Bayesian Analysis (2023) **TBA**, Number TBA, pp. 1–31

Speeding up Inference of Homologous Recombination in Bacteria*

Felipe J Medina-Aguayo[†], Xavier Didelot[‡], and Richard G Everitt[§]

Abstract. Bacteria reproduce clonally but most species recombine frequently, so that the ancestral process is best captured using an ancestral recombination graph. This graph model is often too complex to be used in an inferential setup, but it can be approximated for example by the ClonalOrigin model. Inference in the ClonalOrigin model is performed via a Reversible-Jump Markov Chain Monte Carlo algorithm, which attempts to jointly explore: the recombination rate, the number of recombination events, the departure and arrival points on the clonal genealogy for each recombination event, and the range of genomic sites affected by each recombination event. However, the Reversible-Jump algorithm often performs poorly due to the complexity of the target distribution since it needs to explore spaces of different dimensions. Recent developments in Bayesian computation methodology have provided ways to improve existing methods and code, but are not well-known outside the statistics community. We show how exploiting one of these new computational methods can lead to faster inference under the ClonalOrigin model.

MSC2020 subject classifications: Primary 92D15, 62-08.

Keywords: genetics, recombination, ClonalOrigin, reversible-jump MCMC.

1 Introduction

Recombination is a critical process in evolution, particularly when analysing within-species variation. Bacteria reproduce clonally, but recombination exists in most species, where a donor cell contributes a small segment of its DNA to a recipient cell. This process is analogous to gene conversion in eukaryotes and is typically modelled using an Ancestral Recombination Graph (ARG) model (Hudson, 1990; Griffiths, 1996; Wiuf and Hein, 2000). The simulation of genomic data under this model is relatively easy (Didelot et al., 2009; Brown et al., 2016) but inference of the ancestral process given genomic data is much harder. This inference problem is important for several reasons. Firstly, analysing bacterial genomic data using a phylogenetic method that ignores recombination leads to inaccurate reconstruction of the clonal part of the ancestry (Schierup and Hein, 2000; Hedge and Wilson, 2014). This could in turn lead to misleading results of subsequent analysis, for example when performing a genome-wide association study (Collins and Didelot, 2018) or inferring the dates of ancestors (Didelot et al., 2018).

*Felipe J. Medina-Aguayo and Richard G. Everitt were supported by the UK Biotechnology and Biological Sciences Re-search Council grant BB/N00874X/1. Felipe J. Medina-Aguayo got support from ONRG-RGCOMM grant, and partial funding from CONACYT CB-2016-01-284451 grant.

[†] Instituto Tecnológico Autónomo de México (ITAM), Mexico, felipe.medina@itam.mx

[‡] School of Life Sciences, University of Warwick, UK, Xavier.Didelot@warwick.ac.uk

[§] Department of Statistics, University of Warwick, UK, Richard.Everitt@warwick.ac.uk

https://doi.org/10.1214/23-BA1388

But even more importantly, the recombination process itself is often of major interest due to its great evolutionary potential. The number of substitutions introduced by recombination is often greater than the number of substitutions introduced by *de novo* mutation, since the ratio between these two quantities, denoted r/m, is often estimated to be greater than one (Vos and Didelot, 2009; Didelot and Maiden, 2010). Furthermore, the substitutions introduced by recombination are not random but have already been filtered by the selection process (Castillo-Ramírez et al., 2011). Consequently, recombination has been found to play a major role in many evolutionary processes, for example adaptation to a new host (Sheppard et al., 2013b) or agricultural niche (Sheppard et al., 2013a), speciation (Krause and Whitaker, 2015), evolution of pathogenicity (Dingle et al., 2014) or antibiotic resistance (Perron et al., 2012).

The ClonalOrigin model (Didelot et al., 2010) can be regarded as a good approximation of the aforementioned ARG process, in which recombination events are modelled independently given the clonal genealogy, denoted throughout as $\mathcal{T}$. For completeness, we note the existence of related approximations to the ARG such as the sequential Markov coalescent for eukaryotes (McVean and Cardin, 2005; Marjoram and Wall, 2006) and the bacterial sequential Markov coalescent (De Maio and Wilson, 2017). There are also simpler approximations such as the ClonalFrame model (Didelot and Falush, 2007; Didelot and Wilson, 2015) in which the origin of each recombination event is not modelled.

The major challenge when implementing the ClonalOrigin model is to efficiently explore the joint posterior distribution of the recombination rate (ρ), the number of recombination events (R), their departures ($a_1, \dots, a_R$) and arrivals (b_1,…,b_R) on the clonal genealogy and the sites delimiting the start ($x_1, \dots, x_R$) and end ($y_1, \dots, y_R$) points of each recombination event on the genome. In order to explore such a complex distribution using Markov Chain Monte Carlo (MCMC), one must resort to the Reversible-Jump MCMC (RJMCMC) algorithm (Green, 1995) which mainly aspires exploring spaces of different dimensions. This is the approach that was taken previously in both the original standalone implementation of the ClonalOrigin model (Didelot et al., 2010) and a recent reimplementation (Vaughan et al., 2017) within the BEAST2 framework (Bouckaert et al., 2019).

Unfortunately, as known by computational statisticians, the RJMCMC algorithm usually performs poorly due to the difficulty of proposing good trans-dimensional jumps. Because of this, the number of iterations (and consequently the running time) required by the algorithm for obtaining a reasonable approximation of the posterior distribution may be impractically large. Recent developments in Bayesian computation methodology provide ways of improving existing methods and code, but are not well-known outside the statistics community.

Such state-of-the-art methods belong to the framework of Sequential Monte Carlo (SMC) methods (see e.g. Doucet et al., 2001; Del Moral et al., 2006) where inference is performed using an appropriate sequence of intermediate target distributions leading to the desired one. Some of these methods have been successfully implemented to some extent when inferring coalescent trees as data arrives (see e.g. Dinh et al., 2018; Everitt et al., 2019; Fourment et al., 2018). In other areas of genetics SMC has also proved useful (see e.g. Rasmussen et al., 2014; Smith et al., 2017; Ogundijo and Wang, 2017), where standard MCMC-based inference can be problematic in the sense of diagnosing the convergence of chains, establishing burn-in periods, or avoiding getting trapped in local modes. Instead, when performing inference in a sequential manner, the complexity of the problem is reduced and the desired target distribution may be explored more efficiently.

The ideas presented here are based on the aforementioned SMC methodology that can lead to faster inference in the ClonalOrigin model; nonetheless, these ideas could be in principle applied to other evolutionary models. In particular, the work in Andrieu et al. (2018) (see also the most recent manuscript Andrieu et al. 2021), addresses the use of the aforementioned techniques for improving the convergence properties of the RJMCMC. To be precise, a generalisation of the RJMCMC algorithm is presented, where the use of annealing moves and averages are possible in order to create a chain that has provable smaller asymptotic variance, hence having better convergence properties. Our main contribution is an implementation of the RmJMCMC algorithm from Andrieu et al. (2018), but translated to the inference problem on the ClonalOrigin model. We present several examples where, depending on the configuration used, the resulting chains appears to converge much faster towards the limiting distribution (the desired posterior on the unknown parameters). This faster converges arises primarily from the appealing property of the RmJMCMC algorithm that the multiple moves can be parallelised; this in turn reduces the computational cost involved in computing averages.

2 Methods

Bayesian inference under the ClonalOrigin model

In this section we briefly describe the elements of the ClonalOrigin model, namely the parameters of interest, prior distributions on these parameters and the likelihood function for the data. For more details on assumptions and derivations of formulae please refer to cited references and the supplementary material (Medina-Aguayo et al., 2023).

- We use a coalescent tree to represent the clonal genealogy of n samples ($\mathcal{D} = D_{1:n}$) and denote such a tree by $\mathcal{T} = (\tau, \mathbf{t})$, which is composed of a topology τ and a vector $\mathbf{t} = (t_2, \ldots, t_n)$ of branch lengths. A Kingman's coalescent prior (Kingman, 1982) is assumed for $\mathcal{T}$.

- Let R denote the number of recombination events affecting the DNA sequences, which we assume a priori to be distributed according to a Poisson random variable with mean $\rho\mathcal{L}/2$, where ρ denotes the global recombination rate and $\mathcal{L} = \sum_{i=2}^{n} it_i$ is the total branch length of the tree $\mathcal{T}$, see e.g. Wiuf and Hein (1999) for more details. We also let $\mathcal{L}\,(s,t)$ denote the sum of branch lengths from time s to time t on the tree.

- Each recombination event is formed by the following variables: departure and arrival points on the genealogy, and start and end sites on the genome. These four variables, when referring to the i-th recombination edge, are denoted by a_i, b_i, x_i and y_i, respectively.

- Both variables a_i and b_i are fully determined by a time and lineage on the tree (denoted respectively by $a_i\left[t\right]$ and $a_i\left[l\right]$, respectively). The prior on the arrival point b_i is uniform on the clonal genealogy, and the prior on the departure point a_i given the arrival point b_i corresponds to the coalescent process of the recombinant edge on the clonal genealogy.
- The priors for x_i and y_i are constructed assuming a uniform distribution on the sequence for x_i and a geometric distribution of mean $\delta > 0$ for the difference $y_i - x_i \mid x_i$. When the sequence is made of B blocks comprising a total length of L the priors need to be modified accordingly as in Didelot et al. (2010).
- The entire set of variables describing the recombination events is denoted by

$$\mathcal{R} = \left(R, a_{1:R}, b_{1:R}, x_{1:R}, y_{1:R}\right).$$

- Mutation events occur at rate $\theta/2$ across the genealogy and on existing recombination edges; for simplicity we assume that all substitutions are equally likely, as in the evolutionary model JC69 (Jukes and Cantor, 1969). However, more complex mutation models could be accommodated within our framework such as the General time-reversible model (Tavaré, 1986).

Under the above the assumptions, the full prior for the set of parameters of interest is given by (see the supplementary material for the derivation)

$$\begin{aligned} p_0\left(\rho,\delta,\theta,\mathcal{T},\mathcal{R}\right) &= p_0\left(\mathcal{R}\mid\rho,\delta,\mathcal{T}\right)p_0\left(\mathcal{T}\right)p_{0,\rho}\left(\rho\right)p_{0,\delta}\left(\delta\right)p_{0,\theta}\left(\theta\right)\\ &= \left[\exp\left\{-\frac{\rho T}{2}\right\}\left(\frac{\rho}{2}\right)^R\prod_{i=1}^{R}\exp\left\{-\mathcal{L}\left(b_i\left[t\right],a_i\left[t\right]\right)\right\}p_{0,x}\left(x_i|\delta\right)p_{0,y}\left(y_i|x_i,\delta\right)\right]\\ &\quad\times\exp\left\{-\sum_{i=2}^{n}\binom{i}{2}t_i\right\}p_{0,\rho}\left(\rho\right)p_{0,\delta}\left(\delta\right)p_{0,\theta}\left(\theta\right),\end{aligned}$$

where $p_{0,\rho}, p_{0,\delta}, p_{0,\theta}$ are arbitrary priors for the recombination rate, mean-tract length and mutation rate. For the likelihood computation, first recall that mutation events occur at rate $\theta/2$ across the genealogy and on existing recombination edges; we then assume for simplicity that all substitutions are equally likely (Jukes and Cantor, 1969). The computation of the likelihood function $\mathfrak{L}\left(\mathcal{T},\mathcal{R},\theta;\mathcal{D}\right)$, for any tree $\mathcal{T}=\left(\tau,\mathbf{t}\right)$, set of recombination events $\mathcal{R}$, and mutation rate θ, is done using Felsenstein's pruning algorithm (Felsenstein, 1973, 1981). Briefly, after extracting for each site i the local tree $\mathcal{T}^{(i)}$ with leaves $z_{1:n}^{(i)}$ and internal nodes $z_{n+1:2n-1}^{(i)}$, the contribution of site i to the likelihood is given by the recursion:

$$\mathfrak{L}_i\left(\theta,\mathcal{T}^{(i)};z_{1:n}^{(i)}\right) = \sum_{z_{2n-1}^{(i)}\in\{A,C,G,T\}}\pi\left(z_{2n-1}^{(i)}\right)C\left(z_{2n-1}^{(i)}\right)$$

where

$$C\left(z_k^{(i)}\right)=\begin{cases}\sum_{v_k^{(i)}} C\left(v_k^{(i)}\right) p_{z_k^{(i)},v_k^{(i)}}\left(t^{(i)}_{v_k^{(i)}}\right)\sum_{w_k^{(i)}} C\left(w_k^{(i)}\right) p_{z_k^{(i)},w_k^{(i)}}\left(t^{(i)}_{w_k^{(i)}}\right), & \text{if } k\geq n+1;\\ \mathbb{1}\left(z_k^{(i)}=D_k^{(i)}\right), & \text{if } k\leq n;\end{cases}$$

and $p_{z,w}\left(t\right)$ denotes the transition probability from z to w in t units of time, and π corresponds to the limiting probability of such transitions. See also the supplementary material for more details.

