymbol{x}$ as input, and outputs a vector containing the state of the CTMC at each of the observation times, $\boldsymbol{d}(\boldsymbol{x}) \coloneqq \big(d_1(\boldsymbol{x}), \ldots, d_{|V|}(\boldsymbol{x})\big)$, where $d_v(\boldsymbol{x}) \coloneqq \boldsymbol{x}\left(t_v^{(\text{obs})}\right)$.

To compute the marginal transition probability p_{trans} between a state x and a state x' separated by an interval of time $\Delta \geq 0$, we use the *matrix exponential* of the rate matrix scaled by Δ:

$$p_{\text{trans}}(x'|x, Q, \Delta) \coloneqq \mathbb{P}_Q\left(X(\Delta) = x' | X(0) = x\right) \tag{2.2}$$

$$= \left(\exp(\Delta Q)\right)_{x,x'}. \tag{2.3}$$

In general, there is no closed form expression in terms of the parameters $q_{x,x'}$ for the matrix exponential in Equation (2.3), but pointwise evaluation can be done in time $\mathcal{O}(|\mathcal{X}|^3)$ using numerical methods (Moler and Van Loan, 1978).[1]

To simplify the notation, we assume that $t_1^{(\text{obs})} = 0$.[2] After marginalization, the probability of a discretely observed sample path $\boldsymbol{D} \coloneqq \boldsymbol{d}(\boldsymbol{X})$ becomes:

$$p_{\text{d|Q}}(\boldsymbol{d}|Q) \coloneqq p_{\text{ini}}(d_1) \prod_{e=(v,v+1)\in E} p_{\text{trans}}\left(d_{v+1}|d_v, Q, \Delta_e\right), \tag{2.4}$$

where $\Delta_{(v,v+1)} \coloneqq t_{v+1}^{(\text{obs})} - t_v^{(\text{obs})}$ is the spacing between consecutive observations.

The random variable $\boldsymbol{D}$ is not necessarily observed directly. Instead, we may only have access to a collection of random variables $\boldsymbol{Y} = (Y_1, \ldots, Y_{|V|})$, conditionally independent given $\boldsymbol{D}$, and with an emission probability or density denoted by $f_{\text{emi}}(y_v|d_v)$:

$$f_{\text{d,y|Q}}(\boldsymbol{d}, \boldsymbol{y}|Q) \coloneqq p_{\text{d|Q}}(\boldsymbol{d}|Q) \prod_{v\in V} f_{\text{emi}}(y_v|d_v). \tag{2.5}$$

The density of the data given a rate matrix would naively involve a sum over an exponential set of latent states:

$$f_{\text{y|Q}}(\boldsymbol{y}|Q) = \sum_{d_1\in\mathcal{X}} \cdots \sum_{d_{|V|}\in\mathcal{X}} f_{\text{d,y|Q}}(\boldsymbol{d}, \boldsymbol{y}|Q), \tag{2.6}$$

but we compute the sum in polynomial time using standard HMM inference methods (see for example Kschischang et al. (2006) for a general treatment of the sum-product

[1]For reversible processes, it is typical to use a method based on spectral decompositions, for general processes, the Padé approximant combined with squaring and scaling is a reasonable default choice (Moler and Van Loan, 1978).

[2]We can assume $t_1^{(\text{obs})} = 0$ without loss of generality. If $t_1^{(\text{obs})} > 0$, modify the initial distribution to $p_{\text{ini}}'(d_1) \coloneqq \sum_{x\in\mathcal{X}} p_{\text{ini}}(x) p_{\text{trans}}(d_1|x, Q, t_1^{(\text{obs})})$, and subtract $t_1^{(\text{obs})}$ to all observed times.

algorithm on factor graphs). This is possible since the density $f_{\mathrm{d,y|Q}}(\boldsymbol{d}, \boldsymbol{y}|Q)$ can be decomposed into a chain-shaped graphical model with a structure given by vertices V and edges E (since Equation (2.4) and (2.5) are products of edge-local and vertex local factors). Excluding the cost of matrix exponentiation,[3] this means that computing Equation (2.6) takes time $\mathcal{O}(|V| \cdot |\mathcal{X}|^2)$. The posterior distribution over the discrete latent states, $p_{\mathrm{d|y,Q}}$, is computed in the same way.

2.4 Priors on Q in prior work

A requirement in the design of priors on Q is that the prior density should have its support containing only valid rate matrices. To do this, it is customary to define a density for the off-diagonal entries of Q, which are organized into a vector of non-negative numbers denoted by $\boldsymbol{\lambda} := (q_{x,x'} : (x, x') \in \mathcal{X}^{\text{distinct}})$. We call this parameterization the General Non-Reversible model (GNR), following Baele et al. (2010). We denote the rate matrix corresponding to $\boldsymbol{\lambda}$ by $Q(\boldsymbol{\lambda})$; the diagonal elements are simply formed by negating the sum of the off-diagonal entries of each row, $q_{x,x} = -\sum_{x' \in \mathcal{X}: x \neq x'} q_{x,x'}$.

Various prior densities f_λ on $\boldsymbol{\lambda}$ can be used. For example, in Huelsenbeck et al. (2002), the authors used independent gamma priors with unit mean and precision γ for each $q_{x,x'}, x \neq x'$:

$$f_\lambda(\boldsymbol{\lambda}) = f_{\text{gam}}(\boldsymbol{\lambda}) := \prod_{(x,x') \in \mathcal{X}^{\text{distinct}}} \mathbf{1}[q_{x,x'} > 0] q_{x,x'}^{1/\gamma - 1} \exp(q_{x,x'}/\gamma). \tag{2.7}$$

There are also priors for reversible matrices, these are reviewed in Section 6.1.

2.5 Existing MCMC algorithms

In this section, we review existing MCMC approaches to posterior approximation of CTMC parameters. We view the paths $\boldsymbol{X}$ as nuisance variables which are marginalized using matrix exponentiation. A state of the MCMC algorithm therefore consists of only the variable $\boldsymbol{\lambda}$. We denote the state at MCMC iteration i by $\boldsymbol{\lambda}^{(i)}$.

Given a state $\boldsymbol{\lambda}^{(i)}$ from the previous iteration, existing MCMC algorithms typically propose the next sample $\boldsymbol{\lambda}^{(i+1)}$ using a Metropolis within Gibbs strategy. For example, in MrBayes, a prominent software package for Bayesian phylogenetic inference (Huelsenbeck and Ronquist, 2001; Ronquist et al., 2012), sliding window and multiplier proposals (where the new value of a univariate variable is perturbed additively or multiplicatively at random), as well as Dirichlet proposals (as an independence sampler) are available. We denote the density of the proposal by f_{prop}. The proposed parameters $\boldsymbol{\lambda}'$ are then

[3]While the cost of one matrix exponential is $\mathcal{O}(|\mathcal{X}|^3)$, performing it $|V|$ times (for each of the time intervals Δ_e) can be done faster than $\mathcal{O}(|V| \cdot |\mathcal{X}|^3)$ for reversible CTMCs(Schadt et al., 1998). Under the reversibility condition, reliable numerical methods can be used to obtain a spectral decomposition of Q, which can be reused across all the time intervals, obtaining a running time of $\mathcal{O}(|\mathcal{X}|^3 + |V| \cdot |\mathcal{X}|^2)$ (Moler and Van Loan, 1978).

accepted with probability:

$$\min\left\{1, \frac{f_\lambda(\boldsymbol{\lambda}')}{f_\lambda(\boldsymbol{\lambda}^{(i)})} \frac{f_{\mathrm{y}|\mathrm{Q}}(\boldsymbol{y}|Q(\boldsymbol{\lambda}'))}{f_{\mathrm{y}|\mathrm{Q}}(\boldsymbol{y}|Q(\boldsymbol{\lambda}^{(i)}))} \frac{f_{\mathrm{prop}}(\boldsymbol{\lambda}^{(i)}|\boldsymbol{\lambda}')}{f_{\mathrm{prop}}(\boldsymbol{\lambda}'|\boldsymbol{\lambda}^{(i)})}\right\},$$

in which case $\boldsymbol{\lambda}^{(i+1)} = \boldsymbol{\lambda}'$, otherwise, $\boldsymbol{\lambda}^{(i+1)} = \boldsymbol{\lambda}^{(i)}$. The computational bottleneck is in computing the factor $f_{\mathrm{y}|\mathrm{Q}}$.

2.6 Limitations of previous approaches

The GNR model can lack flexibility in practice: an important issue is the scalability with respect to the size of the state space, $|\mathcal{X}|$. The number of parameters in a GNR model is $\mathcal{O}(|\mathcal{X}|^2)$, so even for a state space of moderate size, there is often not enough data available to obtain a useful posterior distribution on the parameters. One can reduce the number of parameters by enforcing certain agreements (i.e. tying parameters); for example, setting $q_{1,2} = q_{1,3}$ removes one effective degree of parameterization. This can be applied to several groups of pairs of states believed to behave similarly. For example, in order to model DNA evolution, Kimura (1980) used this parameter tying technique to group all the nucleotide transversions. However, parameter tying can be restrictive, such as when several overlapping ways of grouping states are conceivable. For example, each amino acid can be described with several properties including composition, polarity, and volume; Sneath lists as many as 134 potentially useful side chain properties (Sneath, 1966). Using all of these properties by taking intersections of equivalence classes would get us back to an overparameterized model close to the GNR model we were trying to simplify in the first place.

3 Bayesian rate matrix GLMs

To address the issue described in the previous section, we consider the following model, which we call GLM-CTMC, for the off-diagonal entries of a rate matrix $Q^{(\mathrm{nr})}(\boldsymbol{w})$:

$$q^{(\mathrm{nr})}_{x,x'}(\boldsymbol{w}) := \exp\left\{\langle \boldsymbol{w}, \boldsymbol{\varphi}(x,x')\rangle\right\} \underset{\sim}{q}(x,x'), \tag{3.1}$$

constructed from the following quantities:

Natural parameters (weights): $\boldsymbol{w} \in \mathbb{R}^P$, for some fixed integer P.

Sufficient statistic: $\boldsymbol{\varphi} : \mathcal{X}^{\mathrm{distinct}} \to \mathbb{R}^P$, an arbitrary, user-specified function taking a pair of distinct states as input, $\boldsymbol{\varphi}(x,x') := (\varphi_1(x,x'), \ldots, \varphi_P(x,x'))$.

Base measure: $\underset{\sim}{q}$ as an unnormalized mass function assumed to be known. This could come from a simpler model; for simplicity, we set $\underset{\sim}{q} \equiv 1$.

As before, diagonal entries are defined so that each row sums to zero, $q^{(\mathrm{nr})}_{x,x}(\boldsymbol{w}) := -\sum_{x' \in \mathcal{X}: x \neq x'} q^{(\mathrm{nr})}_{x,x'}(\boldsymbol{w})$. The power of this model comes from the flexibility brought by $\boldsymbol{\varphi}$, which allows the user to encode their prior knowledge. How to do this is explained in the next section.

The model is related to Kalbfleisch and Lawless (1985), however this previous work used GLMs for a different purpose, namely the incorporation of covariate dependences into the rate matrix. The two approaches can be combined: covariates can be integrated to our model by including them as input to the sufficient statistic $\boldsymbol{\varphi}$, but we ignore them to simplify the notation.

3.1 Examples

Going back to the amino acid example from the introduction, the P components of the mapping $\boldsymbol{\varphi}$ can be interpreted as representing different side chain properties, designed from biochemical prior knowledge. Following the terminology from machine learning, we call each of these components a *feature*. We now describe some examples of features in the context of amino acid evolutionary modelling. In a first step, we give a description of a single feature. We then progressively generalize to more general mechanisms for feature generation.

