Electronic Journal of Statistics
Vol. 12 (2018) 3365–3396
ISSN: 1935-7524
https://doi.org/10.1214/18-EJS1418

Adaptive MCMC for multiple changepoint analysis with applications to large datasets

Alan Benson and Nial Friel

School of Mathematics and Statistics, University College Dublin and Insight Centre for Data Analytics
e-mail: alan.benson@insight-centre.org; nial.friel@insight-centre.org

Abstract: We consider the problem of Bayesian inference for changepoints where the number and position of the changepoints are both unknown. In particular, we consider product partition models where it is possible to integrate out model parameters for the regime between each changepoint, leaving a posterior distribution over a latent vector indicating the presence or not of a changepoint at each observation. The same problem setting has been considered by Fearnhead (2006) where one can use filtering recursions to make exact inference. However, the complexity of this filtering recursions algorithm is quadratic in the number of observations. Our approach relies on an adaptive Markov Chain Monte Carlo (MCMC) method for finite discrete state spaces. We develop an adaptive algorithm which can learn from the past states of the Markov chain in order to build proposal distributions which can quickly discover where changepoint are likely to be located. We prove that our algorithm leaves the posterior distribution ergodic. Crucially, we demonstrate that our adaptive MCMC algorithm is viable for large datasets for which the filtering recursions approach is not. Moreover, we show that inference is possible in a reasonable time thus making Bayesian changepoint detection computationally efficient.

Keywords and phrases: Adaptive MCMC, changepoint detection, large datasets.

Received March 2017.

Contents

1. Introduction

Changepoint problems arise in many practical instances in statistics, for example, signal processing, financial economics, process monitoring control and DNA sequence analysis. Here we consider chronologically ordered data over a period of time where it is suspected that there may have been some change(s) in the underlying generating process. For changepoints in parametric models, a parameter value (e.g. Gaussian mean or Gaussian precision) applicable to a certain time period may not extend well to another time period. Some examples include the rate of occurrences of coal mining disasters during the 18th and 19th century [15], gene expression sequences [9] and financial time series [3]. In this paper it is shown that analysis of multiple changepoint problems is feasible for larger datasets in a Bayesian setting using adaptive MCMC.

Markov Chain Monte Carlo methods (MCMC) can be used to estimate changepoint locations conditional on a *fixed* number of changepoints, Stephens [18] presents an MCMC method for this problem. When the number of changepoints is unknown, inference is more challenging. This is the problem which we address in this paper. A common approach for state-space dimension traversing is the reversible jump algorithm of Green [6] which performs trans-dimensional MCMC over a set of models, each incorporating a different number of changepoints. A drawback of this algorithm is that it can be difficult to design proposals so that the chain mixes well within and well between all available models. An alternative approach due to Chib [4] compares models with different numbers of changepoints using approximate Bayes Factors from the MCMC output in a post-processing step. The latter method requires MCMC model output for each number of changepoints under consideration.

Fearnhead [5] developed a clever forward-backward algorithm, filtering recursions, which allows one to sample exactly from the posterior distribution of changepoints. The filtering recursions share some similarity to product partition models [1]. The overwhelming advantage of this method is that once the filtering recursions have been calculated, it allows one to draw samples from the posterior using Carpenter's algorithm [2] that exploits the exponentially distributed spacing of order statistics in a uniform distribution.

However, a drawback of filtering recursions is that the algorithm requires a precomputation step to compute the recursions which has a time complexity that is quadratic in the number of observations and thus restricts the amount of data that can be used to perform efficient inference in a reasonable time. Fearnhead [5] offers a solution to this problem that lowers the precision of the recursions in order to make their calculation time approximately linear in the number of observations and the price to pay is it results in an approximate algorithm thereby.

Adaptive Markov Chain Monte Carlo Methods (AMCMC) have recently emerged in an attempt to improve the efficiency of MCMC algorithms. Typically adaptive MCMC uses *on-the-fly* refinement of the proposal distribution, taking information from the past history of the MCMC chain to yield a better mixing algorithm. The adaptive Metropolis algorithm of Haario et al. [8] was one of the earliest adaptive MCMC algorithms using a random walk Metropolis algorithm with an adapted covariance matrix. It is limited to continuous state spaces and to target distributions where a Gaussian proposal is suitable.

Adaptive MCMC methods on discrete state spaces have not yet been widely studied, yet these are very well suited to this methodology. This is because the design of adaptable proposals on discrete state spaces has the advantage that discrete state spaces carry the property of smallness, outlined in Meyn and Tweedie [14], so that simultaneous uniform ergodicity of the proposal kernels is guaranteed, provided that the state space is irreducible and the transition kernel is aperiodic. The second condition necessary for ergodicity of adaptive MCMC, diminishing adaptation, can be satisfied in many ways on a discrete state space leading to widely applicable methods in problems such as variable selection and Bayesian optimisation [12]. Griffin et al. [7] presents an adaptive MCMC algorithm on a discrete state space to carry out variable selection in a model choice setting.

In recent years, the emergence of big data across a vast range of models in statistics and machine learning has lead to the need for methods that can scale well to large datasets. We highlight how our adaptive changepoint approach scales well with an increasing number of observations and an increasing number of changepoints. The size of large datasets can present challenges for non-adaptive MCMC due to the presence of many local modes in the posterior distribution. We show empirically how our algorithm learns to move away from local modes which hinder MCMC.

The remainder of the paper is organised as follows. Section 2 describes multiple changepoint models in a Bayesian framework, Section 4 describes our adaptive changepoint sampler and introduces some advanced adaptation techniques which improve efficiency. We present a brief review in Section 3 of filtering recursions [5]. Section 5 provide a proof of our algorithm and results for three datasets are presented in Section 6 along with comparisons to filtering recursions.

All methods in this paper have been implemented in `C` using the Intel `C` compiler running on an Intel i7 3.40GhZ equipped machine with 16GB of RAM. Code is available on request from the authors.

2. Multiple changepoint models

Consider observed data $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, where observation y_i is observed before observation y_j, for $i < j$. We model $\boldsymbol{y}$ such that each observation y_i arises independently from a likelihood model depending on a parameter $\theta_i \in \Theta$ whose value may or may not change from one observation to the next. The points at which θ_i does change are called changepoints.