The posterior on the full set of parameters is obtained through Bayes' Theorem

$$\pi\left(\rho,\delta,\theta,\mathcal{T},\mathcal{R}\mid\mathcal{D}\right)\propto p_0\left(\rho,\delta,\theta,\mathcal{T},\mathcal{R}\right)\mathfrak{L}\left(\mathcal{T},\mathcal{R},\theta;\mathcal{D}\right). \tag{2.1}$$

However, inferring the whole set of parameters represents a big challenge. One could instead fix one or more parameters and work with the resulting conditional distributions, for example finding a point estimate of $\mathcal{T}$ and then fix it to infer the rest of the parameters. We will consider this incomplete approach for some of the examples presented later where we aim to explore

$$\pi\left(\mathcal{R}\mid\rho,\delta,\theta,\mathcal{T},\mathcal{D}\right)\propto\tilde{\pi}\left(\mathcal{R}\right)=p_0\left(\mathcal{R}\mid\rho,\delta,\mathcal{T}\right)\mathfrak{L}\left(\mathcal{T},\mathcal{R},\theta;\mathcal{D}\right). \tag{2.2}$$

The algorithm

Markov chain Monte Carlo (MCMC) is commonly the method of choice for exploring posterior distributions. If we had access to the marginal posterior for the number of recombination events $\pi\left(R\mid\rho,\delta,\theta,\mathcal{T},\mathcal{D}\right)$, the inference of the remaining parameters could be carried out independently across different values of R. Since such a marginal is not available we need to infer R jointly with the rest of the parameters leading to a posterior that has no fixed dimension. The celebrated Reversible-Jump MCMC (RJMCMC) algorithm (Green, 1995) provides an elegant solution and corresponds to the generalisation of the Metropolis-Hastings (MH) algorithm that allows "jumps" across different dimensions. In our context, these jumps will correspond only to going up or down by one dimension, i.e. given $R\geq 1$ recombination events they can go up to $R+1$ or down to $R-1$. In addition to these trans-dimensional moves, we still need to perform intra-dimensional moves that allow the full exploration of the desired posterior. Therefore, for fixed R we could also perform moves on ρ, δ, θ, $\mathcal{T}$ and $\mathcal{R}\mid R=\{a_{1:R},b_{1:R},x_{1:R},y_{1:R}\}$.

Unfortunately, RJMCMC typically suffers from bad mixing in the sense that the resulting chain converges slowly to the desired posterior; this will in turn require a prohibitively large number of iterations for obtaining accurate answers. Due to recent developments in methodology (Karagiannis and Andrieu, 2013; Andrieu et al., 2018), the acceptance ratio for trans-dimensional moves within the RJMCMC can be understood as an importance sampling estimate of the ratio of two marginal densities, i.e. for an upwards move the ratio corresponds to $\pi\left(R+1\mid\mathcal{D}\right)/\pi\left(R\mid\mathcal{D}\right)$, and using a single importance point (more details on this can be found in the supplementary material). One could then ask whether improving the aforementioned estimate could result in a chain with better mixing. The answer turns out to be positive but with some restrictions. Two straightforward approaches to achieve variance reduction of the estimator are annealed importance sampling (Neal, 2001) and simply using more than one importance point.

In general terms, the annealing procedure creates a smooth bridge between distributions of different dimension; so, in a sense, instead of attempting one jump from R to $R+1$ that has very small chance of being accepted, we attempt to improve the chances of the jump being accepted with $T \geq 2$ perturbations. Clearly, the downside of this approach is the extra cost of performing T perturbations that could still result in a rejection within the MCMC algorithm. Whether annealing provides an advantage or not will entirely depend on the quality of the perturbations. In the ClonalOrigin context, when proposing to add a new recombination event that is in a bad region of the posterior, the perturbations attempt to correct its position before deciding to accept or not.

On the other hand, considering $N \geq 2$ importance points when jumping from R to $R+1$ requires the creation of several proposed recombination events which are used for estimating the aforementioned ratio of marginals. However, if the upwards move is accepted one needs to select which of the proposed recombination events will be retained for the next iteration of the algorithm. To do this, a categorical distribution is used that selects which recombination event will survive according to its weight or contribution to the estimate of the acceptance ratio. Therefore, a recombination event that is more plausible than the rest will have a larger weight and, if the upwards move is accepted, it will have a greater chance of being retained, ready for the next iteration.

Figure 1 illustrates both the annealing and the multiple importance points schemes when $T = 5$ and $N = 4$. The trees in black correspond to the current value of $\mathcal{T}$ on which recombination events are appended. Colours correspond to different stages of the proposed recombination event when perturbed; the idea of annealing is that the proposed event in blue at $t = 1$ will be perturbed in such way that the one in red at $t = T$ has a better chance of being accepted. This process can be repeated N times, possibly in parallel, noting that perturbations could be drastic (as in $n = N$ in the figure) or almost nonexistent (as in $n = 3$); it all depends on the quality of the proposed event at $t = 1$ and the perturbation mechanism. Only one event from those at $t = T$ is selected for the final accept-reject step in the MCMC, but events leading to a higher value in the posterior will have a better chance of being selected and possibly added later on, as discussed previously.

Algorithm 1 describes one step of the full process which was termed Reversible-multiple-Jump MCMC (RmJMCMC) in Andrieu et al. (2018), and where it was firstly introduced and studied. It belongs to the wider class of MH with Asymmetric Acceptance Ratio (MHAAR) algorithms. Here we have adapted RmJMCMC to the ClonalOrigin context. In order to perform Algorithm 1 one requires performing two type of moves, which we have termed the Upwards Annealing Move (UAM) and the Downwards Annealing Move (DAM). These two moves are in fact needed if one desires to perform annealing as described beforehand, i.e. when $T \geq 2$. We refer the reader to

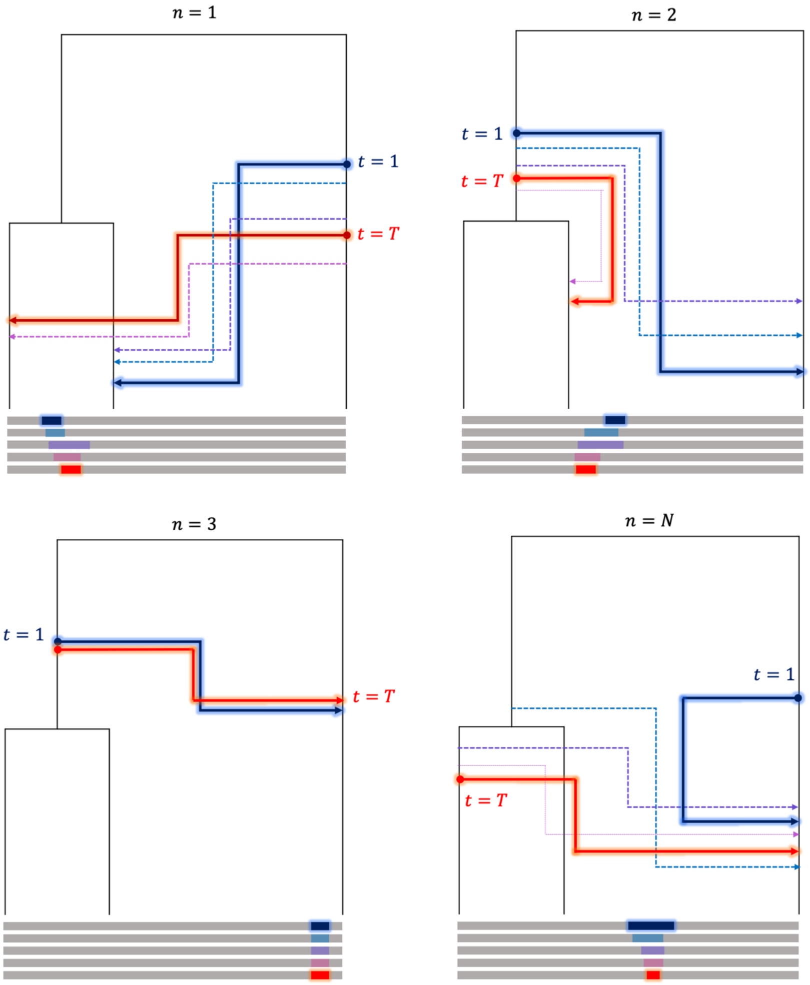


Figure 1: Illustration of annealing and multiple importance points for $T = 5$ and $N = 4$. The black tree represent the clonal genealogy $\mathcal{T}$ on which recombination events (coloured lines) occur. Once multiple recombination events are proposed at time $t = 1$ (blue lines), they are perturbed over $T - 1$ steps obtaining the recombination events at time $t = T$ (red lines). The dashed lines show how the perturbed recombination events go from time $t = 1$ to $t = T$; notice that these perturbations are not necessarily very different at each step. Perturbations also occur on the genome (gray horizontal bars) since a recombination event is also composed of start and end sites on the genome. The gray horizontal bars represent the whole genome, whereas the colours on them indicate the place where each recombination at time t is acting, for $t = 1$ (blue) up to time $t = T$ (red).

```
Algorithm 1 Reversible-multiple-Jump MCMC (RmJMCMC).
NOTATION: Let $E_r = (a_r, b_r, x_r, y_r)$ denote the $r$-th recombination event, and $E_{1:r} = (a_{1:r}, b_{1:r}, x_{1:r}, y_{1:r})$.
REQUIRES: Sequence $\{\gamma_t\}_{t=0}^{T}$ such that $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_T = 1$.
INPUT: Current values for $\rho$, $\delta$, $\theta$, $\mathcal{T}$, and $\mathcal{R} = (R, E_{1:R})$ for $R \geq 0$.
OUTPUT: New value for $\mathcal{R}$.

  • Draw $W \sim \mathrm{Unif}(0,1)$, if $W \leq 1/2$ [attempt and upwards move]:
      1. Generate $N$ recombination events $\left\{E_*^{(n)} = \left(a_*^{(n)}, b_*^{(n)}, x_*^{(n)}, y_*^{(n)}\right)\right\}_{n=1}^{N}$ using the common distribution $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$.
      2. For each $n \in \{1, \ldots, N\}$ perform an UAM as in Algorithm 1 in the supplementary material using as input $\left(\mathcal{R}, E_*^{(n)}\right)$, obtaining $r_T^{(n)}$ and the set of variables $\left(\bar{\mathcal{R}}^{(n)}, J_*^{(n)}\right)$.
      3. Compute $\bar{r}_T = N^{-1} \sum_{n=1}^{N} r_T^{(n)}$ and draw $U \sim \mathrm{Unif}\,(0,1)$.
      4. If $U \leq \bar{r}_T$
           – Draw $K \sim \mathrm{Mult}\left(N, \left(r_T^{(1)}, \ldots, r_T^{(N)}\right)\right)$.
           – Return $\bar{\mathcal{R}}^{(K)}$.
         Otherwise return $\mathcal{R}$.
  • Else if $\mathcal{R} \neq (0, \emptyset)$, rename $E_{1:R}$ and $\mathcal{R} = (R, E_{1:R})$ by $\bar{E}_{1:R}$ and $\bar{\mathcal{R}} = \left(R, \bar{E}_{1:R}\right)$, respectively. Then [attempt a downwards move]:
      1. Sample an index $J_*^{(1)} \in \{1, \ldots, R\}$ from the distribution $\overleftarrow{\varphi}(\cdot \mid \mathcal{R})$, which indicates the recombination index to be deleted.
      2. Delete the recombination event with index $J_*^{(1)}$ from $\bar{E}_{1:R}$ to obtain $E_{1:R-1}$, denoting by $E_*^{(1)} = \bar{E}_{J_*^{(1)}}$.
      3. Perform a DAM as in Algorithm 2 in the supplementary material using as input $\left(\bar{\mathcal{R}}, J_*^{(1)}\right)$, retaining only $r_T^{(1)}$ and discarding the other output variables.
      4. Define $\mathcal{R} = (R-1, E_{1:R-1})$ and generate $N-1$ recombination events $\left\{E_*^{(n)} = \left(a_*^{(n)}, b_*^{(n)}, x_*^{(n)}, y_*^{(n)}\right)\right\}_{n=2}^{N}$ using the common distribution $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$.
      5. For each $n \in \{2, \ldots, N\}$ perform an UAM as in Algorithm 1 in the supplementary material using as input $\left(\mathcal{R}, E_*^{(n)}\right)$, retaining only $r_T^{(n)}$ and discarding the other output variables.
      6. Compute $\bar{r}_T = N^{-1}\left(\left(r_T^{(1)}\right)^{-1} + \sum_{n=2}^{N} r_T^{(n)}\right)$ and draw $U \sim \mathrm{Unif}\,(0,1)$.
      7. If $\bar{r}_T \leq 1/U$ return $\mathcal{R}$, otherwise return $\bar{\mathcal{R}}$.
  • Otherwise, return $\mathcal{R} = (0, \emptyset)$ [there are currently no recombination events available for deletion].
```

the appendix, where we provide a more in depth description of the UAM and DAM algorithms used in the RmJMCMC.

