Bayesian Analysis (2018) **13**, Number 4, pp. 1033–1059

Designing Simple and Efficient Markov Chain Monte Carlo Proposal Kernels

Yuttapong Thawornwattana[*], Daniel Dalquen[*], and Ziheng Yang[*,†‡]

Abstract. We discuss a few principles to guide the design of efficient Metropolis–Hastings proposals for well-behaved target distributions without deeply divided modes. We illustrate them by developing and evaluating novel proposal kernels using a variety of target distributions. Here, efficiency is measured by the variance ratio relative to the independent sampler. The first principle is to introduce negative correlation in the MCMC sample or to reduce positive correlation: *to propose something new, propose something different*. This explains why single-moded proposals such as the Gaussian random-walk is poorer than the uniform random walk, which is in turn poorer than the bimodal proposals that avoid values very close to the current value. We evaluate three new bimodal proposals called Box, Airplane and StrawHat, and find that they have similar performance to the earlier Bactrian kernels, suggesting that the general shape of the proposal matters, but not the specific distributional form. We propose the "Mirror" kernel, which generates new values around the mirror image of the current value on the other side of the target distribution (effectively the "opposite" of the current value). This introduces negative correlations, leading in many cases to efficiency of $> 100\%$. The second principle, applicable to multidimensional targets, is that a sequence of well-designed one-dimensional proposals can be more efficient than a single d-dimensional proposal. Thirdly, we suggest that variable transformation be explored as a general strategy for designing efficient MCMC kernels. We apply these principles to a high-dimensional Gaussian target with strong correlations, a logistic regression problem and a molecular clock dating problem to illustrate their practical utility.

Keywords: asymptotic variance, bimodal kernel, Metropolis–Hastings algorithm, Mirror kernel, molecular clock dating, variable transformation.

1 Introduction

Markov chain Monte Carlo (MCMC) is a class of algorithms for generating samples from a probability distribution π on $X \subset \mathbf{R}^d$, where π may be known up to a normalizing constant. It is widely used to simulate samples from the posterior distribution in Bayesian inference where the normalizing constant is usually intractable. The Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) simulates a discrete-time Markov chain on X with stationary distribution π as follows. Given the chain is currently at x, a potential next state x' is generated from a proposal kernel Q

[*]Department of Genetics, Evolution and Environment, University College London, London, WC1E 6BT, UK

[†]Radcliffe Institute for Advanced Studies, Harvard University, Cambridge, MA 02138, USA

[‡]Corresponding author (z.yang@ucl.ac.uk)

https://doi.org/10.1214/17-BA1084

on X, with density $q(x'|x)$. The chain then moves to x' with probability

$$\alpha(x,x') := \min\left(1, \frac{\pi(x')}{\pi(x)}\frac{q(x|x')}{q(x'|x)}\right). \tag{1}$$

Otherwise it stays at x. The resulting Markov chain has transition kernel P with density

$$p(x'|x) = \begin{cases} q(x'|x)\alpha(x,x') & \text{if } x' \neq x, \\ 1 - \int_X q(x'|x)\alpha(x,x')\,dx & \text{if } x' = x. \end{cases}$$

By construction, this Markov chain is reversible with respect to π, with $\pi(x)p(x'|x) = \pi(x')p(x|x')$ for almost every $x, x' \in X$. If the proposal kernel Q is irreducible and aperiodic, the transition kernel P will also be irreducible, aperiodic and is π-invariant.

A simulated path $(x_n)_{n=1}^N$ of the chain can be used to estimate an expectation under π. Let $f : X \to \mathbf{R}$ be an absolutely integrable function and let $\pi(f) := \mathbf{E}_\pi f(x) = \int_X \pi(x)f(x)\,dx$ denote the expected value of f under π. Then $\pi(f)$ can be estimated by

$$\hat{\pi}(f) := \frac{1}{N}\sum_{n=1}^{N} f(x_n). \tag{2}$$

Provided the Markov chain is ergodic with stationary distribution π, this estimator $\hat{\pi}(f)$ converges to $\pi(f)$ almost surely as $N \to \infty$ (see e.g. Theorem 3 in Tierney (1994)). Moreover, under certain ergodicity assumptions, the central limit theorem holds for $\sqrt{N}\hat{\pi}(f)$ (see e.g. Theorems 4 and 5 in Tierney (1994)), with the asymptotic variance

$$\nu := \lim_{N\to\infty} N\mathrm{Var}(\hat{\pi}(f)) = V_f\left(1 + 2\sum_{k=1}^{\infty}\rho_k\right), \tag{3}$$

where $V_f := \mathrm{Var}_\pi(f(x)) = \mathbf{E}_\pi(f(x) - \mathbf{E}_\pi f(x))^2$ is the variance of $f(x)$ under π, and $\rho_k := \mathrm{Cor}(f(x_n), f(x_{n+k}))$ is the lag-k autocorrelation.

When the state space X is discrete, Peskun (1973) showed that given a transition kernel Q, the choice of the acceptance probability α in (1) is optimal in terms of minimising the asymptotic variance of $\hat{\pi}(f)$. The analogous result for continuous state spaces was provided by Tierney (1998). However, what features the proposal kernel Q should have to minimise the asymptotic variance is not clear. The most common choice of Q is based on the random walk $x' = x + u$ where u has a Gaussian or uniform distribution with variance σ^2. The Langevin proposal $x' = x + \frac{\sigma^2}{2}\nabla_x \log \pi(x) + u$ with $u \sim N(0, \sigma^2)$ makes use of gradient information to bias the proposal towards a local mode of the target. All these proposals involve a step-size parameter σ yet to be specified.

It is well known that a poor choice of σ can adversely affect the mixing and convergence properties of the algorithm. Determining the optimal choice of the step-size parameter σ is an active area of research, known as optimal scaling. Gelman et al.

(1996) estimated the optimal σ (that minimises ν) for the Gaussian random walk $q(x'|x) = N(x'|x, \sigma^2)$ for estimating the mean of the $N(0,1)$ target to be about 2.38, with the corresponding expected acceptance probability

$$P_{\text{jump}} := \int_X \int_X \pi(x)\alpha(x,x')q(x'|x)\,dx'\,dx$$

to be about 0.44. When the target distribution π is d-dimensional with identically distributed components, Roberts et al. (1997) showed that for the Gaussian kernel $q(x'|x) = N(x'|x, \sigma^2 I_d)$, the optimal step size σ that minimises ν is the one that leads to $P_{\text{jump}} \approx 0.234$ as $d \to \infty$. Recent work on optimal scaling covers more complex algorithms such as Multiple-try Metropolis (Bédard et al., 2012), delayed rejection MH (Bédard et al., 2014), Hamiltonian Monte Carlo (HMC) (Beskos et al., 2013), as well as the MH and Metropolis-adjusted Langevin algorithm (MALA) in the infinite-dimensional setting (Beskos et al., 2009; Pillai et al., 2012). However, optimal scaling analysis has been mostly limited to the Gaussian random walk and the Langevin proposal (Roberts and Rosenthal, 1998). Beyond this small collection of proposal kernels, there is no general theory available.

In this work, we address the problem of designing efficient proposal kernels (Q) for the MH algorithm, with each kernel implemented at a nearly optimal scale. Recently, Yang and Rodríguez (2013) empirically demonstrated that using the so-called Bactrian kernels can substantially improve the asymptotic efficiency for a range of univariate target distributions, compared with the uniform random walk, which is in turn more efficient than the Gaussian random walk. We extend this work by proposing several new proposal kernels and evaluate their statistical efficiency at the optimal step size.

In Section 2, we describe the calculation of efficiency and automatic adjustment of the proposal step size (σ). We present new kernels for one-dimensional targets in Section 3, and consider multidimensional targets in Section 4. In Section 5, we apply the new kernels to the Bayesian molecular clock dating problem in molecular phylogenetics. We discuss limitations of our work as well as connections to previous work in Section 6.

2 Targets and efficiency calculation

2.1 Target and proposal distributions

We consider the following five target distributions (Figure 1): (a) Standard normal distribution $N(0,1)$, with mean 0; (b) Mixture of two normal distributions $\frac{1}{4}N(-1,\frac{1}{4}) + \frac{3}{4}N(1,\frac{1}{4})$, with mean $\frac{1}{2}$; (c) Mixture of two t_4 distributions $\frac{3}{4}t_4(-\frac{3}{4},s^2) + \frac{1}{4}t_4(\frac{3}{4},s^2)$, where $s = \frac{1}{8}\sqrt{\frac{37}{2}}$, with mean $-\frac{3}{8}$; (d) Gamma distribution $G(4,2)$, with mean 2; (e) Uniform distribution $U(-\sqrt{3},\sqrt{3})$, with mean 0. Each of the five targets has variance 1. Note that even though the density in (b) has two modes, we focus in this paper on simple targets with a single mode; we do not expect the proposals discussed here to work well when the target has multiple peaks separated by deep valleys.

Note that targets (d) and (e) have a constrained support. Sampling from targets with constrained support is often dealt with using rejection or truncated proposals (or

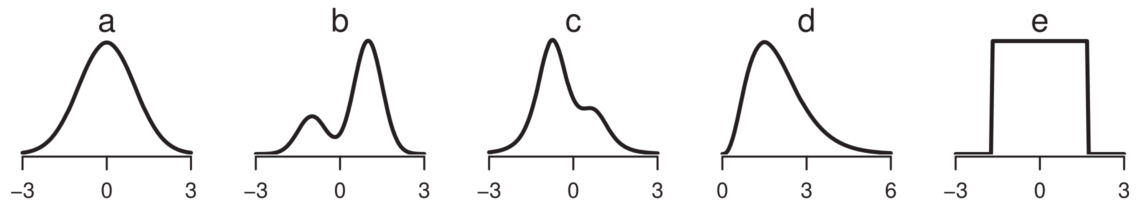


Figure 1: Five target distributions: (a) standard normal $N(0,1)$, (b) mixture of two normals $\frac{1}{4}N(-1,\frac{1}{4})+\frac{3}{4}N(1,\frac{1}{4})$, (c) mixture of two t_4 distributions $\frac{3}{4}t_4(-\frac{3}{4},s^2)+\frac{1}{4}t_4(\frac{3}{4},s^2)$, (d) gamma $G(4,2)$ and (e) uniform $U(-\sqrt{3},\sqrt{3})$.

truncated full conditionals in the context of Gibbs sampling) (Gelfand et al., 1992; Browne, 2006). We note that rejection can be very inefficient if a large proportion of proposed values are discarded, while the truncated variables can be expensive to simulate, often based on the inverse transform method (Devroye, 1986, p. 38). It is simpler and typically more efficient (in terms of the amount of computation involved as well as the asymptotic variance of the estimator) to use reflection (e.g. Yang and Rodríguez (2013)). For example, if x has support on the interval $[a,\infty)$ and if the kernel is symmetric with $q(x'|x)=q(x|x')$, we generate $x' \sim q(x'|x)$, and set $x' \leftarrow 2a-x'$ if $x'<a$. The proposal ratio is 1.

We evaluate five new proposals (Box, Airplane, StrawHat, MirrorU and MirrorN; figures 2 and 3) described in Section 3, together with the uniform, Gaussian and BactrianTriangle proposals from Yang and Rodríguez (2013).

2.2 Efficiency calculation

As in Gelman et al. (1996), we define statistical efficiency of the estimator $\hat{\pi}(f)$ (2) as the ratio of the asymptotic variance of $\hat{\pi}(f)$ for an iid sample to the variance for an MCMC sample of the same size

$$E := \frac{V_f/N}{\nu/N} = \frac{V_f}{\nu} = \left(1+2\sum_{k=1}^{\infty}\rho_k\right)^{-1}. \tag{4}$$

We use the identity function $f=1$ in (2) and estimate the mean of π. We used two methods to estimate E, one based on discretization of the target distribution, and another based on the MCMC sample.

Method 1: Using the transition matrix on a discretized state space The state space $X \subset \mathbf{R}$ is truncated to an interval $[x_L,x_U]$, then discretized into K bins of width $\Delta=\frac{x_U-x_L}{K}$. For each bin $k=1,\ldots,K$, we use the midpoint $x_k := x_L+(k-1/2)\Delta$ as its representative. Then we compute the transition matrix P on the discretized space, and calculate E using a closed form expression from Kemeny and Snell (1960) or Peskun (1973). This method requires an analytic expression of the proposal density $q(x'|x)$. The calculation details are described in Gelman et al. (1996) and Yang and Rodríguez (2013).

Here, we use $x_L = -5$, $x_U = 5$ and $K = 500$ for all targets, except for the mixture of two t_4, where we use $x_L = -10$, $x_U = 10$ and $K = 1000$.

While our focus is on the mixing property of the Markov chain at stationarity, we also consider two measures of the convergence rate of the Markov chain, namely the absolute value of the second largest eigenvalue of P, denoted $|\lambda|_2$, and the largest total variation distance to the target distribution after n steps among all possible starting points, denoted δ_n. We use $n = 8$. These measures are computed on the discretized space. See Yang and Rodríguez (2013).

Method 2: Using MCMC samples The initial positive sequence method of Geyer (1992) is used, which truncates the infinite sum of (4) as soon as $\rho_{k-1} + \rho_k < 0$. This uses an MCMC run and requires the evaluation of the proposal ratio, but not the proposal density. Our experience suggests that small sample sizes (say, $N < 10^5$) do not allow reliable estimation, especially when E is small. We typically use $N = 10^7$ to 10^8, after a burn-in of 10^4 iterations.

We also use the following efficiency measure in the case of one-dimensional targets

$$E_\pi^2 := \mathbf{E}(x_n - x_{n-1})^2 = 2(1-\rho_1)\mathrm{Var}(x),$$

where the expectation is over the joint distribution of x_{n-1} and x_n. This has been used for optimal scaling (Sherlock and Roberts, 2009) and adaptive MCMC (Pasarica and Gelman, 2010). Maximising E_π^2 is equivalent to minimising the first-order autocorrelation ρ_1.

2.3 Tuning of the proposal step size

Ideally, the proposal step size σ should be set to give the optimal efficiency E. A computationally intensive approach is to run the algorithm for a range of σ values and choose the one that gives the highest efficiency, referred to as grid evaluation. This is expensive and may not be practical in real applications. It is used for one-dimensional targets in Section 3. In Section 4, we employ an approach of automatic scale adjustment (Yang and Rodríguez, 2013), where we monitor P_{jump} and use it to adjust σ for a one-dimensional proposal. Note that there is a monotonic decreasing relationship between σ and P_{jump} (larger σ meaning smaller P_{jump}). Specifically, we use the relationship $P_{\text{jump}}(\sigma) = \frac{2}{\pi}\tan^{-1}(2/\sigma)$, for the $N(0,1)$ target and $x'|x \sim N(x, \sigma^2)$ kernel (Gelman et al., 1996), to derive the update formula

$$\sigma^* = \sigma \frac{\tan(\frac{\pi}{2} P_{\text{jump}})}{\tan(\frac{\pi}{2} P^*_{\text{jump}})}, \tag{5}$$

where σ is the current step size, P_{jump} is the observed acceptance proportion, while σ^* and P^*_{jump} are the optimal values. The optimal P_{jump} is around 0.4 for unimodal kernels (including Gaussian and uniform kernels) and 0.3 for bimodal kernels (including Bactrian, Box, Airplane and StrawHat kernels); see Section 3 and Yang and Rodríguez (2013). We update σ several times during the burn-in.

Choice of the step size σ for the Mirror moves is discussed below.

