BMC Bioinformatics

METHODOLOGY ARTICLE

Open Access

Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains

Paula Tataru[*] and Asger Hobolth

Abstract

Background: Continuous time Markov chains (CTMCs) is a widely used model for describing the evolution of DNA sequences on the nucleotide, amino acid or codon level. The sufficient statistics for CTMCs are the time spent in a state and the number of changes between any two states. In applications past evolutionary events (exact times and types of changes) are unaccessible and the past must be inferred from DNA sequence data observed in the present.

Results: We describe and implement three algorithms for computing linear combinations of expected values of the sufficient statistics, conditioned on the end-points of the chain, and compare their performance with respect to accuracy and running time. The first algorithm is based on an eigenvalue decomposition of the rate matrix (EVD), the second on uniformization (UNI), and the third on integrals of matrix exponentials (EXPM). The implementation in R of the algorithms is available at http://www.birc.au.dk/~paula/.

Conclusions: We use two different models to analyze the accuracy and eight experiments to investigate the speed of the three algorithms. We find that they have similar accuracy and that EXPM is the slowest method. Furthermore we find that UNI is usually faster than EVD.

Background

In this paper we consider the problem of calculating the expected time spent in a state and the expected number of jumps between any two states in discretely observed continuous time Markov chains (CTMCs). The case where the CTMC is only recorded at discretely observed time points arises in molecular evolution where DNA sequence data is extracted at present day and past evolutionary events are missing. In this situation, efficient methods for calculating these types of expectations are needed. In particular, two classes of applications can be identified.

The first class of applications is concerned with rate matrix estimation. [1] describes how the expectation-maximization (EM) algorithm can be applied to estimate the rate matrix from DNA sequence data observed in the leaves of an evolutionary tree. The EM algorithm is implemented in the software XRate [2] and has been applied in [3] for estimating empirical codon rate matrices. [1] uses the eigenvalue decomposition of the rate matrix to calculate the expected time spent in a state and the expected number of jumps between states.

The second class of applications is concerned with understanding and testing various aspects of evolutionary trajectories. In [4] it is emphasized that analytical results for jump numbers are superior to simulation approaches and various applications of jump number statistics are provided, including a test for the hypothesis that a trait changed its state no more than once in its evolutionary history and a diagnostic tool to measure discrepancies between the data and the model. [4] assumes that the rate matrix is diagonalizable and that the eigenvalues are real, and applies a spectral representation of the transition probability matrix to obtain the expected number of state changes.

[5] and [6] describe a method, termed substitution mapping, for detecting coevolution of evolutionary traits, and a similar method is described in [7]. The substitution mapping method is based on the expected number

* Correspondence: paula@birc.au.dk
Bioinformatics Research Center, Aarhus University, Aarhus, Denmark

of substitutions while [7] base their statistics on the expected time spent in a state. Furthermore [7] describes an application concerned with mapping synonymous and non-synonymous mutations on branches of a phylogenetic tree and employs the expected number of changes between any two states for this purpose. [8] uses the expected number of state changes to calculate certain labeled evolutionary distances. A labeled evolutionary distance could for example be the number of state changes from or to a specific nucleotide. In [9] substitution mapping is invoked for identifying biochemically constrained sites. In [7] and [8] the summary statistics are calculated using the eigenvalue decomposition method suggested by [1]. In [5,6] and [9] the substitution mapping is achieved using a more direct formula for calculating the number of state changes. In this direct approach an infinite sum must be truncated and it is difficult to control the error associated with the truncation. An alternative is described in [10] where uniformization is applied to obtain the expected number of jumps. [10] uses the expected number of jumps on a branch to detect lineages in a phylogenetic tree that are under selection.

A third algorithm for obtaining the number of changes or time spent in a state is outlined in [11]. The algorithm is based on [12] where a method for calculating integrals of matrix exponentials is described. A natural question arises: which of the three methods (eigenvalue decomposition, uniformization or matrix exponentiation) for calculating conditional expectations of summary statistics for a discretely observed CTMC should be preferred? The aim of this paper is to provide an answer to this question. We describe and compare the three methods. Our implementations in R [13] are available at http://www.birc.au.dk/~paula/. (Furthermore the eigenvalue decomposition and uniformization methods are also available as a C++ class in the bio++ library at http://biopp.univ-montp2.fr/.) The performance and discussion of the algorithms are centered around two applications. The first application is concerned with rate matrix estimation; we estimate the Goldman-Yang codon model [14] using the expectation-maximization algorithm. The second application is based on the labeled distance estimation presented in [8].

Consider a stochastic process $\{X(s): 0 \leq s \leq t\}$ which can be described by a CTMC with n states and an $n \times n$ rate matrix $Q = (q_{cd})$. The off-diagonal entries in Q are non-negative and rows sum to zero, i.e. $q_{cc} = -\sum_{d \neq c} q_{cd} = -q_c$. Maximum likelihood estimation of the rate matrix from a complete observation of the process is straight forward. The likelihood of the process, conditional on the beginning state $X(0)$, is given by (e.g. [15])

$$L(Q; \{X(s) : 0 \leq s \leq t\}) = \exp\left(-\sum_c q_c T_c\right)\left(\prod_{c=1}^{n}\prod_{d \neq c} q_{cd}^{N_{cd}}\right), \quad (1)$$

where T_c is the total time spent in state c and N_{cd} is the number of jumps from c to d. The necessary sufficient statistics for a CTMC are thus the time spent in each state and the number of jumps between any two states. In applications, however, access is limited to DNA data from extant species. The CTMC is discretely observed and we must estimate the mean values of T_c and N_{cd} conditional on the end-points $X(0) = a$ and $X(t) = b$. From [15] we have that

$$\mathbb{E}[T_c|X(0) = a, X(t) = b] = \mathbb{E}[T_c|t, a, b] = \frac{I_{cc}^{ab}(t)}{p_{ab}(t)} \quad (2)$$

$$\mathbb{E}[N_{cd}|X(0) = a, X(t) = b] = \mathbb{E}[N_{cd}|t, a, b] = \frac{q_{cd}I_{cd}^{ab}(t)}{p_{ab}(t)} \quad (3)$$

where $P(t) = (p_{ij}(t)) = e^{Qt}$ is the transition probability matrix and

$$I_{cd}^{ab}(t) = \int_0^t p_{ac}(u)p_{db}(t - u)du. \quad (4)$$

Many applications require a linear combination of certain substitutions or times. Examples include the number of transitions, transversions, synonymous and non-synonymous substitutions. In the two applications described below the statistics of interest is a linear combination of certain substitutions and times. Let therefore C be an $n \times n$ matrix and denote by $\Sigma(C; t)$ the matrix with entries

$$\sum(C; a, b, t) = \sum_{c,d} C_{cd}I_{cd}^{ab}(t). \quad (5)$$

We describe, compare and discuss three methods for calculating $\Sigma(C; t)$. The evaluation of the integrals (4) takes $O(n^3)$ time and therefore a naive calculation, assuming that C contains just one entry different from zero has a $O(n^5)$ running time. Even worse, if C contains $O(n^2)$ entries different from zero, then the naive implementation has a $O(n^7)$ running time. For all three methods our implementations of $\Sigma(C; t)$ run in $O(n^3)$ time.