Consider the possibility of an unknown $k < n$ changepoints in $\boldsymbol{y}$ occurring at positions $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_k\}$. These changepoints partition $\boldsymbol{y}$ into $k+1$ contiguous non-overlapping segments

$$\{(y_1, y_{\tau_1}), (y_{\tau_1+1}, y_{\tau_2}), \ldots, (y_{\tau_k+1}, y_n)\}\,. \tag{2.1}$$

This partitioning of $\boldsymbol{y}$ can be represented by a fixed length latent changepoint indicator vector $\boldsymbol{z} = \{z_1, z_2, \ldots, z_{n-1}\}$ with $z_t = 1$ for each $t \in \boldsymbol{\tau}$ and $z_t = 0$ for each $t \notin \boldsymbol{\tau}$, with the number of changepoints satisfying $k = \sum_{i=1}^{n-1} z_i$. Within segment j, the likelihood has a constant parameter θ_j, $1 \leq j \leq k+1$. The full likelihood across all segments can be expressed as a product of $k+1$ segment likelihoods

$$f(\boldsymbol{y}|\theta_1, \theta_2, \ldots, \theta_{k+1}, \boldsymbol{z}) = \prod_{j=1}^{k+1} \prod_{i=\tau_{j-1}+1}^{\tau_j} f(y_i|\theta_j) \tag{2.2}$$

where $\tau_0 = 0, \tau_{k+1} = n$ and where $f(y_i|\theta_j)$ denotes the likelihood of observation y_i in a segment with parameter θ_j. In a Bayesian formulation the joint posterior distribution for the latent changepoint indicator vector $\boldsymbol{z}$ and segment parameters $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_{k+1}\}$ can be written as a product of the full segment likelihood (2.2) and the priors for $\boldsymbol{z}$ and $\boldsymbol{\theta}$,

$$\begin{aligned}\pi(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{y}) &\propto f(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{z})\pi(\boldsymbol{\theta}|\boldsymbol{z})\pi(\boldsymbol{z}) \\ &= \left(\prod_{j=1}^{k+1} \prod_{i=\tau_{j-1}+1}^{\tau_j} f(y_i|\theta_j)\right) \left(\prod_{j=1}^{k+1} \pi(\theta_j)\right) \pi(\boldsymbol{z}).\end{aligned} \tag{2.3}$$

The dependence of $\boldsymbol{\theta}$ on $\boldsymbol{z}$ is only through the prior multiplicity of $\boldsymbol{\theta}$ $(k+1)$ which sets the dimension of the prior term $\pi(\boldsymbol{\theta}|\boldsymbol{z})$. This shares some similarity to the hierarchical changepoint model used in Green [6] except that it does not condition on the number of changepoints and so this varies over the support of $\boldsymbol{z}$.

The prior for $\boldsymbol{z}$ specifies how the changepoint positions should be distributed prior to the data being observed. A convenient form that captures the gap lengths between changepoints is,

$$\pi(\boldsymbol{z}) = \pi(\tau_1, \dots \tau_k) = g_0(\tau_1)\left(\prod_{j=2}^{k} g(\tau_j - \tau_{j-1})\right)(1 - G(n - \tau_k)),$$

where $g_0(\cdot)$ is the distribution of the distance to the first changepoint, $g(\cdot)$ is the gap distribution for the distance between successive changepoints and $G(\cdot)$ is the cumulative distribution function for $g(\cdot)$. The choice for g can be a negative binomial or its special case, a geometric distribution

$$g(t) = \binom{t-1}{k-1} p^k (1-p)^{t-k}, \qquad g_0(t) = p(1-p)^{t-1}.$$

A more complex prior that minimises the *a priori* clustering of changepoints [6], is specified by the distribution of even order statistics of a draw of size $2k{+}1$ from $(1, \dots, n{-}1)$ without replacement. This prior prevents changepoints occurring at adjacent observations which minimises outliers (degenerate changepoints) being classified as true changepoints.

The priors for θ_j can be chosen to be conjugate to the likelihood, however this is not a requirement. The next section details the collapsing of the joint posterior (2.3) when the prior is conjugate but if it is possible to collapse the joint posterior using another method (e.g. quadrature) this is also feasible for use in our algorithm.

2.1. Collapsing multiple changepoints models

We assume that it is possible to integrate (collapse) out $\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots, \theta_{k+1}\}$ parameters from the posterior (2.3) to leave a discrete state space of changepoint positions. This is also the approach taken by Fearnhead [5]. With an appropriate conjugate prior for $\boldsymbol{\theta}$, the resulting posterior for $\boldsymbol{z}$ is

$$\begin{aligned}\pi(\boldsymbol{z}|\boldsymbol{y}) &\propto \int_{\boldsymbol{\theta}} f(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{z})\pi(\boldsymbol{\theta})\pi(\boldsymbol{z})\, d\boldsymbol{\theta} \\ &= \pi(\boldsymbol{z}) \prod_{j=1}^{k+1}\left(\int_{\theta_j} \prod\nolimits_{i=\tau_{j-1}+1}^{\tau_j} f(y_i|\theta_j)\pi(\theta_j)\, d\theta_j\right) \qquad (2.4) \\ &= \pi(\boldsymbol{z}) \prod_{j=1}^{k+1} \mathrm{P}(\tau_{j-1}+1, \tau_j),\end{aligned}$$

where $\mathrm{P}(\tau_{j-1}+1, \tau_j) = \int_{\theta_j} \prod_{i=\tau_{j-1}+1}^{\tau_j} f(y_i|\theta_j)\pi(\theta_j)\, d\theta_j$ denotes the evidence for segment $(y_{\tau_{j-1}+1}, y_{\tau_j})$. The evidence (marginal likelihood) is the probability of the data observed in that segment after the dependence on the parameter θ_j has been integrated out with respect to its prior. The dependence of $\boldsymbol{\theta}$ on $\boldsymbol{z}$ has been removed; the position of changepoints and the within segment parameter are assumed independent.

2.1.1. *A simple example of collapsing – Poisson Gamma*

Consider the case where the data in segment j can be modelled by a Poisson distribution with parameter $\theta_j > 0$. Placing a Gamma(α, β) prior on each θ_j and integrating out θ_j for $j \in \{1, \dots, k+1\}$, the marginal likelihood for segment (y_a, y_b) is,

$$\begin{aligned} \mathrm{P}(a,b) &= \int_0^\infty \frac{\beta^\alpha}{\Gamma(\alpha)} \theta_j^{\alpha-1} e^{-\alpha\theta_j} \prod_{i=a}^{b} \frac{\theta_j^{y_i}}{y_i!} e^{-\theta_j}\, d\theta_j \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{1}{F_{a:b}} \frac{\Gamma(S_{a:b}+\alpha)}{(b-a+1+\beta)^{S_{a:b}+\alpha}}, \end{aligned} \tag{2.5}$$

where

$$\mathrm{F}_{a:b} = \prod\nolimits_{i=a}^{b} y_i! \qquad \text{and} \qquad \mathrm{S}_{a:b} = \sum\nolimits_{i=a}^{b} y_i.$$

Precomputation of $\mathrm{F}_{1:t}$ and $\mathrm{S}_{1:t}$ for $1 \le t \le n$ and using the following recursions

$$\mathrm{F}_{a:b} = \frac{\mathrm{F}_{1:b}}{\mathrm{F}_{1:a-1}} \qquad \text{and} \qquad \mathrm{S}_{a:b} = \mathrm{S}_{1:b} - \mathrm{S}_{1:a-1},$$

negates the need to store each individual y_i for computation of the marginal likelihood (2.5) which is required to be computed many times in filtering recursions and in our algorithm. Similar precomputations are available for other likelihood models, see Appendix B for the Gaussian distribution mean and precision.