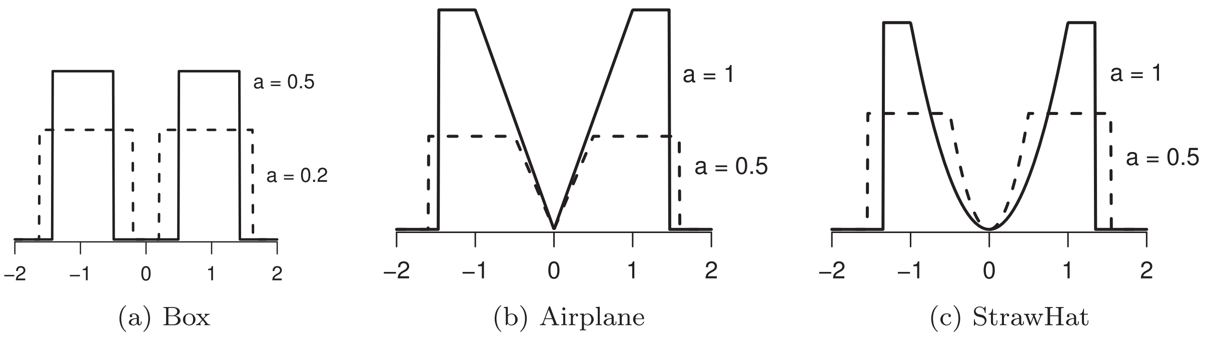


Figure 2: Box, Airplane and StrawHat proposals. Each proposal is a one-parameter family of distributions with parameter a.

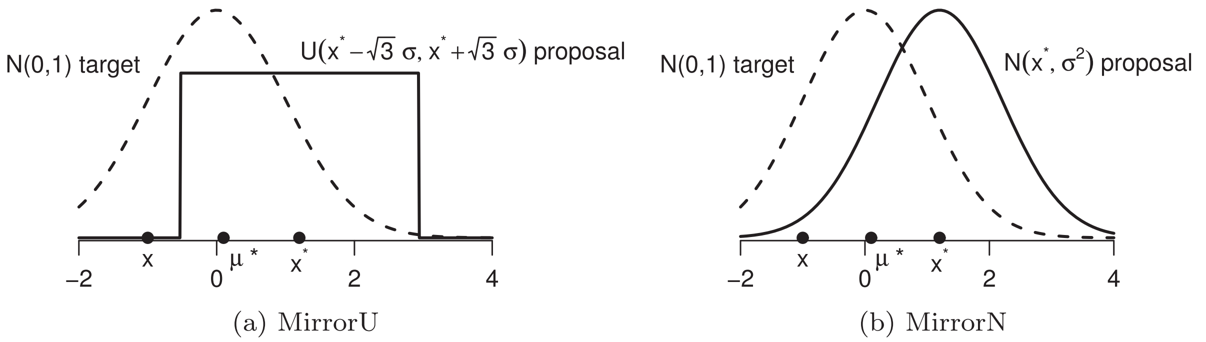


Figure 3: Examples of the proposal distribution for the two Mirror kernels when the current point is $x = -1$ and the estimated "centre" of the target distribution is $\mu^* = 0.1$. The proposal is centred at the mirror point $x^* = 2\mu^* - x$.

3 New one-dimensional proposals

These proposals attempt to reduce the autocorrelation of the Markov chain, thereby improving the precision of the resulting MCMC estimates. One simple approach is to use a bimodal distribution with two modes on both sides of the current position. We describe three such proposals, called *Box*, *Airplane* and *StrawHat* (Figure 2). They have a bimodal shape similar to the Bactrian-type kernels given in Yang and Rodríguez (2013), and are symmetric, with $q(x'|x) = q(x|x')$. We then present a novel family of non-symmetric kernels that induces negative correlations in the Markov chain, called the *Mirror* kernel (Figure 3).

For each of the proposal kernels described below, we first introduce a standard distribution version with zero mean and unit variance. Then given a current point x of the Markov chain, we give the proposal density with mean x and variance σ^2.

3.1 Box

Given x, we generate x' uniformly from two intervals, one on each side of x (Figure 2a). The standard box distribution is $p(y; a) := \frac{1}{2(b-a)}$, $a \le |y| \le b$, where b :

$= \frac{1}{2}(\sqrt{12 - 3a^2} - a)$, and a is a parameter taking values in the interval $[0, 1)$. When $a = 0$, this is $U(-\sqrt{3}, \sqrt{3})$, which is the uniform kernel. In the proposal, we set $x' := x + \sigma y$, where y has the standard box distribution, with density $q(x'|x) = \frac{1}{2\sigma(b-a)}$, $\sigma a \le |x' - x| \le \sigma b$. To sample from $q(x'|x)$, draw $y \sim U(a, b)$ and $u \sim U(0, 1)$. If $u < \frac{1}{2}$, set $y \leftarrow -y$. Then set $x' \leftarrow x + \sigma y$.

3.2 Airplane

The standard Airplane distribution $p(y; a)$ is $\frac{1}{2b-a}$ if $a \le |y| \le b$ and $\frac{1}{2b-a}$ if $|y| < a$, where b is the root of $4b^3 - 12b + 6a - a^3 = 0$ with $b > a$, and $a \in [0, \sqrt{2})$ is a parameter (Figure 2b). The proposal density with mean x and variance σ^2 is $q(x'|x) = \frac{1}{\sigma(2b-a)}$ if $\sigma a \le |x' - x| \le \sigma b$ and $q(x'|x) = \frac{1}{\sigma a(2b-a)}|x' - x|$ if $|x' - x| < \sigma a$. To sample from $q(x'|x)$, draw $u_1, u_2, u_3 \sim U(0, 1)$ independently. If $u_1 < \frac{a}{2b-a}$, set $y \leftarrow a\sqrt{u_2}$, otherwise draw $y \sim U(a, b)$. If $u_3 < \frac{1}{2}$, set $y \leftarrow -y$. Then set $x' \leftarrow x + \sigma y$.

3.3 StrawHat

The standard StrawHat distribution $p(y; a)$ is $\frac{3}{2(3b-2a)}$ if $a \le |y| \le b$ and $\frac{3}{2a^2(3b-2a)}y^2$ if $|y| < a$, where b is the root of $5b^3 - 15b + 10a - 2a^3$ with $b > a$, and $a \in [0, \sqrt{5/3})$ is a parameter. The proposal density $q(x'|x)$ can be derived similarly as for the Airplane kernel. To sample from $q(x'|x)$, draw $u_1, u_2, u_3 \sim U(0, 1)$ independently. If $u_1 < \frac{a}{3b-2a}$, set $y \leftarrow a u_2^{1/3}$, otherwise draw $y \sim U(a, b)$. If $u_3 < \frac{1}{2}$, set $y \leftarrow -y$. Then set $x' \leftarrow x + \sigma y$.

In all three moves (Box, Airplane, StrawHat), when $a = 0$, the move reduces to the uniform kernel. We note that if a is too close to the upper limit, efficiency tends to drop off quickly as σ becomes too large (Figure S1). In practice, we fix a at $a = 0.5$ ($b = 1.43$) for Box, $a = 1$ ($b = 1.47$) for Airplane and $a = 1$ ($b = 1.35$) for StrawHat. Each kernel then has a step size (σ) which can be adjusted to achieve good mixing.

3.4 Mirror

In the Mirror kernel, we generate values around a point on "the other side" of the target distribution that is the "mirror image" of the current point x. Specifically, let μ^* be an estimate of the location of the target such as the mean or median. The proposal kernel is centred at $x^* := 2\mu^* - x$, the point with the same distance from μ^* as the current point x (Figure 3). We consider two variants, using either the uniform or Gaussian distribution. In the *MirrorU* kernel, we have

$$x'|x \sim U(2\mu^* - x - \sqrt{3}\sigma, 2\mu^* - x + \sqrt{3}\sigma), \tag{6}$$

and in the *MirrorN* kernel, we have

$$x'|x \sim N(2\mu^* - x, \sigma^2). \tag{7}$$

Both have mean $2\mu^* - x$ and variance σ^2.

For example, consider the $N(0,1)$ target. If μ^* is the true mean (μ) of the target, optimal asymptotic efficiency (for estimating μ) is achieved by having $\sigma = 0$, in which case $E = \infty$ with $P_{\text{jump}} = 1$. However, in that case, the chain does not sample from the target, and E for estimating other functions may be 0. In general, if μ^* is close to the true mean, one would prefer a small σ to achieve a high efficiency (E), but a small σ may lead to slow convergence to the target distribution. On balance, we suggest two choices of the step size: $\sigma = \hat{s}$ or $\sigma = \frac{1}{2}\hat{s}$, where $\hat{s}$ is the estimated target standard deviation. Both μ^* and $\hat{s}$ are estimated during the burn-in of the MCMC.

If the target support is not the whole real line, the proposed value may lie outside the target support. While one could reject such values, rejection is not workable if all possible proposed values are outside the target support (i.e. when the proposal and target supports do not overlap). Reflection is another possibility but there are two problems. First, reflection would defeat the purpose of moving to the other side of the target. Second, with a small step size, the reverse move $x' \to x$ of the MirrorU move with reflection may not be possible, thus breaking the detailed balance condition. As an example, consider a target with support $[0, \infty)$. For MirrorU (6) with $\mu^* = 1.5$ and step size $\sigma = 1$, suppose the current value is $x = 5$. Then $x^* = 2\mu^* - x = -2$ and $x' \sim U(-3.73, -0.27)$. Suppose the proposed value is $x' = -0.2$, which is reflected to $x' = 0.2$. Now it is not possible to reach $x = 5$ from $x' = 0.2$ in the reverse direction because from $x' = 0.2$, we have $(x')^* = 2.8$ and the proposal is $x \sim U(1.07, 4.53)$.

Instead of rejection or reflection, we transform the target support X onto the real line $\mathbf{R}$ before applying the Mirror move. For instance, if $X = [a, \infty)$, we apply the Mirror move on the transformed variable $y := \log(x - a)$, with the proposal ratio $\frac{q(x|x')}{q(x'|x)} = \frac{x'-a}{x-a}$. For $X = [a, b]$, we use $y := \log \frac{x-a}{b-x}$, with $\frac{q(x|x')}{q(x'|x)} = \frac{(b-x')(x'-a)}{(b-x)(x-a)}$. With these log transformations, the original variable in the X space is multiplied by a random factor, and the Mirror proposal is referred to as *Mirror Multiplier*.

3.5 Results

Figures 4 and 5 show the performance of eight proposal kernels applied to five targets plotted against the proposal step size σ. We observe large variations in efficiency as σ changes, which emphasises the importance of choosing σ to achieve high efficiency. We also note that for the uniform and Gaussian kernels, optimal σ for convergence rate (δ_8 and $|\lambda|_2$) is larger than that for mixing, while the opposite is true for the bimodal kernels.

The Box, Airplane and StrawHat kernels have similar efficiency to the Bactrian-type kernels from Yang and Rodríguez (2013), with Box and StrawHat generally performing slightly better than the BactrianTriangle kernel (Table 1). In addition, all these bimodal kernels are better than the unimodal Gaussian and uniform kernels. The detail of the distributional form appears to be less important. Among the bimodal kernels, we prefer the StrawHat as it tends to achieve high efficiency and it is not very sensitive to the choice of step size.

For the MirrorU and MirrorN kernels, we fixed μ^* to 0.1 for all targets in this Section, except the gamma target, where we used $\mu^* = 1.5$. Using a fixed μ^* allowed us

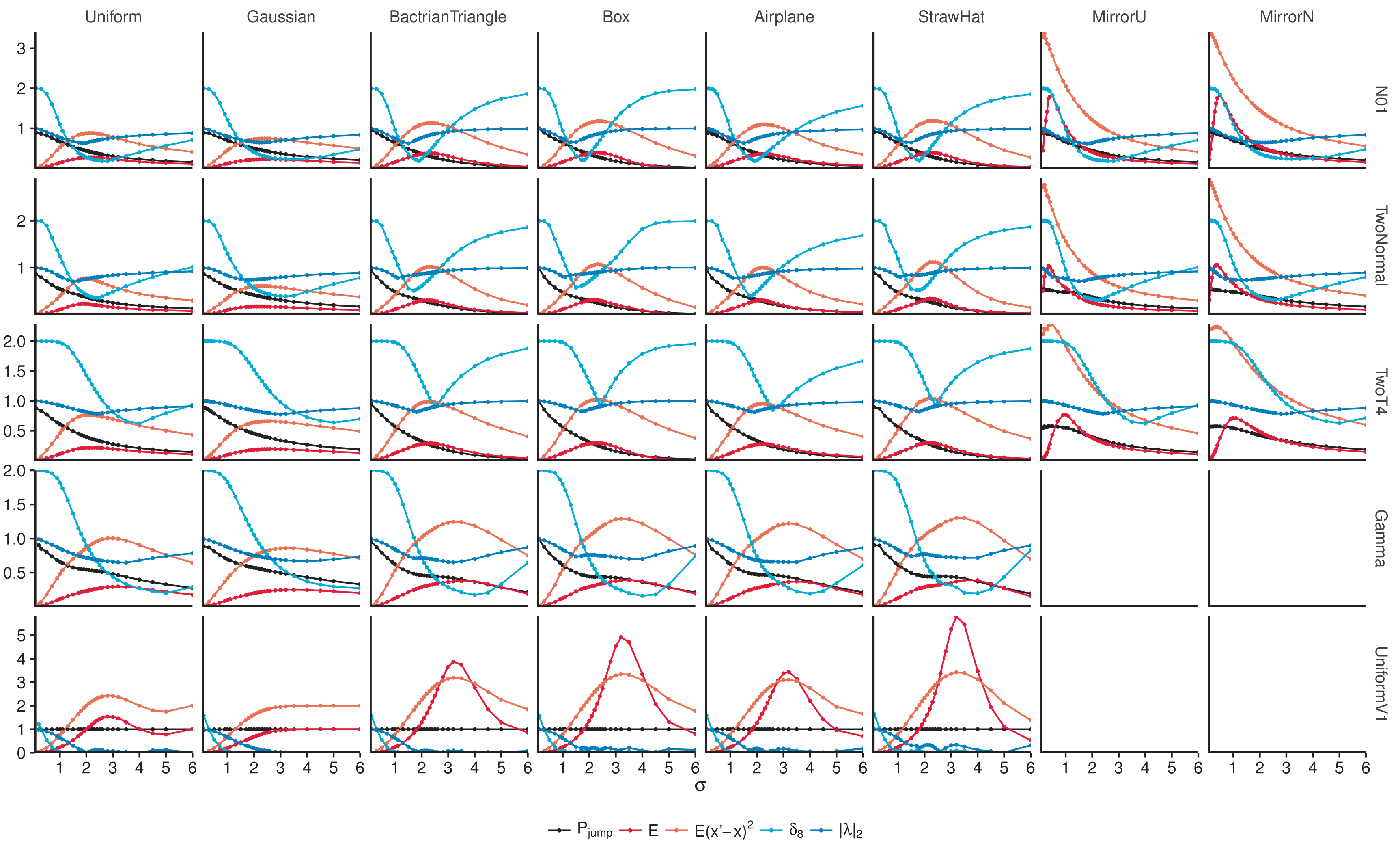


Figure 4: Efficiency (E) of eight proposal kernels for estimating the mean of five target distributions. Parameter: $a = 0.5$ for Box, $a = 1$ for Airplane and StrawHat, and $\mu^* = 0.1$ for MirrorU and MirrorN (the true mean for N01, TwoNormal and TwoT4 is 0, $\frac{1}{2}$ and $-\frac{3}{8}$, respectively). The results for MirrorU and MirrorN kernels for gamma and uniform targets, which require a variable transformation, are in Figure 5.