Results
Applications
Application 1: Rate matrix estimation
Our first application is the problem of estimating the parameters in a CTMC for evolution of coding DNA sequences which we describe using the 61×61 rate matrix (excluding stop codons) given by Goldman and Yang [14]:

$$q_{ij} = \begin{cases} 0 & \text{if there is more than one difference between codons } i \text{ and } j \\ \alpha\kappa\pi_j & \text{if } j \text{ is obtained from } i \text{ by a synonymous transition} \\ \alpha\pi_j & \text{if } j \text{ is obtained from } i \text{ by a synonymous transversion} \\ \alpha\omega\kappa\pi_j & \text{if } j \text{ is obtained from } i \text{ by a non - synonymous transition} \\ \alpha\omega\pi_j & \text{if } j \text{ is obtained from } i \text{ by a non - synonymous transversion} \end{cases} \quad (6)$$

where π is the stationary distribution, κ is the transition/transversion rate ratio, ω is the non-synonymous/synonymous ratio and α is a scaling factor. The stationary distribution π is determined directly from the data using the codon frequencies. We estimate the remaining parameters $\theta = (\alpha, \kappa, \omega)$ using the expectation-maximization (EM) algorithm [16] as described below.

Suppose the complete data **x** is available, consisting of times and types of substitutions in all sites and in all branches of the tree. The complete data log likelihood is, using (1) and (6),

$$\ell(\alpha, \kappa, \omega; \mathbf{x}) = -\alpha L_{\text{s,tv}} - \alpha\omega L_{\text{ns,tv}} - \alpha\kappa L_{\text{s,ts}} - \alpha\kappa\omega L_{\text{ns,ts}} \\ + N\log\alpha + N_{\text{ts}}\log\kappa + N_{\text{ns}}\log\omega, \quad (7)$$

where we use the notation

$$L_{s,ts} = \sum_i T_i \sum_j \pi_i 1((i,j) \in \mathcal{L}_{s,ts}) \text{ and } N_{ts} = \sum_{i,j} N_{ij} 1((i,j) \in \mathcal{L}_{ts}) \quad (8)$$

where e.g.

$$\mathcal{L}_{s,ts} = \{(i,j) : i \text{ and } j \text{ differ at one position and the substitution of } i \text{ with } j \text{ is a synonymous transition}\}.$$

A similar notation applies for $L_{\text{s,tv}}$, $L_{\text{ns,ts}}$, $L_{\text{ns,tv}}$, N_{ns} and N, where the last statistic is the sum of substitutions between all states (i, j) that differ at one position and s, ns, ts and tv subscripts stand for synonymous, non-synonymous, transition and transversion.

The complete data log likelihood can be maximized easily by making the re-parametrization $\beta = \alpha\kappa$. We find that

$$\hat{\alpha} = \frac{N_{\text{tv}}}{L_{\text{s,tv}} + \hat{\omega} L_{\text{ns,tv}}}, \hat{\beta} = \frac{N_{\text{ts}}}{L_{\text{s,ts}} + \hat{\omega} L_{\text{ns,ts}}} \text{ and } \hat{\omega} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad (9)$$

where $a = -L_{\text{ns,tv}} L_{\text{ns,ts}} N_s$, $b = L_{\text{ns,tv}} L_{\text{s,ts}} (N_{\text{ns}} - N_{\text{tv}}) + L_{\text{ns,ts}} L_{\text{s,tv}} (N_{\text{ns}} - N_{\text{ts}})$ and $c = L_{\text{s,tv}} L_{\text{s,ts}} N_{\text{ns}}$.

In reality the data **y** is only available in the leaves and the times and types of substitutions in all sites and all branches of the tree are inaccessible. The EM algorithm is an efficient tool for maximum likelihood estimation in problems where the complete data log likelihood is analytically tractable but full information about the data is missing.

The EM algorithm is an iterative procedure consisting of two steps. In the E-step the expected complete log likelihood

$$G(\theta; \theta_0, \mathbf{y}) = E_{\theta_0}[\ell(\theta; \mathbf{x}) | \mathbf{y}] \quad (10)$$

conditional on the data **y** and the current estimate of the parameters θ_0 is calculated. In the M-step the

parameters are updated by maximizing $G(\theta; \theta_0, \mathbf{y})$. The parameters converge to a local maximum of the likelihood for the observed data.

The expected log likelihood conditional on the data **y** and under the three parameters α, κ and ω is

$$\mathbb{E}[\ell(\alpha, \kappa, \omega; \mathbf{x}) | \mathbf{y}] = -\alpha\mathbb{E}[L_{\text{s,tv}} | \mathbf{y}] - \alpha\omega\mathbb{E}[L_{\text{ns,tv}} | \mathbf{y}] \\ - \alpha\kappa\mathbb{E}[L_{\text{s,ts}} | \mathbf{y}] - \alpha\kappa\omega\mathbb{E}[L_{\text{ns,ts}} | \mathbf{y}] \\ + \mathbb{E}[N | \mathbf{y}]\log\alpha + \mathbb{E}[N_{\text{ts}} | \mathbf{y}]\log\kappa + \mathbb{E}[N_{\text{ns}} | \mathbf{y}]\log\omega. \quad (11)$$

Therefore the E-step requires expectations of linear combinations of waiting times in a set of states and number of jumps between certain states. Because of the Markov property this calculation can be divided in two parts. First we use the peeling algorithm [17,18] to obtain the probability $\mathbb{P}(\gamma_k = a, \beta_k = b | \mathbf{y}, t_k)$ that a branch k of length t_k with nodes γ_k and β_k above and below the branch, respectively, has end-points a and b. Second, we calculate the desired summary statistic by summing over all branches. For example we have

$$\mathbb{E}[L_{\text{s,ts}} | \mathbf{y}] = \sum_{\text{branch } k} \sum_{a,b} \mathbb{P}(\gamma_k = a, \beta_k = b | \mathbf{y}, t_k) \mathbb{E}[L_{\text{s,ts}} | t_k, a, b] \quad (12)$$

$$\mathbb{E}[N_{\text{ts}} | \mathbf{y}] = \sum_{\text{branch } k} \sum_{a,b} \mathbb{P}(\gamma_k = a, \beta_k = b | \mathbf{y}, t_k) \mathbb{E}[N_{\text{ts}} | t_k, a, b]. \quad (13)$$

The E-step thus consists of calculating conditional expectations of linear combinations of times such as $\mathbb{E}[L_{\text{s,ts}} | t_k, a, b]$ and substitutions such as $\mathbb{E}[N_{\text{ts}} | t_k, a, b]$ where $L_{\text{s,ts}}$ and N_{ts} are given by (8). In our application $n = 61$ and the first type of statistics $\mathbb{E}[L_{\text{s,ts}} | t, a, b]$ is (up to a factor $p_{ab}(t)$) on the form (5) with diagonal entries $C_{ii} = \sum_j \pi_j 1((i,j) \in \mathcal{L}_{s,ts})$ and all off diagonal entries equal to zero. The second type of statistics $\mathbb{E}[N_{\text{ts}} | t, a, b]$ is also on the form (5) with off-diagonal entries $C_{ij} = q_{ij} 1((i,j) \in \mathcal{L}_{ts})$ and zeros on the diagonal.

Application 2: Robust distance estimation

The second application is a new approach for estimating labeled evolutionary distance, entitled robust counting and introduced in [8]. The purpose is to calculate a distance that is robust to model misspecification. The method is applied to labeled distances, for example, the synonymous distance between two coding DNA sequences. As it is believed that selection mainly acts at the protein level, synonymous substitutions are neutral and phylogenies built on these type of distances are more likely to reveal the true evolutionary history. The distance is calculated using the mean numbers of labeled substitutions conditioned on pairwise site patterns averaged over the empirical distribution of site patterns observed in the data. In the conventional method the average is done over the theoretical distribution of site patterns. The robustness is therefore

achieved through the usage of more information from the data and less from the model.

Let Q be the rate matrix of the assumed model, $P(t) = e^{Qt}$, the labeling be given through a set of pairs $\mathcal{L}$ and the data be represented by a pairwise alignment $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ of length m. As data only contains information about the product Qt, where t is the time distance between the sequences, we can set $t = 1$.