First, we construct a mapping for the purpose of defining classes of amino acids sharing certain physicochemical properties. For example, we show in Table 1 the definition of a mapping called "size," where size : $\mathcal{X} \to \{\mathrm{Micro}, \mathrm{Big}\}$, based on Barnes et al. (2003). As an illustration, for the amino acid Serine (S), $\mathrm{size(S)} = \mathrm{Micro}$. Also, if $e := (x, x') \in \mathcal{X}^{\mathrm{distinct}}$ is a pair of states, we define $\mathrm{size}((x, x'))$ to be the list obtained by applying the map to the individual elements in e. For example, for the amino acids Phenylalanine (F) and Serine (S), $\mathrm{size}((\mathrm{S}, \mathrm{F})) = (\mathrm{Big}, \mathrm{Micro})$.

Using these definitions, we define our first feature as follows:

$$\varphi_1(x, x') := \mathbf{1}_{(\mathrm{Big},\mathrm{Big})}(\mathrm{size}((x, x'))),$$

The feature $\varphi_1(x, x')$ is equal to one if and only if the size of the amino acid is preserved by the substitution between x and x'. We call this first feature "Big→Big" for short. Note that this ties several entries of rate matrix, since more than one pairs $e_1, e_2, \cdots \in \mathcal{X}^{\mathrm{distinct}}$ are such that $\varphi_1(e_i) \neq 0$. Continuing in this fashion, we can build three additional features similarly, "Big→Micro," "Micro→Big," and "Micro→Micro."

Next, we can use another mapping to get additional features. For example, we consider the map polarity : $\mathcal{X} \to \{\text{Non-polar}, \text{Basic Polar}, \text{Polar}, \text{Acidic Polar}\}$ in Table 1, which encodes a combination of polarity and charge properties. The names of these features follow the same convention, yielding features such as "Acidic→Basic." Following a convention from the machine learning literature, we use the terminology *feature template* for a recipe that yields several features. For example, the function size corresponds to a feature template we denote by SIZE, and polarity, to a feature template we denote by POLARITY. Feature templates can be combined to form bigger models. For example, POLARITYSIZE denotes a sufficient statistic vector that contains the concatenation of the features in POLARITY with those in SIZE.

3.2 Prior distributions

Next, we place a prior on $\boldsymbol{w}$, with density denoted by $f_{\mathrm{w}}(\boldsymbol{w})$. A default choice is a Normal prior with mean zero and precision κ, denoted by $f_{\mathrm{nor}}(\boldsymbol{w})$. As an alternative, taking f_{w} to be the density of a product of independent log-gamma distributions, denoted $f_{\mathrm{lga}}(\boldsymbol{w})$ has the interesting property that the corresponding GLM-CTMC models then become a superset of the Bayesian GNR model reviewed in Section 2.4, in the following sense: if we set $P = |\mathcal{X}|(|\mathcal{X}| - 1)$, order the elements of $\mathcal{X}^{\mathrm{distinct}}$ as $e_1, e_2, \ldots, e_P$, and set $(\boldsymbol{\varphi}(x, x'))_m = \mathbf{1}[(x, x') = e_m]$, then the GLM rate matrix model and the specific GNR model introduced in Section 2.4 induce the same marginal distribution over paths.

Following the terminology of Berg-Kirkpatrick et al. (2010), we call each feature used in the reduction described above a *fine feature*. With fine features, $|\varphi_m^{-1}(\mathbb{R}\backslash\{0\})| = 1$, which means there is exactly one pair of states (x, x') associated with feature m; otherwise, when $|\varphi_m^{-1}(\mathbb{R}\backslash\{0\})| > 1$, we call feature m a *coarse feature*, which means that there is more than one pair of states (x, x') that share the same feature.

Asymptotically, only fine features are needed. However, for a finite sample size, a Bayesian GLM-CTMC model containing both coarse and fine features can dominate a model containing only fine features. We show an example in Section 7.3, using the reversible equivalent of the Bayesian GNR model.

4 Scalable approximate posterior simulation

This section addresses the problem of approximation of the density of the posterior distribution given by:

$$f_{\mathrm{w|y}}(\boldsymbol{w}|\boldsymbol{y}) \propto f_{\mathrm{w}}(\boldsymbol{w}) f_{\mathrm{y|Q}}(\boldsymbol{y}|Q^{(\mathrm{nr})}(\boldsymbol{w})). \tag{4.1}$$

In principle, approximating such posterior could be approached using the existing Metropolis-Hastings-based methods described in Section 2.5. However, the rich parameterization of the GLM-CTMC model can create strong correlations among components of $\boldsymbol{w}$ under the posterior distribution, which is problematic since naive Metropolis-Hastings methods tend to struggle in the face of high-dimensional correlated random vectors (Gilks and Roberts, 1996). We show an example of this poor behaviour in Section 7.4.

These correlations are especially severe in overcomplete representations, i.e. when there are $(x_0, x_0') \in \mathcal{X}^{\mathrm{distinct}}$, $\boldsymbol{w}_0 \in \mathbb{R}^P$, $\boldsymbol{w}_0 \neq 0$, such that $\langle \boldsymbol{w}_0, \boldsymbol{\varphi}(x_0, x_0') \rangle = 0$. When there is a large number of features under consideration (for example, the combination of coarse and fine features described in the previous section) or when there is a rich covariate structure, it becomes increasingly impractical to automatically simplify overcomplete feature sets. In the remainder of this section, we describe our new posterior simulation approach that addresses this issue. Sections 4.1 and 4.2 specialize existing methods to our problem, and Section 4.3 introduces an algorithm based on auxiliary variables which improves the scalability of our method.

4.1 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987) offers a remedy to the computational statistics problem described above. HMC is widely-known as a powerful and efficient sampling algorithm, having been demonstrated to outperform many existing MCMC algorithms, especially in problems with high-dimensional, continuous, and correlated distributions (Chen et al., 2001; Neal, 2010).

HMC requires as input the specification of: (i) a *potential energy* function U, which is the negative logarithm of the target density we wish to sample from (up to an additive constant), in our case,

$$\begin{aligned} U(\boldsymbol{w}) &:= -\log f_{\mathrm{w}|\mathrm{y}}(\boldsymbol{w}|Q^{(\mathrm{nr})}(\boldsymbol{w})) && (4.2)\\ &= -\log f_{\mathrm{w}}(\boldsymbol{w}) - \log f_{\mathrm{y}|\mathrm{Q}}(\boldsymbol{y}|Q^{(\mathrm{nr})}(\boldsymbol{w})) + \text{constant}, && (4.3) \end{aligned}$$

and (ii), a *kinetic energy function*, most typically $K(\boldsymbol{m}) = \frac{1}{2}\boldsymbol{m}^T M^{-1}\boldsymbol{m}$, with momentum vector $\boldsymbol{m}$ and a positive definite *mass matrix* M. For standard HMC, the mass matrix is set to the identity matrix I. We review the algorithm in Supplement 1 and defer the technical details of HMC to existing work (Neal, 2010). Note that the HMC algorithm uses the gradient of the joint log density to guide the exploration of the parameter space. In addition, HMC requires the selection of two free parameters: a step-size $\epsilon > 0$ and a number of leapfrog steps $L \in \{1, 2, \ldots\}$. The accepted guidance is to choose the step-size so as to ensure that the sampler's rejection rate is moderate. It is also preferable to have a large L, since this reduces the random walk behaviour of the sampler (Neal, 2010), but too large an L results in unnecessary computation.

HMC has not been widely adopted in practice, due principally to the sensitivity and difficulty of tuning its hyper-parameters ϵ and L. In fact, tuning HMC has been reported by many experts to be more difficult than tuning other MCMC methods (Ishwaran, 1999; Neal, 2010). Fortunately, a few recent methods address the automated tuning of HMC. Two notable approaches are: No U-Turn Sampler (NUTS), by Hoffman and Gelman (2014), which aims to find the best parameter settings by tracking the sample path and preventing HMC from retracing its steps in this path; and Riemann Manifold Hamiltonian Monte Carlo (RMHMC), by Girolami and Calderhead (2011), which exploits the Riemannian geometry of the problem. In this paper, we follow the approach of *adapting* Markov chains using Bayesian optimization (Mahendran et al., 2012; Hamze et al., 2013; Wang et al., 2013) to improve the convergence rate of HMC. In Wang et al. (2013), the authors demonstrate that this approach outperforms NUTS, and that RMHMC is more accurate, but significantly more expensive. RMHMC also makes additional assumptions, namely that higher moments can be efficiently computed.

As in Neal (2010), we consider a slight variation of the HMC algorithm: instead of performing L leapfrog steps at each iteration i, we perform a random number of leapfrog steps, chosen from the discrete uniform distribution over $\{1, \cdots, L\}$, i.e. $L^{(i)} \sim \mathrm{Uni}(\{1, \ldots, L\})$ steps. This approach amounts to using a mixture of L different HMC transition kernels, thus preserving detailed balance (Andrieu et al., 2003).

We denote the distribution induced by sampling one HMC iteration (Line 3 to 15 in Algorithm 1 in Supplement 1) by:

$$\boldsymbol{W}^{(i+1)}|\boldsymbol{W}^{(i)} \sim \mathrm{HMC}(\,\cdot\,|\boldsymbol{W}^{(i)}; U, L, \epsilon). \tag{4.4}$$

Since we resample an independent momentum $\boldsymbol{m}$ at each step, the momentum can be omitted from the above notation.

4.2 Adaptive Hamiltonian Monte Carlo

In order to adapt the MCMC parameters L and ϵ for HMC, we need to (i) define an objective function and (ii) choose a suitable optimization method.

As pointed out by Pasarica and Gelman (2010), a natural objective function for adaptation is the asymptotic efficiency of the MCMC sampler, $(1+2\sum_{k=1}^{\infty}\rho_k)^{-1}$, where ρ_k is the auto-correlation of the sampler with lag k. Despite its appeal, this measure is problematic because the higher order auto-correlations are hard to estimate. To circumvent this problem, they introduced an objective measure called the Expected Squared Jumping Distance (ESJD), $\mathrm{ESJD}(\gamma) := \mathbb{E}_\gamma\|\boldsymbol{W}^{(i+1)} - \boldsymbol{W}^{(i)}\|^2$, where $\gamma = (\epsilon, L)$ denotes the set of parameters for HMC. In practice, the above intractable expectation with respect to the Markov chain is approximated by an empirical estimator, as outlined in Pasarica and Gelman (2010).

The ESJD measure is efficient in situations where the higher order auto-correlations decrease monotonically with respect to ρ_1. However, it is not suitable for tuning HMC samplers since by increasing the number of leapfrog steps one can almost always generate better samples. What we need is a measure that also takes computing time into consideration. With this goal in mind, Wang et al. (2013) introduced the following objective function: $f(\gamma) := \mathrm{ESJD}(\gamma)/\sqrt{L}$. This function simply normalizes the ESJD by the number of leapfrog steps L, thus taking both statistical efficiency and computation into consideration.

Now that we are armed with an objective function, we need to address the issue of optimization. Bayesian optimization is an efficient gradient-free optimization tool well suited for this expensive black-box function problem. Normalized ESJD involves an intractable expectation that can be approximated by sample averages, where the samples are produced by running HMC for a few iterations. Each set of HMC samples for a specific set of hyper-parameters $\gamma \in \Gamma$ results in a noisy evaluation of the normalized ESJD: $r(\gamma) = f(\gamma) + \varepsilon$, where we assume that the measurement noise is Gaussian $\varepsilon \sim \mathcal{N}(0, \sigma_\eta^2)$.

We approach this problem using standard Bayesian optimization methods based on Gaussian processes. To ensure that the adaptation does not alter HMC's invariant distribution, we stop the adaptation process after a fixed number of steps $N_{\text{adapt}} = 3000$. After this horizon, we fix the maximizers $\epsilon^\star$ and $L^\star$ estimated by the Bayesian optimization algorithm. See Algorithm 1 and Supplement 1 for details. In particular, we discuss the additional computational cost when tuning HMC in Section 1.2 of Supplement 1.