Additionally, it is worth pointing out the fact that the validity of the RmJMCMC algorithm (including the UAM and DAM algorithms) has already been addressed in Andrieu et al. (2018) and in Karagiannis and Andrieu (2013). The novelty of this work is more focused on the adaptation to the ClonalOrigin framework. The implementation in `C++` of the RmJMCMC algorithm is available at: `https://github.com/fjmedinaaguayo/ClonOr_cpp`. We have also created a version of the implementation that is callable from R, found at `https://github.com/maugu/RClonOr`.

Notice that the upwards move in Algorithm 1 agrees with the description in the previous paragraph; however, the downwards move appears to be more intricate. In simple terms, we propose to delete a recombination event but in order to decide whether to accept or not we must test if this deletion is convenient. This is done by generating $N-1$ recombination events and computing the resulting acceptance ratios as if we were trying an upwards move. Low values for the aforementioned ratios would imply that the proposed deletion is possibly a good decision, whereas if many ratios result in high values it might indicate that the deletion is a poor choice. The final decision rule results in a valid algorithm as shown in Andrieu et al. (2018), otherwise the algorithm would not be exact in the sense that it does not target the desired posterior distribution. We provide more intuition in the following subsection of why this is the case. Nonetheless, non-exact or noisy methods have been explored in the past (see e.g. Alquier et al., 2016 and references therein), however we do not discuss them any further as the bias that is introduced is typically difficult to quantify.

One further observation is the obvious increased computational cost of RmJMCMC as opposed to performing only RJMCMC (equivalent to RmJMCMC when $T = N = 1$) or running multiple RJMCMC chains in parallel. Every iteration of RmJMCMC is at least TN times more expensive than one RJMCMC iteration. However, as opposed to running RJMCMC for longer or multiple independent RJMCMC chains, RmJMCMC has provable better convergence properties (Andrieu et al., 2018) that can reduce burn-in times and improve convergence towards a region of high posterior probability. This is discussed in more depth in the Results section where we look at some examples.

Validity of the algorithm

The posterior distribution in the ClonalOrigin model, as described in Didelot et al. (2010) and in the first part of Section 2, cannot be obtained analytically. For this reason the RJMCMC algorithm is needed in order to explore the posterior distribution via simulation; demonstrating that the algorithm reproduces the desired posterior does not seem possible in this context. Nonetheless, the RmJMCMC as presented here can be seen as a particular case of the more general algorithm presented in Andrieu et al. (2018), where the authors show theoretically (together with several tractable examples), that the resulting chain actually converges to the true posterior.

Instead, we now give extra details regarding the connection between the RmJMCMC as presented here and the more general version from Andrieu et al. (2018). The idea behind the RJMCMC algorithm is to explore a posterior of varying dimension, in our context the dimension is determined by the number of recombination events that act on the clonal genealogy $\mathcal{T}$. Suppose there are m active recevs (shorthand for recombination events), we either want to try going up or down in dimension, meaning we either want to add a new recev or delete an existing recev. Let π_m be the posterior distribution when there are m active recevs and let θ_m represent the m recevs, this means that θ_m represents all of the departure and arrival points on the tree and genome of the m recevs, which are ordered according to some specific rule, for example the departure time on the tree. Using our previous notation, θ_m is a vector containing m recevs of the form (a_i, b_i, x_i, y_i).

In order to move up in dimension, we propose a new recombination event, denoted by ϑ, according to the prior p_0, which was discussed in Section 2. Hence, in order to obtain θ_{m+1} we need to rearrange (θ_m, ϑ) according to our ordering rule; this is the purpose of the function $\overrightarrow{\varphi}$ since $(\theta_{m+1}, k) = \overrightarrow{\varphi}(\theta_m, \vartheta)$, where k simply denotes the position in the vector θ_{m+1} where ϑ was inserted. This additional variable ensures that the function is $\overrightarrow{\varphi}$ is invertible, we thus define $\overleftarrow{\varphi} := (\overrightarrow{\varphi})^{-1}$. In order to move down in dimension, say from $m+1$ to m recevs, we simply select uniformly at random one of the $m+1$ available recevs, say the k-th one. Consequently, we can extract ϑ out from θ_{m+1} via $(\theta_m, \vartheta) = \overleftarrow{\varphi}(\theta_{m+1}, k)$.

Therefore, the acceptance ratio for going up in dimension is given by

$$\overrightarrow{r}(\theta_m, \theta_{m+1}) = \frac{\pi_{m+1}(\theta_{m+1})(m+1)^{-1}}{\pi_m(\theta_m) p_0(\vartheta)},$$

where the proposal distribution is

$$q(\theta_m, \theta_{m+1}) = \sum_{k=1}^{m+1} \delta_{(\theta_m^{(1:m)})}(\theta_{m+1}^{(-k)}) \mathbb{1}_{\theta_{m+1}^{(k)} \in (\theta_m^{(k-1)}, \theta_m^{(k)})} p_0(\theta_{m+1}^{(k)}),$$

where $\theta_m^{(k)}$ denotes the k-th element of θ_m and $\theta_{m+1}^{(-k)}$ denotes the subvector from θ_{m+1} without considering the k-th element. In a much simpler form the above proposal is just $q(\theta_m, \theta_{m+1}) = p_0(\vartheta)$, where ϑ is such that θ_{m+1} is equal to (θ_m, ϑ) after reordering. It is straightforward to check that detailed balance is satisfied since

$$\min\{1, \overrightarrow{r}(\theta_m, \theta_{m+1})\} \pi_m(d\theta_m) p_0(d\vartheta) = \min\{1/\overrightarrow{r}(\theta_m; \theta_{m+1}), 1\} \pi_{m+1}(\theta_m + 1)(m+1)^{-1},$$

which also provides the acceptance ratio when attempting to remove a recev. That is, when there are m active recevs, the ratio for going down in dimension is

$$\overleftarrow{r}(\theta_m, \theta_{m-1}) = \frac{\pi_{m-1}(\theta_{m-1}) p_0(\vartheta)}{\pi_m(\theta_m) m^{-1}}.$$

Clearly, if $m = 0$, then the only way to jump in dimension is by going up. These expressions simplify to the ratios appearing in Appendix A from Didelot et al. (2010).

The annealing feature of the algorithm (which is essentially the AISRJMCMC from Karagiannis and Andrieu 2013) states that instead of trying a jump directly one can improve the proposal before the accept/reject step in the following sense:

- Propose $\vartheta \sim p_0(\cdot)$ and obtain $(\theta_{m+1}, k) = \overrightarrow{\varphi}(\theta_m, \vartheta)$.
- Compute, for some $\gamma \in (0, 1)$,
$$\frac{\overrightarrow{\eta_\gamma}}{\overrightarrow{\eta_0}}(\theta_{m+1}, k) = \frac{(\pi_{m+1}(\theta_{m+1})(m+1)^{-1})^{\gamma}(\pi_m(\theta_m)p_0(\vartheta))^{1-\gamma}}{\pi_m(\theta_m)p_0(\vartheta)}.$$
- Perturb (θ_{m+1}, k) using an MCMC kernel $K_\gamma(\theta_{m+1}, k; \cdot)$ targeting $\overrightarrow{\eta_\gamma}$, obtaining (θ^*_{m+1}, k^*).
- Use the following product of ratios in the accept/reject step
$$\overrightarrow{r_\gamma}(\theta_m, \theta^*_{m+1}) = \frac{\overrightarrow{\eta_\gamma}}{\overrightarrow{\eta_0}}(\theta_{m+1}, k) \times \frac{\overrightarrow{\eta_1}}{\overrightarrow{\eta_\gamma}}(\theta^*_{m+1}, k^*).$$

Detailed balance still holds since the MCMC kernel K_γ is reversible w.r.t. $\overrightarrow{\eta_\gamma}$, omitting the overhead arrows for simplicity we have

$$\begin{aligned}
\min\{1,&r_\gamma(\theta_m, \theta^*_{m+1})\}\pi_m(\theta_m)p_0(\vartheta)K_\gamma(\theta_{m+1}, k; \theta^*_{m+1}, k^*) \\
&= \min\{\frac{\eta_0}{\eta_\gamma}(\theta_{m+1}, k), \frac{\eta_1}{\eta_\gamma}(\theta^*_{m+1}, k^*)\}\eta_\gamma(\theta_{m+1}, k)K_\gamma(\theta_{m+1}, k; \theta^*_{m+1}, k^*) \\
&= \min\{\frac{\eta_0}{\eta_\gamma}(\theta_{m+1}, k), \frac{\eta_1}{\eta_\gamma}(\theta^*_{m+1}, k^*)\}\eta_\gamma(\theta^*_{m+1}, k^*)K_\gamma(\theta^*_{m+1}, k^*; \theta_{m+1}, k) \\
&= \min\{1/r_\gamma(\theta_m, \theta^*_{m+1}), 1\}\eta_1(\theta^*_{m+1}, k^*)K_\gamma(\theta^*_{m+1}, k^*; \theta_{m+1}, k) \\
&= \min\{1/r_\gamma(\theta_m, \theta^*_{m+1}), 1\}\pi_{m+1}(\theta^*_{m+1})(m+1)^{-1}K_\gamma(\theta^*_{m+1}, k^*; \theta_{m+1}, k).
\end{aligned}$$

This tells us that in order to obtain a valid algorithm we need to implement also an annealing move for going down in dimension in such way that

$$\overleftarrow{r_\gamma}(\theta_m, \theta^*_{m-1}) = 1/\overrightarrow{r_\gamma}(\theta^*_{m-1}, \theta_m),$$

hence we need

$$\frac{\overleftarrow{\eta_\gamma}}{\overleftarrow{\eta_0}}(\theta_{m-1}, \vartheta) \times \frac{\overleftarrow{\eta_1}}{\overleftarrow{\eta_\gamma}}(\theta^*_{m-1}, \vartheta^*) = \frac{\overrightarrow{\eta_\gamma}}{\overrightarrow{\eta_1}}(\theta_m, k) \times \frac{\overrightarrow{\eta_0}}{\overrightarrow{\eta_\gamma}}(\theta^*_m, k^*),$$

which imposes the following condition

$$\overleftarrow{\eta_\gamma}(\theta_{m-1}, \vartheta) = \overrightarrow{\eta_\gamma}(\theta_m, k).$$

The above implies that

$$(\pi_{m-1}(\theta_{m-1})p_0(\vartheta))^{\overleftarrow{\gamma}}(\pi_m(\theta_m)m^{-1})^{1-\overleftarrow{\gamma}} = (\pi_m(\theta_m)m^{-1})^{\overrightarrow{\gamma}}(\pi_{m-1}(\theta_{m-1})p_0(\vartheta))^{1-\overrightarrow{\gamma}},$$

which translates to $\overleftarrow{\gamma} = 1 - \overrightarrow{\gamma}$, where $\overleftarrow{\gamma}$ is the value used in the downwards move and $\overrightarrow{\gamma}$ is the one used going upwards in dimension. This condition is exactly the imposed in Algorithm 2 in the supplementary material (DAM move). The previous idea can be extended to any number of intermediate steps using a sequence of intermediate targets

$\{\overrightarrow{\eta_{\gamma_t}}\}_{t=1}^{T}$, resulting in a valid algorithm in the sense that it targets the desired posterior, as long as $\overleftarrow{\gamma}_t = \overrightarrow{\gamma}_{T-t}$, for each $t = 0, \dots, T$.