Suppose we observe the complete data consisting of the types of substitutions that occurred in all sites and let $N_{\mathcal{L}} = \sum_{i,j} N_{ij} 1((i,j) \in \mathcal{L})$ be the labeled number of substitutions. A natural labeled distance is given by $d_{\mathcal{L}} = \mathbb{E}(N_{\mathcal{L}})$. The labeled distance is estimated as the average across all sites of the expected number of labeled substitutions conditioned on the observed end points:

$$\widehat{d}_{\mathcal{L}} = \frac{1}{m} \sum_{m}^{s=1} E[N_{\mathcal{L}} | X(0) = \mathbf{y}_{1s}, X(1) = \mathbf{y}_{2s}]$$

$$= \frac{1}{m} \sum_{m}^{s=1} E \left[\sum_{i,j} N_{i,j} 1((i,j) \in \mathcal{L}) | 1, \mathbf{y}_{1s}, \mathbf{y}_{2s} \right]. \quad (14)$$

Therefore this application requires evaluating a sum on the form (5) with off-diagonal entries $C_{ij} = q_{ij} \mathbb{1}((i,j) \in \mathcal{L})$ and zeros on the diagonal.

Algorithms

The calculation of $\Sigma(C; t)$ is based on the integrals $I_{cd}^{ab}(t)$. In this section we present three existing methods for obtaining the integrals and extend them to obtain $\Sigma(C; t)$.

Eigenvalue decomposition (EVD)

When the rate matrix Q is diagonalizable, the computation of transition probabilities $p_{ab}(t)$ and integrals $I_{cd}^{ab}(t)$ can be done via the eigenvalue decomposition (EVD). EVD is a widely used method for calculating matrix exponentials. Let $Q = U\Lambda U^{-1}$ be the eigenvalue decomposition, with $\Lambda = \text{diag}(\lambda_1, ..., \lambda_n)$. It follows that

$$P(t) = e^{Qt} = e^{(U\Lambda U^{-1})t} = Ue^{\Lambda t}U^{-1}. \quad (15)$$

Because Λ is diagonal, $e^{\Lambda t}$ is also diagonal with $(e^{\Lambda t})_{ii} = e^{\lambda_i t}$.

The integral (4) becomes

$$I_{cd}^{ab}(t) = \sum_{i} U_{ai}(U^{-1})_{ic} \sum_{j} U_{dj}(U^{-1})_{jb} J_{ij}(t) \quad (16)$$

$$\text{where } J_{ij}(t) = \begin{cases} te^{\lambda_i t} & \text{if } \lambda_i = \lambda_j \\ \dfrac{e^{\lambda_i t} - e^{\lambda_j t}}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j. \end{cases} \quad (17)$$

Replacing $I_{cd}^{ab}(t)$ with (16) in (5), rearranging the sums and using $A_{cj} = \sum_d C_{cd}U_{dj}, B_{ij} = J_{ij}(t) \sum_c (U^{-1})_{ic}A_{cj}, D_{ib} = \sum_j B_{ij}(U^{-1})_{jb}$ and $\Sigma(C; a, b, t) = \sum_i U_{ai}D_{ib}$ we find

$$\Sigma(C; t) = U[J(t) \circ (U^{-1}CU)]U^{-1} \quad (18)$$

where $\circ$ represents the entry-wise product.

The eigenvalues and eigenvectors might be complex, but they come in complex conjugate pairs and the final result is always real; for more information we refer to the Supplementary Information in [2]. If the CTMC is reversible, the decomposition can be done on a symmetric matrix obtained from Q (e.g. [15]), which is faster and tends to be more robust. Let π be the stationary distribution. Due to reversibility, $\pi_a q_{ab} = \pi_b q_{ba}$, which can be written as $\Pi Q = Q^*\Pi$ where $\Pi = \text{diag}(\pi)$. Let $S = \Pi^{1/2}Q\Pi^{-1/2}$.

We have that

$$S^* = \Pi^{-1/2}Q^*\Pi^{1/2} = \Pi^{-1/2}(Q^*\Pi)\Pi^{-1/2}$$

$$= \Pi^{-1/2}(\Pi Q)\Pi^{-1/2} = \Pi^{1/2}Q\Pi^{-1/2} = S \quad (19)$$

where S^* is the transpose of S. Then S is symmetric. Let Λ, V be its eigenvalues and eigenvectors, respectively. Then $V\Lambda V^{-1} = S = \Pi^{1/2}Q\Pi^{-1/2}$, which implies $Q = (\Pi^{-1/2}V)\Lambda(V^{-1}\Pi^{1/2})$ and it follows that Q has the same eigenvalues as S and $\Pi^{-1/2}V$ for eigenvectors.

The results can be summarized in the following algorithm:

> **Algorithm 1**: EVD
> **Input**: Q, C, t
> **Output**: $\Sigma(C; t)$
> Step 1: Determine eigenvalues λ_i.
>   Determine the eigenvectors U_i for Q and compute U^{-1}.
> Step 2: Determine matrix $J(t)$ from (17).
> Step 3: Determine matrix $\Sigma(C;t)$ from (18).

Uniformization (UNI)

The uniformization method was first introduced in [19] for computing the matrix exponential $P(t) = e^{Qt}$. In [11] it was shown how this method can be used for calculating summary statistics, even for statistics that cannot be written in integral form. Let $\mu = \max_i (q_i)$ and $R = \dfrac{1}{\mu}Q + I$, where I is the identity matrix.

Then

$$P(t) = e^{\mu(R-I)t} = \sum_{m=0}^{\infty} R^m \frac{(\mu t)^m}{m!} e^{-\mu t} = \sum_{m=0}^{\infty} R^m \text{Pois}(m; \mu t) \quad (20)$$

where $\text{Pois}(m; \lambda)$ is the probability of m occurrences from a Poisson distribution with mean λ. Using (20) we also have

$$
\begin{aligned}
I_{cd}^{ab}(t) &= \int_0^t p_{ac}(u)p_{db}(t-u)du \\
&= \int_0^t \left[\sum_{i=0}^{\infty}(R^i)_{ac}\frac{(\mu u)^i}{i!}e^{-\mu u}\right]\left[\sum_{j=0}^{\infty}(R^j)_{db}\frac{(\mu(t-u))^j}{j!}e^{-\mu(t-u)}\right]du \\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}(R^i)_{ac}(R^j)_{db}\frac{\mu^{i+j}}{i!j!}e^{-\mu t}\int_0^t u^i(t-u)^j du \\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}(R^i)_{ac}(R^j)_{db}\frac{\mu^{i+j}}{i!j!}e^{-\mu t}\frac{i!j!}{(i+j+1)!}t^{i+j+1} \\
&= \frac{1}{\mu}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}(R^i)_{ac}(R^j)_{db}\frac{(\mu t)^{i+j+1}}{(i+j+1)!}e^{-\mu t} \\
&= \frac{1}{\mu}\sum_{m=0}^{\infty}\text{Pois}(m+1;\mu t)\sum_{l=0}^{m}(R^l)_{ac}(R^{m-l})_{db}.
\end{aligned}
\tag{21}
$$

Replacing (21) in (5), rearranging the sums and using that $\sum_d C_{cd}(R^{m-l})_{db} = (CR^{m-l})_{cb}$ and $\sum_c (R^l)_{ac}(CR^{m-l})_{cb} = (R^l CR^{m-l})_{ab}$ we derive

$$
\Sigma(C; t) = \frac{1}{\mu}\sum_{m=0}^{\infty}\text{Pois}(m+1;\mu t)\sum_{l=0}^{m}R^l CR^{m-l}.
\tag{22}
$$

The main challenge with this method is the infinite sum and we use (20) to determine a truncation point. In particular if we let $\lambda = \mu t$ and truncate at $s(\lambda)$ we can bound the error using the tail of the Poisson distribution:

$$
\left|p_{ab}(t) - \sum_{m=0}^{s(\lambda)}(R^m)_{ab}\text{Pois}(m;\mu t)\right| = \sum_{m=s(\lambda)+1}^{\infty}(R^m)_{ab}\text{Pois}(m;\mu t) \leq \sum_{m=s(\lambda)}^{\infty}\text{Pois}(m;\mu t).
$$

We have that, for large values of λ, $\text{Pois}(\lambda) \approx \mathbb{N}(\lambda, \lambda)$, where $\mathbb{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. Therefore, for large λ, the error bound

$$
b = \sum_{m=s(\lambda)}^{\infty}\text{Pois}(m;\mu t) \approx 1 - \Phi\left(\frac{s(\lambda)-\lambda}{\sqrt{\lambda}}\right),
$$

where $\Phi(\cdot)$ is the cumulative distribution function for the standard normal distribution. Consequently we can approximate the truncation point $s(\lambda)$ with $\sqrt{\lambda}\Phi^{-1}(1-b) + \lambda$. If $b = 10^{-8}$ we obtain $\Phi^{-1}(1-b) = 5.6$.