Algorithm 1 AHMC

```
1: Given: $\Gamma$, $m$, $k$, $\alpha$, and $\gamma_1$.
2: for $i = 1, 2, \dots,$ do
3:     Run HMC for $m$ iterations with $\gamma_i = (\epsilon_i, L_i)$ (Algorithm 1 in Supplement 1).
4:     Obtain the objective function value $r_i$ using the drawn samples.
5:     Augment the data $\mathcal{D}_i = \{\mathcal{D}_{i-1}, (\gamma_i, r_i)\}$.
6:     if $r_i > \sup_{j \in \{1, \cdots, i-1\}} r_j$ then
7:         $s = \frac{\alpha}{r_i}$
8:     end if
9:     Draw $u \sim \mathcal{U}([0, 1])$
10:    let $p_i = (\max\{i - k + 1, 1\})^{-0.5}$, with $k \in \mathbb{N}^+$.
11:    if $u < p_i$ then
12:        $\gamma_{i+1} := \arg\max_{\gamma \in \Gamma} u(\gamma, s|\mathcal{D}_i)$.
13:    else
14:        $\gamma_{i+1} := \gamma_i$
15:    end if
16: end for
```

4.3 HMC for GLM-CTMC models

The remaining ingredient required by HMC is the gradient of the logarithm of the joint density:

$$\nabla_{\boldsymbol{w}} U(\boldsymbol{w}) \quad = \quad -\nabla_{\boldsymbol{w}} \log f_{\mathrm{w}}(\boldsymbol{w}) - \nabla_{\boldsymbol{w}} \log f_{\mathrm{y}|\mathrm{Q}}(\boldsymbol{y}|Q^{(\mathrm{nr})}(\boldsymbol{w})).$$

The first term is generally trivial, for example, when $f_{\mathrm{w}} = f_{\mathrm{nor}}$, the first term simplifies to $\kappa \boldsymbol{w}$. The second term, which involves the gradient of matrix exponentials, is more challenging, but can still be computed numerically. In previous work, this has been done using a spectral decomposition (Jennrich and Bright, 1976), as automatic differentiation does not apply readily to this problem.[4]

However, computing L leapfrog steps using the method described in Jennrich and Bright (1976) is computationally prohibitive for the models we consider in this paper. More precisely, the method of Jennrich and Bright (1976) has a term of the form $|\mathcal{X}|^3 PL$ in its asymptotic running time. Since much of the previous work has been motivated by frequentist estimators, the number of parameters P has generally been assumed not to be dominant. In contrast, we expect a typical use case of our model to include $\Theta(|\mathcal{X}|^2)$ parameters, since our experimental results show that including all fine features is beneficial in practice (Figure 2). This means that for large state spaces, one MCMC iteration has a running time of $\Theta(L|\mathcal{X}|^5)$, if we were to use existing methods for computing the gradient. This is true even after taking into account the savings coming from reusing spectral decompositions, and careful ordering of vector-matrix product operations—see details in Supplement 2.

[4]For example, the developers of Stan (Stan Development Team, 2014), a popular package for automatic differentiation of statistical applications, do not plan to support automatic differentiation of matrix exponentials in the near future.

In the remaining of this section, we introduce a methodology allowing us to use HMC while avoiding the computational difficulties described above. The key advantage of our method is that it can take advantage of the sparsity in the feature vectors $\boldsymbol{\varphi}(x, x')$. In contrast, previous work has focussed on sparsity of the rate matrix (although we discuss in Section 8 some extensions that can take advantage of both types of sparsity). If we let T_0 denote the running time of the sum-product algorithm (reviewed in Section 2.3), and $s := \sum_{(x,x') \in \mathcal{X}^{\text{distinct}}} \sum_{m=1}^{P} \mathbf{1}[\varphi_m(x, x') \neq 0]$, the total number of non-zero entries across all possible features, then performing L leapfrog steps takes time $\mathcal{O}(Ls + T_0)$ using our method. See Supplement 4 for justification of our running-time analysis. In all the examples we consider in Section 7, $s \in \mathcal{O}(|\mathcal{X}|^2)$.

Assume that $\boldsymbol{w}^{(i-1)}$ denotes the state of the MCMC algorithm at the previous iteration. Our method computes the next state $\boldsymbol{w}^{(i)}$ as follows.

Precomputation: the most expensive parts of the calculations do not need to be recomputed at each leapfrog step within the current MCMC iteration.

1. Construct the HMM described in Section 2.3, and using standard graphical model inference algorithms (Kschischang et al., 2006), compute the forward recursions on the HMM from the previous step, followed by a backward sampling step to impute the values of the latent time series at the discrete set of observation times.

$$\boldsymbol{D}^{(i)} \Big| \boldsymbol{Y}, Q^{(\text{nr})}\left(\boldsymbol{W}^{(i)}\right) \sim p_{\text{d}|\text{y},\text{Q}}\left(\cdot \Big| \boldsymbol{Y}, Q^{(\text{nr})}\left(\boldsymbol{W}^{(i)}\right)\right). \tag{4.5}$$

2. For each pair of consecutive imputed latent states, $d_v^{(i)}, d_{v+1}^{(i)}$, simulate a path $\boldsymbol{X}_e^{(i)}$ of length Δ_e given the two end points $e = (v, v+1)$:

$$\boldsymbol{X}_e^{(i)} \Big| \left(X_{t_v^{(\text{obs})}} = d_v^{(i)}, X_{t_{v+1}^{(\text{obs})}} = d_{v+1}^{(i)} \right) \overset{\propto}{\sim} f_{\text{x}|\text{Q},\Delta}\left(\cdot \Big| Q^{(\text{nr})}\left(\boldsymbol{W}^{(i)}\right), \Delta_e \right). \tag{4.6}$$

Because of the conditioning on the end-point $d_{v+1}^{(i)}$, the density on the right of Equation (4.6) is only specified up to a normalization constant. We address this issue using techniques from the *substitution mapping* literature (Nielsen, 2002; Minin and Suchard, 2008; Rodrigue et al., 2008; Hobolth and Stone, 2009; Tataru and Hobolth, 2011; Rao and Teh, 2013; Irvahn and Minin, 2014), in which a range of methods for end-point simulation have been developed. The method we use is based on uniformization and caching of intermediate quantities. We provide a detailed description of the cached uniformization method in Supplement 3. It is worth noting that cached uniformization can be combined with the work of Tataru and Hobolth (2011) to speed up evaluation of the transition probabilities (a method which additionally is not limited to the reversible case).

3. Compute the sum of the sufficient statistics produced in the previous step, $\boldsymbol{Z}^{(i)} := \sum_e \boldsymbol{z}\left(\boldsymbol{X}_e^{(i)}\right)$.

HMC sampling: perform an HMC step,

$$\boldsymbol{W}^{(i+1)}|\boldsymbol{W}^{(i)}, \boldsymbol{Z}^{(i)} \sim \mathrm{HMC}(\,\cdot\,|\boldsymbol{W}^{(i)}; U_{\boldsymbol{Z}^{(i)}}, L^\star, \epsilon^\star). \tag{4.7}$$

with parameters $\epsilon^\star$ and $L^\star$ tuned using AHMC, and a function evaluation and gradient (assuming a normal prior on the weights) given by:

$$\begin{aligned} U_{\boldsymbol{z}^{(i)}}(\boldsymbol{w}) &= \frac{1}{2}\kappa\|\boldsymbol{w}\|_2^2 - \sum_{x\in\mathcal{X}} h_x^{(i)} q_{x,x}^{(\mathrm{nr})}(\boldsymbol{w}) \qquad (4.8)\\ &\quad - \sum_{(x,x')\in\mathcal{X}^{\text{distinct}}} c_{x,x'}^{(i)} \log\left(q_{x,x'}^{(\mathrm{nr})}(\boldsymbol{w})\right) + \mathrm{cnst} \\ \nabla_{\boldsymbol{w}} U_{\boldsymbol{z}^{(i)}}(\boldsymbol{w}) &= \kappa\boldsymbol{w} + \sum_{(x,x')\in\mathcal{X}^{\text{distinct}}} \boldsymbol{\varphi}(x,x')\left(h_x^{(i)} q_{x,x'}^{(\mathrm{nr})}(\boldsymbol{w}) - c_{x,x'}^{(i)}\right). \qquad (4.9)\end{aligned}$$

Crucially, note that the inner loop, namely computation of Equation (4.9), does *not* involve re-computation of the forward recursions and matrix exponentials (see Supplement 4). This sampling method converges to the desired stationary distribution, namely the distribution with density $f_{\mathrm{w}|\mathrm{y}}(\boldsymbol{w}|\boldsymbol{y})$ given in Equation (4.1). The proof, which is based on an auxiliary variable construction, can be found in Appendix A.

5 Reversible models

The priors we have discussed so far have a support over rate matrices that are not necessarily reversible. In practice, it can be useful to also have priors that only assign positive probability to reversible rate matrices. For example, it is often the case in phylogenetics that non-reversibility is only weakly identifiable (Huelsenbeck et al., 2002), so reversibility is widely assumed in mainstream models.

Another advantages of reversible models is that they allow us to tie the initial distribution p_{ini} shown in Equation (2.1) to the stationary distribution of the CTMC. When there are several time series available, this allows more information coming from the initial values of the time series to be pooled. This also relaxes the assumption made in Section 2.3 that $t_0 = 0$.

In the remainder of this section, we show how to accommodate reversibility constraints within our GLM-CTMC framework.

5.1 Bayesian reversible matrix GLMs

Recall first that a rate matrix is reversible if and only if the off-diagonal entries of the rate matrix can be parameterized as $q_{x,x'} = \theta_{\{x,x'\}}\pi_{x'}$, with $\pi_x = p_{\mathrm{ini}}(x)$ coinciding with the stationary distribution, and $\theta_{\{x,x'\}}$ denoting some non-negative parameters indexed by unordered pairs of distinct states (Tavaré, 1986). We denote the set of unordered distinct pairs of states by $\mathcal{X}^{\text{unordered,dist.}} := \{\{x,x'\} \subset \mathcal{X} : x \neq x'\}$.

To build priors over reversible rate matrices, we consider two collections of exponential families, one for each local decision involved in the generative process (i.e. sampling: initial points, exponentially distributed waiting times, and jumps), defined as follows:

$$\begin{aligned} \theta_{\{x,x'\}}(\boldsymbol{w}) &:= \exp\left\{\langle \boldsymbol{w}, \boldsymbol{\phi}(\{x,x'\})\rangle\right\}\underset{\sim}{\theta}(\{x,x'\}) && (5.1)\\ \pi_x(\boldsymbol{w}) &:= \exp\left\{\langle \boldsymbol{w}, \boldsymbol{\psi}(x)\rangle - A(\boldsymbol{w})\right\}\underset{\sim}{\pi}(x), && (5.2)\\ A(\boldsymbol{w}) &:= \log\sum_{x\in\mathcal{X}} \exp\left\{\langle \boldsymbol{w}, \boldsymbol{\psi}(x)\rangle\right\}, && (5.3)\\ q^{(\text{rev})}_{x,x'}(\boldsymbol{w}) &:= \theta_{\{x,x'\}}(\boldsymbol{w})\pi_{x'}(\boldsymbol{w}), && (5.4) \end{aligned}$$

which in turn depends on the following quantities:

Natural parameters (weights): $\boldsymbol{w} \in \mathbb{R}^P$, for some fixed integer P, which are shared by the θ-exponential family (Equation (5.1)), and the π-exponential family (Equation (5.2)).