3. Change point inference using filtering recursions

Fearnhead [5] provides a filtering recursions approach to inferring changepoint positions. Barry and Hartigan [1] have also used these type of recursive methods for analysis of changepoint problems. We give a brief recap of the filtering recursions method and we will use the method as a comparison to our adaptive changepoint sampler. Some drawbacks of the filtering recursions will also be discussed.

Define for $t = 2, \dots, n$

$$Q(t) = \mathbf{P}(y_t, \dots y_n | \text{changepoint at } t-1)$$

and $Q(1) = \mathbf{P}(y_1, \dots y_n)$. Fearnhead [5] provides a backward recursion for $Q(t)$ as follows, using the marginal likelihood $\mathrm{P}(a, b)$ in (2.4),

$$Q(t) = \left(\sum_{i=t}^{n-1} g(i-t+1)\mathrm{P}(t,i)Q(i+1) \right) + \mathrm{P}(t,n)(1 - G(n-t)).$$

The function $g(\cdot)$ is the gap length distribution between changepoints (for example, geometric) and $G(\cdot)$ is its cumulative distribution function.

Once the $Q(t)$ values have been calculated (normally on the log scale) it is possible to draw samples of size N from the posterior distribution of positions as follows:

1. Initialise all N samples to have a changepoint at $t = 0$, i.e. $\tau_0 = 0$
2. For $t = 0, \dots, n-2$,
   (a) Find n_t, the number of samples for which the last changepoint was at time t.
   (b) If $n_t > 0$, compute the probability distribution for the next changepoint
   $$\mathbf{P}(\tau_j|\tau_{j-1}) = \mathrm{P}(\tau_{j-1}+1, \tau_j)Q(\tau_j+1)g(\tau_j - \tau_{j-1})/Q(\tau_{j-1}+1). \quad (3.1)$$
   (c) Sample n_t times, using Carpenter's algorithm (see Appendix D for details), from $\mathbf{P}(\tau_j|\tau_{j-1})$ and update the n_t samples using a random permutation of the n_t samples.

The filtering recursions approach has the advantage that the design of the method allows one to draw independently from the posterior distribution. Moreover Carpenter's algorithm for sampling the changepoints is fast. This method however has some drawbacks which arise as the dataset increases in size. Firstly, the calculation of the $Q(t)$ values is $\mathcal{O}(n^2)$ as the recursion for each possible ordered pair of points $(i < j)$ must be computed before perfect simulation can begin. This calculation time can be reduced by truncating the $Q(t)$ sums once they fail to grow by a certain amount, Fearnhead [5] suggests 10×10^{-10} and we compare various truncation levels in the results section. The price to pay for this reduced run time is that the truncation introduces an approximation to the recursion algorithm. Secondly, hyperparameters must remain fixed throughout the algorithm as a change in hyperparameters or indeed the inclusion of a hyperprior would require complete recalculation of $Q(t)$. Thirdly, for larger datasets (300 000 observations for the largest example considered in this paper) the transition probabilities in (3.1) have the potential to become numerically unstable, as we outline in Section 6.4.1. We suggest using the exact algorithm, where possible. However for larger (> 100 000 observations) datasets we advocate the use of our adaptive changepoint sampler as it is much more stable, by comparison.

4. Adaptive MCMC changepoint sampler

We now introduce our adaptive MCMC changepoint sampler to sample from the posterior distribution of changepoints (2.4). Wyse and Friel [21] developed an MCMC scheme based on adding and deleting changepoints using samplers similar to those used by Lavielle and Lebarbier [11]. The algorithm of Wyse and Friel [21] turns out to be a special case of our adaptive algorithm when no adaptation occurs and as we shall see, the adaptive MCMC algorithm we develop offers an improvement in efficiency, by comparison.

Sampling over $\boldsymbol{z}$ is a challenging problem as the size of the space scales exponentially with n, leaving brute force enumeration of all $\boldsymbol{z}$ intractable. However, for datasets with few changepoints ($k \ll n$) the realised $\boldsymbol{z}$ vectors will be quite sparse. The design of our algorithm motivates searching element-wise through $\boldsymbol{z}$ identifying which elements (positions) are likely changepoints and those which are not. Positions which are deemed unlikely to be changepoints will tend not to be proposed as changepoint locations and conversely locations which are identified as being locations of changepoints will tend to be proposed more frequently. In this way, our algorithm will facilitate proposed moves to centre around areas of high changepoint activity and move away from areas of low changepoint activity. As we will shortly see, this adaptive algorithm where proposed changepoint locations change over time will by design preserve the ergodicity of the adaptive Markov chain.

We now describe the adaptive algorithm in detail and defer a proof of ergodicity to Section 5.

4.1. Detailed description

At iteration t denote the current state of changepoint locations as $\boldsymbol{z}^{(t)}$. Our algorithm consists of three proposal moves to update the vector $\boldsymbol{z}^{(t)}$. The three proposal moves involve adding a new changepoint to $\boldsymbol{z}^{(t)}$ (*add move*) and deleting a changepoint from $\boldsymbol{z}^{(t)}$ (*delete move*). At each iteration t, one of either the *add move* or the *delete move* is selected with probability p and $1-p$, respectively.

The space of all realisable $\boldsymbol{z}$ vectors is large, having 2^{n-1} elements. It is important therefore to add changepoints in locations of high posterior changepoint probability and delete changepoints in areas of low posterior changepoint probability. Adaptively learning these areas *on-the-fly* provides a route to a scalable inferential framework for large datasets, as we now illustrate.