Another way to determine $s(\lambda)$ is to use R to evaluate $\text{Pois}(m; \lambda)$ for values of m that gradually increase, until the tail is at most $b = 10^{-8}$. Combining these two approaches, we performed a linear regression, approximating the tails from R by $c_1 + c_2\sqrt{\lambda} + c_3\lambda$. We obtained $c_1 = 4.0731$, $c_2 = 5.6469$, $c_3 = 0.9963$ but, in order to be conservative, we use $s(\lambda) = \lceil 4 + 6\sqrt{\lambda} + \lambda \rceil$ where $\lceil x \rceil$ is the smallest integer greater than or equal to x. In Figure 1 we compare the exact truncation value and the linear regression approximation.

The linear regression provides an excellent fit to the tail of the distribution.

In summary we have the following algorithm:

Algorithm 2: UNI
Input: Q, C, t
Output: $\Sigma(C; t)$
Step 1: Determine μ, $s(\mu t)$ and R.
Step 2: Calculate R^m for $2 \leq m \leq s(\mu t)$.
Step 3: Calculate $A(m) = \sum_{l=0}^{m}R^l CR^{m-l}$ for $0 \leq m \leq s(\mu t)$.
    using the recursion $A(m+1) = A(m)R + R^{m+1}C$.
Step 4: Determine $\Sigma(C; t)$ from (22).

Exponentiation (EXPM)

This method for calculating the integral (4) was developed in [12] and emphasized in [11]. Suppose we want to evaluate $\int_0^t e^{Qu}Be^{Q(t-u)}du$, where Q and B are $n \times n$ matrices. To calculate this integral, we use an auxiliary matrix $A = \begin{bmatrix} Q & B \\ 0 & Q \end{bmatrix}$ and the desired integral can be
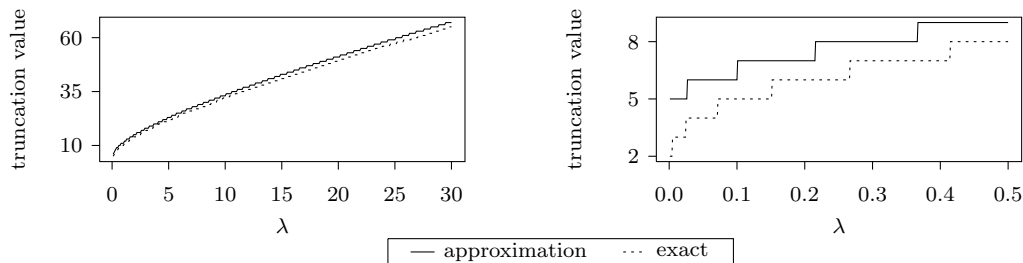
Figure 1 Poisson truncation. Comparison between the exact truncation value and the $\lceil 4 + 6\sqrt{\lambda} + \lambda \rceil$ approximation. In the plot on the left, λ ranges from 0 to 30, while the plot on the right is a zoom-in for values between 0 to 0.5. The plot shows that the approximation is a conservative fit of the Poission tail.

found in the upper right corner of the matrix exponential of A:

$$\int_0^t e^{Qu} B e^{Q(t-u)} du = (e^{At})_{1:n,(n+1):2n}. \qquad (23)$$

We are interested in

$$I_{cd}^{ab}(t) = \int_0^t p_{ac}(u) p_{db}(t-u) du = \int_0^t (e^{Qu})_{ac} (e^{Q(t-u)})_{db} du$$
$$= \left(\int_0^t e^{Qu} \mathbb{1}_{\{(c,d)\}} e^{Q(t-u)} du \right)_{ab} \qquad (24)$$

where $\mathbb{1}_{\{(c,d)\}}$ is a matrix with 1 in entry (c, d) and zero otherwise. We can use this method to determine $I_{cd}^{ab}(t)$ by simply setting $B = \mathbb{1}_{\{(c,d)\}}$, construct the auxiliary matrix A, calculate the matrix exponential of At, and finally read off the integral in entry (a, b) in the upper right corner of the matrix exponential.

Replacing (24) in (5) and rearranging the terms we have

$$\Sigma(C;t) = \int_0^t e^{Qu} \sum_{c,d} C_{cd} \mathbb{1}_{\{(c,d)\}} e^{Q(t-u)} du \quad \text{and} \quad \sum_{c,d} C_{cd} \mathbb{1}_{\{(c,d)\}} = C. \quad (25)$$

Therefore by setting $B = C$ in the auxiliary matrix we obtain $\Sigma(C;t)$.

The EXPM algorithm is as follows:

Algorithm 3: EXPM
Input: Q, C, t
Output: $\Sigma(C; t)$
Step 1: Construct $A = \begin{bmatrix} Q & C \\ 0 & Q \end{bmatrix}$.
Step 2: Calculate the matrix exponential e^{At}.
Step 3: $\Sigma(C; t)$ is the upper right corner of the matrix exponential.

Testing
We implemented the presented algorithms in R and tested them with respect to accuracy and speed.
Accuracy
The accuracy of the methods depends on the size of the rate matrix and the time t. To investigate how these factors influence the result, we used two different CTMCs that allow an analytical expression for (4). The first investigation is based on the Jukes-Cantor model where the rate matrix has uniform rates and variable size n:

$$q_{ij} = \begin{cases} -1 & \text{if } i = j \\ \dfrac{1}{n-1} & \text{if } i \neq j. \end{cases}$$

Q has two unique eigenvalues: 0 with multiplicity 1 and $-\dfrac{n}{n-1}$ with multiplicity $n-1$. We obtain

$$p_{ij}(t) = \begin{cases} \frac{1}{n} + \frac{n-1}{n} \exp\left(-\frac{nt}{n-1}\right) & \text{if } i = j \\ \frac{1}{n} - \frac{1}{n} \exp\left(-\frac{nt}{n-1}\right) & \text{if } i \neq j \end{cases}$$

and $I_{cd}^{ab}(t) = \frac{1}{n^2} \begin{cases} t + t \exp\left(-\frac{nt}{n-1}\right) - \frac{2(n-1)}{n}\left(1 - \exp\left(-\frac{nt}{n-1}\right)\right) & \text{if } a \neq c, d \neq b \\ t + (n-1)^2 t \exp\left(-\frac{nt}{n-1}\right) + \frac{2(n-1)^2}{n}\left(1 - \exp\left(-\frac{nt}{n-1}\right)\right) & \text{if } a = c, d = b \\ t - (n-1)t \exp\left(-\frac{nt}{n-1}\right) + \frac{(n-2)(n-1)}{n}\left(1 - \exp\left(-\frac{nt}{n-1}\right)\right) & \text{otherwise.} \end{cases}$

We compared the result from all three methods against the true value of (5) for size n ranging from 5 to 100, $t = 0.1$ and random binary matrices C. Entries in C are 1 with probability $\frac{1}{2}$. For each fixed size, we generated 5 different matrices C. The average normalized deviation is shown in Figure 2.