Sufficient statistics, coming in two flavours: *bivariate features* $\boldsymbol{\phi} : \mathcal{X}^{\text{unordered,dist.}} \to \mathbb{R}^P$ for the θ-exponential family, taking unordered pairs of states as input; and *univariate features* $\boldsymbol{\psi} : \mathcal{X} \to \mathbb{R}^P$ for the π-exponential family, taking a single state as input. Both types map these contexts into $\mathbb{R}^P$.

Log-normalization: $A : \mathbb{R}^P \to (0,\infty)$ for the π-exponential family. This is just a sum over $|\mathcal{X}|$ elements. Note that the θ-exponential family is a family of measures rather than a probability distribution, therefore there is no need for log-normalization in this case.

Base measures: $\underset{\sim}{\theta}$ and $\underset{\sim}{\pi}$ are known probability mass functions. As in the non-reversible case, these could come from a simpler model. For simplicity, we assume they are identically equal to one to simplify the notation.

5.2 Posterior simulation for reversible models

We follow essentially the same strategy described in Section 4 to approximate the posterior distribution in the reversible case, with the following two modifications. First, we augment the sufficient statistic to include the number of series N_x that are initialized at state x: $\boldsymbol{Z}^{(\text{rev})} := (\boldsymbol{N}, \boldsymbol{C}, \boldsymbol{H})$, where $\boldsymbol{N} := (N_1, \ldots, N_{|\mathcal{X}|})$. Second, we need to modify the expression involved in the calculation of the gradient. We show the derivation of the gradient in Supplement 5.

6 Extension to phylogenetic trees

In this section, we extend our methodology to phylogenetic trees within our GLM-CTMC framework and outline the posterior simulations for phylogenetic models.

6.1 Notation and background on Bayesian phylogenetic inference

Phylogenetics is an important area of application of CTMCs, as the core of many modern phylogenetic models is a CTMC encoding evolution of discrete structures over time—in particular, characters or groups of characters from biological sequences.

Assume that a certain biological sequence has related homologs in a set of organisms, and that these sequences have been aligned (i.e., that for each character in the sequence, we know which characters in the other sequences share a common ancestral character). This sequence can be subsampled so that the evolution of one group of aligned nucleotides or amino acids is approximately independent (or assumed to be independent) to the evolution of any other group included in the dataset. Each such group is called a *site*. The continuous time aspect of the CTMC is used as a continuum approximation of overlapping generations. The states of the CTMC can be nucleotides, amino acids, or more complex structures such as groups of adjacent or interacting sites evolving in a non-independent fashion.

In order to incorporate the phylogenetic tree into our model, we generalize $\mathcal{T}$, which was until now a real segment, into a branching process where line segments are organized into a tree. The phylogenetic tree is based on a discrete directed tree (V, E) called the topology, and a collection of real numbers $\boldsymbol{\Delta} := (\Delta_e : e \in E)$ associated to each edge $e \in E$, called the branch lengths. Formally, we define $\mathcal{T} := \{r\} \sqcup \mathcal{T}_r$, where r is the root, $\sqcup$ denotes a disjoint union, and for all $v \in V$, we set $\mathcal{T}_v := \sqcup_{v':(v,v') \in E}((0, \Delta_e) \sqcup \{v'\} \sqcup \mathcal{T}_{v'})$. The vertices $V \subset \mathcal{T}$ consists in the speciation points and leaves of the tree. Note that we use the same notation as the chain graphical model in Section 2, this highlights the fact that the tree graphical model and the chain graphical model play similar roles. A sample path $\boldsymbol{X} := (X(t) : t \in \mathcal{T})$ on a phylogenetic tree can be simulated recursively as follows:

1. Generate the value $X(r)$ at the root r of the tree (V, E) according to an initial distribution p_{ini}.
2. Given the root of any subtree $v \in V$ and the state $X(v)$ at that node, independently generate the value at each child v' of v as follows:
   (a) simulate a CTMC on one edge, $(X(t) : t \in (0, \Delta_e])$ for a time given by Δ_e, where $e = (v \to v')$,
   (b) set $X(v')$ to the final point of this process, and recurse at v'.

There is one such process $\boldsymbol{X}_k$ for each site $k \in \{1, \ldots, K\}$, and all of these processes share the same tree $\mathcal{T}$. We view the topology and branch lengths as random variables, $\boldsymbol{\tau} := (\boldsymbol{\Delta}, V, E)$ and assume that a prior is defined on those variables, for example uniform over the topologies and independent exponential for each of the branch lengths. The processes $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K$ are conditionally independent given the tree and the evolutionary parameters.

We assume for simplicity that the observations take the form of the value of the process for each leaf $V_{\text{leaves}} \subset V$ and each site, $\boldsymbol{Y} := (Y_1, \ldots, Y_{|V_{\text{leaves}}|})$. Previous work typically uses reversible evolutionary models based on a parameterization of the form $q_{x,x'} = \pi_{x'}\theta_{\{x,x'\}}$. A popular model consists of using a Dirichlet prior on π, and independent gamma priors on θ. This parameterization is known as the General Time Reversible model (GTR).

6.2 Phylogenetic GLM-CTMC model

Since we want a reversible process, we base our evolutionary parameter model on the construction introduced in Section 5. Reversibility in a phylogenetic context means that the location of the root is unidentifiable. We therefore view the topology as undirected, but root the tree arbitrarily when required by computations.

Another source of unidentifiability is that multiplying all the branch lengths Δ_e by a constant while dividing all parameters $\theta_{\{x,x'\}}$s by the same constant keeps the likelihood constant (this can be easily seen in Equation (2.4)). It is therefore common to add a normalization constraint to the rate matrices in the phylogenetic setup, for examples enforcing that the expected number of changes per site is one on a branch of unit length. We follow this convention, making the following amendment to the model of Section 5: first, we define a normalization coefficient β,

$$\beta(\boldsymbol{w}) = -\left(\sum_{x\in\mathcal{X}} \pi_x(\boldsymbol{w}) q^{(\mathrm{rev})}_{x,x}(\boldsymbol{w})\right)^{-1}, \tag{6.1}$$

from which we construct a normalized rate matrix, $q^{(\mathrm{norm})}_{x,x'}(\boldsymbol{w}) \coloneqq \beta(\boldsymbol{w}) q^{(\mathrm{rev})}_{x,x'}(\boldsymbol{w})$.

6.3 Posterior simulation for phylogenetic models

We now discuss how to generalize the techniques of Section 5 to the phylogenetic setup. Since the tree $\boldsymbol{\tau}$ is unknown, we need to augment our MCMC sampler, so that a state in the MCMC chain now has the form $(\boldsymbol{W}^{(i)}, \boldsymbol{\tau}^{(i)})$.

The MCMC moves for this sampler alternate between (1), resampling the weights $\boldsymbol{W}$ while keeping the tree $\boldsymbol{\tau}$ fixed; and (2), resampling the tree $\boldsymbol{\tau}$ while keeping the weights fixed. Weight resampling, Step (1), is performed using the method described in Section 4, with two minor differences. First, the gradient of the normalized reversible rate matrix is modified as described in Section 6.2. Second, the graphical model (V, E) is now a tree instead of a chain. The posterior inference methods for chains can be extended to trees in a straightforward fashion (see Felsenstein (1981) for a description tailored to phylogenetic inference).

Tree resampling, Step (2), can be performed by first forming $Q = Q^{(\mathrm{norm})}(\boldsymbol{w})$ from the weights $\boldsymbol{w}$ we condition on, and then using standard tree resampling methods. These tree resampling methods are based on Metropolis-Hastings local operators on topology and branch lengths. See a comparison of these methods in Lakner et al. (2008). Our package implements stochastic neighbour interchange and multiplicative branch length perturbations. We use a spectral decomposition method to calculate the transition probabilities p_{trans}.

7 Numerical examples

As an illustration of how a typical modeller would interact with our framework, we develop in this section three novel rate matrix models for amino acid data by incorporating various biochemical properties of amino acids. We investigate the behaviour of these models on synthetic and real data. We also perform experiments to assess the computational efficiency of the method, as well as the effectiveness of the automated HMC adaptation method. All the datasets used in the experiments are available at `https://github.com/zhaottcrystal/CTMC-via-HMC`.

7.1 Construction of amino acid evolutionary models

Modelling amino acid evolution is more complex than modelling single nucleotides for the simple reason that the rate matrix is of size 20-by-20 rather than 4-by-4. While the GTR model is popular for modelling nucleotide sequences, there is often not enough data in one protein family to naively rely on the GTR model for amino acid evolutionary modelling.

Several approaches have been proposed to sidestep this issue. The most common strategy is to pool data from several families. This can be done either inside or outside of a CTMC framework (in the latter case, only closely related sequence pairs are used to reduce the probability of multiple replacements) (Dayhoff et al., 1978; Jones et al., 1992; Adachi and Hasegawa, 1996; Yang et al., 1998; Adachi et al., 2000; Müller and Vingron, 2000; Whelan and Goldman, 2001; Dimmic et al., 2002; Kosiol et al., 2007; Le and Gascuel, 2008; Dang et al., 2010). More recent work has introduced non-parametric methods (Dimmic et al., 2000; Lartillot and Philippe, 2004; Huelsenbeck et al., 2008; Murrell et al., 2011; Le et al., 2012) building on previous work based on mixture models (Goldman et al., 1996; Lio and Goldman, 1999).

We take a different approach, and explore in this section how physicochemical properties can be used to design structured priors over substitution rate matrices. This approach is in many ways complementary to existing work on protein evolution modelling. For example, our model could be used as a base measure for the Dirichlet process mixture model of Huelsenbeck et al. (2008). Conversely, several existing models from the literature, for example the work on mechanistic codon models (Doron-Faigenboim and Pupko, 2007; Miyazawa, 2011a,b, 2013) could be translated into features useable in our framework, and easily combined with the physicochemical features described here. We relied on a large literature studying the physicochemical similarities between amino acids to design a set of potentially useful features. Sneath (1966) and Grantham (1974) created models that incorporate polarity, molecular volume, and chemical composition, and allow distances to be calculated, although these were not based on a CTMC. Dan Graur (2000) pointed out that hydrophobicity and polarity were well preserved during the evolutionary process.

The features we use are similar to those described in Section 3.1, with a few modification required to make the model reversible. First, if $\{x, x'\} \in \mathcal{X}^{\text{unordered,dist.}}$ is a *set* of two distinct states, we define size$(\{x, x'\})$ to be the set obtained by applying the map

Size

Micro	A, G, S, P, C, T, N, D
Big	R, E, Q, H, I, L, K, M, F, W, Y, V

Polarity/charge

Non-polar	A, C, G, I, L, M, F, P, W, V
Basic Polar	R, H, K
Polar	N, Q, S, T, Y
Acidic Polar	D, E

Table 1: Size and polarity/charge properties of amino acids

to the individual elements in $\{x, x'\}$. For example, for the amino acids Phenylalanine (F) and Serine (S), $\text{size}(\{\text{S}, \text{F}\}) = \{\text{Big}, \text{Micro}\}$. Using this definition, an example of a reversible bivariate feature is given by:

$$\phi_1(\{x, x'\}) := \mathbf{1}_{\{\text{Big},\text{Big}\}}(\text{size}(\{x, x'\})),$$

Note that we do not take the directionality into account. We call this first feature "BigBig" for short. Continuing in this fashion, we can build two additional features similarly, "BigMicro," and "MicroMicro." This defines the reversible feature template Size. We construct the reversible feature template Polarity similarly.

Another feature template that we consider is the Gtr template, obtained with the same recipe as above applied to the identity map. This is a special case of the fine features described in Section 3. The feature template PolaritySizeGtr consists in the concatenation of Gtr with PolaritySize. The features we have defined so far are all bivariate. For the univariate features (those concerning the stationary distribution rather than the dynamics), we always use fine features in this section. The reason is that the stationary distribution over the 20 amino acids is relatively easy to estimate from the typical dataset size. We implemented our framework in Java, with open source code available at `https://github.com/zhaottcrystal/conifer`. We first tested our implementation using several strategies, including numerical checks of the gradient against the pointwise value of the objective function, as well as the prior/prior-posterior simulator test of Geweke (2004).