We associate with $\boldsymbol{z}^{(t)}$ two iteration dependent selection weight vectors $\boldsymbol{a}^{(t)} = \{a_1^{(t)} \ldots, a_{n-1}^{(t)}\}$ and $\boldsymbol{d}^{(t)} = \{d_1^{(t)} \ldots, d_{n-1}^{(t)}\}$. We remark that these weights correspond to how often the algorithm should pick a particular point. This is different to the approach of Griffin et al. [7] where the vectors are used as inclusion probabilities for variable selection. If a changepoint is proposed to be added, a position i (having $z_i^{(t)} = 0$) will be selected as the add position with probability $a_i^{(t)} / \sum_{\{j,z_j=0\}} a_j^{(t)}$. If a changepoint is proposed to be deleted, some position i (having $z_i^{(t)} = 1$) will be selected as the deletion position with probability $d_i^{(t)} / \sum_{\{j,z_j=1\}} d_j^{(t)}$. If the relevant add or delete move is accepted then the selected element i of $\boldsymbol{z}^{(t+1)}$ will be toggled, otherwise $\boldsymbol{z}^{(t+1)}$ does not change from $\boldsymbol{z}^{(t)}$.

The probability of accepting or rejecting the moves described above will depend on the relative change in the marginal likelihood of the segment added or deleted around position i. Let a be the changepoint immediately before i and b the changepoint immediately after i. The addition of a changepoint at position i would cause the segment that contains position i to be split into two new segments (y_{a+1}, y_i) and (y_{i+1}, y_b). The deletion of a changepoint at position i would cause the two segments created by the changepoint at i to merge into one segment (y_{a+1}, y_b). All other segments remain the same. The marginal likelihood ratios are thus

$$\text{Add Move} \rightarrow \frac{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)}{\mathrm{P}(a+1,b)} \qquad \text{Delete Move} \rightarrow \frac{\mathrm{P}(a+1,b)}{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)}. \tag{4.1}$$

The two moves are summarised clearly in Figure 1.

Move 4.1: Add a Changepoint

```
Calculate $\boldsymbol{a}_+^{(t)} = \sum_{\{j,z_j=0\}} \boldsymbol{a}_j^{(t)}$ and
    $\boldsymbol{d}_+^{(t)} = \sum_{\{j,z_j=1\}} \boldsymbol{d}_j^{(t)}$.
Select $i$ with $z_i = 0$ with prob. $\boldsymbol{a}_i^{(t)}/\boldsymbol{a}_+^{(t)}$.
Accept to toggle $z_i = 1 - z_i$ with probability
    $\min(1, \alpha_{\text{add}})$, where
    $\alpha_{\text{add}} = \frac{\pi(\boldsymbol{z}')}{\pi(\boldsymbol{z})} \frac{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)}{\mathrm{P}(a+1,b)} \frac{1-p}{p} \frac{\boldsymbol{d}_i^{(t)}/(\boldsymbol{d}_i^{(t)}+\boldsymbol{d}_+^{(t)})}{\boldsymbol{a}_i^{(t)}/\boldsymbol{a}_+^{(t)}}.$
```

Move 4.2: Delete a Changepoint

```
Calculate $\boldsymbol{d}_+^{(t)} = \sum_{\{j,z_j=1\}} \boldsymbol{d}_j$ and
    $\boldsymbol{a}_+^{(t)} = \sum_{\{j,z_j=0\}} \boldsymbol{a}_j^{(t)}$.
Select $i$ with $z_i = 1$ with prob. $\boldsymbol{d}_i^{(t)}/\boldsymbol{d}_+^{(t)}$.
Accept to toggle $z_i = 1 - z_i$ with probability
    $\min(1, \alpha_{\text{del}})$, where
    $\alpha_{\text{del}} = \frac{\pi(\boldsymbol{z}')}{\pi(\boldsymbol{z})} \frac{\mathrm{P}(a+1,b)}{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)} \frac{p}{1-p} \frac{\boldsymbol{a}_i^{(t)}/(\boldsymbol{a}_i^{(t)}+\boldsymbol{a}_+^{(t)})}{\boldsymbol{d}_i^{(t)}/\boldsymbol{d}_+^{(t)}}.$
```

FIG 1. *Adaptive MCMC changepoint sampler moves, the add move is performed with probability* p *and the delete move with probability* $1 - p$.

This is the basis of our changepoint sampler. We are now left to describe the adaptation scheme used to update the $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ vectors during the algorithm, using the past history of the add and selected moves. This is a crucial part of the algorithm as these parameters decide where to place changepoints and remove changepoints in an efficient manner. This is described in the following section.

4.2. Adaptation of the selection weights $\boldsymbol{a}^{(t)}$ *and* $\boldsymbol{d}^{(t)}$

The MCMC algorithm of Wyse and Friel [21] selects positions i for addition and deletion uniformly at random from all the valid $n-1$ positions. This is equivalent to having constant vectors $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ which do not vary with iteration t. The adaptive method we use, proposes to update $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ using information from previously accepted *add* and *delete* moves. The scheme for adaptation is given in Figure 2. The strategy is to target the acceptance rate of the *add* and *delete* moves to an overall target acceptance rate α_{target} by updating the $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ at each iteration. The updates are performed on the log scale to ensure that the weights remain positive.

Adaptation Scheme
At iteration t:

1. If an *add* move at point i has been accepted then update only the $a_i^{(t)}$ parameter as follows
$$\log(a_i^{(t+1)}) = \log(a_i^{(t)}) + \frac{h}{t/n}\left(\alpha_{\text{add}} - \alpha_{\text{target}}\right).$$
2. If a *delete* move at point i has been accepted then update only the $d_i^{(t)}$ parameter as follows
$$\log(d_i^{(t+1)}) = \log(d_i^{(t)}) + \frac{h}{t/n}\left(\alpha_{\text{del}} - \alpha_{\text{target}}\right).$$

Parameters
h - Initial Adaptation ($h > 0$)
t/n - Monte Carlo time, iterations (t) per number of datapoints (n)

Fig 2. *Adaptation scheme to update the vectors* $\boldsymbol{a}^{(t)}$ *and* $\boldsymbol{d}^{(t)}$.

This adaptation scheme is different from Griffin et al. [7] in that there is no restriction on $0 < \boldsymbol{a}_i < 1$ or $0 < \boldsymbol{d}_i < 1$ as these are unnormalised selection weights and not probabilities. The parameter h controls the initial intensity of the adaptation, we find values $<< 1$ work well.