The second CTMC is the HKY model:

$$Q = \begin{pmatrix} \cdot & \kappa\pi_G & \pi_C & \pi_T \\ \kappa\pi_A & \cdot & \pi_C & \pi_T \\ \pi_A & \pi_G & \cdot & \kappa\pi_T \\ \pi_A & \pi_G & \kappa\pi_C & \cdot \end{pmatrix}$$
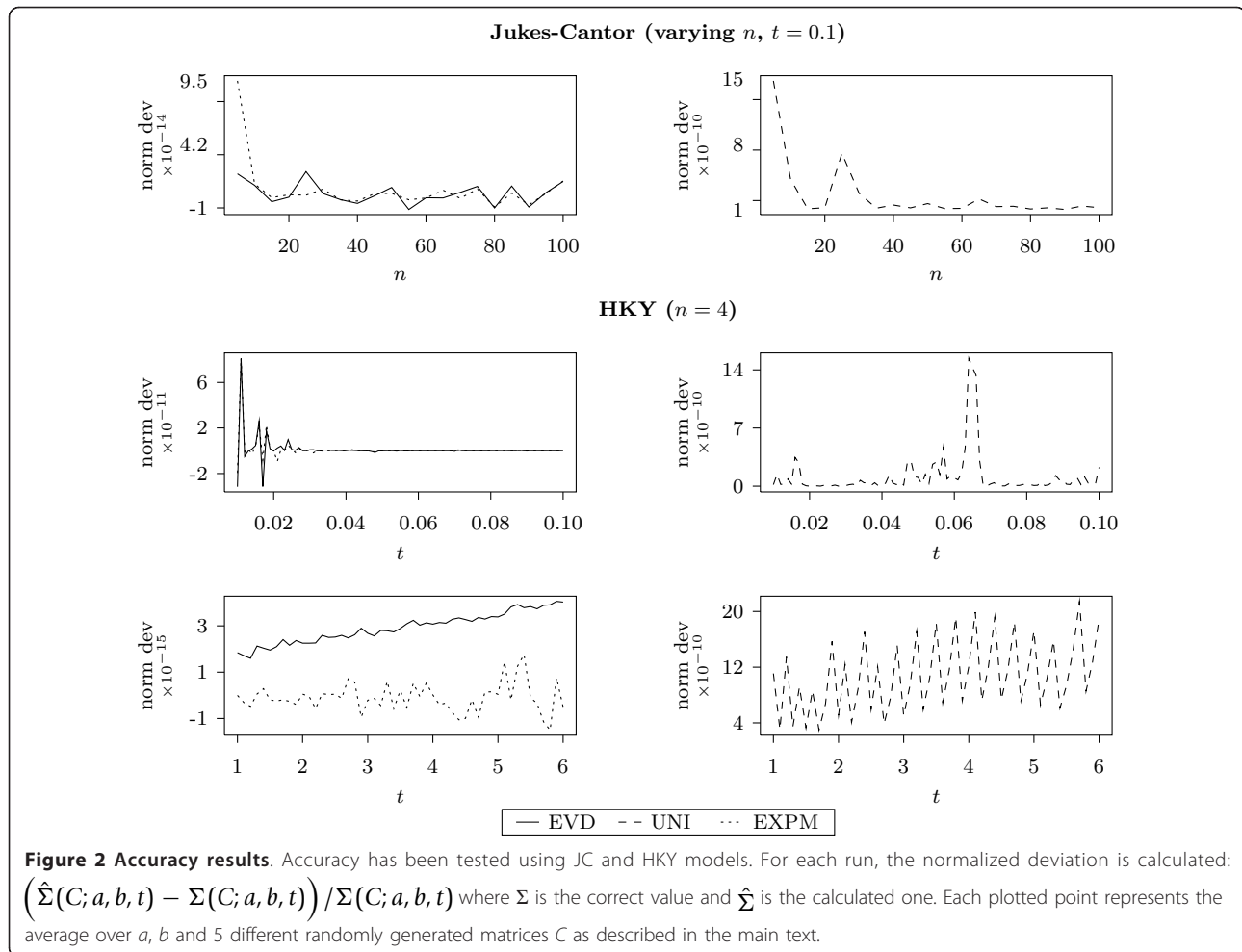
where $\pi = (0.2, 0.2, 0.3, 0.3)$ is the stationary distribution and $\kappa = 2.15$ is the transition/transversion rate ratio. This rate matrix has an analytical result for (4) which can be obtained through the eigenvalue decomposition. The eigenvalues and eigenvectors of Q are

$$\lambda = (0, -1, -\pi_Y\kappa - \pi_R, -\pi_R\kappa - \pi_Y) \qquad \text{where } \pi_R = \pi_A + \pi_G \text{ and } \pi_Y = \pi_C + \pi_T,$$

$$U = \begin{pmatrix} 1 & -\frac{\pi_Y}{\pi_R} & 0 & -\frac{\pi_G}{\pi_A} \\ 1 & -\frac{\pi_Y}{\pi_R} & 0 & 1 \\ 1 & 1 & -\frac{\pi_T}{\pi_C} & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \qquad U^{-1} = \begin{pmatrix} \pi_A & \pi_G & \pi_C & \pi_T \\ -\pi_A & -\pi_G & \frac{\pi_C\pi_R}{\pi_Y} & \frac{\pi_T\pi_R}{\pi_Y} \\ 0 & 0 & -\frac{\pi_C}{\pi_Y} & \frac{\pi_C}{\pi_Y} \\ -\frac{\pi_A}{\pi_R} & \frac{\pi_A}{\pi_R} & 0 & 0 \end{pmatrix}.$$

From this, using the symbolic operations in Matlab [20], we obtained the final analytic expression for (4). Using this model we compared for all three methods the true value of (5) for various values of t and randomly generated binary matrices C. For each t we generated 5 different matrices C. The average normalized deviation is shown in Figure 2.

In both cases, all methods showed good accuracy as the normalized deviation was no bigger than 3×10^{-9}. We also note that EXPM tended to be the most precise while UNI provided the worst approximation. To further investigate the accuracy, we performed calculations on randomly generated reversible rate matrices: we first obtained the stationary distribution from the Dirichlet distribution with shape parameters equal to 1, then all entries q_{ij} with $i \geq j$ from the exponential distribution with parameter 1 and finally calculated the remaining entries using the reversibility property. In all

Figure 2 Accuracy results. Accuracy has been tested using JC and HKY models. For each run, the normalized deviation is calculated: $\left(\hat{\Sigma}(C; a, b, t) - \Sigma(C; a, b, t)\right)/\Sigma(C; a, b, t)$ where Σ is the correct value and $\hat{\Sigma}$ is the calculated one. Each plotted point represents the average over a, b and 5 different randomly generated matrices C as described in the main text.

the runs the relative difference between EVD, UNI and EXPM was less than 10^{-5}. This indicated that all three methods have a similar performance in a wide range of applications.

Speed

Partition of computation Assume we need to evaluate $\Sigma(C; t)$ for a fixed matrix C and multiple time points $t \in \{t_1,...t_k\}$. In each iteration of the EM-algorithm in Application 1 we need this type of calculation while in order to calculate the labeled distance in Application 2 just one time point is required. Using EVD (Algorithm 1) we do the eigenvalue decomposition (Step 1) once and then, for each time point t_i, we apply Step 2 and Step 3. The eigenvalue decomposition, achieved through the R function *eigen*, has a running time of $O(n^3)$. In Step 2 we determine $J(t)$ and this takes $O(n^2)$ time. Step 3 has a running time of $O(n^3)$ due to the matrix multiplications.