7.2 Setup for synthetic data experiments

We use a weight configuration of the PolaritySizeGtr model described in the previous section to get a rate matrix, which is in turn used to generate synthetic amino acid sequences evolving along a phylogenetic tree. We want to show that our method can estimate the held-out rate matrix used to generate the data accurately. We also use this controlled dataset to demonstrate the computational efficiency of our HMC algorithm relative to the classic MCMC techniques.

Using the package from Paradis et al. (2004), we simulate an unrooted bifurcating tree with 10 tips. We use a uniform distribution on $[0, 1]$ on the branch lengths. The tree topology is generated according to Aldous' Markov branching model (Aldous, 1996). The weights for the Gtr features are generated from a uniform distribution ranging from -1 to 1. The values of the weights for the other bivariate polarity/charge, and size features (the coarse features) are listed in Supplement 10. The magnitude and sign of these weights are selected by looking at the posterior weight distributions on real data. We do this since the magnitudes on the coarse features affect the maximum possible benefit that can be gained from using coarse features. The values we use are consistent with the posterior weights values obtained from real data in Figure 5. For example, we show in Figure 5 that both the lower and upper quantiles for the weights of features corresponding to changes between polar amino acids like "AcidicPolar" are supported within the positive real line so that we choose positive weights for them in our experiments. Similarly, the lower and upper quantile are negative for the weights corresponding to the feature "BigMicro" related to the change between different size categories. This motivates us to select negative values for them. As for the univariate feature weights, we use a uniform distribution over -0.2 and 0.2.

We evaluate the accuracy of the rate matrix reconstructions using three error measures. The first one looks at the stationary distribution of the estimated rate matrix, and compares it to the stationary distribution of the held-out rate matrix (that is, the one used to generate the dataset). To compare the two stationary distributions, we compute the Kullback-Leibler (KL) divergence. The second measure is based on relative biases in entries of the marginal transition probability matrices, $p_t(x, x') = \exp(Qt)_{x,x'}$. The relative bias is defined as $\hat{r}(x, x') \coloneqq (\hat{p}_t(x, x') - p_t(x, x'))/p_t(x, x')$, where $p_t(x, x')$ and $\hat{p}_t(x, x')$ are the held-out (respectively, estimated) transition probabilities between states x and x' at the end points of a branch of length t. We show results for $t = 1$ in the main paper, and other values in Supplement 9. Finally, we also measure error using a more direct, but less interpretable metric, the Root-Mean-Square Error (RMSE) on individual rates. We take into account all entries in Q except the diagonal entries, which are redundant because of the constraint that each diagonal entry $Q(x, x)$ is equal to the negative sum of off-diagonal entries of row x.

7.3 Accuracy of rate matrix reconstruction on synthetic data

In this section, we assess the accuracy of our inferred parameters on a synthetic dataset generated under PolaritySizeGtr model. We select the same number of sites (415) and sequences (641) as in our real data experiments (Section 7.5). We summarize the posterior distribution into a single reconstructed rate matrix via the mean of the rate matrix posterior samples.

First, we observed that the stationary distribution reconstruction is very accurate (as measured by the KL divergence). Using 100,000 MCMC iterations with 10,000 as the burnin period, we obtain a KL divergence of 0.0003. See Supplement 9 for a visualization interpreting this result, showing the high quality of the reconstruction in terms of the induced stationary distribution. This confirms our intuition that the stationary distribution is not the challenging part of the problem.

To get a more complete picture of accuracy, we look at the reconstruction error in terms of the relative biases of individual transition probabilities. The $20 \times 20 - 20 = 380$ off-diagonal transition probabilities each have a different relative bias, so we summarize them using a histogram, shown in Figure 1. Values higher than zero correspond to an over-estimation of the transition probability, and values lower than zero correspond

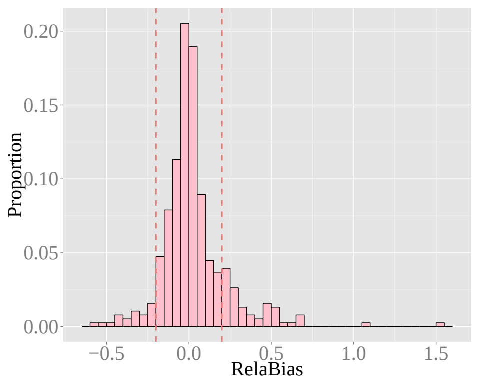

Figure 1: Histogram of the relative biases (described in Section 7.2) of the 380 off-diagonal transition probability matrix ($t = 1$). The two dotted lines label -0.2 and 0.2.

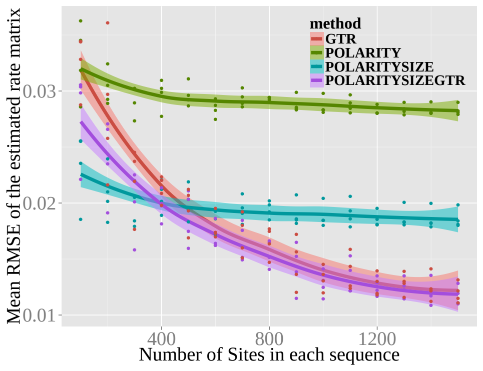

Figure 2: Mean RMSE of estimated rate matrices are shown for the four models described in Section 7.1. The confidence bands are obtained from the Loess smoothing method using a quadratic polynomial regression.

to an under-estimation. The 5% 15%, 25%, 50%, 75% 85% and 95% quantiles of the relative bias without the diagonal elements in the transition probability matrix under $t = 1$ are -0.211, -0.112, -0.062, 0.000, 0.082, 0.176 and 0.423, indicating that 80% of the estimated values range from 21.1% smaller to 17.6% larger than the true values. We find the true transition probability for the elements with relative biases greater than one are 0.0029 and 0.0021 and the estimates are 0.0061 and 0.0053 corresponding to amino acids with low stationary distribution in the synthetic dataset. This validates that large relative biases come from the small transition probabilities. We provide the histograms of the relative bias under $t = 2, 3$ and 5 in Supplement 9. When $t = 5$, we find almost all elements of the relative bias are within 20% difference from the true one.

Next, we investigate the dependency of the estimates on the number and type of features used in the model, and on the size of the dataset. We create 60 synthetic datasets, each based on 10 synthetic sequences related by a shared phylogenetic tree under a PolaritySizeGtr model. The tree topology is the same as the remaining synthetic data experiments in our paper. We generate four independent datasets with 1500 sites. We let the number of sites in each sequence increase from 100 to 1400 with a step size 100 by subsampling from the four independent datasets with 1500 sites. We apply four different models to each of these 60 datasets: Gtr, Polarity, PolaritySize, and PolaritySizeGtr (see Section 7.1). The number of MCMC iterations with the HMC move is 100,000 and we choose a burn-in period of 10,000. Each iteration contains several leapfrog steps. We perform a Geweke diagnostic test (Geweke, 1992) to assess MCMC mixing; the results of this test are in Supplement 10. The averaged wall times (in minutes) of the four independent datasets using the 10 synthetic sequences with 1500 sites under Gtr, Polarity, PolaritySize, PolaritySizeGtr are 2021.9, 1944.5, 1424.5, and 2071.2, respectively. These experiments are run on a cluster of Intel Xeon

E5430 quad-core processors, running at 2.66 GHz.

To summarize the performance of each model on each dataset by a single number, we use the RMSE metric, described in Section 7.2. We plot the mean accuracies in Figure 2 as a function of the number of sites. We also show 95% confidence bands computed using the smoothing method Local Polynomial Regression Fitting (Loess) (Chambers and Hastie, 1991) based on four repeated measures for each site. The smoothed value obtained from the Loess fit is determined by its neighbouring data points using a quadratic polynomial in the regression.

Under a naive asymptotic analysis, one might expect the Gtr model to be unchallengeable. The results show that for a wide range of finite, realistic dataset sizes, it is advantageous to add coarse features to the model. Of course, this is true only when the values of the coarse feature weights are non-zero. Fortunately, our experiments in Section 7.5 show that our model has high confidence that the magnitudes of the weights of several of our physicochemical coarse features are indeed large in a dataset of proteins in the protein kinase domain family. Reassuringly, we also see from Figure 2 that the performance of all models improve as more data becomes available. As expected, for the Polarity and PolaritySize models, this performance increase tapers off since these two models are misspecified in this setup. However, PolaritySize is very effective for small datasets.

We also compare our implementation to MrBayes, a widely-used Bayesian phylogenetics package (Huelsenbeck and Ronquist, 2001; Ronquist et al., 2012). We select a specific set of features, described in Supplement 8, that match the degrees of freedom in MrBayes' GTR configuration. We show in the same Supplement that even in this specific configuration, our model yields a different rate matrix prior compared to MrBayes'. Our experiments therefore focus on measuring the effect that the difference in priors has on the posterior distributions. To do so, we simulate a tree with 10 leaves and 5000 sites using the same strategy as described in Section 7.2. We fix the tree topology and branch lengths to the correct values for both MrBayes and our method. The results are in the Supplement, Figure 5 and 6, and validate that the posterior does not appear to be sensitive to the form of the distribution f_{w} on the individual weights.

7.4 Evaluation of computational efficiency

Next, we seek to assess the computational gains brought by our sampling algorithm. To do so, we simulate a tree with 10 leaves and 2000 sites, using the same strategy as described in Section 7.2. We focus on a single model, PolaritySizeGtr, and compare several samplers. As a baseline, we first perform experiments using a Metropolis-Hastings algorithm with a Normally distributed proposal distribution. Since we observed the performance of this baseline to be highly sensitive to the bandwidth of the proposal, we repeat the experiment with a wide range of bandwidths. We also run an adaptive MCMC algorithm from Roberts and Rosenthal (2009), namely a Normal Proposal Metropolis-Hastings (NMH) algorithm based on a joint Normal proposal distribution with an empirical covariance estimated from the MCMC run. The proposal distribution is given at iteration n by

$$Q_n(\boldsymbol{x}, \cdot) = \begin{cases} N(\boldsymbol{x}, (0.1)^2 I_d/d) & \text{for} \quad n \leqslant 2d, \\ (1-\beta)N(\boldsymbol{x}, (2.38)^2\Sigma_n/d) + \beta N(\boldsymbol{x}, (0.1)^2 I_d/d) & \text{for} \quad n > 2d, \end{cases}$$

where $\boldsymbol{x}$ is the estimate in the previous iteration, d is the dimension of the parameters, I_d is an identity matrix of dimension d and Σ_n is the current empirical estimate of the covariance matrix based on the run so far and β is a small positive constant (we take β=0.05, as suggested in Roberts and Rosenthal (2009)). It is known from Roberts et al. (1997) and Roberts et al. (2001) that the proposal $N(x, (2.38)^2\Sigma/d)$ is optimal in a particular large-dimensional context. We compared these baseline samplers to our tuning-free sampling method, which combines auxiliary variables and AHMC (see Section 4).

We show a trace plot in Figure 8 of Supplement 9 of the weights related to a randomly chosen feature "AD" which represents the GTR bivariate feature for amino acid "A" and "D" using the NMH kernels with bandwidth 0.01 denoted as NMH 0.01. The trace plot exhibits clear signs of poor mixing. Then we show the trace obtained from the HMC running in the same amount of time (2267 minutes) in the same plot. Within each MCMC iteration, the number of leapfrog steps L is 18 with a step-size ϵ of 0.0085. However, it is clear from the trace plot that the NMH is severely autocorrelated, whereas HMC efficiently explores the space.