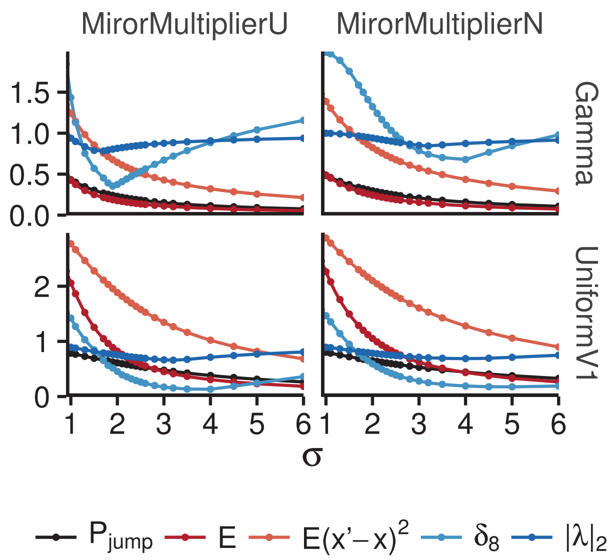


Figure 5: Efficiency (E) of the Mirror Multiplier kernels for estimating the mean of the gamma and uniform target distributions. For gamma target (mean 2), we use $\mu^* = 1.5$; i.e. we apply the Mirror kernel to $\log x$, with mean 0.563 and $\log \mu^* = 0.405$. For uniform target (mean 0), we use $\mu^* = 0.1$; i.e. we apply the Mirror kernel to $\log \frac{x-a}{b-x}$, with mean 0 and $\log \frac{\mu^*-a}{b-\mu^*} = 0.116$.

to optimise the step length σ and obtain smooth efficiency curves (Figure 4) without averaging over many simulation replicates. The two Mirror kernels generally achieve several-fold improvements in efficiency, and are "super-efficient", with $E > 1$, in most cases (Table 1). In practical applications, we suggest setting $\mu^* = \hat{\mu}$ and $\sigma = \hat{s}$ or $\sigma = \frac{1}{2}\hat{s}$, with both the target mean μ and standard deviation s estimated during the burn-in (Section 3.4). If the estimated mean is closer to the true mean than the fixed μ^* used in our experiments, performance will be better as well. For the $N(0,1)$ target, the efficiency, averaged over 10 replicates, is 1.290 for $\sigma = \hat{s}$, and 2.815 for $\sigma = \frac{1}{2}\hat{s}$ for MirrorN, compared with $E = 1.82$ when $\mu^* = 0.1$ is fixed and σ is optimised in Table 1.

We note that the ranking of the proposal kernels is largely the same across these five targets (Table 1), suggesting that this pattern may hold for fairly arbitrary targets. For the Box, Airplane and StrawHat kernels, the optimal P^*_{jump} is reasonably stable across targets evaluated, and we suggest using the automatic scale adjustment (5) for setting the proposal step size σ, with $P^*_{\text{jump}} = 0.3$.

Finally, to assess whether the efficiency ordering of the kernels depends on the specific function estimated, we consider estimating a tail probability of the normal target $N(0,1)$. For estimating the probability $\mathbf{P}(x > 2.3263) = 0.01$, the same ordering of the kernels holds as for estimation of the target mean, with comparable optimal σ (Figure 6a). The highest efficiency is $E \approx 0.4$, achieved by the two Mirror kernels. Similar results were obtained for estimating the probability $\mathbf{P}(x > 1.2815) = 0.1$ (Figure 6b), but with a generally narrower range of σ achieving the maximum efficiency. The MirrorU kernel was the most efficient, with $E \approx 0.5$.

Kernel	optimal σ	P_{jump}	E	E_{π}^{2}	ρ_1	δ_8	$\lvert\lambda\rvert_2$
			Target $N(0,1)$				
Uniform	2.2	0.405	0.276	0.879	0.560	0.230	0.671
Gaussian	2.5	0.426	0.228	0.744	0.628	0.286	0.657
BactrianTriangle ($m = 0.95$)	2.3	0.304	0.377	1.131	0.434	0.442	0.829
Box ($a = 0.5$)	2.3	0.290	0.394	1.150	0.410	0.608	0.857
Airplane ($a = 1$)	2.2	0.334	0.360	1.096	0.452	0.296	0.789
StrawHat ($a = 1$)	2.2	0.308	0.395	1.188	0.406	0.488	0.838
MirrorU ($\mu^* = 0.1$)	0.5	0.821	1.823	2.815	−0.408	1.828	0.865
MirrorN ($\mu^* = 0.1$)	0.5	0.828	1.824	2.884	−0.442	1.840	0.880
			Target $\frac{1}{4}N(-1,\frac{1}{4})+\frac{3}{4}N(1,\frac{1}{4})$				
Uniform	1.9	0.385	0.227	0.771	0.614	0.454	0.746
Gaussian	2.2	0.388	0.171	0.608	0.696	0.501	0.750
BactrianTriangle ($m = 0.95$)	2.2	0.271	0.303	1.010	0.495	0.705	0.880
Box ($a = 0.5$)	2.2	0.261	0.308	1.057	0.472	0.806	0.894
Airplane ($a = 1$)	2.2	0.283	0.304	1.004	0.498	0.603	0.863
StrawHat ($a = 1$)	2.2	0.269	0.339	1.114	0.443	0.693	0.878
MirrorU ($\mu^* = 0.1$)	0.35	0.525	1.045	2.503	−0.252	1.983	0.884
MirrorN ($\mu^* = 0.1$)	0.35	0.525	1.058	2.534	−0.267	1.980	0.893
			Target $\frac{3}{4}t_4(-\frac{3}{4},s^2)+\frac{1}{4}t_4(\frac{3}{4},s^2)$				
Uniform	2.2	0.366	0.218	0.760	0.620	1.276	0.794
Gaussian	2.6	0.377	0.192	0.659	0.670	1.157	0.791
BactrianTriangle ($m = 0.95$)	2.3	0.276	0.289	0.986	0.507	1.054	0.881
Box ($a = 0.5$)	2.3	0.254	0.296	1.025	0.488	1.014	0.894
Airplane ($a = 1$)	2.2	0.295	0.277	0.954	0.523	1.147	0.852
StrawHat ($a = 1$)	2.2	0.272	0.300	1.041	0.480	1.086	0.884
MirrorU ($\mu^* = 0.1$)	1.0	0.550	0.769	1.922	0.039	1.964	0.925
MirrorN ($\mu^* = 0.1$)	1.0	0.542	0.710	1.964	0.018	1.960	0.931
			Target $G(4,2)$				
Uniform	3.2	0.464	0.297	0.998	0.501	0.388	0.652
Gaussian	3.5	0.463	0.249	0.856	0.572	0.450	0.674
BactrianTriangle ($m = 0.95$)	3.5	0.403	0.378	1.241	0.379	0.213	0.665
Box ($a = 0.5$)	3.5	0.398	0.392	1.284	0.358	0.200	0.702
Airplane ($a = 1$)	3.5	0.412	0.371	1.209	0.395	0.224	0.654
StrawHat ($a = 1$)	3.5	0.414	0.388	1.302	0.349	0.206	0.717
			Target $U(-\sqrt{3},\sqrt{3})$				
Uniform	2.8	1	1.537	2.425	−0.212	0.000	0.216
Gaussian	∞	1	1.000	2.000	0.000	0.000	0.000
BactrianTriangle ($m = 0.95$)	3.2	1	3.875	3.190	−0.595	0.022	0.604
Box ($a = 0.5$)	3.2	1	4.916	3.346	−0.673	0.060	0.682
Airplane ($a = 1$)	3.2	1	3.439	3.107	−0.554	0.013	0.562
StrawHat ($a = 1$)	3.2	1	5.801	3.421	−0.710	0.091	0.719

Table 1: Efficiency and convergence rate of proposal kernels for estimating the mean of the five 1-D target distributions (all have variance 1). NOTE: Parameter μ^* for the Mirror kernels was chosen arbitrarily to be 0.1 and not optimised; see text.

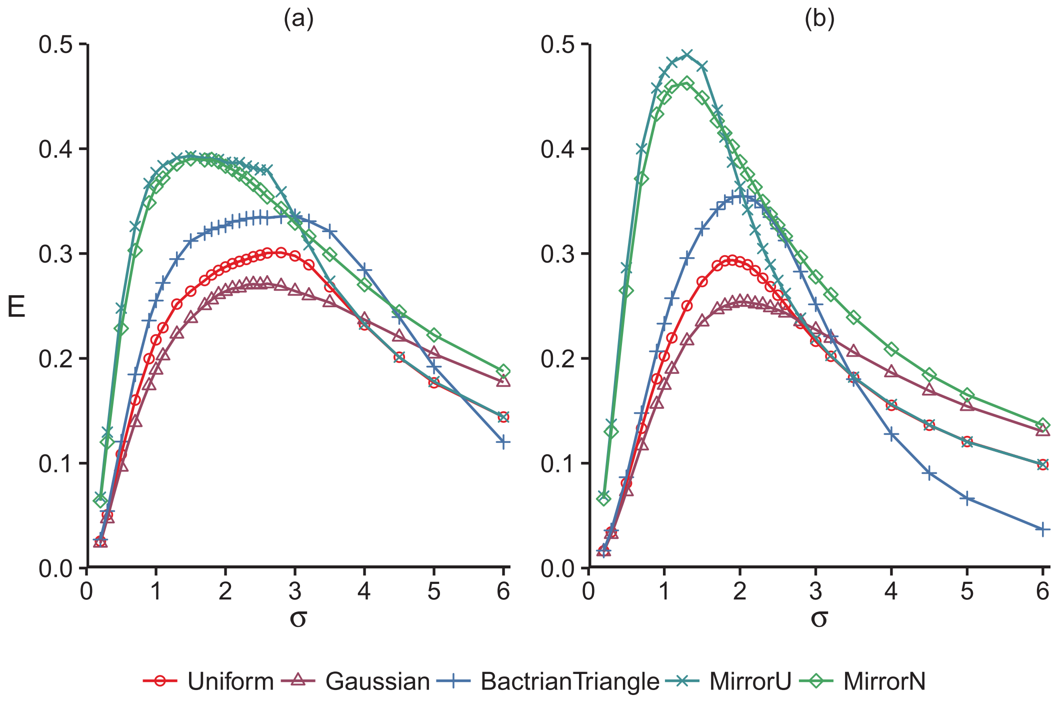


Figure 6: Efficiency (E) of five proposal kernels for estimating the tail probability of the normal distribution $N(0,1)$: $\mathbf{P}(x > 2.3263) = 0.01$ (a) and $\mathbf{P}(x > 1.2815) = 0.1$ (b). For MirrorU and MirrorN kernels, μ^* is fixed to 0.1.

4 Multidimensional target distributions

4.1 Multivariate Gaussian target using multidimensional uniform and Mirror kernels

We extend the one-dimensional uniform and MirrorN kernels to multiple dimensions for the $N_d(0, I)$ target, obtaining optimal scaling, optimal efficiency and P_{jump} (Table 2). For the uniform kernel, we consider the Cube and Sphere extensions in multi-dimensions. For MirrorN, we consider two variants, MirrorN1 with $x'|x \sim N(x^*, \hat{\Sigma})$ and MirrorN$\frac{1}{2}$ with $x'|x \sim N(x^*, \frac{1}{4}\hat{\Sigma})$, where $x^* = 2\mu^* - x$, with μ^* and $\hat{\Sigma}$ to be the estimated target mean and variance. Efficiency is calculated by averaging over 10 replicates.

We find that the Cube and Sphere kernels are more efficient than the Gaussian kernel for $d = 1, 2, 3, 4$, but both are very similar to the Gaussian when $d > 4$. The MirrorN1 and MirrorN$\frac{1}{2}$ kernels are several times more efficient than Gaussian, Cube and Sphere kernels for $d \leq 10$, with MirrorN$\frac{1}{2}$ being over twice more efficient than MirrorN1. Note that these MirrorN moves evaluated in Table 2 are d-dimensional moves. In comparison, the efficiency of one-dimensional MirrorU and MirrorN is higher than 100% whatever the dimension of the target is (Table S1).

In the supplementary material (Thawornwattana et al., 2017), we also perform detailed simulations for the two-dimensional case, comparing the two-dimensional extensions of the uniform kernel (the Square and Disc), with one-dimensional Gaussian, uniform and MirrorU kernels.

d	Gaussian kernel			Cube kernel			Sphere kernel			MirrorN1 kernel		MirrorN$\frac{1}{2}$ kernel	
	optimal σ	E	P_{jump}	optimal σ	E	P_{jump}	optimal σ	E	P_{jump}	E	P_{jump}	E	P_{jump}
1	2.40	0.233	0.441	2.20	0.276	0.416	2.20	0.276	0.416	1.290	0.706	2.815	0.846
2	1.70	0.136	0.352	1.64	0.155	0.317	1.62	0.157	0.315	0.869	0.552	2.118	0.756
3	1.39	0.098	0.316	1.36	0.107	0.284	1.36	0.109	0.279	0.643	0.453	1.727	0.694
4	1.25	0.076	0.279	1.20	0.081	0.266	1.18	0.083	0.262	0.501	0.382	1.418	0.635
5	1.10	0.062	0.275	1.07	0.065	0.259	1.07	0.067	0.252	0.375	0.314	1.259	0.601
6	1.00	0.053	0.266	0.98	0.055	0.252	0.98	0.056	0.245	0.297	0.266	1.100	0.567
7	0.93	0.047	0.261	0.91	0.047	0.249	0.91	0.048	0.242	0.251	0.233	0.970	0.533
8	0.87	0.041	0.255	0.85	0.041	0.245	0.85	0.042	0.240	0.190	0.191	0.861	0.503
9	0.80	0.037	0.261	0.80	0.037	0.245	0.80	0.037	0.237	0.160	0.170	0.749	0.469
10	0.74	0.034	0.267	0.76	0.033	0.243	0.76	0.034	0.236	0.129	0.145	0.683	0.444

Table 2: Optimal step size (σ) and asymptotic efficiency (E) for the Gaussian target $N_d(0, I)$ and five proposal kernels. The results for the Gaussian kernel are from Table 1 in Gelman et al. (1996). For MirrorN1 and MirrorN$\frac{1}{2}$, the proposal covariance is $\hat{\Sigma}$ and $\frac{1}{4}\hat{\Sigma}$, respectively, where $\hat{\Sigma}$ is the estimated target covariance from the burn-in. The results are averaged over 10 replications.

Kernel	Proposal σ	P_{jump}	E	ρ_1
1DTransfGaussian (true Σ)	Automatic	0.392	0.225	0.631
1DTransfGaussian (estimated Σ)	Automatic	0.401	0.228	0.624
1DTransfMirrorU1 (true Σ)	$\sigma = s$	0.674	1.072	−0.034
1DTransfMirrorU1 (estimated Σ)	$\sigma = \hat{s}$	0.675	1.031	−0.016
1DTransfMirrorU$\frac{1}{2}$ (true Σ)	$\sigma = \frac{1}{2}s$	0.830	2.433	−0.423
1DTransfMirrorU$\frac{1}{2}$ (estimated Σ)	$\sigma = \frac{1}{2}\hat{s}$	0.821	2.319	−0.397
HMC (Stan)	Automatic	0.894	0.00682	0.983

Table 3: Efficiency for estimating the mean of the first component of the target $N_{100}(0, \Sigma)$.