Having established the nature of the annealing moves we now turn to the multiple jump aspect of the algorithm. Having m active recevs, we propose N possible new recevs, specifically we propose $\vartheta^{(1:N)} := \{\vartheta^{(n)}\}_{n=1}^{N}$, each of them coming from the prior p_0; this in turn produces N different acceptance ratios $\{\overrightarrow{r}^{(n)}\}_{n=1}^{N}$. A way to condense all of them is to average their values, hence obtaining an averaged acceptance ratio

$$\overrightarrow{r_N} = \frac{1}{N}\sum_n \overrightarrow{r}^{(n)}.$$

Similarly, when trying to delete recevs, one could propose N possible candidates out of the existing m recevs obtaining N different integers $\left\{k^{(n)}\right\}_{n=1}^{N}$, where each $k^{(n)}$ was selected uniformly from $\{1, \dots, m\}$. Once again, this produces N different acceptance ratios $\{\overleftarrow{r}^{(n)}\}_n$, which could be condensed into an averaged acceptance ratio

$$\overleftarrow{r_N} = \frac{1}{N}\sum_n \overleftarrow{r}^{(n)}.$$

However, plugging these averaged ratios into a MH algorithm does not produce in general a chain with the correct limiting posterior, see e.g. Alquier et al. (2016). Nonetheless, using a slight modification of the previous idea will in fact produce an exact algorithm, as shown in Andrieu et al. (2018). Suppose we use $\overrightarrow{r_N}$ as acceptance ratio into an MH algorithm for going up in dimension, the acceptance probability can be expressed as follows

$$\begin{aligned}
\min\{1, \overrightarrow{r_N}\} &= \min\{1, \frac{1}{N}\sum_n \overrightarrow{r}^{(n)}\} \\
&= \min\{1, \frac{N^{-1}\overrightarrow{r}^{(j)}}{\frac{\overrightarrow{r}^{(j)}}{\sum_n \overrightarrow{r}^{(n)}}}\} \\
&= \min\{1, \frac{N^{-1}\pi_{m+1}(\theta_{m+1}^{(j)})(m+1)^{-1}}{\pi_m(\theta_m)p_0(\vartheta^{(j)})\frac{\overrightarrow{r}^{(j)}}{\sum_n \overrightarrow{r}^{(n)}}}\} \\
&= \min\{1, \frac{\pi_{m+1}(\theta_{m+1}^{(j)})(m+1)^{-1}\prod_{l\neq j} p_0(\vartheta^{(l)})N^{-1}}{\pi_m(\theta_m)\prod_l p_0(\vartheta^{(l)})\frac{\overrightarrow{r}^{(j)}}{\sum_n \overrightarrow{r}^{(n)}}}\} \\
&=: \min\{1, \frac{\pi_{m+1}(\theta_{m+1})\overleftarrow{Q_N}(\theta_{m+1}, \theta_m)}{\pi_m(\theta_m)\overrightarrow{Q_N}(\theta_m, \theta_{m+1})}\}.
\end{aligned}$$

The proposal $\overrightarrow{Q}_N$ states how N new recevs are generated (independently and according to the prior) and then only one is selected for constructing $\theta_{m+1}^{(j)}$, in this case the j-th one (according the unnormalised weight $\overrightarrow{r}^{(j)}$). The other proposal $\overleftarrow{Q}_N$ states the reverse move, a recombination is selected out of the existing $m+1$ for deletion, this is equivalent to sample uniformly a single ϑ from θ_{m+1}. However, recall that there are other $N-1$ recevs involved in the algorithm, these are proposed independently from the prior as stated by the product of $N-1$ terms. The only thing left to decide is the value of the superscript of the recev that will be deleted, this is done uniformly as the term N^{-1} indicates.

Having found the above kernels we can write the acceptance probability when attempting to delete a recev. This is simply

$$\begin{aligned}
&\min\{1, \frac{\pi_{m-1}(\theta_{m-1})\overrightarrow{Q_N}(\theta_{m-1}, \theta_m)}{\pi_m(\theta_m)\overleftarrow{Q}_N(\theta_m, \theta_{m-1})}\} \\
&= \min\{1, \frac{\pi_{m-1}(\theta_{m-1}) \prod_l p_0(\vartheta^{(l)}) \frac{\overrightarrow{r}^{(j)}}{\sum_n \overrightarrow{r}^{(n)}}}{\pi_m(\theta_m^{(j)})m^{-1} \prod_{l \neq j} p_0(\vartheta^{(l)}) N^{-1}}\} \\
&= \min\{1, \frac{\pi_{m-1}(\theta_{m-1}) p_0(\vartheta^{(j)}) \frac{\overrightarrow{r}^{(j)}}{\sum_n \overrightarrow{r}^{(n)}}}{\pi_m(\theta_m^{(j)})m^{-1} N^{-1}}\} \\
&= \min\{1, \frac{N \overrightarrow{r}^{(j)}}{\overrightarrow{r}^{(j)} \sum_n \overrightarrow{r}^{(n)}}\} \\
&= \min\{1, \frac{1}{\overrightarrow{r_N}}\}.
\end{aligned}$$

Notice that the previous probability coincides with the expression used in Algorithm 1, where a deletion would proceed as follows: select uniformly a recombination to be deleted out of the m existing ones, which produces a ratio $\overleftarrow{r}^{(j)}$ for going down in dimension. Considering the remaining recevs, propose $N-1$ new ones according to the prior, which produces $N-1$ ratios for going up in dimension, i.e. $\{\overrightarrow{r}^{(-j)}\}$. Select uniformly the superscript j (out of N possibilities) to the recev selected for deletion, and finally accept the deletion if an independent uniform r.v. is less than or equal to $1/\overrightarrow{r_N} = N/\sum_n \overrightarrow{r}^{(n)}$, recalling that $\overrightarrow{r}^{(j)} = 1/\overleftarrow{r}^{(j)}$.

Some final implementation details are in order, in either move (adding or deleting a recev) there is no need to sample the superscript j before deciding whether to accept or not. The only expression needed for making a decision is the value of $\overrightarrow{r_N}$, once a decision is made the superscript can be obtained using the unnormalised weights $\{\overrightarrow{r}^{(n)}\}$ (when adding a recev) or uniformly out of N possible values (when deleting a recev). Lastly, notice that the value of the superscript j is not needed in a further iteration of the MCMC chain. This is because the selected recev and the other $N-1$ dummy events are only needed for computing the acceptance ratio $\overrightarrow{r_N}$; in the end, if a deletion is accepted, the new state of the chain corresponds simply to the preserved recevs. Hence, the value of j is always 1 in Algorithm 1, as it is irrelevant in further iterations.

Flavours of RmJMCMC

In order to implement Algorithm 1, we must specify the sampling auxiliary distributions $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$ and $\overleftarrow{\varphi}(\cdot \mid \mathcal{R})$ (which could both depend on other parameters, e.g. $\rho, \delta, \theta, \mathcal{T}$). In the following section we present results using simple choices, the joint prior on (a, b, x, y) for the distribution $\overrightarrow{\varphi}$ and a discrete uniform on the set $\{1, \ldots, R\}$ for $\overleftarrow{\varphi}$ assuming there are R active recombination events. In mathematical terms, the associated densities are

$$\overrightarrow{\varphi}\,(a, b, x, y) = \exp\left\{-\mathcal{L}\left(b\left[t\right], a\left[t\right]\right)\right\} p_{0,x}\left(x|\delta\right) p_{0,y}\left(y|x, \delta\right),$$

and $\overrightarrow{\varphi}\,(j \mid R) = R^{-1}$ for $j \in \{1, \ldots, R\}$. The previous choices greatly simplify the computation of the output ratios in Algorithms 1 and 2 in the supplementary material, which involve mainly ratios of likelihood functions.

Additionally, we must define the way to perturb recombination events at every small step in the annealing process. This is done using an MCMC algorithm as explained in Algorithms 1 and 2 in the supplementary material; hence the perturbation is fully defined once we select a proposal distribution for $\mathcal{R}$ fixing the value of R, i.e. we need to perturb at least one recombination event within the existing R events. For the examples in the following section we choose to perturb (using the prior as proposal) only the newly created recombination event when going upwards, or the recombination event to be deleted when going downwards, for both cases this event is denoted by E_* in Algorithms 1 and 2 in the supplementary material. Doing this provides a very simple expression for the acceptance ratio in the MCMC steps within the annealing that involves only ratios of likelihood functions.

Finally, we must decide the number of annealing steps T, the sequence of real numbers $\{\gamma_t\}_{t=0}^{T}$ and the value of replicates N. In the examples that follow we use different values for T and N which involve different costs and running times. As mentioned earlier, the computational cost increased as T and N increases, however an appealing property of RmJMCMC is that loops involving N (either upwards or downwards) can be performed in parallel, this may lead to higher efficiency when computing running times. Due to this, the value of N is commonly determined by the number of cores available in the computer or server. For the required sequence $\{\gamma_t\}_{t=0}^{T}$ we simply choose a linear interpolation $\gamma_t = t/T$.

We want to emphasise that the choice of $\overrightarrow{\varphi}$ and $\overleftarrow{\varphi}$ were made in accordance to the original ClonalOrigin implementation from Didelot et al. (2010); whereas the choices for the proposals within the MCMC and the elements of the sequence $\{\gamma_t\}_{t=0}^{T}$ were made for convenience. However, the way RmJMCMC was formulated permits the use of more complex approaches that could provide better results in terms of efficiency. Some of these improvements are briefly described in the final section and are devoted to future work.

3 Results

This section compares the RmJMCMC algorithm and the standard RJMCMC for the ClonalOrigin model. We start with some toy simulated experiments, moving later on to results using real, and previously studied, datasets. When considering more than one importance point (i.e. $N > 1$), we are running a parallelised version of the algorithm in all of the examples that follow.

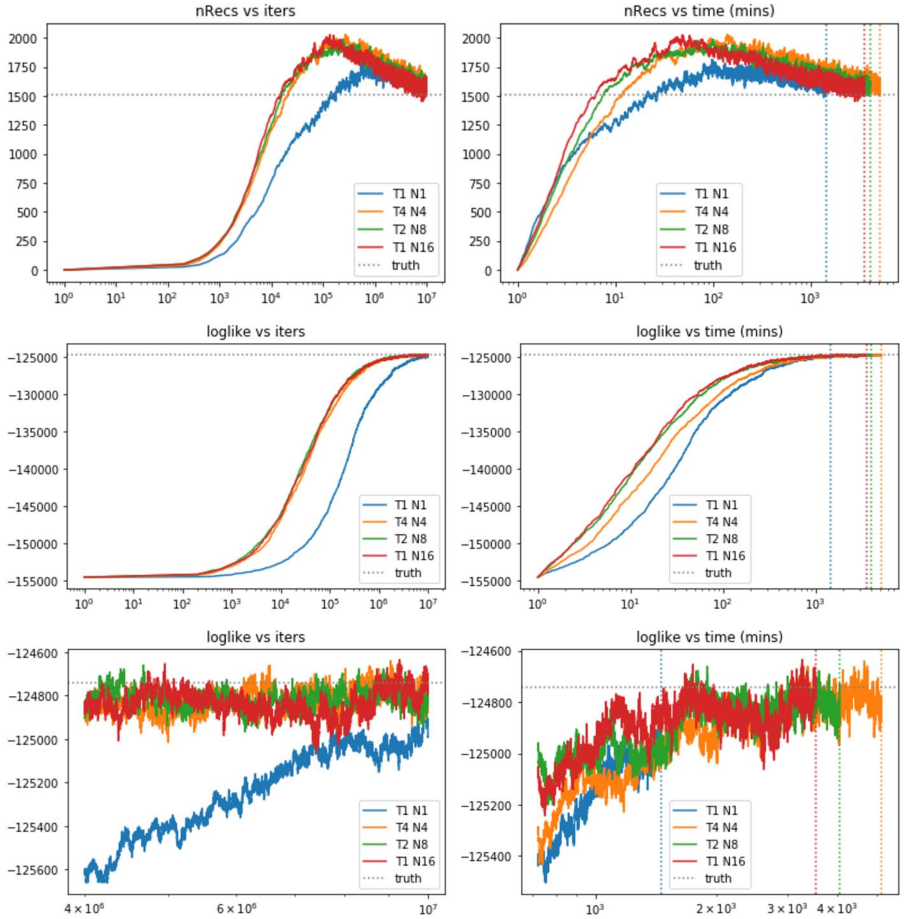


Figure 2: Trace plots for the total number of recombinations and for the log-likelihood for different combinations of T and N and using 10 million iterations. Plots on the left correspond to values vs iteration number, those on the right are vs running time. Grey dotted line corresponds to ground truth, coloured dotted lines indicate time when algorithm stopped. Bottom row: trace plots of the log-likelihood for the last 6 million iterations (left) and after 12 hours of running time (right).