If instead we apply UNI (Algorithm 2), we run Steps 1-3 for the largest time point $\max(t_i)$ and then, for each time point t_i, we apply Step 4. Steps 1-3 take $O(s(\mu\max(t_i))) n^3$ time, and Step 4 takes $O(s(\mu t_i)n^2)$ time for each

$i \in \{1,..., k\}$. Therefore, even though the total time for both methods is $O(n^3)$, the addition of one time point contributes with $O(n^3)$ for EVD, but only $O(s(\mu t)n^2)$ for UNI. Recall that the constant $s(\mu t)$ is the truncation point for the infinite sum in the uniformization method.

In the case of EXPM (Algorithm 3) we need to calculate the matrix exponential at every single time point. We used the *expm* R package [21] with the *Higham08* method. This is a Padé approximation combined with an improved scaling and squaring [22] and balancing [23]. The running time is $O(n^3)$.

Table 1 provides an overview of the running times for each of the methods. The algorithms are divided into precomputation and main computation where the precomputation consists of steps that must be executed only once, while in the main computation we calculate the value of $\Sigma(C;t)$ for every time point under consideration.

Experiments We tested the speed of the algorithms in six experiments based on the presented applications and two more experiments using a non-reversible matrix.

Table 1 Running time complexity

Method	EVD	UNI	EXPM
Precomputation			
Steps	1	1-3	none
Order	$O(n^3)$	$O(s(\mu t)n^3)$	
Main Computation			
Steps	2-3	4	1-3
Order	$O(n^3)$	$O(s(\mu t)n^2)$	$O(n^3)$

Table 2 Experimental design

GY

Experiment		2			3	
k	t_k	μt_k	$s(\mu_{tk})$	t_k	μt_k	$s(\mu_{tk})$
1	0.0017	0.0045	5	0.1	0.2668	8
2	0.0032	0.0085	5	0.2	0.5337	9
3	0.0046	0.0124	5	0.3	0.8005	11
4	0.0061	0.0163	5	0.4	1.0674	12
5	0.0076	0.0202	5	0.5	1.3342	13
6	0.0090	0.0241	5	0.6	1.6010	14
7	0.0105	0.0281	6	0.7	1.8679	15
8	0.0120	0.0320	6	0.8	2.1347	15
9	0.0135	0.0359	6	0.9	2.4015	16
10	0.0150	0.0398	6	1.0	2.6684	17

GTR

Experiment		5			6	
k	t_k	μt_k	$s(\mu t_k)$	t_k	μt_k	$s(\mu t_k)$
1	0.1760	0.2668	8	0.1	0.1516	7
2	0.3520	0.5337	9	0.6	0.9098	11
3	0.5280	0.8005	11	1.1	1.6680	14
4	0.7039	1.0674	12	1.6	2.4262	16
5	0.8798	1.3342	13	2.1	3.1844	18
6	1.0558	1.6010	14	2.6	3.9426	20
7	1.2318	1.8679	15	3.1	4.7008	22
8	1.4077	2.1347	15	3.6	5.4590	24
9	1.5837	2.4015	16	4.1	6.2172	26
10	1.7597	2.6684	17	4.6	6.9754	27

UNR

Experiment		7			8	
k	t_k	μt_k	$s(\mu t_k)$	t_k	μt_k	$s(\mu_{tk})$
1	0.0379	0.1516	7	0.1	0.4	9
2	0.2275	0.9098	11	0.6	2.4	16
3	0.4170	1.6680	14	1.1	4.4	21
4	0.6066	2.4262	16	1.6	6.4	26
5	0.7961	3.1844	18	2.1	8.4	30
6	0.9857	3.9426	20	2.6	10.4	34
7	1.1752	4.7008	22	3.1	12.4	38
8	1.3648	5.4590	24	3.6	14.4	42
9	1.5543	6.2172	26	4.1	16.4	45
10	1.7439	6.9754	27	4.6	18.4	49

The table shows the time points t_k, μ_{tk} and the approximation of the Poission tail $s(\mu_{tk})$. For experiment 2, t_k spanned the interval that contains the 10 longest branch lengths from the phylogeny of the 16 HIV *pol* sequences. In experiment 3 we started at 0.1 and ended at 1. We wished to design experiment 5 such that the corresponding $s(\mu_{tk})$ was the same as $s(\mu_{tk})$ from experiment 3. This allowed us to illustrate the relative performance of the methods when running on different sizes of the rate matrix. Experiment 6 was done on time points starting from 0.1 and ending at 4.6. As before, we wished to design experiment 7 such $s(\mu_{tk})$ corresponded to experiment 6. Experiment 8 used the same t_k as experiment 6.

GY The first experiment corresponded to running the EM algorithm on real data consisting of DNA sequences from the HIV *pol* gene described in [24]. HIV has been extensively studied with respect to selection pressure and drug resistance and in [24] the authors document convergent evolution in *pol* gene caused by drug resistance mutations. The observed data y was a multiple codon alignment of the sequences. For simplicity, we did not consider the columns with gaps or ambiguous nucleotides. To compare the performance of the methods as a function of the size of the data set, we applied the EM algorithm for 15 data sets containing from 2 up to 16 sequences each, extracted from the HIV *pol* gene data. For each set we assumed the sequences were related according to a fixed tree; we have reconstructed the phylogenies in Mega [25] using the Jukes-Cantor model and Neighbor-Joining. We ran the EM algorithm until all three parameters converged. Experiments two and three used the previously estimated matrix Q given by (6) with $\alpha = 10.5$, $\kappa = 4.27$ and $\omega = 0.6$. We let $C_{ij} = q_{ij}$ and $C_{ii} = 0$, corresponding to calculating the total number of expected substitutions $\mathbb{E}[N|t, a, b]$, and computed the value of $\Sigma(C; t_k)$ for 10 equidistant sorted time points t_k with $1 \le k \le 10$ (Table 2).

GTR In the fourth experiment we estimated the robust labeled distance of two sequences, using the same setup as in [8]. For each considered evolutionary distance t between 0.1 and 1, we generated 50 pairwise sequence data sets of length 2000 which have evolved for time t under the general time reversible (GTR) model with

$$Q = \begin{pmatrix} \cdot & r_1\pi_G & r_2\pi_C & r_3\pi_T \\ r_1\pi_A & \cdot & r_4\pi_C & r_5\pi_T \\ r_2\pi_A & r_4\pi_G & \cdot & r_6\pi_T \\ r_3\pi_A & r_5\pi_G & r_6\pi_C & \cdot \end{pmatrix}$$

where $r = (0.5, 0.3, 0.6, 0.2, 0.3, 0.2)$ and $\pi = (0.2, 0.2, 0.3, 0.3)$. For labeling, we considered the jumps to and from nucleotide A, leading to $C_{ij} = q_{ij}$ if i or j represents nucleotide A. For each data set, we estimated the GTR parameters as described in [8] and calculated the robust distance. Experiments 5 and 6 used the same GTR matrix and $C_{ij} = q_{ij}$ if i or j represents nucleotide A and zero otherwise, and computed the

value of $\Sigma(C; t_k)$ for 10 equidistant sorted time points t_k with $1 \le k \le 10$ (Table 2).