4.2 Hundred-dimensional Gaussian distribution

To demonstrate the scalability of our approach to high dimensions, we consider the $N_{100}(0, \Sigma)$ target where Σ^{-1} is generated from a Wishart distribution with identity scale matrix and 100 degrees of freedom. The target distribution used has many strong correlations, with 1627 out of 4950 pairs of variables having correlations with magnitude greater than 0.99.

We compare the one-dimensional Gaussian and MirrorU kernels. For MirrorU, the parameter μ^* is set to the target mean estimated during the burn-in, and the component-specific proposal step size σ is set to either $\hat{s}$ or $\frac{1}{2}\hat{s}$ where $\hat{s}$ is the estimated standard deviation of the component in the target. These two proposals are referred to as MirrorU1 and MirrorU$\frac{1}{2}$, respectively. We use the whitening transformation

$$y = \Sigma^{-1/2}x \tag{8}$$

to remove correlations among the components and rescale all the components to have variance 1. This transformation requires the target's covariance matrix Σ, which can be estimated during the burn-in.

We use 10^5 iterations of burn-in and 10^7 iterations of the main chain. If estimation of Σ is required, we initialise Σ with the identity matrix and update every 10^4 iterations (thus ten rounds of update in total). The final covariance matrix used by the sampler is based on the last 10^4 burn-in samples. For the Gaussian kernel, we use automatic tuning of proposal step size (5) with optimal $P_{\text{jump}} = 0.4$.

For this problem, the MirrorU1 and MirrorU$\frac{1}{2}$ kernels give about four-fold and ten-fold increase in efficiency compared with the Gaussian kernel (Table 3). Efficiency is similar whether the true or estimated variances are used, illustrating that the approach of estimating the variance is practical. Stan does not perform well and takes about 100 times longer than the Gaussian and MirrorU kernels.

4.3 Bayesian logistic regression

Next, we apply the MirrorU kernel to the Bayesian logistic regression analysis of the German credit dataset. The same dataset was used by Girolami and Calderhead (2011) to demonstrate several state-of-the-art MCMC algorithms, namely MALA, HMC and their Riemannian manifold versions. We also include for comparison the Stan algorithm (version 2.15.1) (Carpenter et al., 2017), which implements HMC with automatic tuning. Note that MALA and HMC require the first derivatives, while their manifold versions additionally require the Fisher information matrix as well as its derivatives. The target distribution is

$$\begin{aligned} p(\theta|x, y) &\propto p(\theta) \prod_{n=1}^{N} p(y_n|x_n, \theta) \\ &\propto \exp\left(-\frac{1}{2\alpha}\theta^\top\theta + \sum_{n=1}^{N} y_n(\theta^\top x_n) - \sum_{n=1}^{N} \log(1 + e^{\theta^\top x_n}) \right), \end{aligned}$$

where θ is a vector of an intercept term and 24 regression coefficients, x_n is a vector of 24 normalized predictors (with zero mean and unit variance), $y_n \in \{0, 1\}$ is an indicator for a good credit risk, and $N = 1000$. We give each component of θ an independent Gaussian prior $N(0, \alpha)$ with $\alpha = 100$, following Girolami and Calderhead (2011). Each chain is run for 10^7 iterations after 10^4 burn-in iterations. For MALA, MCMC and the manifold versions, we use the Matlab implementation of Girolami and Calderhead (2011) and run for 10^6 iterations.

From Table 4, the multidimensional MALA and HMC proposals are worse than the simple one-dimensional 1DUniform and are comparable to the 1DGaussian kernel. The manifold versions of MALA and HMC are much better than all those four, and Stan performs the best. The MirrorU$\frac{1}{2}$ kernel has comparable efficiency to manifold HMC and Stan, achieving super-efficiency ($E > 1$) for most of the 25 parameters, while taking less time. We note that the Mirror kernel requires estimation of the target mean and variance, but is otherwise very simple to implement, and does not require any fine-tuning. MALA and HMC require estimation of the target variance, and the manifold versions in addition need higher derivatives or Fisher information. In complex models where analytic expressions of the required derivatives are not available, automatic differentiation may be used to evaluate derivatives at machine precision, but with increasing running time. In addition, MALA, HMC and their manifold versions all have at least one parameter that requires tuning. Thus the Mirror kernel is simpler to implement.

However, manifold MALA, HMC and manifold HMC give consistent efficiency across dimensions, while for the Mirror kernel, some components can have much lower efficiency than the rest.

5 Application to phylogenetics

We apply the proposal kernels studied above to a Bayesian inference problem of estimating species divergence time and evolutionary rate using molecular sequence data from two species. The dataset is the 12S rRNA gene from the mitochondrial genome of human and orangutan from Horai et al. (1995), summarized as $x = 90$ differences out of $n = 948$ sites.

Parameter	Kernel							
	1DUniform	1DMirrorU$\frac{1}{2}$	1DGaussian	MALA	HMC	Manifold MALA	Manifold HMC	HMC (Stan)
1	0.255	0.678	0.143	0.075	0.144	0.710	1.099	1.402
2	0.260	1.245	0.166	0.088	0.149	0.732	1.120	1.532
3	0.253	1.153	0.088	0.049	0.153	0.752	1.093	1.346
4	0.244	1.572	0.144	0.074	0.151	0.743	1.098	1.459
5	0.239	1.071	0.082	0.045	0.153	0.762	1.092	1.325
6	0.260	1.196	0.175	0.082	0.149	0.746	1.090	1.549
7	0.259	1.634	0.149	0.080	0.156	0.739	1.090	1.505
8	0.253	1.236	0.195	0.118	0.150	0.745	1.094	1.634
9	0.250	1.320	0.147	0.082	0.155	0.758	1.071	1.566
10	0.280	1.270	0.076	0.038	0.155	0.737	1.099	1.303
11	0.269	1.086	0.114	0.059	0.154	0.731	1.089	1.454
12	0.241	1.802	0.193	0.125	0.154	0.756	1.102	1.596
13	0.252	1.266	0.141	0.073	0.151	0.743	1.098	1.458
14	0.266	1.625	0.184	0.105	0.151	0.738	1.088	1.608
15	0.261	1.244	0.115	0.059	0.153	0.739	1.084	1.420
16	0.245	0.161	0.184	0.051	0.131	0.712	1.143	1.433
17	0.260	1.548	0.181	0.109	0.153	0.741	1.092	1.566
18	0.253	1.314	0.155	0.062	0.149	0.717	1.103	1.534
19	0.263	0.473	0.070	0.034	0.147	0.700	1.105	1.289
20	0.266	0.601	0.070	0.036	0.149	0.713	1.092	1.292
21	0.273	1.293	0.039	0.020	0.158	0.736	1.103	1.102
22	0.269	1.535	0.038	0.020	0.153	0.735	1.102	1.105
23	0.250	1.134	0.119	0.065	0.150	0.737	1.109	1.351
24	0.266	1.455	0.054	0.026	0.153	0.725	1.087	1.155
25	0.273	1.006	0.054	0.026	0.152	0.723	1.087	1.149
Running time (s), C	936	936	937	n/a	n/a	n/a	n/a	2263
Running time (s), Matlab	n/a	n/a	1981	299	4649	7069	25039	n/a

Table 4: Efficiency for estimating the mean of the posterior distribution for the logistic regression problem. Running time (in seconds) is for 10^6 iterations for all kernels. The 1D Gaussian kernel is implemented in both C and Matlab, and indicates a 2-fold difference in running time between the two languages.

5.1 Model

The evolutionary process at each site is modelled as a continuous time Markov process on the four nucleotides (T, C, A, and G) with the transition rate matrix $Q = \{q_{ij}\}$, with $q_{ij} = \lambda$ for any $i \neq j$ (Jukes and Cantor, 1969). The substitution rate for each nucleotide is thus $r = 3\lambda$ per time unit, which is defined as one million years (Myrs). The transition probability matrix is $P_t = \{P_t(i,j)\}$, with

$$P_t(i,j) = \begin{cases} \frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}tr} & \text{if } i = j, \\ \frac{1}{4} - \frac{1}{4}e^{-\frac{4}{3}tr} & \text{if } i \neq j. \end{cases}$$

Given the data of x differences at n sites, the likelihood is

$$p(x|t,r) = \left(\frac{1}{16} + \frac{3}{16}e^{-\frac{8}{3}tr}\right)^{n-x}\left(\frac{1}{16} - \frac{1}{16}e^{-\frac{8}{3}tr}\right)^{x}.$$

This is a function of the genetic distance $\theta := 2tr$, but not of t and r individually. The maximum likelihood estimate of θ is $\hat{\theta} = \frac{3}{4}\log(\frac{3n}{3n-4x}) = 0.1015$, and the 95% likelihood interval is $(0.0817, 0.1245)$.

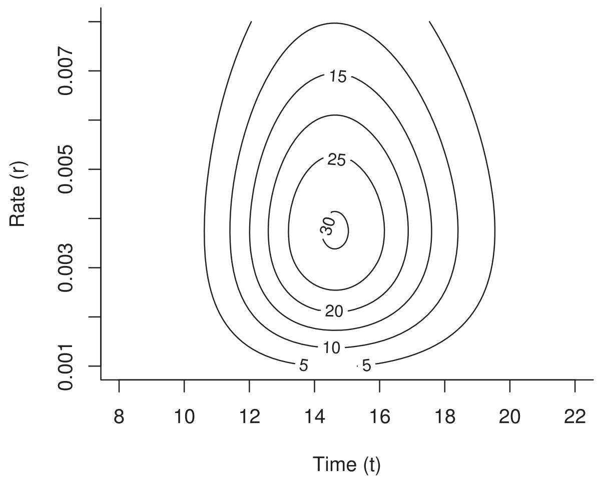
We assign gamma priors $t \sim G(40, 40/15)$, with mean 15 Myrs and 95% interval (10.7, 20.0), and $r \sim G(4, 800)$, with mean 0.005 substitutions per million years, and 95% interval (0.0014, 0.0110) (Figure 7a). The posterior distribution is

$$\begin{aligned} p(t,r|x) &\propto p(y|t,r)p(t)p(r) \\ &\propto \left(\frac{1}{16} + \frac{3}{16}e^{-\frac{8}{3}tr}\right)^{n-x}\left(\frac{1}{16} - \frac{1}{16}e^{-\frac{8}{3}tr}\right)^{x} t^{39}e^{-(40/15)t}r^{3}e^{-800r}. \end{aligned} \tag{9}$$

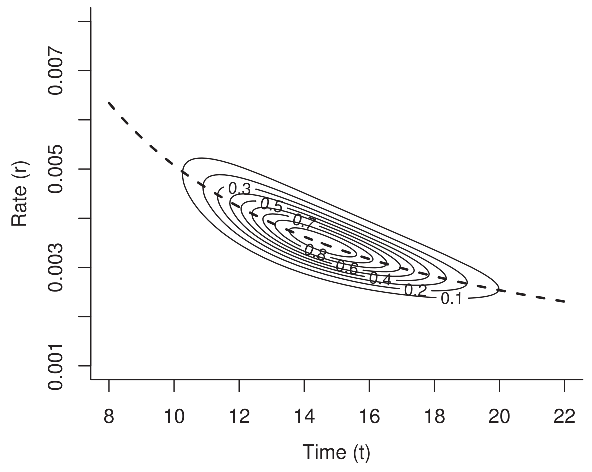
We sample from this posterior $p(t,r|x)$ (Figure 7b) using MCMC algorithms with different proposal schemes, and compare their efficiencies for estimating the posterior means of t and r.

5.2 MCMC algorithms for posterior inference

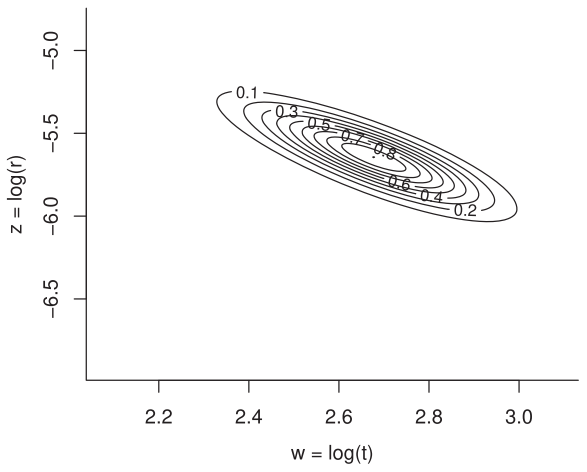
Since the uniform proposal is generally more efficient than the Gaussian proposal, we consider seven proposal kernels (A1-7) based on the uniform and MirrorU kernels and five state-of-the-art MCMC algorithms: MALA, HMC, HMC (Stan), manifold MALA, manifold HMC (A8-A12), which are based on a multivariate Gaussian proposal. The derivatives and Fisher information matrices required by algorithms A8-A12 are derived using the unnormalized posterior (9); these quantities are tractable but tedious (see supplementary material). We use variable transformations to deal with correlations and/or scale differences of the target variables (Figure 7b). Depending on the transformation used, each algorithm has component-specific scaling parameters. Specifically, σ_t and σ_r are standard deviations of proposals on t and r; σ_w and σ_z are for $w := \log t$ and $z := \log r$ (Figure 7c); σ_x and σ_y are for $x := \log(tr)$ and $y := \log(t/r)$ (Figure 7d). The details for tuning these step-size parameters are summarised in Table 5, and explicit steps for each algorithm are provided in the supplementary material.



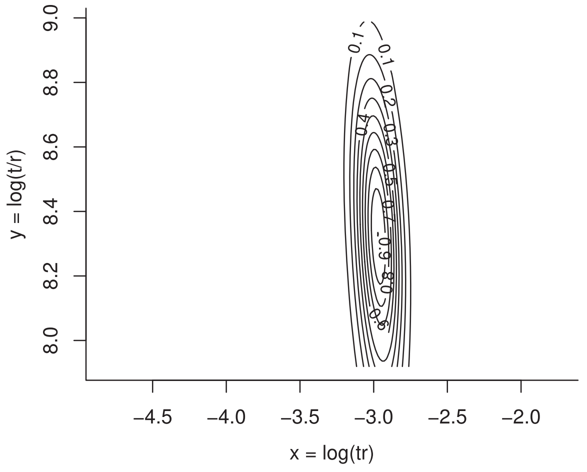
(a) Prior $p(t, r)$



(b) Posterior $p(t, r|x)$



(c) Transformed posterior: $w \coloneqq \log(t)$, $z \coloneqq \log(r)$



(d) Transformed posterior: $x \coloneqq \log(tr)$, $y \coloneqq \log(t/r)$

Figure 7: (a) Prior $p(t, r)$ and (b) posterior $p(t, r|x)$ distributions for the molecular clock dating problem. The dashed curve in the posterior indicates the values of (t, r) for which $2tr = \hat{\theta} = 0.1015$ (see text). (c) and (d) are different transformations of (b). All plots are based on the same ranges of values of t and r.

Algorithm A1 1D Uniform on t, r.

Algorithm A2 1D Uniform on w, z.

Algorithm A3 2D Uniform on w, z.

Algorithm A4 1D Uniform on w, z with whitening transformation (8).

Algorithm A5 1D Uniform on x, y.

Algorithm A6 1D MirrorU on x, y. **A6a** 1D MirrorU1 on x, y. **A6b** 1D MirrorU$\frac{1}{2}$ on x, y.