Application to simulated data

We present several simulated examples in order to explore the scalability and usability of RmJMCMC. We compare the settings when $N = T = 1$ (corresponding to the RJMCMC algorithm), when $T = 4$ and $N = 4$, when $T = 2$ and $N = 8$, and when $T = 1$ and $N = 16$. The efficiency between these configurations is compared using an estimation of the effective sample size (ESS) for a quantity of interest, as is commonly done in MCMC (Robert and Casella, 2004). In simple terms, the ESS indicates how

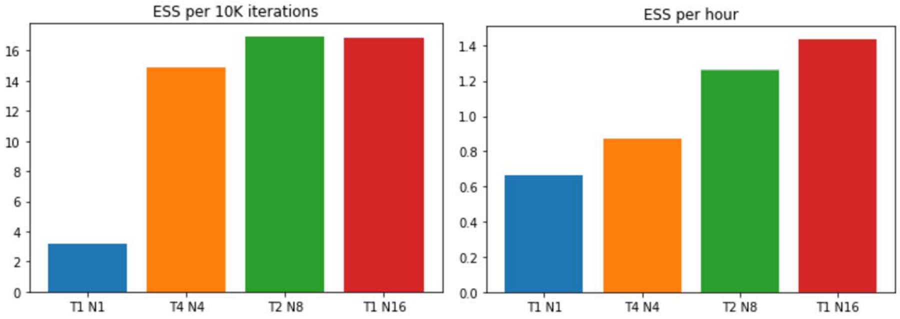


Figure 3: Effective sample sizes per 10 thousand iterations (left) and per hour of running time (right).

many independent samples were obtained from the total number of iterations for which the algorithm was run, and if the chain exhibits large correlation then the ESS will typically be much smaller than the number of iterations.

Throughout this section we consider a "typical setting" where data was simulated using a mean tract length of $\delta = 236$, a mutation rate per site $\theta_s = 0.03$, and a recombination rate per site $\rho_s = 0.005$. These three parameters remain fixed, and consequently the inference is carried only for the number of recombination events and their locations on the genome and on the tree, i.e. the target distribution is given by (2.2).

Example 1. Typical setting: 50 sequences, length 50 kilobase pairs (Kb), mean tract length $\delta = 236$, global mutation rate $\theta = 1500$ ($\theta_s = 0.03$), global recombination rate $\rho = 250$ ($\rho_s = 0.005$).

Figure 2 contains traceplots for the number of recombination events (top row) and for the value of the log-likelihood (middle and bottom rows). Observe that the schemes for which $N > 1$ have a similar performance when comparing against iteration number (plots on the left); this is not surprising since the annealing moves (those involving T) are perturbed using an MCMC algorithm with a proposal equal to the prior, and also the multiple instances (those involving N) are generated using the prior. However, the computational burden is different in each setting since for a fixed value of NT the algorithm could be parallelised using N cores but still requires $T - 1$ serial iterations of the annealing process. Hence, for this case, the setting where $T = 1$ and $N = 16$ should perform best when taking into account the running time. The plots on the right incorporate this information and notice that all the schemes still seem to perform better than the standard RJMCMC chain (in blue) as they reach a region where the likelihood is high in a shorter amount of time, this can also be seen more clearly in the plots from the bottom row. It is worth noting that despite the RJMCMC chain being closer to the true number of recombination events, it is not a reliable indicator of whether the chain has converged since the value of the log-likelihood is still in a transient phase.

Figure 3 compares the ESS fusing the values of the log-likelihood for the different schemes. Notice that when the running time is not taken into account (left plot) the

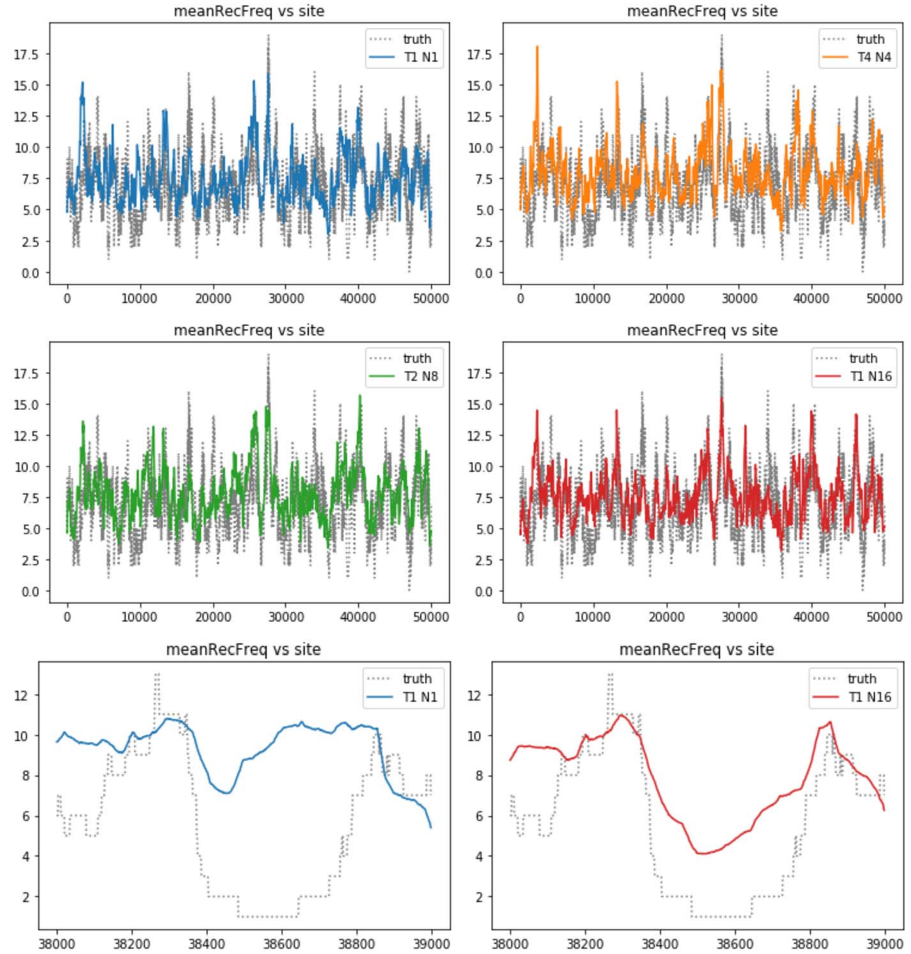


Figure 4: Average recombination frequency vs site number for different combinations of T and N. Grey dotted line corresponds to the true recombination frequency.

three settings of RmJMCMC (those with $NT = 16$) have very similar ESS, as expected from the left plots of Figure 3. The right plot is in line with the conjecture that the setting when $T = 1$ and $N = 16$ is the most efficient, observing that all three still outperform the standard RJMCMC chain. When a chain has not reached stationarity one must be careful when attempting to draw conclusions from the ESS, this is the case for the standard RJMCMC chain shown in blue. Nevertheless, using annealing moves ($T > 1$) or multiple importance points ($N > 1$) is beneficial as the corresponding chains appear to converge more rapidly towards the limiting distribution.

Figures 4 and 5 compare other quantities of interest. Figure 4 shows the recombination frequencies across the sequence of length 50K depicting the active number of

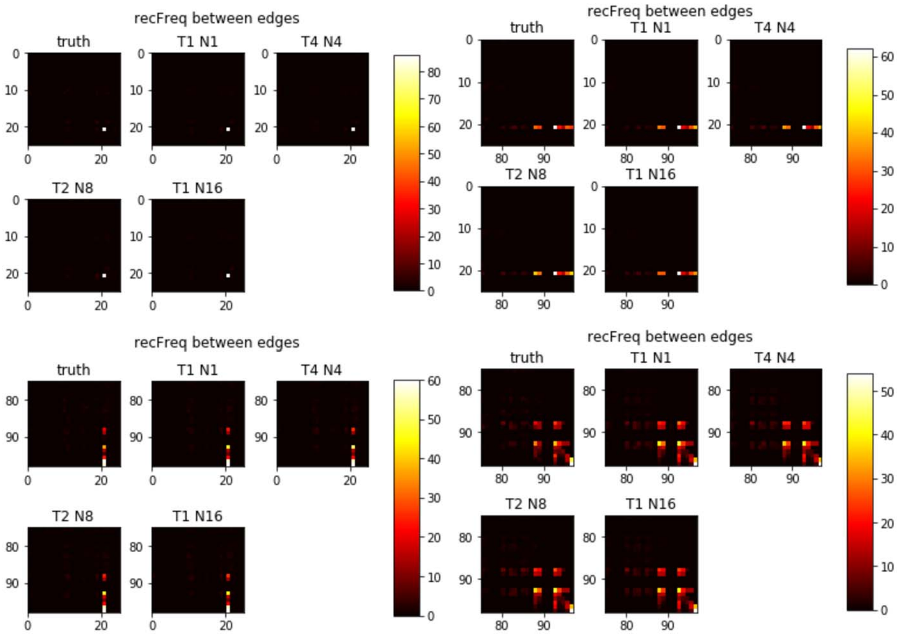


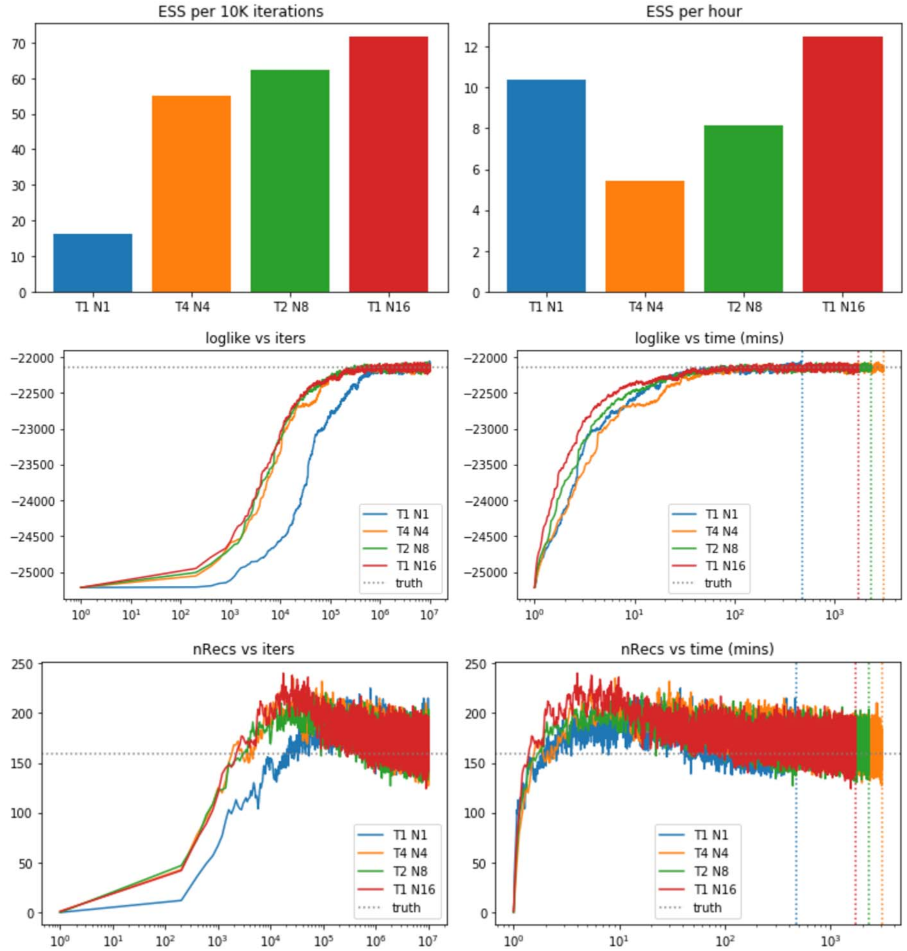
Figure 5: Average recombination frequency between edges, y-axis indicates the edge number where a recombination departed, x-axis denotes the edge number where a recombination landed. The plots correspond to the corners of the 98x99 matrices containing the recombination frequencies between edges.

recombinations affecting each site. Despite the apparent similarity of the results for the different schemes (top and middle rows), when looking at a smaller scale there are important deviances from the truth (bottom plots) that are direct consequence of a bad mixing from the chain. Settings with large values of NT have greater chances of better exploring the complicated state space in the ClonalOrigin model.

Figure 5 also shows recombination frequencies, but between edges in the tree. Coloured squares indicate pair of edges on the tree joined by several recombination events, depending on the scale. In this case, the differences between the schemes and against the ground truth are negligible.

Example 2. Typical setting: 50 sequences, length 10Kb, mean tract length $\delta = 236$, global mutation rate $\theta = 300$ ($\theta_s = 0.03$), global recombination rate $\rho = 50$ ($\rho_s = 0.005$).