We quantitatively evaluate the computational efficiency by monitoring the Effective Sample Size (ESS) per second. This measure provides an estimate of the number of independent samples that the chain generates; such a measure is needed as MCMC produces correlated samples, and a higher number of steps per second does not necessarily mean a higher efficiency, as the samples could be more correlated. We use the package from Plummer et al. (2006) to compute the ESS. We calculate the ESS per second for each feature weight and report the extrema and quantile ESS per second across the different feature weights.

We repeat each experiment four times and average the results. We show the results in Figure 3. We scale all results by 10^4 seconds for ease of presentation. See Supplement 10 for results on a wider range of different bandwidth. Our method achieves a higher efficiency compared to classical NMH kernels with all the bandwidth tested, as well as compared to the adaptive NMH algorithm. Even when comparing our sampler to the "oracle" NMH baseline, the minimum ESS per second is superior by an order of magnitude, and the quantiles, by a factor of at least two. Our sampler outperforms adaptive NMH by a factor of four in terms of the minimum ESS per second.

Next, we investigate the effectiveness of our Bayesian optimization adaptation scheme from Section 4.2. We simulate a tree with 10 leaves and amino acid sequences with 500 sites. We adapt the parameters L and ϵ for HMC using the techniques of AHMC described in Section 4.2. We obtain $L = 31$ and $\epsilon = 0.02473$ after adaptation. To assess the quality of these values, we repeat the same experiment several times without adaptation, on a grid of parameters centered at $L = 31$ and $\epsilon = 0.02473$; see Figure 4. Each cell in the plot reflects the average over ESS per second across all feature weights,

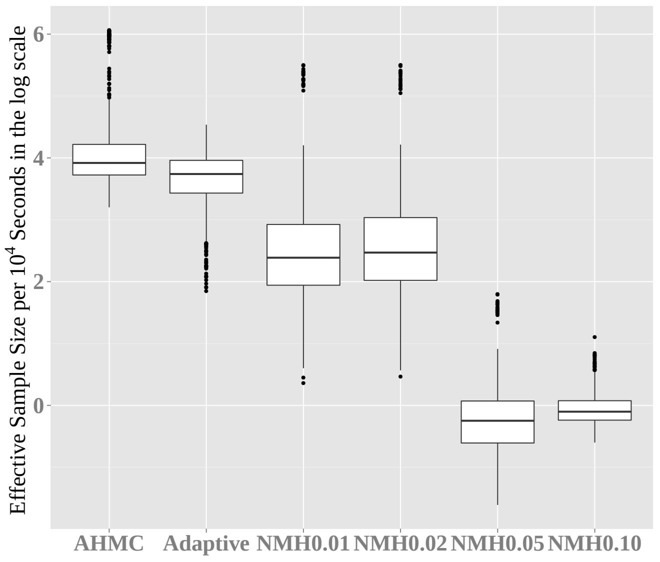

Figure 3: Effective sample size per 10^4 seconds in the log scale using AHMC, an adaptive NMH with a joint proposal guided by an estimated covariance matrix and NMH with bandwidths of 0.01, 0.02, 0.05, 0.10, where NMH x stands for a Normal proposal Metropolis Hastings algorithms with proposal bandwidth x.

displayed in the log scale. The results show that the performance of the combination of tuning parameters identified by our adaptation scheme is close to optimal. Moreover, we found that the cost of the adaptive Bayesian optimization is negligible in all cases we encountered. While the cost depends on the total number of optimization steps, we found that a small number of steps was sufficient in practice (Figure 4), with a cost on the order of 2–3 minutes. By comparison the wall-clock time for running 100,000 MCMC iterations was 2,262 minutes.

We also include a comparison of the computational efficiency of the HMC sampling scheme with and without the auxiliary variables described in Section 4.3 and Appendix A. In both setups, we set $\epsilon = 5 \times 10^{-7}$ and $L = 30$. The results, shown in Supplement 10, show that the HMC sampling scheme with auxiliary variable outperforms an HMC algorithm without auxiliary variables by more than one order of magnitude. As before, performance is measured by ESS per second.

7.5 Kinase domain data

We apply our methods to the protein kinase domain family, which is found in many different proteins. To create an alignment, we use the jackhmmer program, part of the HMMER 3.0 software suite (Eddy, 1998), to search the non-redundant protein sequence

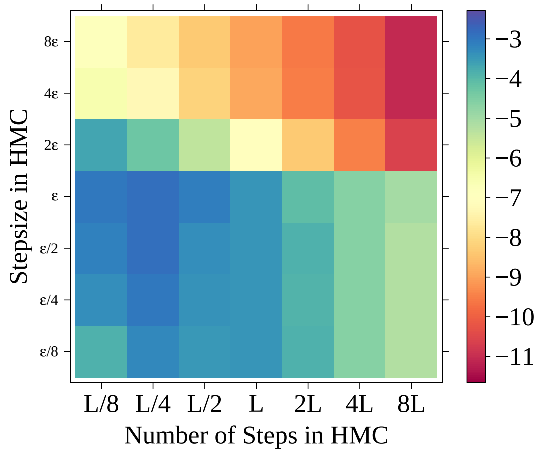

Figure 4: ESS per second (displayed in log scale) across all features obtained by running HMC with tuning parameters coming from a grid centered at the values ϵ and L coming from AHMC.

database for homologs. The starting sequence is selected to be the kinase domain present in the mouse Titin protein, taken from the PDB file 1TKI, which also includes some of the flanking sequence. The final alignment contains 641 sequences and 415 sites. To initialize the phylogenetic tree used in our MCMC algorithm, we ran MrBayes (Ronquist et al., 2012), which has a more complete and efficient set of tree resampling moves, until the average standard deviation of split frequencies was under 0.1.

We first estimate the rate matrix using a model containing the PolaritySize set of features. In this situation, only thirty-three parameters need to be estimated, including thirteen weights corresponding to bivariate features and twenty related to the univariate features. We run the MCMC chain for $100,000$ iterations, including a burn-in period of $10,000$ iterations. We use the Heidelberg and Welch Diagnostic (Heidelberger and Welch, 1981, 1983) to test if the sampled values come from a stationary distribution. The test is passed for the thirty-three parameters which indicates the number of MCMC iterations is sufficient. In Figure 5, we show the approximation of the posterior distribution of the weights for each bivariate feature obtained from the post burn-in iterations. Recall that when a weight is positive, its feature has an accelerating effect on the substitution rate, and when the weight is negative, the feature has a decelerating effect.

We expect that substitutions among amino acids with similar properties would be favoured. From Figure 5, we see that the 95% Highest Posterior Density (HPD) interval are supported within the positive real line for two weights corresponding to the "AcidicPolar" and "BasicBasic" features, and the 95% HPD interval are supported

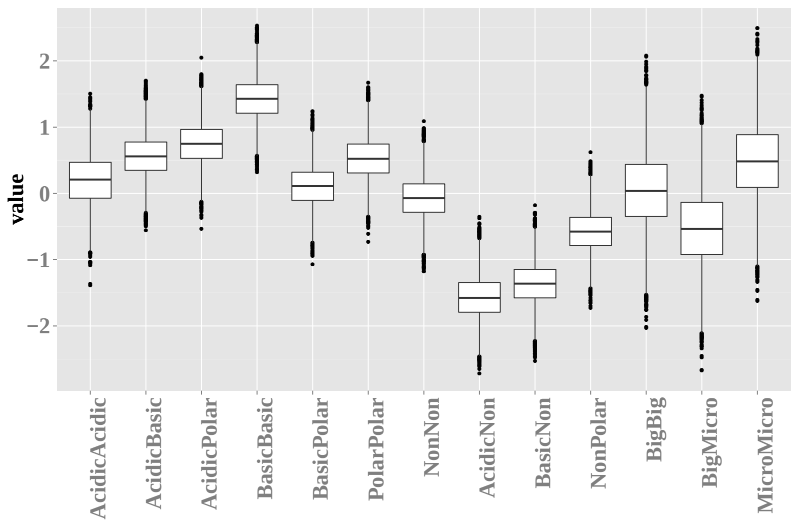

Figure 5: Approximation of the posterior distribution of the weights corresponding to coarse features under the PolaritySize model. The naming convention on the abscissa is described in Section 7.1.

within the negative real line for weights corresponding to the “AcidicNonPolar” and “BasicNonPolar” features. We performed the same experiment with the Polarity-SizeGtr model, and obtained similar results. See Supplement 9.

We also estimated the exchangeable coefficients $\theta_{\{x,x'\}}$ between each pair of amino acids. The exchangeable coefficients represent the rate of substitution between two amino acids after factoring out the effect of their stationary distribution. Our results shown in Figure 6 are qualitatively consistent with the substitution patterns found by previous work. For example, comparing with Barnes et al. (2003), the exchangeable coefficients between tryptophan (W) and other amino acids are relatively small compared with other states, which is expected based on tryptophan’s unique properties. Moreover, arginine (R) and lysine (K) have a high exchangeable coefficient, which is expected because of their similar charge properties.

8 Discussion

Our work shows promise for estimating large rate matrices for which prior knowledge is known. One well-suited application would be the modelling of the co-evolution of groups of interacting amino acid residues. Features could be used to model both the evolution of each individual amino acid (driven by mechanistic codon constraints and physico-chemical considerations), as well as structural constraints acting on several amino acids simultaneously.

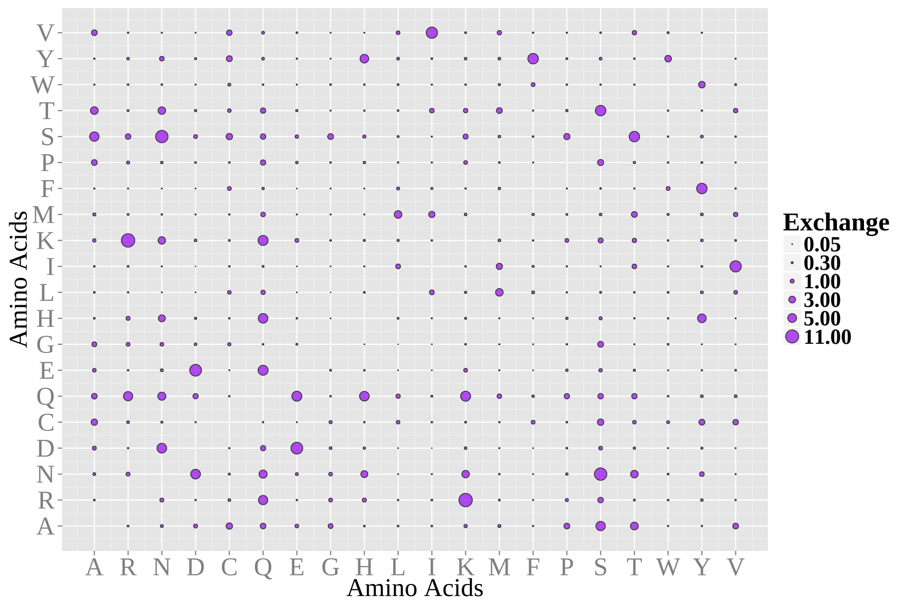

Figure 6: Bubble plot of exchangeable coefficients. A bigger bubble indicates a large exchangeable coefficient value.

An important challenge with such models, which we have not addressed here, is the computation of matrix exponentials over large matrices. Fortunately, recent advances in the field could be combined with our method (Sidje and Stewart, 1999; Higham, 2009; Hajiaghayi et al., 2014), in particular, methods that can exploit sparsity in the rate matrix (Didier et al., 2009; Zhang et al., 2010; Rao and Teh, 2013; Irvahn and Minin, 2014)—note that the base measures in our GLM-CTMC model can be used to create sparsity.