	Kernel	Proposal step size (σ)	Running time (s)	Time (t)			Rate (r)		
				c	P_{jump}	E	c	P_{jump}	E
A1	1D Uniform on t, r	Automatic	26	1.29	0.396	0.054	1.24	0.405	0.052
A2	1D Uniform on w, z	Automatic	28	1.46	0.403	0.055	1.33	0.388	0.054
A3	2D Uniform on w, z	$\sigma_w \leftarrow \hat{s}_w \times 2.2 \times \frac{1.7}{2.4}$, $\sigma_z \leftarrow \hat{s}_z \times 2.2 \times \frac{1.7}{2.4}$	22	1.76	0.206	0.079	1.55	0.206	0.078
A4	1D Uniform on w, z with whitening	Automatic	24	2.18	0.412	0.265	2.22	0.395	0.263
A5	1D Uniform on x, y	Automatic	26	2.15	0.401	0.284	2.16	0.399	0.211
A6a	1D MirrorU1 on x, y	$\sigma_x \leftarrow \hat{s}_x, \sigma_y \leftarrow \hat{s}_y$	38	1.07	0.621	0.970	0.96	0.646	0.621
A6b	1D MirrorU$\frac{1}{2}$ on x, y	$\sigma_x \leftarrow \frac{1}{2}\hat{s}_x, \sigma_y \leftarrow \frac{1}{2}\hat{s}_y$	38	0.48	0.762	1.168	0.46	0.766	0.411
A7	1D MirrorU$\frac{1}{2}$ on w, z with whitening	$\sigma_w \leftarrow \frac{1}{2}, \sigma_z \leftarrow \frac{1}{2}$	27	0.48	0.829	2.308	0.49	0.823	1.802
A8	MALA	Manual	30	1.48	0.617	0.600	1.47	0.617	0.591
A9	HMC	Manual	55	3.44	0.882	1.143	2.46	0.882	1.117
A10	HMC (Stan)	Automatic	1056	2.60	0.949	0.298	2.61	0.949	0.291
A11	Manifold MALA	Manual	162	n/a	0.631	0.571	n/a	0.631	0.568
A12	Manifold HMC	Manual	2727	n/a	0.939	1.715	n/a	0.939	1.670

Table 5: Efficiency of twelve kernels for the molecular clock dating problem. The scaling factor $c = \sigma/s$ is the ratio of the proposal standard deviation σ over the target standard deviation s. NOTE: s_w and s_z are the standard deviations of $w := \log t$ and $z := \log r$; s_x and s_y are the standard deviations of $x = \log(tr)$ and $y = \log(t/r)$, $\hat{\mu}$ denotes the estimate of the true mean μ, and $\hat{s}$ denotes the estimate of the true standard deviation s. The running time is an average from 10 replications. The scaling factors for A6a and A6b are not exactly 1 and 0.5 because the variances estimated during burn-in involve inaccuracies. For manifold MALA and manifold HMC, the scaling factor depends on the current position of the Markov chain.

Algorithm A7 1D MirrorU$\frac{1}{2}$ on w, z with whitening transformation (8).

Algorithm A8 MALA with preconditioning on w, z.

Algorithm A9 HMC on w, z.

Algorithm A10 HMC (Stan) on w, z.

Algorithm A11 Manifold MALA on w, z.

Algorithm A12 Manifold HMC on w, z.

Note that A4 and A7 use generic logarithm and whitening transformations to deal with correlations and scale differences, while A5 and A6 use certain features of the model (namely the fact that the likelihood depends on tr only) to design efficient transformations or search direction.

For each kernel, we simulate a Markov chain for 5×10^7 iterations, after a burn-in of 8×10^4 iterations. The estimates of the two marginal posterior means (and the 2.5th and 97.5th percentiles) are identical for all algorithms: 14.58 (10.5, 19.4) for t and 0.00361 (0.0025, 0.0051) for r, while the efficiency of the algorithms varies by nearly 40 folds (Table 5).

When the target's covariance structure is not taken into account, the efficiency achieved is less than 10%. The one-dimensional uniform proposals on t and r and on $\log t$ and $\log r$ (A1 and A2, respectively) are very inefficient, with $E \approx 5\%$, even less efficient than the two-dimensional uniform kernel (A3). This is not surprising as both pairs (t, r) and $(\log t, \log r)$ are highly correlated (correlation about -0.8), as is expected from the fact that the likelihood depends on the product tr only. Removing the correlation and adjusting for the scale differences between the target variables via the whitening transformation (8) (A4) improves the efficiency significantly. An alternative and computationally cheaper way to reduce the correlation is to use the transformation $x = \log(tr)$ and $y = \log(t/r)$ (A5), based on our knowledge of the model. This reduces the correlation to -0.28, and yielded a similar efficiency boost as A4.

The MirrorU kernels A6 and A7 show a superior performance to the uniform kernel using the same transformation (A4 and A5) with no extra computational cost. Independent simulations with different estimates of the target mean and variance suggest that efficiency is stable around the values given in Table 5 (supplementary material). Both MALA (A8) and manifold MALA (A11) perform better than the uniform kernel (A4) ($E \approx 60\%$), but do not beat the MirrorU kernel. HMC (A9) and manifold HMC (A12) also give super-efficient estimates, but at greater computational and implementation cost. Stan (A10) does not perform as well as other variants of HMC (A9, A12). In terms of efficiency per second, all variants of the MirrorU kernel outperform manifold MALA, manifold HMC and Stan by a substantial margin (Table 5). Finally, although well-tuned MALA and HMC also give good efficiency-per-time results, the need for high-order derivatives and manual tuning of the step-size parameters makes them challenging to implement.

6 Discussion

6.1 Measures of performance

We have compared the mixing efficiencies of different MH proposals, measured by the asymptotic variance for estimating a function defined on the target distribution (such as the mean or tail probability). As the efficiency of the kernel may depend on the function or target (Mira, 2001), we have included several targets in our evaluation. We note that the rankings of the proposal kernels are largely the same for all the targets we evaluated, suggesting the existence of some general principles that may apply to fairly arbitrary targets.

Besides the mixing efficiency, another useful measure is the rate of convergence of the Markov chain to the stationary distribution (such as δ_8 in Table 1). This rate should affect the desired length of the burn-in. We consider the convergence rate to be less important than the mixing efficiency because the burn-in is typically a small fraction of the MCMC run, and because a kernel efficient for mixing tends to also be good for convergence. For example, the uniform kernel converges faster and mixes more efficiently than the Gaussian kernel (Table 1). It is also cheaper to simulate than the Gaussian kernel. For the Mirror kernel, a small step size gives an estimate with a lower asymptotic variance, but causes slower convergence. It is thus preferable to use large steps during the burn-in for fast convergence, and small steps afterwards for fast mixing.

In practical MCMC applications, the computational and implementational costs are of major concern. We note that the computational cost may depend on hardware and software implementation details, as well as the specific inference problem. For example, certain one-dimensional moves may not change the likelihood and are thus computationally efficient, such as the change to $y = t/r$ when $x = tr$ is fixed in the molecular clocking dating example. Our analyses of the logistic regression and the molecular clock dating examples suggest that the Mirror moves are simpler to implement and run faster than the manifold MALA and HMC kernels. We leave it to the algorithm developer to assess the computational cost of different proposals in their specific applications.

6.2 Comparison with other MCMC algorithms

Several MCMC algorithms have been proposed to improve mixing by suppressing the diffusive behaviour of random walk proposals in which every iteration tends to take a small step in a random direction. We discuss a few that are related to our work.

The idea of proposing values on the other side of the distribution appeared in the literature before. For instance, the overrelaxation method (Adler, 1981; Barone and Frigessi, 1990) is a Gibbs sampler for Gaussian conditionals that makes a move to the other side of each component's full conditional. The update for the component i is

$$x'_i = \mu_{i|-i} + \alpha(x_i - \mu_{i|-i}) + \sqrt{\sigma^2_{i|-i}(1-\alpha^2)}z, \qquad z \sim N(0,1),$$

where $\mu_{i|-i}$ and $\sigma^2_{i|-i}$ are the conditional mean and variance of x_i given all other variables x_{-i}, and $\alpha \in (-1, 1)$ is a user-specified parameter. Choosing $\alpha \in (-1, 0)$ will make a

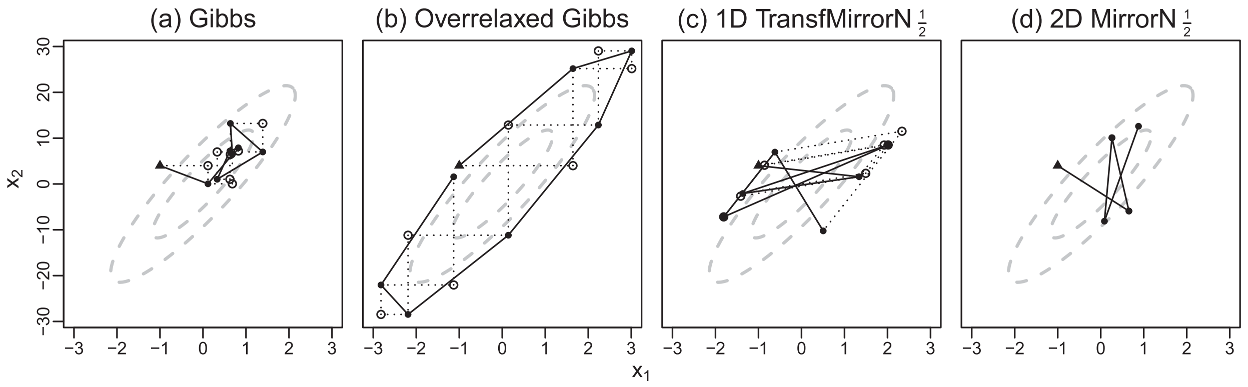


Figure 8: Sample path from a few steps of four algorithms for sampling from $N_2(0, \Sigma)$, with $\Sigma = \left(\begin{smallmatrix} 1 & 9 \\ 9 & 100 \end{smallmatrix}\right)$: (a) standard Gibbs sampler, (b) overrelaxed Gibbs sampler ($\alpha = -0.98$), (c) MH using 1D TransfMirrorN$\frac{1}{2}$ kernel, and (d) MH using 2D MirrorN$\frac{1}{2}$ kernel. The first three consist of a sequence of two one-dimensional moves, while the last one is a single two-dimensional move. The 1D TransfMirrorN$\frac{1}{2}$ kernel applies the MirrorN kernel $y'_i|y_i \sim N(2(\hat{\Sigma}^{-1/2}\mu^*)_i - y_i, \frac{1}{4})$, $i = 1, 2$, on $y = \hat{\Sigma}^{-1/2}x$, where $x = (x_1, x_2)$ is the target variable, and μ^* and $\hat{\Sigma}$ are estimated mean and covariance matrix of the target from the burn-in as described in Section 4.2. The 2D MirrorN$\frac{1}{2}$ kernel proposes $x'|x \sim N(2\mu^* - x, \frac{1}{4}\hat{\Sigma})$. Triangle = starting point $(-1, 4)$; filled circle = state of the Markov chain; empty circle = intermediate step (for the one-dimensional moves). Two ellipses enclose the 50% and 90% probability mass of the target.

move to the other side of the full conditional distribution of x_i. The Markov chain does not move to the other side of the target in one step, but instead moves along the density contour (Figure 8b), with higher-order autocorrelations oscillating between positive and negative signs (Figure 9). This results in cancellations of autocorrelations in (3), yielding a lower asymptotic variance than the standard Gibbs sampler in certain cases. In contrast, the Mirror is a general MH proposal kernel that moves to the other side of the target in one step, giving a negative first-order autocorrelation (Figure 9). In addition, its implementation does not require the knowledge of the full conditionals. The mirror reflection of the current state through a centre point as an MH proposal kernel to induce negative correlations has been suggested by Tierney (1994, Section 4.3.3), who referred to it as an antithetic variate method, but theoretical analysis and empirical comparisons have been lacking.

In the antithetic coupling method (Hammersley and Morton, 1956; Frigessi et al., 2000), two Markov chains are constructed with one to be the mirror reflection of the other. Combining the two chains yields a low-variance estimate. In contrast, the Mirror kernel introduces negative correlations within a chain rather than between chains.

HMC is another method that aims to propose a value away from the current position, in the direction of the peak of the target. A proposal is generated by simulating a trajectory of the so-called Hamiltonian dynamics. It requires computation of the first derivative of the log target density, and the tuning of its parameters is currently a topic of research (Wang et al., 2013; Hoffman and Gelman, 2014). MALA is an MH algorithm

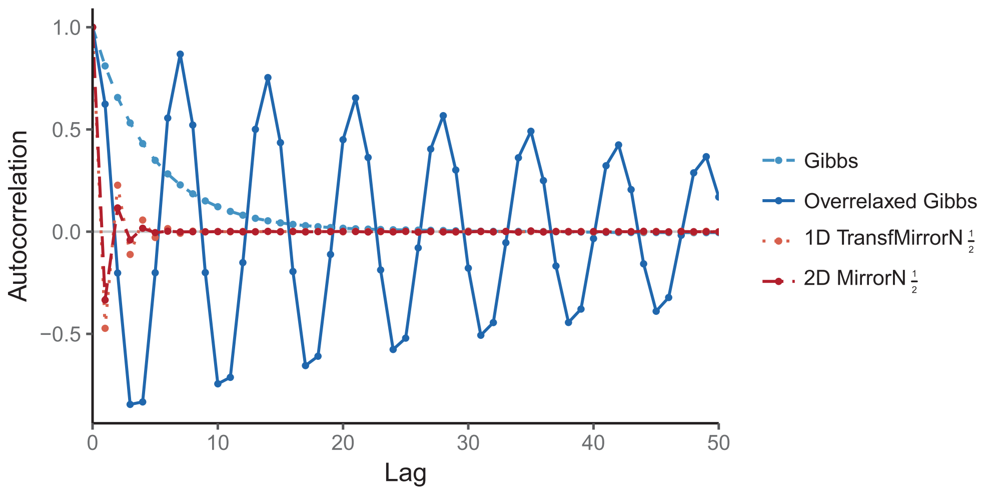


Figure 9: Autocorrelation function for the four proposal kernels of Figure 8, calculated using 10^6 iterations after a burn-in of 8,000 iterations. The efficiency for the four kernels is 0.104, 11.127, 2.784 and 2.122.

that uses the Langevin proposal (see Section 1) and can be viewed as a special case of HMC. For the $N(\mu, s^2)$ target, choosing step size $\sigma = 2s$ gives the MALA update $x'|x \sim N(2\mu - x, 4s^2)$, which is equivalent to the MirrorN kernel with a fixed step size of $2s$.

6.3 Parametrisation, variable transformation and efficiency for different functions

Parametrisation of the target distribution or variable transformation is a useful approach to designing efficient MCMC samplers. We have illustrated this with several transformations that deal with correlations and/or scales of the variables. We note that using different functions f in (2) to evaluate MCMC mixing efficiency for the same target π is equivalent to using different transformations or target densities but the same function (such as the mean). Given that the ranking of kernels does not appear to be sensitive to the target used or the function to be estimated, a useful approach may be to transform the variable into a density for which efficient proposal kernels are known, and design proposals for the original variables accordingly.