UNR In the last two experiments we used the same setup as in experiments 5 and 6 but with a different matrix

and time points (Table 2). As the speed of EVD is influenced by the type of the model, we decided to employ a non-reversible matrix. We chose the unrestricted model and carefully set the rates such that the matrix has a complex decomposition:

$$
Q = \begin{pmatrix} -4 & 2 & 1 & 1 \\ 0 & -3 & 2 & 1 \\ 1 & 0 & -3 & 2 \\ 2 & 1 & 1 & -4 \end{pmatrix}.
$$

Figure 3 shows the results. For experiments 1 and 4, the plots show the recorded running time under each set-up (different number of sequences or different evolutionary distance). For the remaining experiments each plot starts with the running time of the precomputation which, for UNI, is done on the largest time point t_{10}. Then, at position k, we plot the cumulative running time for precomputation and the evaluation of $\Sigma(C; t_i)$ for all $i \leq k$. Since EVD and EXPM have running times that are independent of t_k, the running times for these two algorithms are the same in experiments 2 and 3, 5 and 6, and 7 and 8. Even more, as EXPM is dependent only on the size of the matrix, the running times in experiments 5-8 are the same. We observe that in all our experiments EXPM is the slowest method. Deciding if EVD or UNI is faster depends on the size and type of the matrix, the number of time points and the values of $s(\mu t)$. As the main computation for UNI has a running time of $O(n^2)$ as opposed to $O(n^3)$ for EVD (Table 1), this method should have an increased advantage when the rate matrix is bigger. This means that if many time points are considered, then UNI is generally the faster method. Importantly, we note that the EVD precomputation tends to be faster than the UNI precomputation. We remark that, in the first experiment, UNI proved to be the fastest method while, in the fourth experiment, UNI became slower with the increase of the evolutionary distance between the sequences and it was only faster than EVD for small distances (< 0.2). By setting t_k in an appropriate manner (Table 2), we have the same running time for UNI and EXPM for experiment 7 compared to experiment 6. Due to the fact that in experiment 7 we used the UNR matrix, EVD is slower as opposed to experiment 6. In this case, the difference is observable but not very big, but as the size of the matrix increases, this discrepancy increases too. We also note that the difference between the reversible and non-reversible cases is enough to make UNI faster than EVD in the latter case.

Discussion

The EVD algorithm assumes that the rate matrix is diagonalizable. However, a direct calculation of the integral (4) in the non-diagonalizable case is actually possible using the Jordan normal form for the rate matrix. Let $Q = PJP^{-1}$ where J is the Jordan normal form of Q and P consists of the generalized eigenvectors (we recognize that we used P and J for other quantities earlier but for this discussion this should not cause any confusion and we prefer to use standard notation), i.e. J has a block diagonal form $J = \text{diag}(J_1, ..., J_\kappa)$ where $J_k = \lambda_k I + N$ is a matrix with λ_k on the diagonal and 1 on the superdiagonal. We have

$$
e^{Qt} = P\text{diag}(e^{J_1 t}, \dots, e^{J_\kappa t})P^{-1}, \tag{26}
$$

and noting that N is a nilpotent matrix with degree d_k (equal to the size of block J_k) we obtain

$$
e^{J_k t} = e^{t\lambda_k}e^{tN} = e^{\lambda_k t}\left(I + tN + \frac{t^2}{2}N^2 + \ldots + \frac{t^{d_k-1}}{(d_k-1)!}N^{d_k-1}\right). \tag{27}
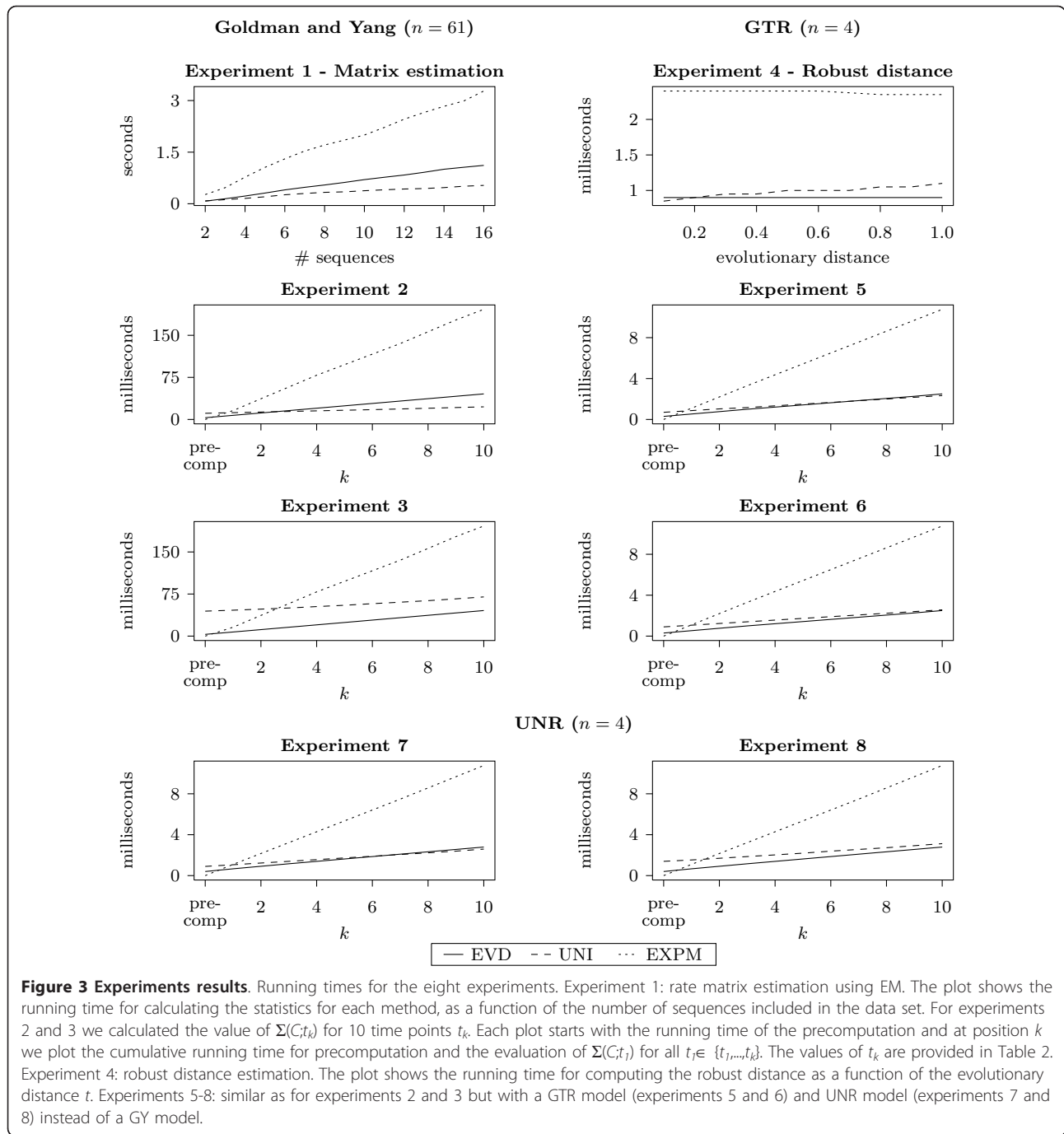$$

In order to calculate the integral (4) the expressions (26) and (27) are used. It is evident that this procedure is feasible but also requires much bookkeeping.

In [26] an extension of uniformization, adaptive uniformization, is described for calculating transition probabilities in a CTMC. The basic idea is to perform a local uniformization instead of a global uniformization of the rate matrix and thereby have fewer jumps in the jump process. [26] considers a model with rate matrix

$$
Q = \begin{pmatrix} -3v & 3v & 0 & 0 \\ \mu & -(\mu+2v) & 2v & 0 \\ 0 & \mu & -(\mu+v) & v \\ 0 & 0 & 0 & 0 \end{pmatrix}
$$

(state 4 is an absorbing state). If this process starts in state 1 then the first jump is to state 2 and the second is from state 2 to either state 1 or state 3. This feature can be taken into account by having a so-called adaptive uniformized (AU) jump process where the rate for the first jump is $3v$, for the second is $\mu + 2v$ and, assuming $\mu + v > 3v$, the rate for the third jump is $\mu + v$. From the third jump the rate in the AU jump process is $\mu + 2v$ as in the standard uniformized jump process. The AU jump process has a closed-form expression for the jump probabilities (it is a pure birth process) but is of course more complicated than a Poisson jump process. The advantage is that the AU jump process exhibits fewer jumps. This procedure could very well be useful for codon models where the set of states that the process can be in after one or two jumps are limited because only one nucleotide change is allowed in each state change.