This example is similar to the previous one, except for the length of the sequence which is now 10Kb. The global mutation and recombination rates have been scaled appropriately leaving the per-site rates the same as before. Figure 6 contains some plots of interest, we observe once more an improvement of the ESS and faster convergence of the log-likelihood without considering running times. However, when time is taken into account (right plots) running RmJMCMC seems to be worth it only when $T = 1$ and


ESS per 10K iterations

ESS per hour

loglike vs iters

loglike vs time (mins)

nRecs vs iters

nRecs vs time (mins)

Figure 6: Top row: Effective sample sizes per 10 thousand iterations (left) and per hour of running time (right). Middle and bottom row: Trace plots for the total number of recombinations (bottom) and for the log-likelihood (middle) for different combinations of T and N and using 10 million iterations.

$N = 16$. Having shorter sequences makes the inference problem less expensive, so the schemes where $T > 1$ do not seem to add any benefit.

Example 3. Typical setting: Comparison across schemes with different sequence lengths and different number of sequences. For all cases $\delta = 236$, $\theta_s = 0.03$, $\rho_s = 0.005$.

Here we compare how the ESS is affected as the length of the sequences becomes larger and as the number of sequences increases. Figure 7 presents the results for lengths of 10Kb, 20Kb and 50Kb. From the plots in the bottom row, the inverse of the ESS appears to be linear as the length of the sequence increases, more specifically, doubling

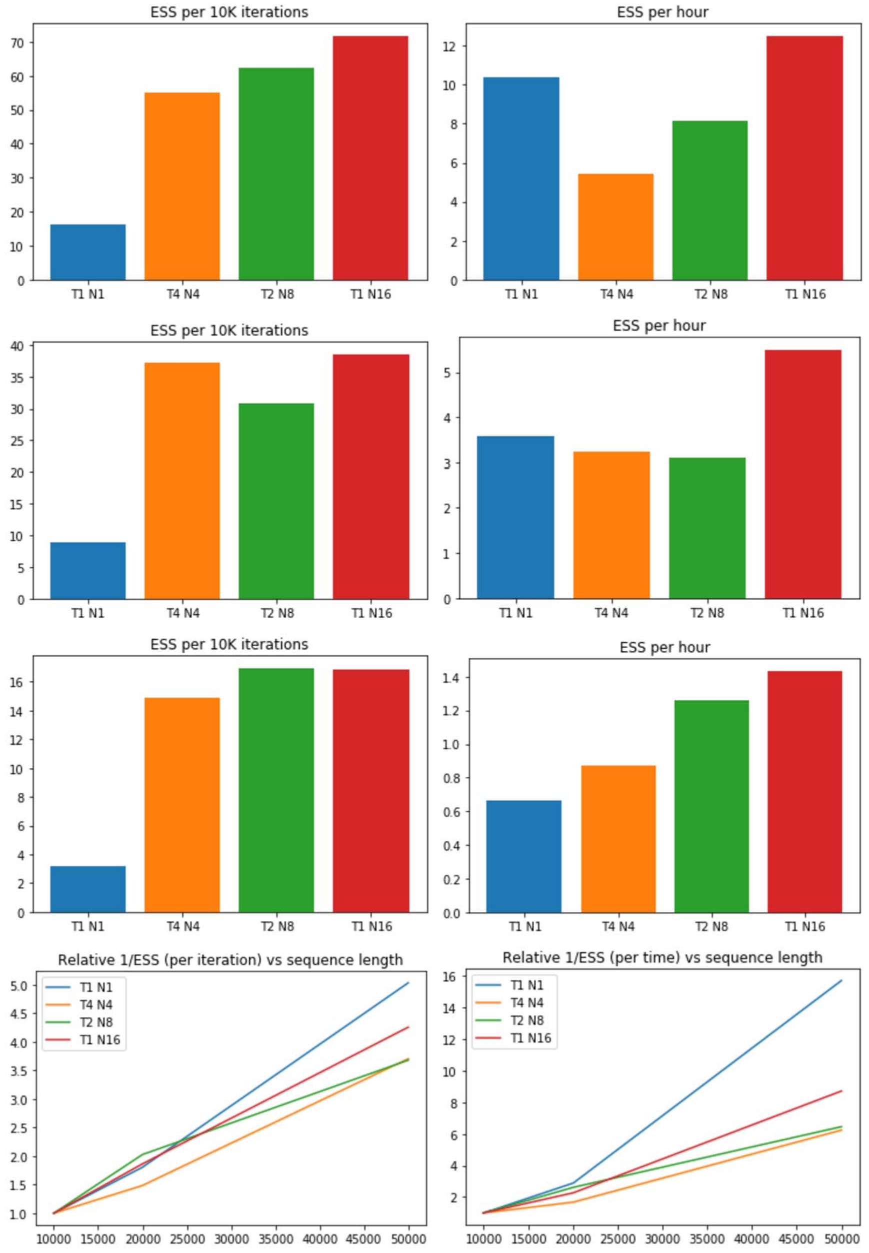


Figure 7: ESS comparison for 10Kb, 20Kb and 50Kb (plots in rows 1-3 , respectively) considering 50 sequences in all cases. Bottom plots: increase of the inverse of the ESS as a function of sequence length.

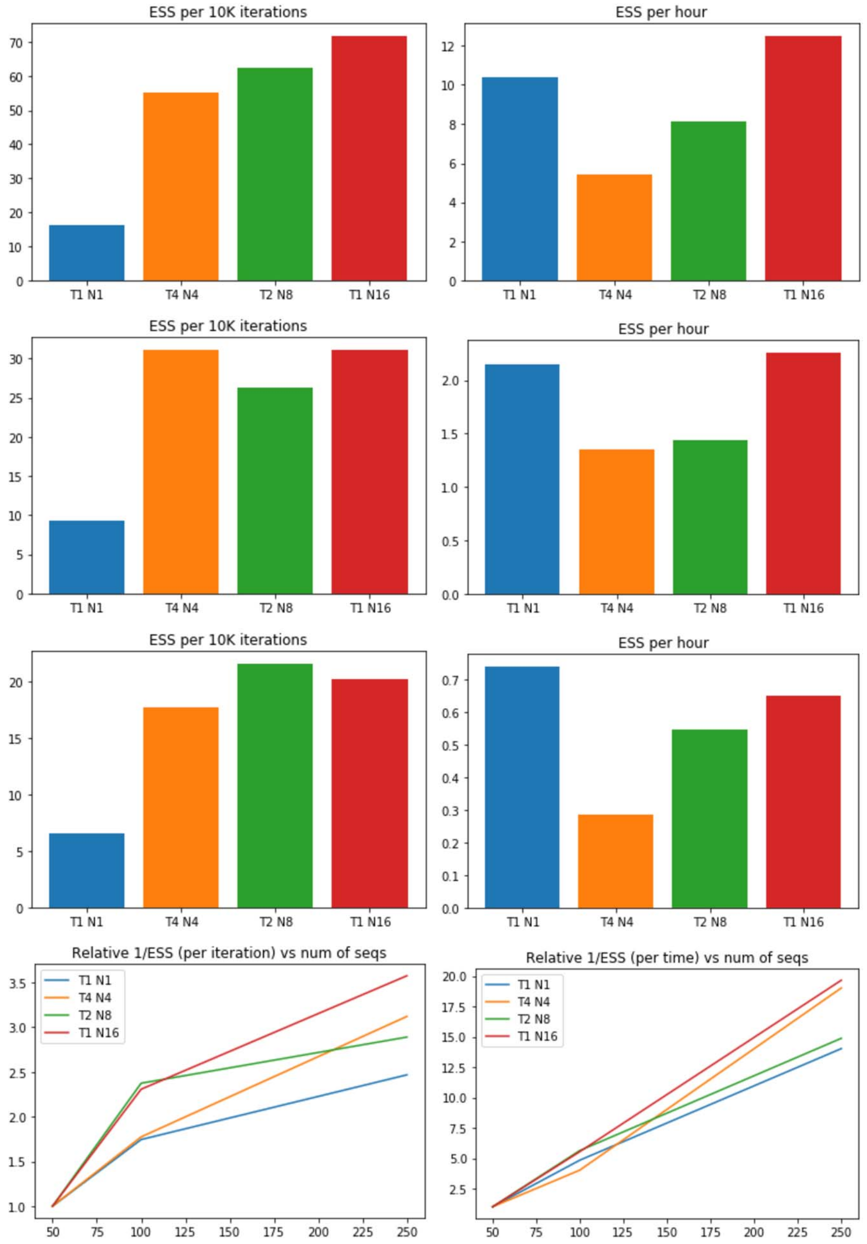


Figure 8: ESS comparison for 50, 100 and 250 sequences (plots in rows 1-3 , respectively) considering a length of 10K in all cases. Bottom plots: increase of the inverse of the ESS as a function of the number of sequences.

the sequence length halves the ESS. Interestingly, the reduction of ESS for the RJMCMC chain (in blue) when taking into account the running time seems more drastic than the other schemes.

In contrast, increasing the number of sequences seems to have a logarithmic effect on the inverse of the ESS as seen in Figure 8; however, when considering running times, the effect becomes linear. For this case, the method with less reduction of ESS is the RJMCMC chain. Thus, RmJMCMC seems to outperform RJMCMC when having larger sequences rather than when having lots of short sequences.

3.1 Application to *Escherichia coli* data

Here we present results for a dataset of 27 sequences from *Escherichia coli* on two regions with different recombination intensity; this dataset has been previously studied using ClonalOrigin (Didelot et al., 2012). Similarly as in the simulated examples, we have fixed the clonal genealogy $\mathcal{T}$ and also some of the global parameters. The specific details on how these fixed values were obtained can be found in Didelot et al. (2012).

Example 4. *E. coli* data on region of length 27.1Kb. The fixed parameters considered here are the clonal genealogy $\mathcal{T}$ and the mutation rate per site $\theta_s = 0.0125$. Therefore, the inference is performed on the set of recombination events $\mathcal{R}$ and the global parameters ρ and δ, i.e. the target distribution arises from (2.1) by fixing $\mathcal{T}$ and θ accordingly.

Figure 9 presents some preliminary results on a region with a moderate recombination frequency of ρ_s near 0.01. Observe that, even though increasing N does not seem to improve the value of the ESS adjusted for running time, the log-likelihood trace plot appears to indicate there is some advantage of taking $N > 1$ during the burn-in phase. Interestingly, the trace plots in the bottom row show that when $N > 1$ the chain appears to favour large values for ρ during the burn-in stage, whereas for δ might take different routes towards the apparent high-posterior region. Clearly, further simulations are needed to decide whether RmJMCMC offers a clear advantage.

Performing full inference can be quite challenging since the global parameters have a strong influence in the number and length of recombination events. We thus run further simulations but reducing the complexity by fixing the global parameter δ to $\delta = 542$. Doing this allow us to obtain convergent chains in a decent amount of time and with moderate file sizes. In Figure 10 we observe the results when δ is fixed. As commented before, the ESS adjusted for time (top-left plot) does not tell the whole story; in fact, the chain when $N = 24$ reaches regions where the likelihood is much higher (top-right plot). This effect does not seem to affect neither the inferred number of recombination events nor the recombination rate ρ (middle plots). However, the mean recombination frequencies (bottom-left plot) seem to be quite different, in certain regions, depending on the value of N; in particular, when $N = 12$ or $N = 24$ the chains seem to infer a region of very high recombination near the site 2000, whereas the chain $N = 1$ does not seem to reflect this. Finally, there seems to be almost no variation in the inferred recombination frequencies between edges (bottom-right plot).

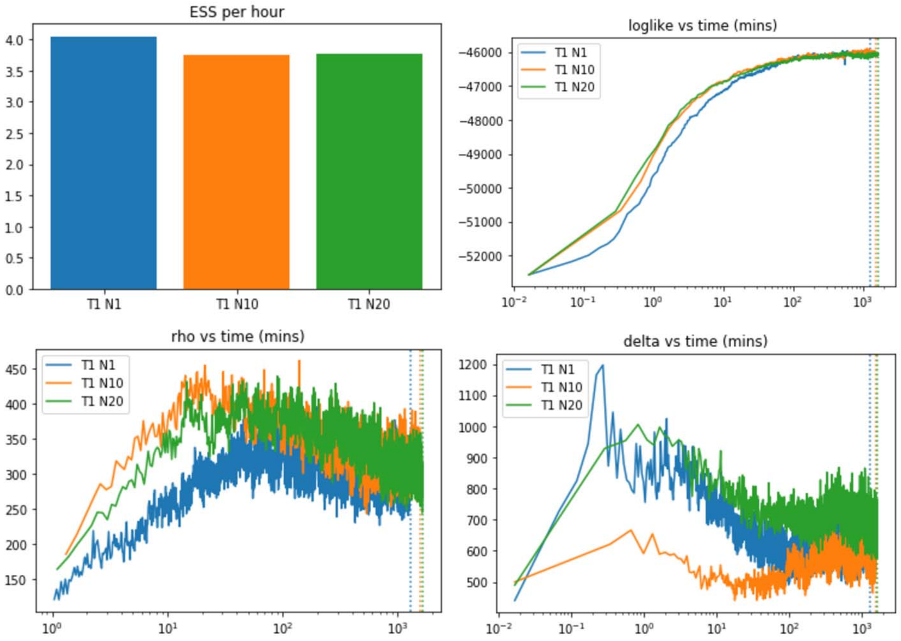


Figure 9: *E. coli* data on region of length 27.1Kb allowing moves on δ and ρ. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; bottom-left: trace plots for ρ; bottom-right: trace plots for δ.