4.2.1. A note on non uniform sampling for selection weights

The $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ weights, once normalised appropriately using $\boldsymbol{a}_+^{(t)}$ and $\boldsymbol{d}_+^{(t)}$ (see Figure 1), must be sampled from to propose elements of $\boldsymbol{z}^{(t)}$ for toggling. Discrete random variate generation for non-uniform probability vectors presents an extra level of complexity. In the case of Wyse and Friel [21] with no adaptation, selection of elements for toggling is $\mathcal{O}(1)$ and is extremely efficient. To take advantage of the adaptive proposals the algorithm requires an efficient non-uniform sampler.

A naïve implementation of non-uniform sampling from the $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ vectors involves building a cumulative distribution of the values, $\mathcal{O}(n)$ time, and then sampling from this by binary lookup, $\mathcal{O}(\log_2 n)$ time. This is significantly slower and may even detriment the use of the adaptive algorithm in the first instance. A method due to Walker [20] overcomes this problem by precomputing lookup tables called alias tables in $\mathcal{O}(n)$ time and then sampling in $\mathcal{O}(1)$ time. A numerically stable implementation of Walker's method that overcomes numerical errors is due to Vose [19]. A discussion of the alias method is given in Appendix C.

Using alias tables we can get quite close to uniform sampling efficiency. Note that Matias et al. [13] allows updating alias tables in less than $\mathcal{O}(n)$ time, however this imposes restrictions on the magnitude of the change in weights at each adaptation step.

4.3. Advanced adaptation techniques

In this section some advanced techniques are presented to improve the efficiency of the adaptive method. It is possible to implement thresholding of the $\boldsymbol{a}^{(t)}$ values so that only some of the values use alias tables. Dual adaptation is used by Griffin et al. [7] to simultaneously update $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ after an accepted move. We modify this to our adaptation scheme. These advanced techniques allow the algorithm to be computationally efficient while performing the adaptive updates. Many issues with adaptive MCMC can arise due to adapting too quickly. These issues are discussed in Łatuszyński and Rosenthal [10].

4.3.1. Advanced adaptation 1: Thresholding of non-changepoints

Many of the $\boldsymbol{a}_i$ values won't significantly change in magnitude over the course of the algorithm. This is due to the update of the $\boldsymbol{a}_i$ values only being performed on acceptance of a changepoint and for points far away from changepoints the $\boldsymbol{a}_i$ will rarely change. Computational time is still spent embedding these small $\boldsymbol{a}_i$ in the rebuilding of alias tables each time any $\boldsymbol{a}_i$ changes. This problem is not as pronounced for the $\boldsymbol{d}_i$ values as we assume that there are many more non-changepoints than changepoints in a dataset.

To take advantage of the low number of changepoints, we propose to split the points that are not changepoints into two groups, one with high posterior probability of being added, G_{active}, and the other with a low posterior probability of being added, G_{inactive}. The membership of each group is mutually exclusive and is determined by a threshold parameter $\boldsymbol{a}_{\text{cutoff}}$. All points begin in G_{inactive} and as the $\boldsymbol{a}_i$ values are adapted, points with $\boldsymbol{a}_i > \boldsymbol{a}_{\text{cutoff}}$ move to G_{active}. The other points remain in G_{inactive} and are assumed to have a flat weight of $\boldsymbol{a}_{\text{inactive}} < \boldsymbol{a}_{\text{cutoff}}$ which means they can be sampled without the use of alias tables (equivalent to uniform sampling within G_{inactive}). Each element of G_{inactive} will retain its true underlying $\boldsymbol{a}_i$ value but this will only be used for sampling if and when it moves into G_{active}. The thresholding will modify the algorithm slightly and the modifications to the acceptance probabilities are shown in Figure 3.

4.3.2. Advanced adaptation 2: Dual adaptation

As can be seen in the description of the moves, knowledge of α_{add} allows one to also calculate α_{del} quite easily. Griffin et al. [7] uses this idea to perform a double or dual adaptation of both $\boldsymbol{a}^{(t)}$ and $\boldsymbol{d}^{(t)}$ at each acceptance in the algorithm

Move 4.3: Add (with threshold)

```
Calculate $\boldsymbol{a}^{(t)}_{\text{active}} = \sum_{\{j|z_j=0, j \in G_{\text{active}}\}} \boldsymbol{a}^{(t)}_j$ and
    $\boldsymbol{d}^{(t)}_{+} = \sum_{\{j, z_j=1\}} \boldsymbol{d}^{(t)}_j$
Select $i$ with $z_i = 0$ with prob. $\boldsymbol{a}^{(t)}_i / \boldsymbol{a}^{(t)}_{+}$
Accept to toggle $z_i = 1 - z_i$ with probability
    $\alpha_{\text{add}} = \min(1, \hat{\alpha}_{\text{add}})$

        $\hat{\alpha}_{\text{add}} = \frac{\pi(\boldsymbol{z}')}{\pi(\boldsymbol{z})} \frac{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)}{\mathrm{P}(a+1,b)} \frac{1-p}{p} \times \frac{\boldsymbol{d}^{(t)}_i / (\boldsymbol{d}^{(t)}_i + \boldsymbol{d}^{(t)}_{+})}{\widehat{\boldsymbol{a}}^{(t)}_i / (\boldsymbol{a}^{(t)}_{\text{active}} + \boldsymbol{a}_{\text{inactive}} |G_{\text{inactive}}|)}$

    where $\widehat{\boldsymbol{a}}^{(t)}_i = \boldsymbol{a}^{(t)}_i$ if $i \in G_{\text{active}}$ or
    $\widehat{\boldsymbol{a}}^{(t)}_i = \boldsymbol{a}_{\text{inactive}}$ otherwise.
```

Move 4.4: Delete (with threshold)

```
Calculate $\boldsymbol{d}^{(t)}_{+} = \sum_{\{j, z_j=1\}} \boldsymbol{d}^{(t)}_j$ and
    $\boldsymbol{a}^{(t)}_{\text{active}} = \sum_{\{j|z_j=0, j \in G_{\text{active}}\}} \boldsymbol{a}^{(t)}_j$
Select $i$ with $z_i = 1$ with prob. $\boldsymbol{d}^{(t)}_i / \boldsymbol{d}^{(t)}_{+}$
Accept to toggle $z_i = 1 - z_i$ with probability
    $\alpha_{\text{add}} = \min(1, \hat{\alpha}_{\text{add}})$

        $\alpha_{\text{del}} = \frac{\pi(\boldsymbol{z}')}{\pi(\boldsymbol{z})} \frac{\mathrm{P}(a+1,b)}{\mathrm{P}(a+1,i)\mathrm{P}(i+1,b)} \frac{p}{1-p} \times \frac{\widehat{\boldsymbol{a}}^{(t)}_i / (\widehat{\boldsymbol{a}}^{(t)}_i + \boldsymbol{a}^{(t)}_{\text{active}} + \boldsymbol{a}_{\text{inactive}} |G_{\text{inactive}}|)}{\boldsymbol{d}^{(t)}_i / (\boldsymbol{d}^{(t)}_{+})}$

    where $\widehat{\boldsymbol{a}}^{(t)}_i = \boldsymbol{a}^{(t)}_i$ if $i \in G_{\text{active}}$ or
    $\widehat{\boldsymbol{a}}^{(t)}_i = \boldsymbol{a}_{\text{inactive}}$ otherwise.
```

FIG 3. *Adjusted moves for use with thresholding of $\boldsymbol{a}_i$ values. Note that $|G_{inactive}|$ denotes the cardinality of the inactive set.*

rather than updating only one of these vectors. The dual adaptation approach is applied without thresholding to the updates in Figure 2 and is described in Appendix E.