In an application concerned with modeling among-site rate variation, [27] applies the uniformization procedure (20) to calculate the transition probabilities instead of the eigenvalue decomposition method (15). [27] shows,

Figure 3 Experiments results. Running times for the eight experiments. Experiment 1: rate matrix estimation using EM. The plot shows the running time for calculating the statistics for each method, as a function of the number of sequences included in the data set. For experiments 2 and 3 we calculated the value of $\Sigma(C;t_k)$ for 10 time points t_k. Each plot starts with the running time of the precomputation and at position k we plot the cumulative running time for precomputation and the evaluation of $\Sigma(C;t_l)$ for all $t_l \in \{t_1,...,t_k\}$. The values of t_k are provided in Table 2. Experiment 4: robust distance estimation. The plot shows the running time for computing the robust distance as a function of the evolutionary distance t. Experiments 5-8: similar as for experiments 2 and 3 but with a GTR model (experiments 5 and 6) and UNR model (experiments 7 and 8) instead of a GY model.

in agreement with our results, that uniformization is a faster computational method than eigenvalue decomposition.

The presented methods are not the only ones for calculating the desired summary statistics. For example, in [5] it is suggested to determine the expected number of jumps from the direct calculation

$$p_{ab}(t)E[N_{cd}|t,a,b] = \int_0^t (e^{Qs})_{ac} q_{cd} (e^{Q(t-s)})_{ac} ds$$
$$= \sum_\infty^{i=0} \sum_\infty^{j=0} (Q^i)_{ac} q_{cd} (Q^j)_{db} \int_0^t \frac{s^i(t-s)^j}{i!j!} ds$$
$$= \sum_\infty^{k=1} \frac{t^k}{k!} \sum_{k-1}^{m=0} (Q^m)_{ac} q_{cd} (Q^{k-m-1})_{db},$$

where the infinite sum is truncated at $k = 10$. The problem with this approach is that it is difficult to

bound the error introduced by the truncation. In UNI a similar type of calculation applies but the truncation error can be controlled.

Conclusion

Recall that EVD assumes that the rate matrix is diagonalizable and this constraint means that EVD is less general than the other two algorithms. We have shown in the Discussion how a direct calculation of the integral (4) is actually still possible but requires much bookkeeping. On top of being less general, EVD is dependent on the type of the matrix: reversible or non-reversible. We have shown how this discrepancy can make EVD slower than UNI even when the state space has size of only 4.

We found that the presented methods have similar accuracy and EXPM is the most accurate one. With respect to running time, it is not straightforward which method is best. We found that both the eigenvalue decomposition (EVD) and uniformization (UNI) are faster than the matrix exponentiation method (EXPM). The main reason for EVD and UNI being faster is that they can be decomposed into a precomputation and a main computation. The precomputation only depends on the rate matrix for EVD while for UNI it also depends on the largest time point and the matrix C. We also remark that EXPM involves the exponentiation of a matrix double in size. UNI is particularly fast when the product μt is small because in this case only a few terms in the sum (22) are needed.

Authors' contributions
PT extended the existing methods to linear combinations of statistics, implemented the algorithms and performed the testing. AH conceived the study and guided the development and evaluation of the methods. Both authors wrote the paper. All authors read and approved the final manuscript.

References
1. Holmes I, Rubin GM: **An expectation maximization algorithm for training hidden substitution models.** *J Mol Bio* 2002, **317**:753-764.
2. Klosterman PS, Holmes I: **XRate: a fast prototyping, training and annotation tool for phylo-grammars.** *BMC Bioinf* 2006, **7**:428.
3. Kosiol C, Holmes I, Goldman N: **An empirical codon model for protein sequence evolution.** *Mol Biol Evol* 2007, **24**:1464-79.
4. Minin VN, Suchard MA: **Counting labeled transitions in continuous-time Markov models of evolution.** *J Math Biol* 2008, **56**:391-412.
5. Dutheil J, Pupko T, Jean-Marie A, Galtier N: **A Model-Based Approach for Detecting Co-evolving Positions in a Molecule.** *Mol Biol Evol* 2008, **22**:1919-1928.
6. Dutheil J, Galtier N: **Detecting groups of co-evolving positions in a molecule: a clustering approach.** *BMC Evol Biol* 2007, **7**:242.
7. Minin VN, Suchard MA: **Fast, accurate and simulation-free stochastic mapping.** *Phil Trans R Soc B* 2008, **363(1512)**:3985-3995.
8. O'Brien JD, Minin VN, Suchard MA: **Learning to count: robust estimates for labeled distances between molecular sequences.** *Mol Biol Evol* 2009, **26**:801-814.
9. Dutheil J: **Detecting site-specific biochemical constraints through substitution mapping.** *J Mol Evol* 2008, **67**:257-65.
10. Siepel A, Pollard KS, Haussler D: **New methods for detecting lineage-specific selection.** *Proceedings of the 10th International Conference on Research in Computational Molecular Biology (RECOMB)* 2006, 190-205.
11. Hobolth A, Jensen JL: **Summary statistics for end-point conditioned continuous-time Markov chains.** *J Appl Prob* 2011, **48**:1-14.
12. Van Loan CF: **Computing integrals involving the matrix exponential.** *IEEE Transactions on Automatic Control* 1978, **23**:395-404.
13. R Development Core Team: R: **A Language and Environment for Statistical Computing.**[http://www.R-project.org], R Foundation for Statistical Computing.
14. Goldman N, Yang Z: **A Codon-based Model of Nucleotide Substitution for Protein-coding DNA Sequences.** *Mol Biol Evol* 1994, **11**:725-736.
15. Hobolth A, Jensen JL: **Statistical Inference in Evolutionary Models of DNA Sequences via the EM Algorithm.** *Stat App Gen Mol Biol* 2005, **4**:18.
16. Dempster AP, Laird NM, Rubin DB: **Maximum Likelihood from Incomplete Data via the EM Algorithm.** *J R Statist Soc B* 1977, **39**:1-38.
17. Yap VB, Speed T: **Estimating Substitution Matrices.** In *Statistical Methods in Mol Evolution.* Edited by: Nielsen R. Springer; 2005:420-422.
18. Felsenstein J: **Evolutionary trees from DNA sequences: a maximum likelihood approach.** *J Mol Evol* 1981, **17**:368-376.
19. Jensen A: **Markov chains as an aid in the study of Markov processes.** *Skand Aktuarietidskr* 1953, **36**:87-91.
20. MATLAB R2010a. Natick, Massachusetts: The MathWorks Incorporated.
21. Goulet V, et al: **expm: Matrix exponential.**[http://CRAN.R-project.org/package=expm].
22. Higham J: **The Scaling and Squaring Method for the Matrix Exponential Revisited.** *SIAM Review* 2003, **51**:747-764.
23. Stadelmann M: **Matrixfunktionen. Analyse und Implementierung.** *Master thesis* ETH Zurich, Mathematics Department; 2009.
24. Lemey P, et al: **Molecular footprint of drug-selective pressure in a human immunodeficiency virus transmission chain.** *J Virol* 2005, **79**:11981-11989.
25. Tamura K, et al: **MEGA5: Molecular Evolutionary Genetics Analysis using Maximum Likelihood, Evolutionary Distance, and Maximum Parsimony Methods.** *Mol Biol Evol Advance Access* 2011.
26. Van Moorsel APA, Sanders WH: **Adaptive uniformization.** *Stochastic Models* 1994, **10**:619-647.
27. Mateiu L, Rannala B: **Inferring complex DNA substitution processes on phylogenies using uniformization and data augmentation.** *Systematic Biol* 2006, **55**:259-269.