A second challenge in estimating large models is data scarcity. One strategy to address this is the use of several datasets simultaneously. Again, this can be handled easily in our framework, including the modelling of hierarchical and non-hierarchical structures across datasets. One could use distinct covariates where the underlying rate matrix model is allowed to vary while sharing statistical strengths across different families and/or different subtrees of the tree of life.

We expect the performance improvements brought by our AHMC methods over (adaptive) NMH methods to increase in these these higher dimensional models. In Neal (2010), an asymptotic analysis supports that the computational requirements for HMC scale as $\mathcal{O}(d^{5/4})$, compared to $\mathcal{O}(d^2)$ for an optimally tuned NMH method. Moreover, using the Riemann manifold HMC approach of Girolami and Calderhead (2011), it may be possible to combine the advantages of our method with some of the gains over naive NMH brought by the covariance adaptation baseline of Section 7.4. More generally, our method can benefit from various classical methods and recent advances from the HMC literature, for example methods to decrease rejection rates via non-constant trajectory lengths (Hoffman and Gelman, 2014; Sohl-Dickstein et al., 2014), partial refreshment methods (Horowitz, 1991), splitting methods to exploit tractable Hamiltonian subcomponents (Sexton and Weingarten, 1992) and several other techniques reviewed in Neal (2010).

Our framework could also be applied to continuous time Bayesian network models (Nodelman et al., 2002), where a graph specifies a sparsity structure in the rate matrix. Rao and Teh (2013) has constructed an auxiliary variable Gibbs sampling algorithm for such models based on the uniformization method. This method could be combined with the GLM representation of the structured rate matrices to perform Bayesian inference on the parameters of continuous time Bayesian network models.

Appendix A: Correctness of the sampling scheme

We show that the sampling algorithm described in Section 4.3 converges to the desired stationary distribution, namely the distribution with density $f_{\mathrm{w}|\mathrm{y}}(\boldsymbol{w}|\boldsymbol{y})$ (Equation (4.1)). The key step consists in defining the appropriate auxiliary variable $\boldsymbol{Z}$. In general, auxiliary variables can be used in an MCMC framework in two ways: either by directly augmenting the state space of the Markov chain, or as a transient variable used to define a kernel, but discarded right after its instantiation. As we show here, our situation falls in the second case, a fact that becomes crucial in Section 6.[5]

Consider the joint distribution

$$f_{\mathrm{w,x,y,z}}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \;:=\; f_{\mathrm{w}}(\boldsymbol{w}) f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{x}|Q(\boldsymbol{w})) f_{\mathrm{y}|\mathrm{d}}(\boldsymbol{y}|\boldsymbol{d}(\boldsymbol{x})) \mathbf{1}[\boldsymbol{z}=\boldsymbol{z}(\boldsymbol{x})], \tag{A.1}$$

$$f_{\mathrm{y}|\mathrm{d}}(\boldsymbol{y}|\boldsymbol{d}) \;:=\; \prod_{v\in V} f_{\mathrm{emi}}(y_v|d_v). \tag{A.2}$$

It is easy to check that we recover the target model of Section 3 after marginalization of $\boldsymbol{x}$ and $\boldsymbol{z}$. The conditional densities $f_{\mathrm{z}|\mathrm{w,y}}$ and $f_{\mathrm{w}|\mathrm{z,y}}$ are defined in the obvious way from the above joint density. Moreover, since $\boldsymbol{z}$ is sufficient to compute $f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{x}|Q(\boldsymbol{w}))$, then the following is well defined when $\boldsymbol{z} \in \boldsymbol{z}(S(|\mathcal{T}|))$: $f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{z}|Q(\boldsymbol{w})) := f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{x}_{\boldsymbol{z}}|Q(\boldsymbol{w}))$, where $\boldsymbol{x}_{\boldsymbol{z}}$ is such that $\boldsymbol{z}(\boldsymbol{x}_{\boldsymbol{z}}) = \boldsymbol{z}$.

Next, we define a Markov chain $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \dots$ on the state space $\mathbb{R}^P$. We construct a Markov transition kernel T using a sequence of sub-steps: first, augmentation of the state space to accommodate auxiliary variables, second, transition in the augmented space, and third, projection back to the target space $\mathbb{R}^P$. We show that each of these sub-steps keeps $f_{\mathrm{w}|\mathrm{y}}(\boldsymbol{w}|\boldsymbol{y})$ invariant.

We denote the sub-steps as follows: $\boldsymbol{w} \xmapsto{T_1} (\boldsymbol{w},\boldsymbol{z}) \xmapsto{T_2} (\boldsymbol{w}',\boldsymbol{z}) \xmapsto{T_3} \boldsymbol{w}'$. Each kernel in this decomposition is defined as follows: T_1 samples the auxiliary variable $\boldsymbol{Z}$ given the data $\boldsymbol{Y}$ and current parameters $\boldsymbol{W}$. It has density $T_1(\boldsymbol{w}',\boldsymbol{z}'|\boldsymbol{w}) := \mathbf{1}[\boldsymbol{w}=\boldsymbol{w}'] f_{\mathrm{z}|\mathrm{w,y}}(\boldsymbol{z}|\boldsymbol{w},\boldsymbol{y})$. Sampling from this density is outlined in Section 4.3, with additional details provided in Supplement 3. T_2 performs one (or several) Metropolis-within-Gibbs step(s) on $\boldsymbol{w}$, keeping $\boldsymbol{z}$ fixed. More precisely, this Metropolis-within-Gibbs step uses the HMC algorithm of Section 4. We show below that it is invariant with respect to the conditional distribution $f_{\mathrm{w}|\mathrm{z,y}}$. T_3 deterministically projects the pair back to the original space, retaining only the weights $\boldsymbol{w}$.

It remains to show that the formulae for $U_{\boldsymbol{z}^{(i)}}$ and $\nabla_{\boldsymbol{w}} U_{\boldsymbol{z}^{(i)}}$ in Section 4.3 correspond to the

[5]More precisely, if $\boldsymbol{Z}$ were part of the MCMC state space, we would have to develop new topology resampling operators that condition on both $\boldsymbol{W}$ and $\boldsymbol{Z}$. This is not the case here: we only need to condition on $\boldsymbol{W}$, which allows us to reuse existing phylogenetic MCMC operators.

negative logarithm of the conditional density $f_{\mathrm{w}|\mathrm{z},\mathrm{y}}$: [6]

$$\begin{aligned}
\log f_{\mathrm{w}|\mathrm{z},\mathrm{y}}(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{y}) &= \log \int\limits_{\boldsymbol{x}:\boldsymbol{z}(\boldsymbol{x})=\boldsymbol{z}} f_{\mathrm{w},\mathrm{x},\mathrm{y},\mathrm{z}}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} + \mathrm{cnst} && (A.3)\\
&= \log \int\limits_{\boldsymbol{x}:\boldsymbol{z}(\boldsymbol{x})=\boldsymbol{z}} f_{\mathrm{w}}(\boldsymbol{w}) f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{x}|Q(\boldsymbol{w})) f_{\mathrm{y}|\mathrm{d}}(\boldsymbol{y}|\boldsymbol{d}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} + \mathrm{cnst} && (A.4)\\
&= \log f_{\mathrm{w}}(\boldsymbol{w}) f_{\mathrm{x}|\mathrm{Q}}(\boldsymbol{z}|Q(\boldsymbol{w})) + \underbrace{\log \int\limits_{\boldsymbol{x}:\boldsymbol{z}(\boldsymbol{x})=\boldsymbol{z}} f_{\mathrm{y}|\mathrm{d}}(\boldsymbol{y}|\boldsymbol{d}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}}_{\text{constant with respect to } \boldsymbol{w}} + \mathrm{cnst}. && (A.5)
\end{aligned}$$

Now, plugging in Equation (2.1) into the above expression, we obtain Equation (4.8). For the gradient:

$$\begin{aligned}
\nabla_{\boldsymbol{w}} U_{\boldsymbol{z}}(\boldsymbol{w}) &= \nabla_{\boldsymbol{w}} \left[\frac{1}{2}\kappa \|\boldsymbol{w}\|_2^2 - \sum_{x\in\mathcal{X}} h_x q^{(\mathrm{nr})}_{x,x}(\boldsymbol{w}) - \sum_{(x,x')\in\mathcal{X}^{\mathrm{distinct}}} c_{x,x'} \log\left(q^{(\mathrm{nr})}_{x,x'}(\boldsymbol{w})\right)\right] && (A.6)\\
&= \kappa\boldsymbol{w} - \sum_{x\in\mathcal{X}} h_x \nabla_{\boldsymbol{w}} \left[\sum_{x'\in\mathcal{X}:x\neq x'} q^{(\mathrm{nr})}_{x,x'}(\boldsymbol{w})\right] - \sum_{(x,x')\in\mathcal{X}^{\mathrm{distinct}}} c_{x,x'}\boldsymbol{\varphi}(x,x') && (A.7)\\
&= \kappa\boldsymbol{w} + \sum_{(x,x')\in\mathcal{X}^{\mathrm{distinct}}} \boldsymbol{\varphi}(x,x')\left(h_x q^{(\mathrm{nr})}_{x,x'}(\boldsymbol{w}) - c_{x,x'}\right). && (A.8)
\end{aligned}$$

This coincides with Equation (4.9).

References

Adachi, J. and Hasegawa, M. (1996). "Model of amino acid substitution in proteins encoded by mitochondrial DNA." *Journal of Molecular Evolution*, 42(4): 459–468.

Adachi, J., Waddell, P. J., Martin, W., and Hasegawa, M. (2000). "Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA." *Journal of Molecular Evolution*, 50(4): 348–358.

Aldous, D. (1996). "Probability distributions on cladograms." In *Random discrete structures*, 1–18. Springer.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). "An Introduction to MCMC for Machine Learning." *Machine Learning*, 50(1): 5–43.

Baele, G., Van de Peer, Y., and Vansteelandt, S. (2010). "Using non-reversible context-dependent evolutionary models to study substitution patterns in primate non-coding sequences." *Journal of Molecular Evolution*, 71(1): 34–50.

Barnes, M. R., Gray, I. C., et al. (2003). *Bioinformatics for geneticists*, volume 2. Wiley Hoboken, NJ.

Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). "Painless Unsupervised Learning with Features." In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL10)*, volume 8, 582–590.

[6]In Equation (A.3), the densities and integrals are defined with respect to a reference measure on CTMC sample paths containing finite numbers of jumps.

Chambers, J. M. and Hastie, T. J. (1991). *Statistical models in S*. CRC Press, Inc.

Chen, L., Qin, Z., and Liu, J. S. (2001). "Exploring Hybrid Monte Carlo in Bayesian Computation." *Sigma*, 2: 2–5.

Clayton, D. G. (1991). "A Monte Carlo method for Bayesian inference in frailty models." *Biometrics*, 47: 467–485.

Dan Graur, W.-H. L. (2000). *Fundamentals of Molecular Evolution*. Sinauer Associates.

Dang, C. C., Le, Q. S., Gascuel, O., and Le, V. S. (2010). "FLU, an amino acid substitution model for influenza proteins." *BMC Evolutionary Biology*, 10(1): 99.

Dayhoff, M., Schwartz, R., and Orcutt, B. (1978). *Atlas of Protein Sequences and Structure*, volume 5, chapter A model of evolutionary change in proteins, 345–352. Silver Springs: Natl. Biomed. Res. Found.

Didier, F., Henzinger, T. A., Mateescu, M., and Wolf, V. (2009). "Fast adaptive uniformization of the chemical master equation." In *High Performance Computational Systems Biology, 2009. HIBI'09. International Workshop on*, 118–127. IEEE.

Dimmic, M. W., Mindell, D. P., and Goldstein, R. A. (2000). "Modeling evolution at the protein level using an adjustable amino acid fitness model." *Pacific Symposium on Biocomputing*, 18–29.