Example 5. *E. coli* data on region of length 3.4Kb. Here we consider a shorter region but for which we know the recombination frequency is higher. The fixed parameters considered once again are the clonal genealogy $\mathcal{T}$ and the mutation rate per site $\theta_s = 0.0125$, we have also fixed the tract length $\delta = 542$ due to the high levels of recombination. The inference is thus performed on the set of recombination events $\mathcal{R}$ and the global parameter ρ, i.e. the target distribution arises from (2.1) by fixing $\mathcal{T}$, θ and δ accordingly.

Observe from Figure 11 that this is a more challenging region for the *E. coli* data. In particular, observe that apparently the setting $N = 12$ does not provide an advantage in terms of reaching a high likelihood region in a shorter time; however, when looking at the trace plots for the number of recombination events and for ρ it does appear to have an improvement. In this case ESS plots have not been presented since clearly the chain has not reached stationarity, even after nearly 21 days of running the algorithms. Recall that accurate ESS values are based on the fact that the chains have converged. Observe that the recombination rate in this region is much higher than in the previous example, with ρ_s reaching values around 0.17. Due to the difficulty of this example, further simulations are needed in order to have a clearer picture of the benefits, if any, from running a RmJMCMC algorithm.

Further simulations were run fixing the value $\rho = 670$, which is roughly the value

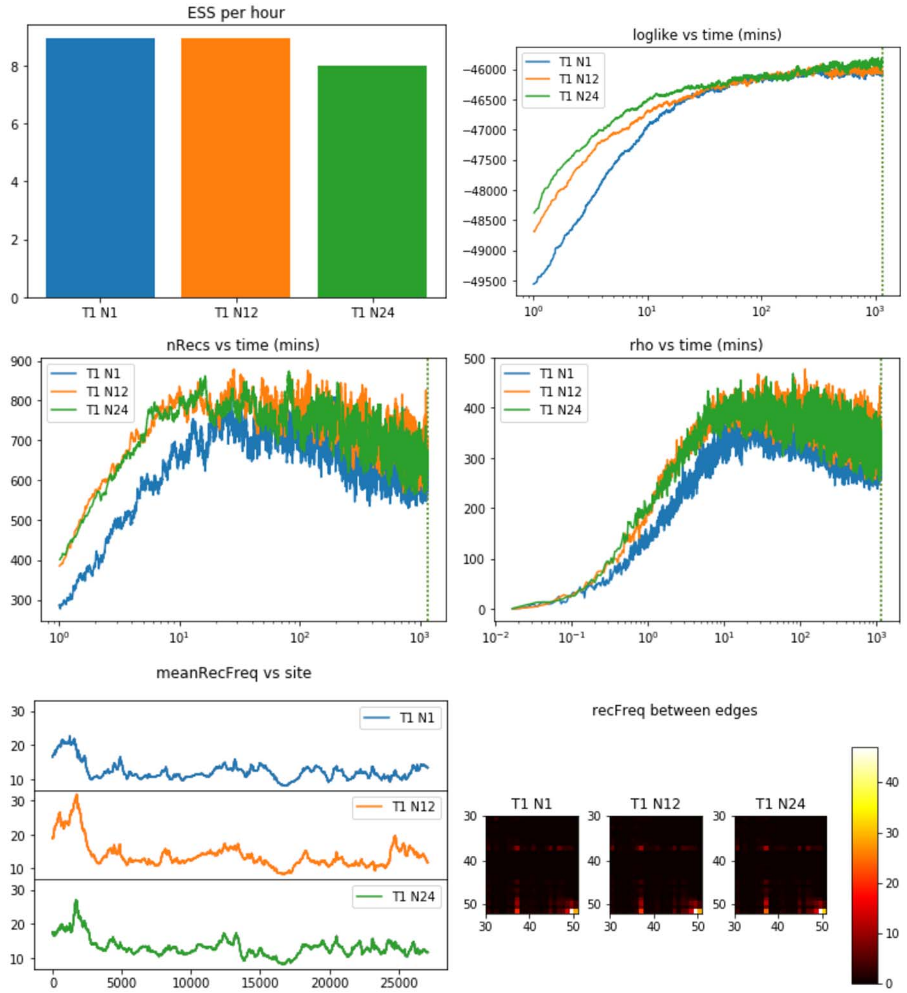


Figure 10: *E. coli* data on region of length 27.1Kb allowing moves on ρ and fixing $\delta = 542$. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; middle-left: trace plots for the number of recombination events; middle-right trace plots for ρ; bottom-left: comparison of mean recombination frequencies per site; bottom-right: mean recombination frequencies between edges.

reached by the chains during the last hundreds of iterations. Once more this reduces the complexity of the inference problem in order to obtain sensible estimates in a manageable amount of time. Figure 12 shows the results when ρ and δ are fixed observing that in this case the ESS adjusted by time does seem to improve for the chains with multiple jump. All chains seem to reach similar regions in the case when looking at the trace plots for the log-likelihood and for number of recombination events; however, the mean

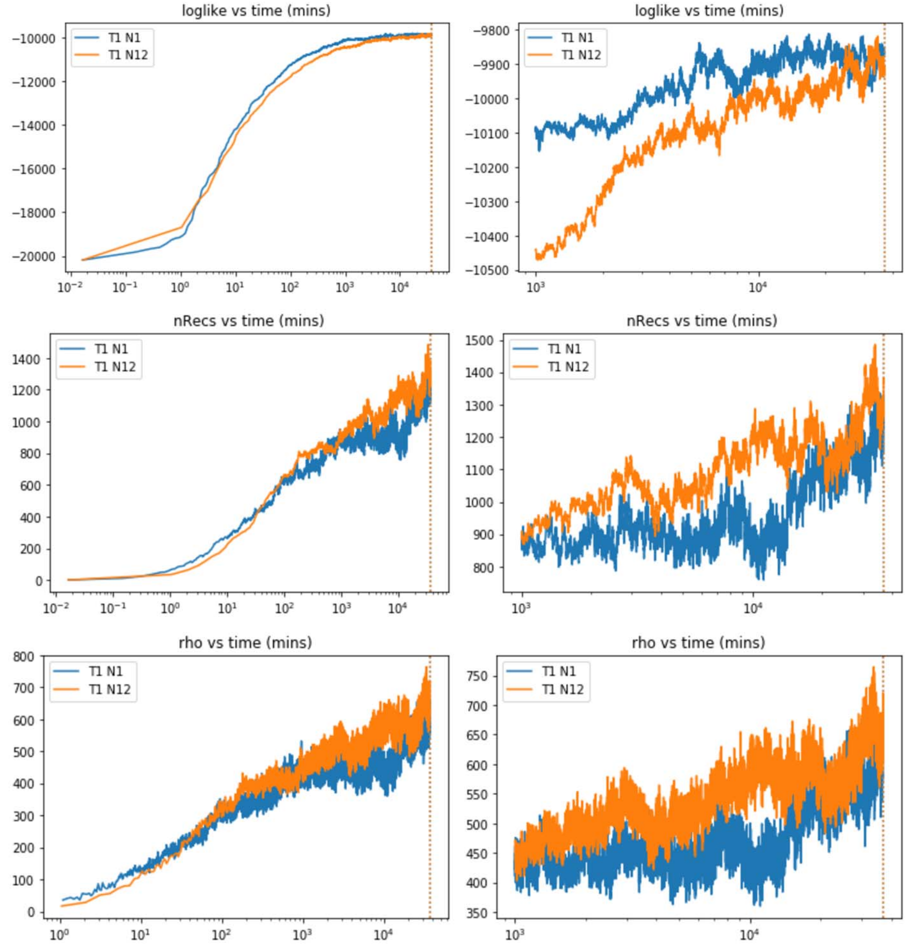


Figure 11: *E. coli* data on region of length 3.4Kb. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; bottom-left: trace plots for ρ; bottom-right: trace plots for δ.

recombination frequencies suggest that the chains when $N = 12$ and $N = 56$ provide similar answers but quite different to $N = 1$.

4 Discussion

In this study we have shown that the RmJMCMC algorithm has the potential of speeding up the convergence in the ClonalOrigin model (Didelot et al., 2010) towards a region of high likelihood, hence reducing the number of iterations and running time when compared to a standard RJMCMC. The version used in the examples presented

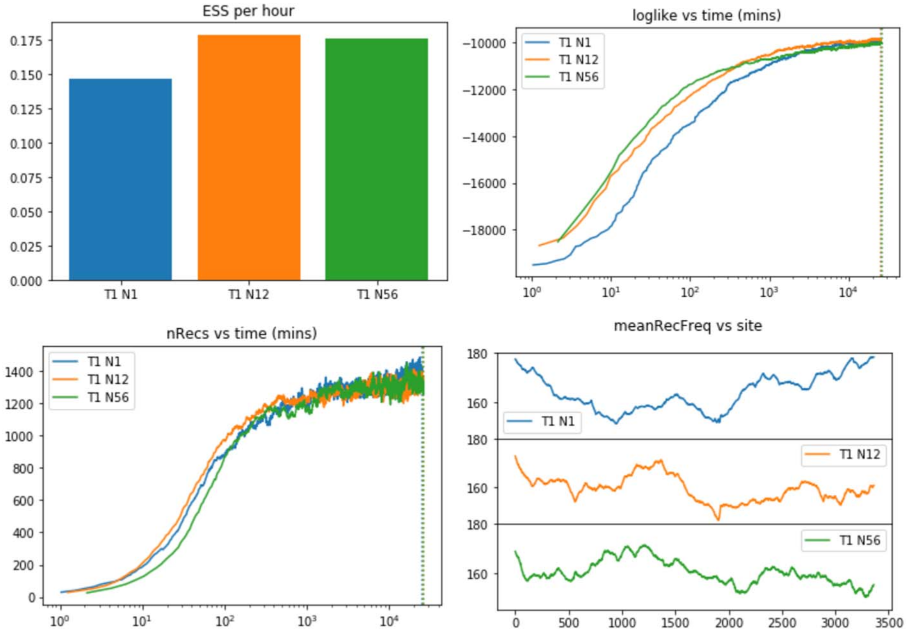


Figure 12: *E. coli* data on region of length 3.4Kb fixing $\rho = 670$ and $\delta = 542$. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; bottom-left: trace plots for the number of recombination events; bottom-right: comparison of mean recombination frequencies per site.

here is possibly the simplest that one could implement, namely using the prior distribution for proposing and perturbing recombination events. However, other alternatives are possible that may lead to better results, for example splitting or merging existing recombination events, or modifying the prior accordingly if recombination is more likely to happen in some parts of the tree (Ansari and Didelot, 2014) or in some specific regions of the genome (Yahara et al., 2014).

Similarly, different perturbations in the annealing step may be considered since in the presented examples there is no clear advantage when taking $T > 1$. Alternatives include random-walk style proposals for some (or all) of the variables involved in the proposed recombination event (a_i, b_i, x_i, y_i). In this respect, within the annealing process, one should also consider perturbing all of the existing recombination events and not just the recently proposed one. Finally, it is worth pointing out that the number of cores used in the presented examples was not particularly large; nonetheless we were able to observe significant improvements in some cases. It would be interesting to perform a much larger study where several hundred cores are at hand.

Here the RmJMCMC algorithm has been presented in the context of a model of bacterial recombination, but in principle it could be implemented to other models in population genetics where the dimension of the posterior is not fixed, for example the inference of demographic history (Opgen-Rhein et al., 2005), the estimation of transmission networks (Didelot et al., 2017) or the selection of a substitution site model (Bouckaert and Drummond, 2017).