To find a good proposal $q(x'|x)$ for the target $\pi_X(x)$, we use a one-to-one transformation $y = T(x)$ so that the resulting density $\pi_Y(y)$ resembles a simple density for which an efficient proposal $q(y'|y)$ is known. The X- and Y-chains are then coupled, in the sense that if the initial states are the same with $y_0 = T(x_0)$ and if the same sequence of random numbers is used to run the two chains, then $y_n = T(x_n)$ for all $n \geq 1$. Estimating $\mathbf{E}_{\pi_X}(f(x))$ using the X-chain sample $(x_n)_{n=1}^N$ is then the same as estimating $\mathbf{E}_{\pi_Y}(f(T^{-1}(y)))$ using the Y-chain sample $(y_n)_{n=1}^N$. Thus finding an efficient proposal kernel for a given target is equivalent to finding a good variable transformation or parametrisation. It is then profitable to study the mixing efficiency for estimating various functions for simple targets such as the uniform. Viewed in this light, our early

Kernel	P_{jump}	E	E_π^2	ρ_1
Exp(1) target, Exp(1)-CDF transform				
Uniform ($\sigma_x = 2.5$)	0.408	0.161	0.589	0.705
TransfUniform ($\sigma_y = 2.8$)	1.000	1.298	2.283	−0.142
TransfBactrianTriangle ($m = 0.95, \sigma_y = 3.2$)	1.000	2.014	2.820	−0.410
TransfStrawHat ($a = 1, \sigma_y = 3.2$)	1.000	2.026	2.950	−0.474
Folded Gaussian target $N_+(0,1)$, Exp(1)-CDF transform				
Uniform ($\sigma_x = 2.3$)	0.392	0.213	0.259	0.643
TransfUniform ($\sigma_y = 2.8$)	0.839	1.075	0.755	−0.039
TransfBactrianTriangle ($m = 0.95, \sigma_y = 3.2$)	0.834	1.919	0.961	−0.322
TransfStrawHat ($a = 1, \sigma_y = 3.2$)	0.847	2.224	1.013	−0.394
$N(0,1)$ target, t_2-CDF transform				
Uniform ($\sigma_x = 2.2$)	0.405	0.275	0.879	0.561
TransfUniform ($\sigma_y = 2.8$)	0.832	0.959	1.961	0.020
TransfBactrianTriangle ($m = 0.95, \sigma_y = 3.2$)	0.836	1.592	2.471	−0.236
TransfStrawHat ($a = 1, \sigma_y = 3.2$)	0.846	1.680	2.548	−0.274
$N(0,1)$ target, logistic-CDF transform				
TransfUniform ($\sigma_y = 2.8$)	0.739	0.875	1.880	0.060
TransfBactrianTriangle ($m = 0.95, \sigma_y = 3.2$)	0.710	1.292	2.268	−0.134
TransfStrawHat ($a = 1, \sigma_y = 3.2$)	0.752	1.459	2.382	−0.191

Table 6: Efficiency for estimating the mean of three target distributions. The transformation $y = e^{-x}$ is used for Exp(1) and folded Gaussian, and the transformations $y \sim t_2$ and $y \sim$ logistic are used for $N(0,1)$.

observation that different proposal kernels with the same general shape have similar performances is equivalent to the observation that the ranking of proposals is insensitive to the target or function used.

Consider the target $x \sim \text{Exp}(1/\mu)$ with mean μ. Then $y = e^{-x/\mu} \sim U(0,1)$. From Table 1, the uniform kernel $y'|y \sim U(y - \frac{w}{2}, y + \frac{w}{2})$ with reflection at $w = 2.8$ achieves $E = 1.537$ for estimating $\mathbf{E}(y)$. Transformed onto the original variable x, the move is as follows. Set $y = e^{-x/\mu}$, sample $y'|y \sim U(y - \frac{w}{2}, y + \frac{w}{2})$ and reflect so that $y' \in (0,1)$. Then set $x' = -\mu \log y'$. The acceptance probability is

$$\alpha(x, x') = \min\left(1, \mathrm{e}^{(x'-x)/\mu} \times \frac{\pi(x')}{\pi(x)}\right), \tag{10}$$

which equals 1. This algorithm gives $E = 1.298$ for estimating $\mathbf{E}(x) = \mathbf{E}(-\mu \log y)$ (Table 6). This is good performance since w was optimized for estimating $\mathbf{E}(y)$ instead of $\mathbf{E}(x)$. Even higher efficiency is achieved by using bimodal kernels such as BactrianTriangle or the new StrawHat on y (Table 6).

Next, we use the same transformation $y = e^{-x/\mu}$ to sample from the folded Gaussian

$\pi(x) \propto \exp(-\frac{1}{2}x^2), x > 0$, to estimate $\mathbf{E}(x) = 0.7979$. The acceptance probability is given by (10) although this does not equal 1. The uniform kernel on y gives $E = 1.075$ (Table 6). This is good because Exp(1) has only a passing resemblance to the folded Gaussian. Again bimodal kernels such as BactrianTriangle and StrawHat give even higher efficiency (Table 6).

Lastly, we consider two generic transformations for targets with support on the real line. We sample from $x \sim N(0, 1)$ using uniform, BactrianTriangle or StrawHat kernel on $y = h((x - \hat{\mu})/\hat{s})$ where h is the CDF of the t_2 or logistic distribution, and $\hat{\mu}$ and $\hat{s}$ are estimates of the mean and standard deviation of the target from the burn-in. For both transformations, the uniform kernel gives E close to 1, for estimating $\mathbf{E}(x) = 0$, whereas BactrianTriangle and StrawHat kernels give $E > 1$ (Table 6).

Supplementary Material

Supplementary Material of "Designing Simple and Efficient Markov Chain Monte Carlo Proposal Kernels" (DOI: 10.1214/17-BA1084SUPP; .pdf). (I) Efficiency curves for Box, Airplane and StrawHat kernels for a range of a values. (II) Two-dimensional Gaussian target example. (III) MCMC algorithms for the phylogenetic problem. (IV) Effect of μ^* on efficiency.

References

Adler, S. L. (1981). "Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions." *Physical Review. D. Particles and Fields*, 23: 2901–2904. 1053

Barone, P. and Frigessi, A. (1990). "Improving stochastic relaxation for Gaussian random fields." *Probability in the Engineering and Informational Sciences*, 4(3): 369–389. 1053

Bédard, M., Douc, R., and Moulines, E. (2012). "Scaling analysis of multiple-try MCMC methods." *Stochastic Processes and Their Applications*, 122(3): 758–786. MR2891436. 1035

Bédard, M., Douc, R., and Moulines, E. (2014). "Scaling analysis of delayed rejection MCMC methods." *Methodology and Computing in Applied Probability*, 16(4): 811–838. MR3270597. 1035

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). "Optimal tuning of the hybrid Monte Carlo algorithm." *Bernoulli*, 19(5A): 1501–1534. 1035

Beskos, A., Roberts, G., and Stuart, A. (2009). "Optimal scalings for local Metropolis-Hastings chains on nonproduct targets in high dimensions." *The Annals of Applied Probability*, 19(3): 863–898. 1035

Browne, W. J. (2006). "MCMC algorithms for constrained variance matrices." *Computational Statistics & Data Analysis*, 50(7): 1655–1677. 1036

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). "Stan: a probabilistic programming language." *Journal of Statistical Software*, 76(1): 1–32. 1047

Devroye, L. (1986). *Non-uniform random variate generation*. Springer-Verlag, New York. 1036

Frigessi, A., Gåsemyr, J., and Rue, H. (2000). "Antithetic coupling of two Gibbs sampler chains." *Annals of Statistics*, 28(4): 1128–1149. MR1810922. 1054

Gelfand, A. E., Smith, A. F. M., and Lee, T.-M. (1992). "Bayesian analysis of constrained parameter and truncated data problems using Gibbs sampling." *Journal of the American Statistical Association*, 87(418): 523–532. 1036

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996). "Efficient Metropolis jumping rules." In *Bayesian statistics, 5*, 599–607. Oxford Univ. Press, New York. 1034, 1036, 1037, 1045

Geyer, C. J. (1992). "Practical Markov chain Monte Carlo." *Statistical Science*, 7(4): 473–483. 1037

Girolami, M. and Calderhead, B. (2011). "Riemann manifold Langevin and Hamiltonian Monte Carlo methods." *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 73(2): 123–214. With discussion and a reply by the authors. MR2814492. 1046, 1047

Hammersley, J. M. and Morton, K. W. (1956). "A new Monte Carlo technique: antithetic variates." *Mathematical Proceedings of the Cambridge Philosophical Society*, 52: 449–475. MR0080984. 1054

Hastings, W. K. (1970). "Monte Carlo sampling methods using Markov chains and their applications." *Biometrika*, 57(1): 97–109. 1033

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: 1593–1623. MR3214779. 1054

Horai, S., Hayasaka, K., Kondo, R., Tsugane, K., and Takahata, N. (1995). "Recent African origin of modern humans revealed by complete sequences of hominoid mitochondrial DNAs." *Proceedings of the National Academy of Sciences of the United States of America*, 92(2): 532–536. 1047

Jukes, T. H. and Cantor, C. R. (1969). "Evolution of protein molecules." In Munro, H. N. (ed.), *Mammalian Protein Metabolism*, volume 3, 21–132. Academic Press, New York. 1049

Kemeny, J. G. and Snell, J. L. (1960). *Finite Markov chains*. D. Van Nostrand Co., Inc. 1036

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). "Equation of state calculations by fast computing machines." *The Journal of Chemical Physics*, 21(6): 1087–1092. 1033

Mira, A. (2001). “Efficiency of finite state space Monte Carlo Markov chains.” *Statistics & Probability Letters* , 54(4): 405–411. MR1861386. 1053

Pasarica, C. and Gelman, A. (2010). “Adaptively scaling the Metropolis algorithm using expected squared jumped distance.” *Statist. Sinica*, 20(1): 343–364. 1037

Peskun, P. H. (1973). “Optimum Monte-Carlo sampling using Markov chains.” *Biometrika*, 60: 607–612. 1034, 1036

Pillai, N. S., Stuart, A. M., and Thiéry, A. H. (2012). “Optimal scaling and diffusion limits for the Langevin algorithm in high dimensions.” *The Annals of Applied Probability*, 22(6): 2320–2356. MR3024970. 1035

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). “Weak convergence and optimal scaling of random walk Metropolis algorithms.” *The Annals of Applied Probability*, 7(1): 110–120. MR1428751. 1035

Roberts, G. O. and Rosenthal, J. S. (1998). “Optimal scaling of discrete approximations to Langevin diffusions.” *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 60(1): 255–268. MR1625691. 1035

Sherlock, C. and Roberts, G. (2009). “Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets.” *Bernoulli*, 15(3): 774–798. MR2555199. 1037

Thawornwattana, Y., Dalquen, D., and Yang, Z. (2017). “Supplementary Material of the “Designing Simple and Efficient Markov Chain Monte Carlo Proposal Kernels”.” *Bayesian Analysis*. doi: `https://doi.org/10.1214/17-BA1084SUPP`. 1044

Tierney, L. (1994). “Markov chains for exploring posterior distributions.” *Annals of Statistics*, 22(4): 1701–1762. MR1329166. 1034, 1054

Tierney, L. (1998). “A note on Metropolis-Hastings kernels for general state spaces.” *The Annals of Applied Probability*, 8(1): 1–9. 1034

Wang, Z., Mohamed, S., and de Freitas, N. (2013). “Adaptive Hamiltonian and Riemann Manifold Monte Carlo.” In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 1462–1470. 1054

Yang, Z. and Rodríguez, C. E. (2013). “Searching for efficient Markov chain Monte Carlo proposal kernels.” *Proceedings of the National Academy of Sciences of the United States of America*, 110(48): 19307–19312. MR3153956. 1035, 1036, 1037, 1038, 1040

Acknowledgments

This study was supported by a grant from Biotechnological and Biological Sciences Research Council (BBSRC) to Z.Y., and in part by the Radcliffe Institute for Advanced Study at Harvard University. Y.T. was supported by the Royal Thai Government Scholarship.

Supplementary Material

I Efficiency curves for Box, Airplane and StrawHat kernels

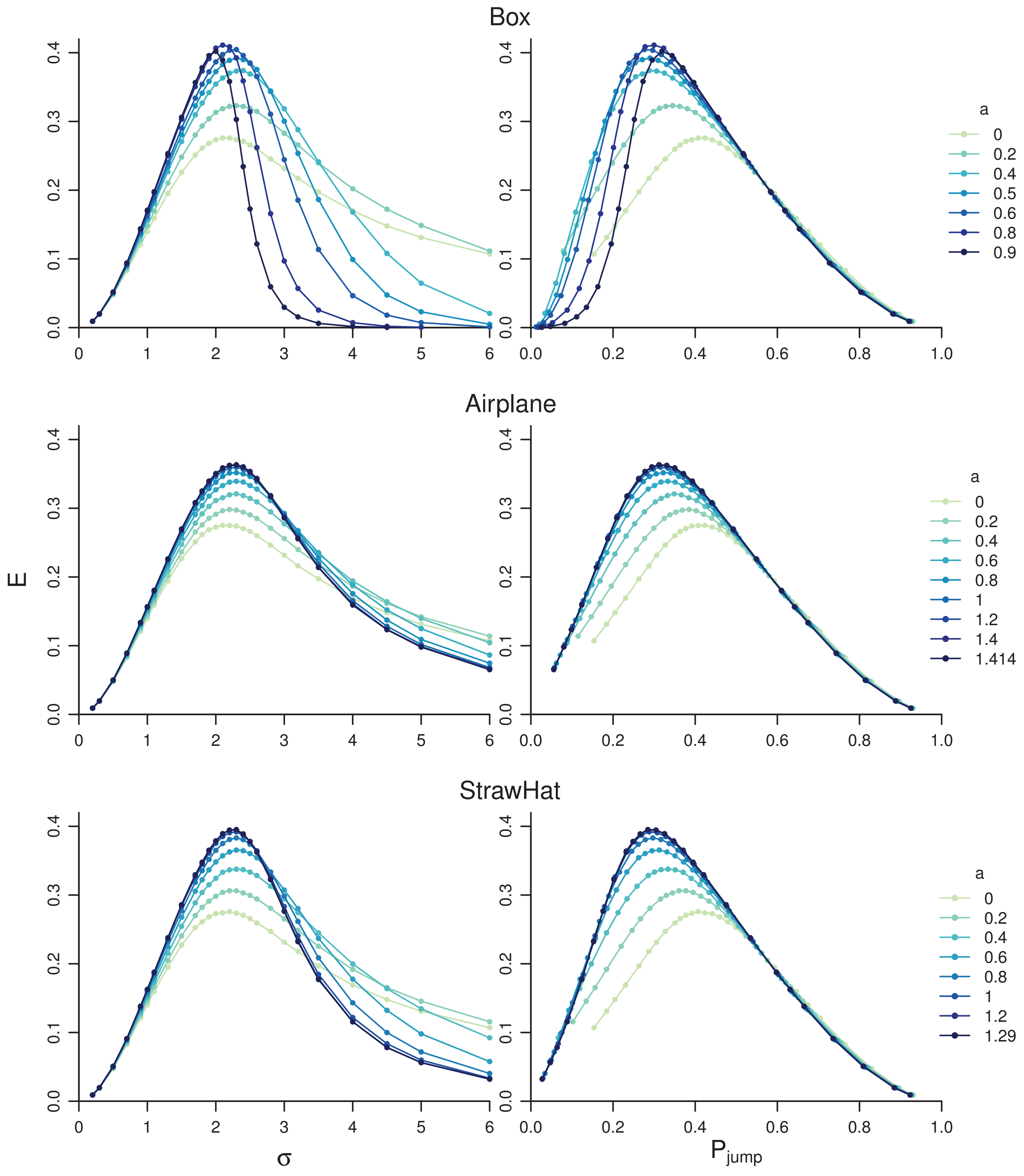


Figure S1: Effect of parameter a of the Box, Airplane and StrawHat kernels on efficiency for estimating the mean of $N(0,1)$.