Dimmic, M. W., Rest, J. S., Mindell, D. P., and Goldstein, R. A. (2002). "rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny." *Journal of Molecular Evolution*, 55(1): 65–73.

Doron-Faigenboim, A. and Pupko, T. (2007). "A combined empirical and mechanistic codon model." *Molecular Biology and Evolution*, 24(2): 388–397.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). "Hybrid Monte Carlo." *Physics Letters B*, 195(2): 216–222.

Eddy, S. R. (1998). "Profile hidden Markov models." *Bioinformatics*, 14(9): 755–763.

Felsenstein, J. (1981). "Evolutionary trees from DNA sequences: a maximum likelihood approach." *Journal of Molecular Evolution*, 17(6): 368–376.

Geweke, J. (1992). "Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments." In *Bayesian Statistics*, volume 4, 169–193. Citeseer.

— (2004). "Getting it right: Joint distribution tests of posterior simulators." *Journal of the American Statistical Association*, 99(467): 799–804.

Gilks, W. R. and Roberts, G. O. (1996). *Markov chain Monte Carlo in practice*, chapter Strategies for Improving MCMC, 89–114. Chapman and Hall/CRC.

Girolami, M. and Calderhead, B. (2011). "Riemann manifold Langevin and Hamiltonian Monte Carlo methods." *Journal of the Royal Statistical Society: Series B*, 73(2): 123–214.

Goldman, N., Thorne, J. L., and Jones, D. T. (1996). "Using evolutionary trees in protein secondary structure prediction and other comparative sequence analyses." *Journal of Molecular Biology*, 263(2): 196–208.

Grantham, R. (1974). "Amino acid difference formula to help explain protein evolution." *Science*, 185(4154): 862–864.

Hajiaghayi, M., Kirkpatrick, B., Wang, L., and Bouchard-Côté, A. (2014). "Efficient Continuous-Time Markov Chain Estimation." In *International Conference on Machine Learning (ICML)*, volume (In Press).

Hamze, F., Wang, Z., and de Freitas, N. (2013). "Self-Avoiding Random Dynamics on Integer Complex Systems." *ACM Transactions on Modeling and Computer Simulation*, 23(1): 9:1–9:25.

Hasegawa, M., Kishino, H., and Yano, T. (1985). "Dating of the human-ape splitting by a molecular clock of mitochondrial DNA." *Journal of Molecular Eevolution*, 22: 160–174.

Heidelberger, P. and Welch, P. D. (1981). "A spectral method for confidence interval generation and run length control in simulations." *Communications of the ACM*, 24(4): 233–245.

— (1983). "Simulation run length control in the presence of an initial transient." *Operations Research*, 31(6): 1109–1144.

Higham, N. J. (2009). "The scaling and squaring method for the matrix exponential revisited." *SIAM review*, 51(4): 747–764.

Hobolth, A. and Stone, E. A. (2009). "Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution." *The Annals of Applied Statistics*, 3(3): 1204.

Hoffman, M. D. and Gelman, A. (2014). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research*, 15: 1351–1381.

Horowitz, A. M. (1991). "A generalized guided Monte Carlo algorithm." *Physics Letters B*, 268: 247–252.

Huelsenbeck, J. P., Bollback, J. P., and Levine, A. M. (2002). "Inferring the root of a phylogenetic tree." *Systematic Biology*, 51(1): 32–43.

Huelsenbeck, J. P., Joyce, P., Lakner, C., and Ronquist, F. (2008). "Bayesian analysis of amino acid substitution models." *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1512): 3941–3953.

Huelsenbeck, J. P. and Ronquist, F. (2001). "MrBayes: Bayesian inference of phylogenetic trees." *Bioinformatics*, 17(8): 754–755.

Ibrahim, J. G., Chen, M.-H., and Sinha, D. (2005). *Bayesian survival analysis*. Wiley Online Library.

Irvahn, J. and Minin, V. N. (2014). "Phylogenetic Stochastic Mapping Without Matrix Exponentiation." *Journal of Computational Biology*, 21(9): 676–690.

Ishwaran, H. (1999). "Applications of Hybrid Monte Carlo to Bayesian Generalized Linear Models: Quasicomplete Separation and Neural Networks." *Journal of Computational and Graphical Statistics*, 8(4): 779–799.

Jennrich, R. I. and Bright, P. B. (1976). "Fitting systems of linear differential equations using computer generated exact derivatives." *Technometrics*, 18(4): 385–392.

Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992). "The rapid generation of mutation data matrices from protein sequences." *Computer Applications in the Biosciences: CABIOS*, 8(3): 275–282.

Jukes, T. and Cantor, C. (1969). *Evolution of Protein Molecules*. New York: Academic Press.

Kalbfleisch, J. and Lawless, J. F. (1985). "The analysis of panel data under a Markov assumption." *Journal of the American Statistical Association*, 80(392): 863–871.

Kay, B. R. (1977). "Proportional hazard regression models and the analysis of censored survival data." *Applied Statistics*, 26(3): 227–237.

Kimura, M. (1980). "A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences." *Journal of Molecular Evolution*, 16: 111–120.

Kosiol, C., Holmes, I., and Goldman, N. (2007). "An empirical codon model for protein sequence evolution." *Molecular Biology and Evolution*, 24(7): 1464–1479.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2006). "Factor Graphs and the Sum-product Algorithm." *IEEE Trans. Inf. Theor.*, 47(2): 498–519.

Lakner, C., Van Der Mark, P., Huelsenbeck, J. P., Larget, B., and Ronquist, F. (2008). "Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics." *Systematic Biology*, 57(1): 86–103.

Lartillot, N. and Philippe, H. (2004). "A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process." *Molecular Biology and Evolution*, 21(6): 1095–1109.

Le, S. Q., Dang, C. C., and Gascuel, O. (2012). "Modeling protein evolution with several amino acid replacement matrices depending on site rates." *Molecular Biology and Evolution*, 2921–2936.

Le, S. Q. and Gascuel, O. (2008). "An improved general amino acid replacement matrix." *Molecular Biology and Evolution*, 25(7): 1307–1320.

Lio, P. and Goldman, N. (1999). "Using protein structural information in evolutionary inference: transmembrane proteins." *Molecular Biology and Evolution*, 16(12): 1696–1710.

Mahendran, N., Wang, Z., Hamze, F., and Freitas, N. D. (2012). "Adaptive MCMC with Bayesian optimization." In *International Conference on Artificial Intelligence and Statistics*, 751–760.

Minin, V. N. and Suchard, M. A. (2008). "Counting labeled transitions in continuous-time Markov models of evolution." *Journal of Mathematical Biology*, 56(3): 391–412.

Miyazawa, S. (2011a). "Advantages of a mechanistic codon substitution model for evolutionary analysis of protein-coding sequences." *PloS one*, 6(12): e28892.

— (2011b). "Selective constraints on amino acids estimated by a mechanistic codon substitution model with multiple nucleotide changes." *PloS one*, 6(3): e17244.

— (2013). "Superiority of a mechanistic codon substitution model even for protein sequences in Phylogenetic analysis." *BMC evolutionary biology*, 13(1): 257.

Moler, C. and Van Loan, C. (1978). "Nineteen dubious ways to compute the exponential of a matrix." *SIAM review*, 20(4): 801–836.

Müller, T. and Vingron, M. (2000). "Modeling amino acid replacement." *Journal of Computational Biology*, 7(6): 761–776.

Murrell, B., Weighill, T., Buys, J., Ketteringham, R., Moola, S., Benade, G., Du Buisson, L., Kaliski, D., Hands, T., and Scheffler, K. (2011). "Non-negative matrix factorization for learning alignment-specific models of protein evolution." *PloS one*, 6(12): e28898.

Neal, R. M. (2010). "MCMC using Hamiltonian dynamics." *Handbook of Markov Chain Monte Carlo*, 54: 113–162.

Neuts, M. (1981). *Matrix-Geometric solutions in stochastic models*. The Johns Hopkins University Press.

Nielsen, R. (2002). "Mapping mutations on phylogenies." *Systematic biology*, 51: 729–739.

Nodelman, U., Shelton, C., and Koller, D. (2002). "Continuous time Bayesian networks." In *Uncertainty in AI*, volume 18.

Paradis, E., Claude, J., and Strimmer, K. (2004). "APE: analyses of phylogenetics and evolution in R language." *Bioinformatics*, 20(2): 289–290.

Pasarica, C. and Gelman, A. (2010). "Adaptively scaling the Metropolis algorithm using expected squared jumped distance." *Statistica Sinica*, 20(1): 343.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, 6(1): 7–11.
URL http://CRAN.R-project.org/doc/Rnews/

Rao, V. and Teh, Y. W. (2013). "Fast MCMC sampling for Markov jump processes and extensions." *The Journal of Machine Learning Research*, 14(1): 3295–3320.

Roberts, G. O., Gelman, A., Gilks, W. R., et al. (1997). "Weak convergence and optimal scaling of random walk Metropolis algorithms." *The Annals of Applied Probability*, 7(1): 110–120.

Roberts, G. O. and Rosenthal, J. S. (2009). "Examples of adaptive MCMC." *Journal of Computational and Graphical Statistics*, 18(2): 349–367.

Roberts, G. O., Rosenthal, J. S., et al. (2001). “Optimal scaling for various Metropolis-Hastings algorithms.” *Statistical Science*, 16(4): 351–367.

Rodrigue, N., Philippe, H., and Lartillot, N. (2008). “Uniformization for sampling realizations of Markov processes: applications to Bayesian implementations of codon substitution models.” *Bioinformatics*, 24(1): 56–62.

Ronquist, F., Teslenko, M., van der Mark, P., Ayres, D. L., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M. A., and Huelsenbeck, J. P. (2012). “MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space.” *Systematic biology*, 61(3): 539–542.

Schadt, E. E., Sinsheimer, J. S., and Lange, K. (1998). “Computational advances in maximum likelihood methods for molecular phylogeny.” *Genome Research*, 8(3): 222–233.

Sexton, J. C. and Weingarten, D. H. (1992). “Hamiltonian evolution for the hybrid Monte Carlo method.” *Nuclear Physics B*, 380: 665–677.

Sidje, R. B. and Stewart, W. J. (1999). “A numerical study of large sparse matrix exponentials arising in Markov chains.” *Computational Statistics and Data analysis*, 29(3): 345–368.

Sneath, P. (1966). “Relations between chemical structure and biological activity in peptides.” *Journal of Theoretical Biology*, 12(2): 157–195.

Sohl-Dickstein, J., Mudigonda, M., and DeWeese., M. (2014). “Hamiltonian Monte Carlo without detailed balance.” In *International Conference on Machine Learning (ICML-14)*, volume 31, 719–726.

Stan Development Team (2014). “Stan: A C++ Library for Probability and Sampling, Version 2.5.0.”
URL http://mc-stan.org/

Tataru, P. and Hobolth, A. (2011). “Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains.” *BMC Bioinformatics*, 12: 465–475.

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

Wang, Z., Mohamed, S., and de Freitas, N. (2013). “Adaptive Hamiltonian and Riemann Manifold Monte Carlo Samplers.” In *International Conference on Machine Learning (ICML)*, 1462–1470. JMLR W&amp;CP 28 (3): 14621470, 2013.

Whelan, S. and Goldman, N. (2001). “A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach.” *Molecular Biology and Evolution*, 18(5): 691–699.

Yang, Z., Nielsen, R., and Hasegawa, M. (1998). “Models of amino acid substitution and applications to mitochondrial protein evolution.” *Molecular Biology and Evolution*, 15(12): 1600–1611.

Zhang, J., Watson, L. T., and Cao, Y. (2010). "A modified uniformization method for the solution of the chemical master equation." *Computers and Mathematics with Applications*, 59(1): 573–584.