5. Proof of ergodicity for the adaptive MCMC algorithm

There are two parts to proving ergodicity for an adaptive MCMC algorithm on a discrete state space $\mathcal{X}$. The first establishes the notion of simultaneous uniform ergodicity and the second establishes diminishing adaptation. An adaptive MCMC algorithm which satisfies both of these conditions is ergodic by Theorem 1 of Rosenthal and Roberts [16].

5.1. Simultaneous uniform ergodicity

We first recap the definition of uniform ergodicity for a Markov chain, the equivalent Doeblin's condition and simultaneous uniform ergodicity for transition kernels on a state space $\mathcal{X}$.

Definition 5.1. (Uniform ergodicity) A Markov chain on a state space $\mathcal{X}$ with a transition kernel $P(x, \cdot)$ is called *uniformly ergodic* if

$$\sup_{x \in \mathcal{X}} \|P^n(x, \cdot) - \pi()\|_{\text{TV}} \to 0 \text{ as } n \to \infty.$$

where $\|\cdot\|_{\text{TV}}$ is the total variation norm.

An equivalent definition by Theorem 16.0.2 [14] states that there exists some $r > 1$ and $R < \infty$ such that $\forall x \in \mathcal{X}$

$$\|P^n(x, \cdot) - \pi()\|_{\text{TV}} \leq Rr^{-n}.$$

This implies that the convergence takes place at a geometric rate independent of the starting point $x_0 \in \mathcal{X}$ of the algorithm.

Uniform ergodicity is generally difficult to prove directly using Definition 5.1. Instead uniform ergodicity can be more easily checked by equivalence to Doeblin's condition on $\mathcal{X}$. This equivalence is shown in Theorem 16.0.2 of Meyn and Tweedie [14] and is repeated here.

Theorem 5.1. *(Doeblin's Condition) Suppose that Doeblin's Condition holds (as defined in Meyn and Tweedie [14, p 396]) so that there exists a probability measure ϕ on the measurable space $(\mathcal{X}, \sigma\{\mathcal{X}\})$ with the property that for some m, some constant measure $\rho < 1$, some $\beta > 0$ and for a set $A \in \sigma\{\mathcal{X}\}$*

$$\phi(A) > \rho \Longrightarrow P^m(x, A) > \beta$$

*then the chain under transition kernel $P^m(x, \cdot)$ is **uniformly ergodic**.*

Proof. See Theorem 16.2.3 of Meyn and Tweedie [14] and relevant lemmas. □

Finally [16] define the notion of simultaneous uniform ergodicity for a collection of transition kernels indexed by $\gamma \in \Gamma$. This definition is repeated here.

Definition 5.2. (Simultaneous Uniform Ergodicity) A collection of transition kernels indexed by $\gamma \in \Gamma$ exhibit simultaneous uniform ergodicity if $\forall\, \gamma \in \Gamma$ and $\forall\, x \in X$

$$\left\|P_\gamma^n(x, \cdot) - \pi()\right\|_{\text{TV}} \leq R_\gamma r_\gamma^{-n}, \text{ where } R_\gamma < \infty \text{ and } r_\gamma > 1 \text{ for all } \gamma \in \Gamma$$

where $\|\cdot\|_{\text{TV}}$ is the total variation norm.

Remarks. *The uniform ergodicity parameters R_γ and r_γ for each kernel may depend on γ but not on the states $x \in \mathcal{X}$ as otherwise uniform ergodicity would not hold.*

Verifying multiple Doeblin's Conditions is equivalent to verifying uniform ergodicity for all kernels $P^n_\gamma(x,\cdot)$, $\gamma \in \Gamma$. This in turn guarantees simultaneous uniform ergodicity. We will now prove simultaneous uniform ergodicity for the adaptive changepoint sampler.

Theorem 5.2. *(Simultaneous uniform ergodicity of the adaptive changepoint sampler) Let $\mathbf{\Gamma}^{(t)} = (\boldsymbol{a}^{(t)}, \boldsymbol{d}^{(t)})$ be the set of adaptive weights at iteration t and let $\boldsymbol{z}^{(t)}$ be the current state of the chain. Then for all t the transition kernel using the weights $\mathbf{\Gamma}^{(t)}$, $P_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)},\cdot)$ is uniformly ergodic.*

Proof. As we are working over a finite discrete state space, $\boldsymbol{Z}$, the transition kernel of our changepoint sampler, $P_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$, can be viewed as a 1-step transition probability from state $\boldsymbol{z}^{(t)}$ to $\boldsymbol{z}'$. Take the measure $\phi(\boldsymbol{z})$ in Doeblin's Condition to be the posterior distribution over $\boldsymbol{z} \in \boldsymbol{Z}$, $\pi(\boldsymbol{z}|\boldsymbol{y})$. The measure $\phi(\boldsymbol{z})$ is always positive since the prior for $\boldsymbol{z}$ allows for all 2^{n-1} values of $\boldsymbol{z}$ to occur with non-zero probability. Doeblin's condition on a finite state space amounts to showing that for some m, the mth power of the transition probability matrix $P^m_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$ has all positive entries for all states $\boldsymbol{z}^{(t)}, \boldsymbol{z}' \in \boldsymbol{Z}$, i.e. that it is possible to transition, with non-zero probability, to any state from any other state in m moves. This is equivalent to showing that the Markov chain over $\boldsymbol{Z}$ is both *irreducible* and *aperiodic*, which we now establish.