II Two-dimensional Gaussian target distributions

We consider two bivariate Gaussian targets $N_2(0, I)$ and $N_2(0, \Sigma)$ with $\Sigma = \left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right)$ with $\rho = 0.9$. For $N_2(0, I)$, we compare proposals that are either a two-dimensional distribution, or a cycle of two one-dimensional distributions. For $N_2(0, \Sigma)$, we also use a variable transformation to deal with the correlation.

The proposal kernels considered are as follows. Let $x = (x_1, x_2)$ be the current value and $x' = (x_1', x_2')$ be the proposed value from $q(x_1', x_2'|x_1, x_2)$.

- Two-dimensional proposals on $\mathbf{R}^2$:
  - K1. Gaussian:
    $$q(x_1', x_2'|x_1, x_2) = N(x_1', x_2'|(x_1, x_2), \sigma^2 I_2).$$
    This is a symmetric kernel, with $q(x_1', x_2'|x_1, x_2) = q(x_1, x_2|x_1', x_2')$. Thus the proposal ratio is 1.
  - K2. Square:
    $$q(x_1', x_2'|x_1, x_2) = \frac{1}{12\sigma^2} 1_S(x_1, x_2)$$
    where $S \coloneqq \{(u, v) \in \mathbf{R}^2 : x_1 - \sqrt{3}\sigma \le u \le x_1 + \sqrt{3}\sigma, x_2 - \sqrt{3}\sigma \le v \le x_2 + \sqrt{3}\sigma\}$ is the square of side length $2\sqrt{3}\sigma$, centred at (x_1, x_2), and $1_S(x_1, x_2) :=$ 1 if $(x_1, x_2) \in S$, and 0 otherwise; this has mean (x_1, x_2) and covariance matrix I_2. To generate (x_1', x_2') from this kernel, draw $x_i' \sim U(x_i - \sqrt{3}\sigma, x_i + \sqrt{3}\sigma)$ independently for $i = 1, 2$. The proposal ratio is 1.
  - K3. Disc:
    $$q(x_1', x_2'|x_1, x_2) = \frac{1}{4\pi\sigma^2} 1_S(x_1, x_2)$$
    where $S \coloneqq \{(u, v) \in \mathbf{R}^2 : (x_1 - u)^2 + (x_2 - v)^2 \le 4\sigma^2\}$ is the disc of radius 2σ, centred at (x_1, x_2); this has mean (x_1, x_2) and covariance I_2. To generate (x_1', x_2') from this kernel, first draw $r \sim U(0, 1)$ and $\theta \sim U(0, 2\pi)$, then set $x_1' \coloneqq x_1 + 2\sigma\sqrt{r}\cos\theta$ and $x_2' \coloneqq x_2 + 2\sigma\sqrt{r}\sin\theta$. The proposal ratio is 1.

- Two one-dimensional proposals (one for each coordinate):
  - K4. Two 1D uniform proposals:
    1. First, draw $u \sim U(x_1 - \sqrt{3}\sigma, x_1 + \sqrt{3}\sigma)$ and set $(x_1', x_2) = (u, x_2)$ with probability $\min\left(1, \frac{\pi(u, x_2)}{\pi(x_1, x_2)}\right)$, otherwise set $(x_1', x_2) = (x_1, x_2)$. The proposal ratio is 1.
    2. Then, draw $v \sim U(x_2 - \sqrt{3}\sigma, x_2 + \sqrt{3}\sigma)$ and set $(x_1', x_2') = (x_1', v)$ with probability $\min\left(1, \frac{\pi(x_1', v)}{\pi(x_1', x_2)}\right)$, otherwise set $(x_1', x_2') = (x_1', x_2)$. The proposal ratio is 1.
  - K5. Two 1D Gaussian proposals. This is similar to the uniform one (K4), but with $u \sim N(x_1, \sigma^2)$ and $v \sim N(x_2, \sigma^2)$ instead.

For the $N_2(0, \Sigma)$ target, we consider four additional proposal kernels, based on the whitening transformation (8).

- K6. 2D Gaussian with transformation. Generate $y' \sim N(y, \sigma^2 I)$ and set $x' = \Sigma^{1/2} y'$. The proposal ratio is 1.
- K7. Two 1D Uniform with transformation. This is similar to K4, but uses the transformed variable y.
- K8. Two 1D Gaussian with transformation. This is similar to K7, but uses the Gaussian proposal instead of uniform.
- K9. Two 1D MirrorU with transformation. This is similar to K7, but uses the MirrorU proposal instead of uniform, with $\mu^* = 0.1$ for both components.

For the $N_2(0, I)$ target, the two two-dimensional versions of the uniform kernel, Square2D (K2) and Disc2D (K3), are more efficient than Gaussian2D (K1) (Table S1). This is apparently due to the fact that Gaussian2D is more concentrated on points close to the current point so that the samples are more strongly correlated. The efficiency is almost doubled when two one-dimensional proposals are used instead of a single two-dimensional move (compare Two1DUniform (K4) with Square2D and Disc2D, and Two1DGaussian (K5) with Gaussian2D in Table S1). The optimal σ and efficiency for these kernels agree with those for the $N(0, 1)$ target in Table 1. Note, however, that this improvement in statistical efficiency comes at an extra cost in computation that scales with the target's dimensionality. If the target is d-dimensional, a sequence of d 1D moves requires d evaluations of the target density and d MH acceptance steps instead of just one.

For the $N_2(0, \Sigma)$ target, applying kernels K1-K5 directly gives poor results, with efficiency of only $2-6\%$, compared with over 10% for the $N_2(0, I)$ target (Table S1 and Figure S2). This inefficiency is because these proposals fail to account for the high correlation ($\rho = 0.9$) between the variables in the target. When the correlation is removed via the whitening transformation (8) in TransfGaussian2D (K6), we recover the same efficiency of 0.134 as achieved by Gaussian2D (K1) on $N_2(0, I)$ target. The same pattern applies to the one-dimensional moves with transformation (8): Two1DTransfUnif (K7) achieves the same efficiency as Two1DUnif (K4) on $N_2(0, I)$ ($E = 0.276$), and Two1DTransfGaussian (K8) achieves the same efficiency as Two1DGaussian (K5) on $N_2(0, I)$ ($E = 0.228$). Note that simply using one-dimensional proposals without transformation can yield worse performance than the corresponding two-dimensional moves as correlations make it more difficult to make a large move along the axis-aligned directions. Finally, the Two1DTransfMirrorU (K9) kernel is several times more efficient than any other kernel considered.

III MCMC algorithms for the phylogenetic problem

	Kernel	$N_2(0, I)$					$N_2(0, \Sigma)$				
		optimal σ	P_{jump}	E	E_π^2	ρ_1	optimal σ	P_{jump}	E	E_π^2	ρ_1
K1	Gaussian2D	1.7	0.352	0.134	0.544	0.762	1.8	0.159	0.043	0.174	0.913
K2	Square2D	1.7	0.300	0.155	0.475	0.728	1.5	0.167	0.060	0.236	0.882
K3	Disc2D	1.7	0.294	0.156	0.552	0.724	1.5	0.153	0.048	0.192	0.904
K4	Two1DUniform	2.2	0.407	0.276	0.880	0.560	1.0	0.393	0.030	0.166	0.917
K5	Two1DGaussian	2.5	0.430	0.228	0.744	0.628	1.0	0.456	0.024	0.141	0.930
K6	TransfGaussian2D			n/a			1.7	0.352	0.134	0.475	0.762
K7	Two1DTransfUnif			n/a			2.2	0.407	0.276	0.880	0.560
K8	Two1DTransfGaussian			n/a			2.5	0.430	0.228	0.743	0.629
K9	Two1DTransfMirrorU ($\mu^* = 0.1$)			n/a			0.4	0.852	1.825	2.987	-0.494

Table S1:: Efficiency for estimating the mean of two-dimensional Gaussian targets.

NOTE: The $N_2(0, \Sigma)$ target has covariance $\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$. Efficiency ($E$) is for estimating the mean of the first component $\mu_1 = \mathbf{E}(x_1)$.

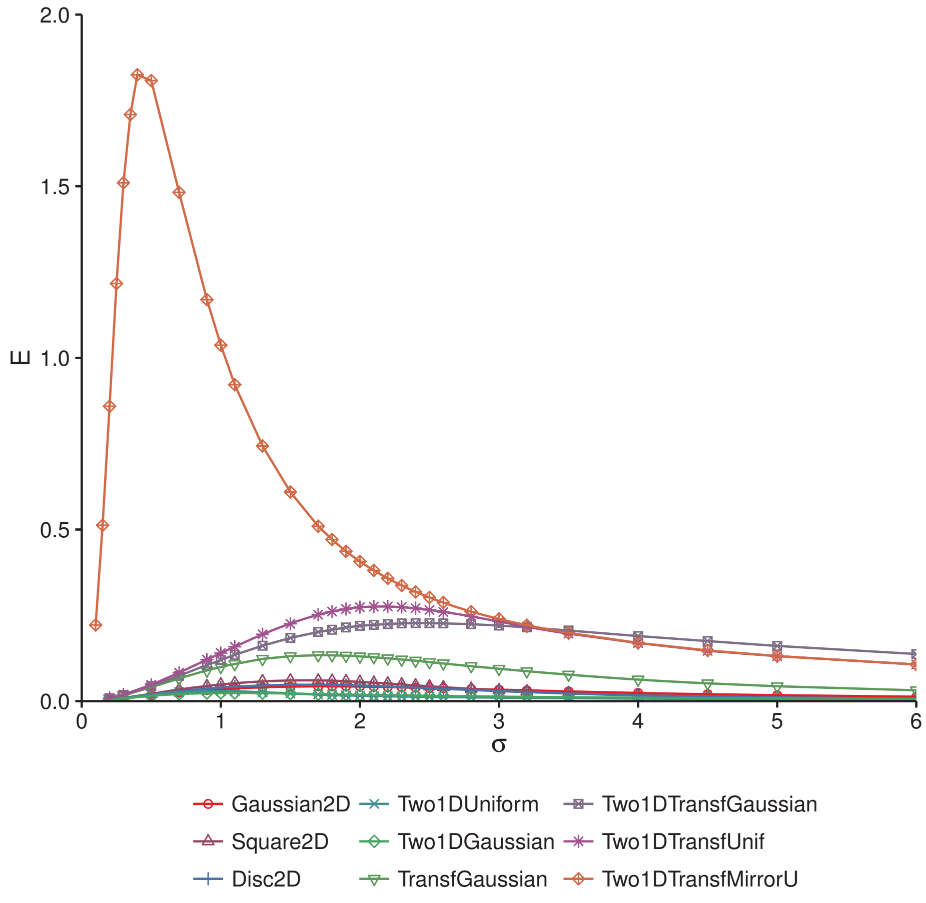


Figure S2: Efficiency of proposal kernels for the $N_2(0,\Sigma)$ target, with $\Sigma = \left(\begin{smallmatrix} 1 & 0.9 \\ 0.9 & 1 \end{smallmatrix}\right)$.

Algorithm A1 (1D Uniform on t, r). The algorithm consists of two MH steps. In step 1, draw $t'|t \sim U(t - \sigma_t\sqrt{3}, t + \sigma_t\sqrt{3})$. If $t' < 0$, set $t' \leftarrow -t'$ (proposal ratio is 1). In step 2, draw $r'|r \sim U(t - \sigma_r\sqrt{3}, t + \sigma_r\sqrt{3})$. If $r' < 0$, set $r' \leftarrow -r'$ (proposal ratio is 1).

The step-size parameters σ_t and σ_r are automatically tuned to achieve $P_{\text{jump}} = 0.4$ (see Section 2.3).

Algorithm A2 (1D Uniform on w, z). The algorithm consists of two MH steps. In step 1, draw $u \sim U(-\sqrt{3}, \sqrt{3})$ and set $t' \leftarrow te^{\sigma_w u}$ (proposal ratio is $\frac{t'}{t}$). In step 2, draw $v \sim U(-\sqrt{3}, \sqrt{3})$ and set $r' \leftarrow re^{\sigma_z v}$ (proposal ratio is $\frac{r'}{r}$).

The step-size parameters σ_w and σ_z are tuned to achieve $P_{\text{jump}} = 0.4$.

Algorithm A3 (2D Uniform on w, z). The algorithm uses a single two-dimensional proposal. First, draw $u \sim U(-\sqrt{3}, \sqrt{3})$ and set $t' \leftarrow te^{\sigma_w u}$. Then draw $v \sim U(-\sqrt{3}, \sqrt{3})$ and set $r' \leftarrow re^{\sigma_z v}$. The proposal ratio is $\frac{t'r'}{tr}$.

It is hard to adjust two step-sizes σ_w and σ_z in one proposal. We use $\sigma_w = s_w \times 2.2 \times \frac{1.7}{2.4}$ and $\sigma_z = s_z \times 2.2 \times \frac{1.7}{2.4}$, where s_w and s_z are the standard deviations of $w = \log t$ and $z = \log r$, estimated during burn-in. Here, 2.4 and 1.7 are optimal scales in 1D and 2D for the Gaussian kernel, while 2.2 is the optimal scale for the uniform kernel in 1D (tables 1 and S1).

Algorithm A4 (1D Uniform on w, z with whitening transformation (8)). Let $w := \log t, z := \log r$. Let $\widehat{\Sigma}$ denote the estimated covariance matrix of (w, z) during the burn-in. The algorithm consists of two one-dimensional MH steps on w and z.

1. Set $\binom{w}{z} \leftarrow \binom{\log t}{\log r}$ and $\binom{\tilde{w}}{\tilde{z}} \leftarrow \widehat{\Sigma}^{-1/2}\binom{w}{z}$. Draw $u \sim U(-\sqrt{3}, \sqrt{3})$ and set $\tilde{w}' \leftarrow \tilde{w} + \sigma_w u$ and $\tilde{z}' \leftarrow \tilde{z}$. Then set $\binom{w'}{z'} \leftarrow \widehat{\Sigma}^{1/2}\binom{\tilde{w}'}{\tilde{z}'}$ and $\binom{t'}{r'} \leftarrow \binom{e^{w'}}{e^{z'}}$. The proposal ratio is $\frac{t'r'}{tr}$.
2. Set $\binom{w}{z} \leftarrow \binom{\log t}{\log r}$ and $\binom{\tilde{w}}{\tilde{z}} \leftarrow \widehat{\Sigma}^{-1/2}\binom{w}{z}$. Draw $v \sim U(-\sqrt{3}, \sqrt{3})$ and set $\tilde{w}' \leftarrow \tilde{w}$ and $\tilde{z}' \leftarrow \tilde{z} + \sigma_z v$. Then set $\binom{w'}{z'} \leftarrow \widehat{\Sigma}^{1/2}\binom{\tilde{w}'}{\tilde{z}'}$ and $\binom{t'}{r'} \leftarrow \binom{e^{w'}}{e^{z'}}$. The proposal ratio is $\frac{t'r'}{tr}$.

The step-size parameters σ_w and σ_z are tuned to achieve $P_{\text{jump}} = 0.4$.

Algorithm A5 (1D Uniform on $x := \log(tr), y := \log(t/r)$). The algorithm consists of two one-dimensional MH steps on x and y.