Supplementary Material

Supplementary Material to "Speeding up Inference of 2 Homologous Recombination in Bacteria" (DOI: 10.1214/23-BA1388SUPP; .pdf).
Details on priors: Full details of prior distributions used on the full set of parameters.
Likelihood computation: Details on how to compute the likelihood function using the JC69 model.
RJMCMC as an importance estimator within MCMC: Brief explanation of how the acceptance ratio in the RJMCMC can be understood as an unbiased estimator of the ratio in the ideal algorithm.
Annealing Moves: Details of the upwards and downwards moves used in the main algorithm.

References

Alquier, P., Friel, N., Everitt, R., and Boland, A. (2016). "Noisy Monte Carlo: convergence of Markov chains with approximate transition kernels". *Statistics and Computing*, 26(1-2): 29–47. MR3439357. doi: `https://doi.org/10.1007/s11222-014-9521-x`. 9, 12

Andrieu, C., Doucet, A., Yıldırım, S., and Chopin, N. (2018). "On the utility of Metropolis-Hastings with asymmetric acceptance ratio". URL `http://arxiv.org/abs/1803.09527` 3, 5, 6, 9, 12

Andrieu, C., Yıldırım, S., Doucet, A., and Chopin, N. (2021). "Metropolis-Hastings with Averaged Acceptance Ratios". URL `https://arxiv.org/abs/2101.01253` 3

Ansari, M. A. and Didelot, X. (2014). "Inference of the Properties of the Recombination Process from Whole Bacterial Genomes". *Genetics*, 196: 253–265. 26

Bouckaert, R., Vaughan, T. G., Fourment, M., Gavryushkina, A., Heled, J., Denise, K., Maio, N. D., Matschiner, M., Ogilvie, H., Plessis, L., and Popinga, A. (2019). "BEAST 2.5: An Advanced Software Platform for Bayesian Evolutionary Analysis". *PLoS Comput. Biol.*, 15(4): e1006650. 2

Bouckaert, R. R. and Drummond, A. J. (2017). "bModelTest: Bayesian Phylogenetic Site Model Averaging and Model Comparison". *BMC Evolutionary Biology*, 17(1):42. MR2712977. 27

Brown, T., Didelot, X., Wilson, D. J., and De Maio, N. (2016). "SimBac: simulation of whole bacterial genomes with homologous recombination". *Microb. Genomics*, 2: 10.1099/mgen.0.000044. 1

Castillo-Ramírez, S., Harris, S. R., Holden, M. T. G., He, M., Parkhill, J., Bentley, S. D., and Feil, E. J. (2011). “The Impact of Recombination on dN/dS within Recently Emerged Bacterial Clones”. *PLoS Pathogens*, 7(7): e1002129. 2

Collins, C. and Didelot, X. (2018). “A phylogenetic method to perform genome-wide association studies in microbes that accounts for population structure and recombination”. *PLoS Computational Biology*, 14(2): e1005958. 1

De Maio, N. and Wilson, D. J. (2017). “The Bacterial Sequential Markov Coalescent”. *Genetics*, 206(1): 333–343. 2

Del Moral, P., Doucet, A., and Jasra, A. (2006). “Sequential Monte Carlo samplers”. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3): 411–436. MR2278333. doi: https://doi.org/10.1111/j.1467-9868.2006.00553.x. 2

Didelot, X., Croucher, N. J., Bentley, S. D., Harris, S. R., and Wilson, D. J. (2018). “Bayesian inference of ancestral dates on bacterial phylogenetic trees”. *Nucleic Acids Research*, 46: e134. 1

Didelot, X. and Falush, D. (2007). “Inference of bacterial microevolution using multilocus sequence data”. *Genetics*, 175(3): 1251–66. 2

Didelot, X., Fraser, C., Gardy, J., and Colijn, C. (2017). “Genomic Infectious Disease Epidemiology in Partially Sampled and Ongoing Outbreaks”. *Molecular Biology and Evolution*, 34: 997–1007. 27

Didelot, X., Lawson, D., Darling, A., and Falush, D. (2010). “Inference of homologous recombination in bacteria using whole-genome sequences”. *Genetics*, 186(4): 1435–1449. 2, 4, 9, 10, 14, 25

Didelot, X., Lawson, D. J., and Falush, D. (2009). “SimMLST: simulation of multi-locus sequence typing data under a neutral model”. *Bioinformatics*, 25(11): 1442–4. 1

Didelot, X. and Maiden, M. C. J. (2010). “Impact of recombination on bacterial evolution”. *Trends in Microbiology*, 18(7): 315–322. 2

Didelot, X., Méric, G., Falush, D., and Darling, A. E. (2012). “Impact of homologous and non-homologous recombination in the genomic evolution of Escherichia coli”. *BMC Genomics*, 13(1): 256. 22

Didelot, X. and Wilson, D. J. (2015). “ClonalFrameML: Efficient Inference of Recombination in Whole Bacterial Genomes”. *PLoS Computational Biology*, 11(2): e1004041. 2

Dingle, K. E., Elliott, B., Robinson, E., Griffiths, D., Eyre, D. W., Stoesser, N., Vaughan, A., Golubchik, T., Fawley, W. N., Wilcox, M. H., Peto, T. E. A., Walker, A. S., Riley, T. V., Crook, D. W., and Didelot, X. (2014). “Evolutionary History of the Clostridium difficile Pathogenicity Locus”. *Genome Biol. Evol.*, 6: 36–52. 2

Dinh, V., Darling, A. E., and Matsen, F. A. (2018). “Online Bayesian phylogenetic inference: Theoretical foundations via sequential Monte Carlo”. *Systematic Biology*. 2

Doucet, A., Freitas, N., and Gordon, N. (2001). “An Introduction to Sequential Monte Carlo Methods.” In *Sequential Monte Carlo Methods in Practice*, 3–14. New York, NY: Springer New York. MR1847784. doi: https://doi.org/10.1007/978-1-4757-3437-9_1. 2

Everitt, R. G., Culliford, R., Medina-Aguayo, F., and Wilson, D. J. (2019). “Sequential Monte Carlo with transformations”. *Statistics and Computing*. MR4065225. doi: https://doi.org/10.1007/s11222-019-09903-y. 2

Felsenstein, J. (1973). “Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters”. *Systematic Biology*, 22(3): 240–249. 4

Felsenstein, J. (1981). “Evolutionary trees from DNA sequences: A maximum likelihood approach”. *Journal of Molecular Evolution*, 17(6): 368–376. 4

Fourment, M., Claywell, B. C., Dinh, V., McCoy, C., Matsen IV, F. A., and Darling, A. E. (2018). “Effective online Bayesian phylogenetics via sequential Monte Carlo with guided proposals”. *Systematic Biology*, 67(3): 490–502. 2

Green, P. J. (1995). “Reversible jump Markov chain Monte Carlo computation and Bayesian model determination”. *Biometrika*, 82(4): 711–732. MR1380810. doi: https://doi.org/10.1093/biomet/82.4.711. 2, 5

Griffiths, R. C. (1996). “Ancestral inference from samples of DNA sequences with recombination”. *Journal of Computational Biology*. 1

Hedge, J. and Wilson, D. J. (2014). “Bacterial phylogenetic reconstruction from whole genomes is robust to recombination but demographic inference is not”. *MBio*, 5(6): 6–9. 1

Hudson, R. R. (1990). “Gene genealogies and the coalescent process”. In *Oxford Surveys in Evolutionary Biology*. 1

Jukes, T. H. and Cantor, C. R. (1969). “Evolution of protein molecules BT - Mammalian protein metabolism”. In *Mammalian protein metabolism*. 4

Karagiannis, G. and Andrieu, C. (2013). “Annealed Importance Sampling Reversible Jump MCMC Algorithms”. *Journal of Computational and Graphical Statistics*, 22(3): 623–648. MR3173734. doi: https://doi.org/10.1080/10618600.2013.805651. 5, 9, 10

Kingman, J. F. C. (1982). “The coalescent”. *Stochastic Processes and their Applications*. MR0671034. doi: https://doi.org/10.1016/0304-4149(82)90011-4. 3

Krause, D. J. and Whitaker, R. J. (2015). “Inferring speciation processes from patterns of natural variation in microbial genomes”. *Systems Biology*, 64(6): 926–935. 2

Marjoram, P. and Wall, J. D. (2006). “Fast “coalescent” simulation”. *BMC Genetics*. 2

McVean, G. A. and Cardin, N. J. (2005). “Approximating the coalescent with recombination". *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459): 1387–1393. 2

Medina-Aguayo, F. J., Didelot, X., and Everitt, R. G. (2023). "Supplementary Material for "Speeding up Inference of Homologous Recombination in Bacteria"." *Bayesian Analysis*. doi: https://doi.org/10.1214/23-BA1388SUPP. 3

Neal, R. M. (2001). "Annealed Importance Sampling". *Statistics and computing*, 11(2): 125–139. MR1837132. doi: https://doi.org/10.1023/A:1008923215028. 6

Ogundijo, O. E. and Wang, X. (2017). "A sequential Monte Carlo approach to gene expression deconvolution". *PLOS ONE*, 12(10): 1–31. 2

Opgen-Rhein, R., Fahrmeir, L., and Strimmer, K. (2005). "Inference of Demographic History from Genealogical Trees Using Reversible Jump Markov Chain Monte Carlo". *BMC Evolutionary Biology*, 5: 6. 27

Perron, G. G., Lee, A. E., Wang, Y., Huang, W. E., and Barraclough, T. G. (2012). "Bacterial recombination promotes the evolution of multi-drug-resistance in functionally diverse populations". *Proc. R. Soc. B Biol. Sci.*, 279(1733): 1477–1484. 2

Rasmussen, D. A., Volz, E. M., and Koelle, K. (2014). "Phylodynamic Inference for Structured Epidemiological Models". *PLOS Computational Biology*, 10(4): 1–16. 2

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer Texts in Statistics. New York, NY: Springer New York. MR2080278. doi: https://doi.org/10.1007/978-1-4757-4145-2. 15

Schierup, M. H. and Hein, J. (2000). "Consequences of recombination on traditional phylogenetic analysis". *Genetics*, 156(2): 879–91. 1

Sheppard, S. K., Didelot, X., Jolley, K. A., Darling, A. E., Pascoe, B., Meric, G., Kelly, D. J., Cody, A., Colles, F. M., Strachan, N. J. C., Ogden, I. D., Forbes, K., French, N. P., Carter, P., Miller, W. G., McCarthy, N. D., Owen, R., Litrup, E., Egholm, M., Affourtit, J. P., Bentley, S. D., Parkhill, J., Maiden, M. C. J., and Falush, D. (2013a). "Progressive genome-wide introgression in agricultural Campylobacter coli". *Molecular Ecology*, 22: 1051–1064. 2

Sheppard, S. K., Didelot, X., Meric, G., Torralbo, A., Jolley, K. A., Kelly, D. J., Bentley, S. D., Maiden, M. C. J., Parkhill, J., and Falush, D. (2013b). "Genome-wide association study identifies vitamin B5 biosynthesis as a host specificity factor in Campylobacter". *Proceedings of the National Academy of Sciences of the United States of America*, 110(29): 11923–7. 2

Smith, R., Ionides, E., and King, A. (2017). "Infectious Disease Dynamics Inferred from Genetic Data via Sequential Monte Carlo". *Molecular Biology and Evolution*, 34(8): 2065–2084. 2

Tavaré, S. (1986). "Some probabilistic and statistical problems in the analysis of DNA sequences". *Lectures on mathematics in the life sciences*, 17(2): 57–86. MR0846877. 4

Vaughan, T. G., Welch, D., Drummond, A. J., Biggs, P. J., George, T., and French, N. P. (2017). "Inferring Ancestral Recombination Graphs from Bacterial Genomic Data". *Genetics*, 205(2): 857–870. 2

Vos, M. and Didelot, X. (2009). "A comparison of homologous recombination rates in bacteria and archaea". *The ISME Journal*, 3(2): 199–208. 2

Wiuf, C. and Hein, J. (1999). "Recombination as a point process along sequences". *Theoretical Population Biology*. 3

Wiuf, C. and Hein, J. (2000). "The coalescent with gene conversion". *Genetics*, 155(1): 451–62. 1

Yahara, K., Didelot, X., Ansari, M. A., Sheppard, S. K., and Falush, D. (2014). "Efficient Inference of Recombination Hot Regions in Bacterial Genomes". *Molecular biology and evolution*, 31: 1593–605. 26

Acknowledgments

FJM-A and RGE were supported by the UK Biotechnology and Biological Sciences Research Council grant BB/N00874X/1. FJM-A also acknowledges support from an ONRG-RGCOMM grant, and partial funding from CONACYT CB-2016-01-284451 grant.