Denote the overall minimum value of any element of $\boldsymbol{a}^{(t)}$ or $\boldsymbol{d}^{(t)}$ by $\epsilon > 0$. The existence of a minimum follows from the adaptation scheme in Figure 2, where it is not possible for any $\boldsymbol{a}^{(t)}$ or $\boldsymbol{d}^{(t)}$ to reach 0 when started from a positive value. Define a distance function $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$ which counts the number of positions at which $\boldsymbol{z}^{(t)}$ and $\boldsymbol{z}'$ differ, e.g. $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$ for the case of a proposed add/delete move.

The 1-step transition probability of our algorithm from state $\boldsymbol{z}^{(t)}$ to $\boldsymbol{z}'$ using the add/delete proposal distribution $q_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$ is written,

$$\begin{aligned} P^1_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') &= q_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')\alpha_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') \\ &\quad + \delta_{\boldsymbol{z}^{(t)}}(\boldsymbol{z}')\left(1 - \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}^{(t)}} q_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*)\alpha_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*)\right). \end{aligned} \tag{5.1}$$

The proposal distribution $q_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$ is positive if and only if $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$. To establish *irreducibility* and *aperiodicity* across the entire state space, we must consider 3 possible cases for $\boldsymbol{z}'$. These are $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$, $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') > 1$ and $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 0$, where the first two cases will establish *irreducibility* and the final case establishes *aperiodicity*.

For $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$, the proposal distribution of add/delete moves, $q_{\mathbf{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$, is lower bounded by $\dfrac{\epsilon}{\omega^{(t)}} > 0$, where $\omega^{(t)}$ normalises the $\boldsymbol{a}^{(t)}$ or $\boldsymbol{d}^{(t)}$ weights depending on which of the add or delete move is taking place. This implies that the 1-step transition probability also has a lower bound since

$$\begin{aligned}
P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') &\geq \frac{\epsilon}{\omega^{(t)}} \alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') \\
&= \frac{\epsilon}{\omega^{(t)}} \min\left\{1, \frac{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}', \boldsymbol{z}^{(t)})}{\pi(\boldsymbol{z}^{(t)}|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')}\right\} \\
&= \frac{\epsilon}{\omega^{(t)}} \pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}', \boldsymbol{z}^{(t)}) \\
&\quad \times \min\left\{\frac{1}{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}', \boldsymbol{z}^{(t)})}, \frac{1}{\pi(\boldsymbol{z}^{(t)}|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}')}\right\} \\
&\geq \frac{\epsilon}{\omega^{(t)}} \pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}', \boldsymbol{z}^{(t)}).
\end{aligned}$$

This inequality holds since $\pi(\boldsymbol{z}^{(t)}|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') < 1$ and $\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}', \boldsymbol{z}^{(t)}) < 1$. Finally

$$\begin{aligned}
P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') &\geq \frac{\epsilon}{\omega^{(t)}} \min_{\boldsymbol{z}} \pi(\boldsymbol{z}|\boldsymbol{y}) \left(\frac{\epsilon}{\omega^{(t)}}\right) \\
&= \left(\frac{\epsilon}{\omega^{(t)}}\right)^2 \min_{\boldsymbol{z}} \pi(\boldsymbol{z}|\boldsymbol{y}) > 0.
\end{aligned}$$

Therefore the 1-step transition probability is positive for states $\boldsymbol{z}^{(t)}$ and $\boldsymbol{z}'$ when $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$.

For $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') > 1$, the 1-step transition probability $P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \cdot)$ must be iterated $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}')$ times to ensure $P^{d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}')}_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') > 0$. The maximum distance between any two states in $\boldsymbol{Z}$ is $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = n - 1$ which occurs when a transition between $\boldsymbol{z}^{(t)}$, the vector with all entries set to 0, and $\boldsymbol{z}'$, the vector with all entries set to 1, takes place. Thus iterating the kernel at least $n - 1$ times ensures *irreducibility* of the Markov chain on $\boldsymbol{Z}$.

The final case to consider is when $d_H(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 0$, i.e. a transition from $\boldsymbol{z}^{(t)}$ to itself. If $P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^{(t)}) > 0$ then the state $\boldsymbol{z}^{(t)}$ is said to be *aperiodic*. Any *irreducible* Markov chain on a finite state space is *aperiodic* provided there exists at least one *aperiodic* state. We now show that at least one *aperiodic* state exists in $\boldsymbol{Z}$. Consider the 1-step transition probability from $\boldsymbol{z}^{(t)}$ to itself which from (5.1) is,

$$P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^{(t)}) = \left(1 - \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}^{(t)}} q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*) \alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*)\right).$$

$P^1_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^{(t)})$ will be strictly positive provided we can find some state $\boldsymbol{z}^*$ such that $q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*) > 0$ and $\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*) < 1$. If all $\boldsymbol{z}^* \in \boldsymbol{Z}$, with $q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}^*) > 0$, have $\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \boldsymbol{z}') = 1$, we simply consider some other starting state $\boldsymbol{z}'' \neq \boldsymbol{z}^{(t)}$ until an *aperiodic* state is found. The possibility of $\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}'', \boldsymbol{z}^*) = 1$ for all pairs of states $\boldsymbol{z}'', \boldsymbol{z}^* \in \boldsymbol{Z}$ with $q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}'', \boldsymbol{z}^*) > 0$ would imply that no *aperiodic* state exists but this leads to a contradiction as we now show. A global acceptance probability of exactly 1 for all states implies that any $\boldsymbol{z}^*$ proposed from $q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}'', \boldsymbol{z}^*)$ is always accepted and never rejected i.e that $q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}'', \boldsymbol{z}^*)$ is the full conditional distribution for the particular component $\boldsymbol{z}_i''$ of $\boldsymbol{z}''$ being updated. This is not the case however since our proposal distribution of add/delete moves is not a full conditional distribution. We conclude therefore that some *aperiodic* state exists in $\boldsymbol{Z}$.

Thus the Markov chain is *irreducible* and *aperiodic* and $P^k_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^{(t)}, \cdot)$ has all positive entries for some $k \geq n-1$, thus satisfying Doeblin's condition and from this uniform ergodicity for each $\boldsymbol{\Gamma}^{(t)}$. □

5.2. Diminishing adaptation

The second part of the proof is to verify diminishing adaptation for $P_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}, \cdot)$, $\forall t$. Recall the definition of diminishing adaptation [16]

Definition 5.3. (Diminishing adaptation) A series of transition kernels indexed by t, $P_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}, \cdot)$, are said to obey diminishing adaptation if

$$\lim_{t\to\infty} \sup_{\boldsymbol{z}} \|P_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}, \cdot) - P_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}, \cdot)\| = 0.$$

For this section of the proof, the two other definitions needed are the concept of Lipschitz and bi-Lipschitz continuity of a real-valued function.