1. Set $x \leftarrow \log(tr)$ and $y \leftarrow \log(t/r)$. Draw $u \sim U(-\sqrt{3}, \sqrt{3})$ and set $x' \leftarrow x + \sigma_x u$ and $y' \leftarrow y$. Then set $t' \leftarrow e^{\frac{x'+y'}{2}}$ and $r' \leftarrow e^{\frac{x'-y'}{2}}$. The proposal ratio is $\frac{t'r'}{tr}$.

2. Set $x \leftarrow \log(tr)$ and $y \leftarrow \log(t/r)$. Draw $v \sim U(-\sqrt{3}, \sqrt{3})$ and set $x' \leftarrow x$ and $y' \leftarrow y + \sigma_y v$. Then set $t' \leftarrow e^{\frac{x'+y'}{2}}$ and $r' \leftarrow e^{\frac{x'-y'}{2}}$. The proposal ratio is $\frac{t'r'}{tr} = 1$.

The step-size parameters σ_x and σ_y are tuned to achieve $P_{\text{jump}} = 0.4$.

Algorithm A6 (1D MirrorU on x, y). The algorithm consists of two one-dimensional MH steps on x and y.

1. Set $x \leftarrow \log(tr)$ and $y \leftarrow \log(t/r)$. Draw $x'|x \sim U(2\mu_x^* - x - \sigma_x\sqrt{3}, 2\mu_x^* - x + \sigma_x\sqrt{3})$ and set $y' \leftarrow y$. Then set $t' \leftarrow e^{\frac{x'+y'}{2}}$ and $r' \leftarrow e^{\frac{x'-y'}{2}}$. The proposal ratio is $\frac{t'r'}{tr}$.
2. Set $x \leftarrow \log(tr)$ and $y \leftarrow \log(t/r)$. Draw $y'|y \sim U(2\mu_y^* - y - \sigma_y\sqrt{3}, 2\mu_y^* - y + \sigma_y\sqrt{3})$ and set $x' \leftarrow x$. Then set $t' \leftarrow e^{\frac{x'+y'}{2}}$ and $r' \leftarrow e^{\frac{x'-y'}{2}}$. The proposal ratio is $\frac{t'r'}{tr} = 1$.

Here, μ_x^*, μ_y^* are set to the estimated means $\hat{\mu}_x, \hat{\mu}_y$ of x and y, respectively, and σ_x, σ_y are set to either $\hat{s}_x, \hat{s}_y$ (A6a) or $\frac{1}{2}\hat{s}_x, \frac{1}{2}\hat{s}_y$ (A6b), where $\hat{s}_x$ and $\hat{s}_y$ are the estimated standard deviations of x and y from the burn-in sample.

Algorithm A7 (1D MirrorU on w, z with whitening transformation). Let $\widehat{\Sigma}$ denote the estimated covariance matrix of (w, z) during burn-in. The algorithm consists of two MH steps.

1. Set $\binom{w}{z} \leftarrow \binom{\log t}{\log r}$ and $\binom{\tilde{w}}{\tilde{z}} \leftarrow \widehat{\Sigma}^{-1/2}\left(\binom{w}{z} - \binom{\hat{\mu}_w}{\hat{\mu}_z}\right)$. Draw $u \sim U(-\sqrt{3}, \sqrt{3})$ and set $\tilde{w}' \leftarrow -\tilde{w} + \sigma_w u$ and $\tilde{z}' \leftarrow \tilde{z}$. Then set $\binom{w'}{z'} \leftarrow \widehat{\Sigma}^{1/2}\binom{\tilde{w}'}{\tilde{z}'} + \binom{\hat{\mu}_w}{\hat{\mu}_z}$ and $\binom{t}{r} \leftarrow \binom{e^w}{e^z}$. The proposal ratio is $\frac{t'r'}{tr}$.
2. Set $\binom{w}{z} \leftarrow \binom{\log t}{\log r}$ and $\binom{\tilde{w}}{\tilde{z}} \leftarrow \widehat{\Sigma}^{-1/2}\left(\binom{w}{z} - \binom{\hat{\mu}_w}{\hat{\mu}_z}\right)$. Draw $v \sim U(-\sqrt{3}, \sqrt{3})$ and set $\tilde{w}' \leftarrow \tilde{w}$ and $\tilde{z}' \leftarrow -\tilde{z} + \sigma_z v$. Then set $\binom{w'}{z'} \leftarrow \widehat{\Sigma}^{1/2}\binom{\tilde{w}'}{\tilde{z}'} + \binom{\hat{\mu}_w}{\hat{\mu}_z}$ and $\binom{t}{r} \leftarrow \binom{e^w}{e^z}$. The proposal ratio is $\frac{t'r'}{tr}$.

The step-size parameters σ_w and σ_z are set to $\frac{1}{2}$.

Algorithm A8 (MALA with preconditioning on w, z). The algorithm uses a single two-dimensional proposal.

1. Set $(w, z) \leftarrow (\log t, \log r)$. Draw $(w', z') \sim N(m(w, z), \varepsilon^2 A)$, where

$$m(w, z) := (w, z) + \frac{\varepsilon}{2} A \nabla \log p(w, z|x),$$

and the first derivatives are derived using (9). Then set $(t', r') \leftarrow (e^{w'}, e^{z'})$. The proposal ratio is $\frac{N((w,z)|(w',z') + \frac{\varepsilon^2}{2} A \nabla \log p(w',z'|x), \varepsilon^2 A)}{N((w',z')|(w,z) + \frac{\varepsilon^2}{2} A \nabla \log p(w,z|x), \varepsilon^2 A)} \frac{t'r'}{tr}$.

The scalar step-size parameter ε is tuned manually to achieve the highest efficiency. The matrix A is set to $\frac{1}{(\det \widehat{\Sigma})^{1/2}}\widehat{\Sigma}$, where $\widehat{\Sigma}$ is denotes the estimate of the target's covariance matrix from burn-in samples, following Marshall and Roberts (2012).

Algorithm A9 (HMC on w, z). Let $L_{\max}$ be the upper bound on the number of leapfrog steps and let ε be the leapfrog step-size. Let $\widehat{\Sigma}$ denote the estimated covariance matrix of (w, z) from burn-in. This algorithm uses a single two-dimensional proposal.

1. Set $(w, z) \leftarrow (\log t, \log r)$. Draw an auxiliary variable $\phi \sim N(0, \widehat{\Sigma})$. Set $(w', z') \leftarrow (w, z)$ and $\phi' \leftarrow \phi$. Draw $L \sim U\{1, \ldots, L_{\max}\}$. For $\ell = 1, \ldots, L$, (a) set $\phi' \leftarrow \phi' + \frac{\varepsilon}{2}\nabla \log p(w', z')$, (b) set $(w', z') \leftarrow (w', z') + \varepsilon\widehat{\Sigma}^{-1}\phi'$, and (c) set $\phi' \leftarrow \phi' + \frac{\varepsilon}{2}\nabla \log p(w', z')$. Then set $(t', r') \leftarrow (e^{w'}, e^{z'})$. The proposal ratio is $\frac{N(\phi'|0,\widehat{\Sigma})}{N(\phi|0,\widehat{\Sigma})}\frac{t'r'}{tr}$.

The parameters $L_{\max}$ and ε are tuned manually to achieve the highest efficiency.

Algorithm A10 (HMC (Stan) on w, z). NUTS algorithm. See Hoffman and Gelman (2014) for description.

Algorithm A11 (Manifold MALA on w, z). This algorithm uses a single two-dimensional proposal.

1. Set $(w, z) \leftarrow (\log t, \log r)$. Draw $(w', z') \sim N(m(w, z), \varepsilon^2 G^{-1}(w, z))$ where

$$m(w, z) := (w, z) + \varepsilon^2 \left(\frac{1}{2} G^{-1}(w, z) \nabla \log p(w, z|x) + \Omega(w, z) \right),$$

$$G(w, z) := -\mathbf{E}_{p(x|w,z)} \nabla^2_{(w,z)} \log p(x|w, z) - \nabla^2_{(w,z)} \log p(w, z)$$

(the Fisher information matrix of the likelihood plus the negative Hessian of the log prior density), and

$$\Omega(w, z) := \frac{1}{2} G^{-1} \begin{pmatrix} \operatorname{tr}(G^{-1}\partial_w G) \\ \operatorname{tr}(G^{-1}\partial_z G) \end{pmatrix} - \sum_{j=w,z} (G^{-1}\partial_j G) G^{-1}_{\cdot,j}.$$

Set $(t', r') \leftarrow (e^{w'}, e^{z'})$. The proposal ratio is $\frac{N((w,z)|m(w',z'),\varepsilon^2 G^{-1}(w',z'))}{N((w',z')|m(w,z),\varepsilon^2 G^{-1}(w,z))}\frac{t'r'}{tr}$.

The step-size parameter ε is tuned manually to achieve the highest efficiency.

Algorithm A12 (Manifold HMC on w, z). Let $L_{\max}$ be the upper bound on the number of leapfrog steps and let ε be the leapfrog step-size. Let M be the number of fixed point iterations for the generalized leapfrog integrator from Girolami and Calderhead (2011). This algorithm uses a single two-dimensional proposal.

1. Set $(w, z) \leftarrow (\log t, \log r)$. Draw an auxiliary variable $\phi \sim N(0, G)$. Set $(w', z') \leftarrow (w, z)$ and $\phi' \leftarrow \phi$. Draw $L \sim U\{1, \ldots, L_{\max}\}$. For $\ell = 1, \ldots, L$,
   (a) Set $\tilde{\phi} \leftarrow \phi'$. For $m = 1, \ldots, M$, set
   $$\tilde{\phi} \leftarrow \phi' + \frac{\varepsilon}{2}\left(\nabla \log p(w', z'|x) - \frac{1}{2}\text{tr}(G^{-1}\nabla G) + \frac{1}{2}\tilde{\phi}^\top G^{-1}(\nabla G)G^{-1}\tilde{\phi}\right).$$
   Then set $\phi' \leftarrow \tilde{\phi}$.
   (b) Set $(\tilde{w}, \tilde{z}) \leftarrow (w', z')$. For $m = 1, \ldots, M$, set
   $$(\tilde{w}, \tilde{z}) \leftarrow (w', z') + \frac{\varepsilon}{2}\left(G^{-1}(w', z') + G^{-1}(\tilde{w}, \tilde{z})\right)\phi'.$$
   Then set $(w', z') \leftarrow (\tilde{w}, \tilde{z})$.
   (c) Set
   $$\phi' \leftarrow \phi' + \frac{\varepsilon}{2}\left(\nabla \log p(w', z'|x) - \frac{1}{2}\text{tr}(G^{-1}\nabla G) + \frac{1}{2}\phi'^\top G^{-1}(\nabla G)G^{-1}\phi'\right).$$

   Then set $(t', r') \leftarrow (e^{w'}, e^{z'})$. The proposal ratio is $\frac{N(\phi'|0, G(w',z'))}{N(\phi|0, G(w,z))}\frac{t'r'}{tr}$.

The parameters $L_{\max}$ and ε are tuned manually to achieve the highest efficiency, and M is fixed to 3.

Note that the parameters of the model are t and r, as are the state of the Markov chain. Transformed variables w and z or x and y are used to design efficient moves in the t-r space.

IV Effect of μ^* on efficiency

We used the burn-in to estimate the means and variances of the posterior distribution. To assess the impact of this estimation on the efficiency of the chain, we performed 100 independent runs of algorithm A6b for the phylogenetic example. The means (μ_x, μ_y) and standard deviations (s_x, s_y) are estimated using four rounds during the burn-in, with each round consisting of 20,000 iterations. The estimates are then used to construct the mirror move. From Figure S3, we see that the mean and variance estimates from the burn-in are reasonably accurate, with mean efficiency 1.165 for t and 0.497 for r, slightly higher than values shown in Table 5.

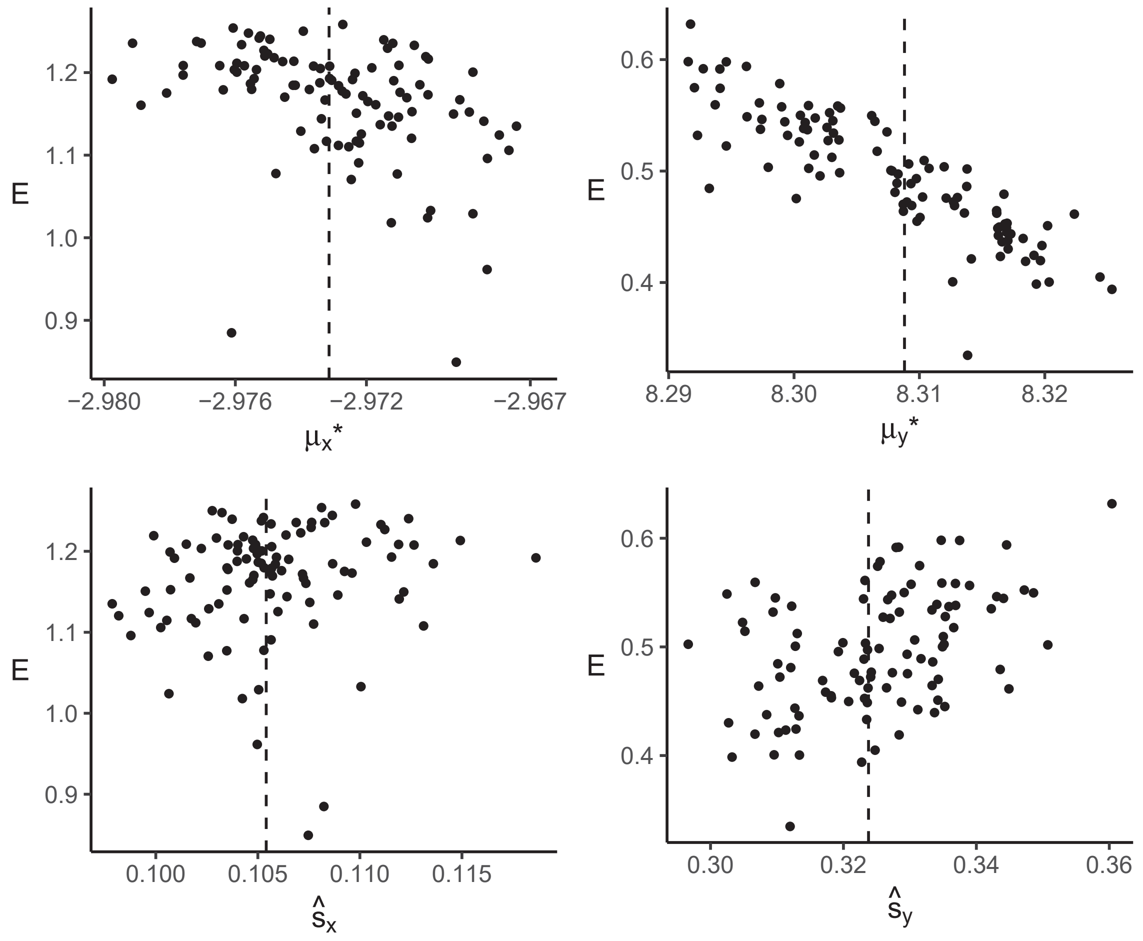


Figure S3: Efficiency (E) for estimating t (left column) and r (right column) over 100 replicate runs of kernel A6b in the phylogenetic example, plotted as a function of μ_x^*, μ_y^*, $\hat{s}_x$, and $\hat{s}_y$ estimates obtained from the burn-in.