Definition 5.4. (Lipschitz continuity) A function f is Lipschitz if there exists $K > 0$ such that

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$$

By the Mean Value Theorem this is equivalent to the function f having a bounded first derivative.

Definition 5.5. (bi-Lipschitz continuity) A function f is bi-Lipschitz if f and its inverse f^{-1} are both Lipschitz and thus one has

$$\frac{1}{K}|x_1 - x_2| \leq |f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$$

where $K > 0$ is the Lipschitz constant of f and the inverse constant of f^{-1}.

Theorem 5.3. *The adaptive changepoint sampler satisfies diminishing adaptation.*

Proof. For a 1-step transition from $\boldsymbol{z}$ to $\boldsymbol{z}'$, define $\Delta^{(t)}$ to be the absolute difference in the transition kernels between iteration t and $t+1$.

$$\Delta^{(t)} = |P_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}, \boldsymbol{z}') - P_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}, \boldsymbol{z}')| \tag{5.2}$$

There are two cases to consider. Firstly when $\boldsymbol{z} \neq \boldsymbol{z}'$ (**Case 1**) and secondly when $\boldsymbol{z} = \boldsymbol{z}'$ (**Case 2**).

Case 1: For $\boldsymbol{z} \neq \boldsymbol{z}'$

$$\begin{aligned}
\Delta^{(t)} &= \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}')\alpha_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}') - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}')\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}') \right| \\
&= \Bigg| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}') \min\left\{1, \frac{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}',\boldsymbol{z})}{\pi(\boldsymbol{z}|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}')}\right\} \\
&\qquad - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}') \min\left\{1, \frac{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}',\boldsymbol{z})}{\pi(\boldsymbol{z}|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}')}\right\} \Bigg| \\
&= \Bigg| \min\left\{ q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}'), \frac{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}',\boldsymbol{z})}{\pi(\boldsymbol{z}|\boldsymbol{y})} \right\} \\
&\qquad - \min\left\{ q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}'), \frac{\pi(\boldsymbol{z}'|\boldsymbol{y}) q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}',\boldsymbol{z})}{\pi(\boldsymbol{z}|\boldsymbol{y})} \right\} \Bigg|. \qquad (5.3)
\end{aligned}$$

It can be checked by simple algebra that the absolute difference of two minimum operators has the following upper bound,

$$|\min\{A,B\} - \min\{C,D\}| \leq |A - C| + |B - D|. \qquad (5.4)$$

Applying this upper bound directly to (5.3) leaves

$$\Delta^{(t)} \leq \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}') - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}') \right| + \frac{\pi(\boldsymbol{z}'|\boldsymbol{y})}{\pi(\boldsymbol{z}|\boldsymbol{y})} \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}',\boldsymbol{z}) - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}',\boldsymbol{z}) \right|$$

Case 2: For $\boldsymbol{z} = \boldsymbol{z}'$

$$\Delta^{(t)} = \left| \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}} \left(q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}^*)\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}^*) - q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}^*)\alpha_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}^*) \right) \right|,$$

applying the triangle inequality and multiplying by $|-1|$ leaves

$$\Delta^{(t)} \leq \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}} \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}^*)\alpha_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}^*) - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}^*)\alpha_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}^*) \right| \qquad (5.5)$$

and applying (5.4) to each term of (5.5) leaves

$$\begin{aligned}
\Delta^{(t)} \leq & \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}} \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}^*) - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}^*) \right| \\
&+ \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}} \frac{\pi(\boldsymbol{z}^*|\boldsymbol{y})}{\pi(\boldsymbol{z}|\boldsymbol{y})} \left| q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}^*,\boldsymbol{z}) - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}^*,\boldsymbol{z}) \right| \qquad (5.6)
\end{aligned}$$

In both **Case 1** and **Case 2**, diminishing adaptation requires that $\Delta^{(t)} \to 0$ as $t \to \infty$ which is true if

$$|q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z},\boldsymbol{z}') - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z},\boldsymbol{z}')| \to 0 \text{ and } |q_{\boldsymbol{\Gamma}^{(t+1)}}(\boldsymbol{z}',\boldsymbol{z}) - q_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}',\boldsymbol{z})| \to 0, \text{ as } t \to \infty$$

This amounts to showing that $|\boldsymbol{a}_i^{(t+1)} - \boldsymbol{a}_i^{(t)}| \to 0$ for all $\boldsymbol{a}_i^{(t)} \in \boldsymbol{a}^{(t)}$ (or equivalently $|\boldsymbol{d}_i^{(t+1)} - \boldsymbol{d}_i^{(t)}| \to 0$ for all $\boldsymbol{d}_i^{(t)} \in \boldsymbol{d}^{(t)}$), as $t \to \infty$. Without loss of generality consider only the $\boldsymbol{a}_i^{(t)}$ adaptive weight. The general form of the update scheme for $\boldsymbol{a}_i^{(t)}$ is

$$\log(\boldsymbol{a}_i^{(t+1)}) = \log(\boldsymbol{a}_i^{(t)}) + \frac{h}{t/n}(\alpha_{\text{add}} - \alpha_{\text{target}})$$

and as $t \to \infty$, with $h < \infty$ and $0 < \alpha_{\text{target}} < 1$

$$\left|\log(\boldsymbol{a}_i^{(t+1)}) - \log(\boldsymbol{a}_i^{(t)})\right| \to 0. \tag{5.7}$$

To prove (5.7) implies $|\boldsymbol{a}_i^{(t+1)} - \boldsymbol{a}_i^{(t)}| \to 0$, we must prove that the log function is bi-Lipschitz. Since $\log(x)$ and its inverse, $\exp(x)$, have a bounded first derivative and provided $0 < x < \infty$ which will be satisfied by the existence of $\epsilon > 0$, log is bi-Lipschitz (Definition (5.5)). Therefore

$$|\boldsymbol{a}_i^{(t+1)} - \boldsymbol{a}_i^{(t)}| \leq K \left|\log(\boldsymbol{a}_i^{(t+1)}) - \log(\boldsymbol{a}_i^{(t)})\right| \to 0, \tag{5.8}$$

and so diminishing adaptation for $P_{\boldsymbol{\Gamma}^{(t)}}(\boldsymbol{z}, \cdot)$ is established. □

6. Results

We will now demonstrate our adaptive algorithm on a number of datasets, varying in size from a small to a large number of observations.

1. **Well Log Drilling data -** a small dataset to demonstrate the equivalence of filtering recursions and the adaptive changepoint sampler.
2. **Channel Noise data -** a moderately sized simulated dataset that takes minutes of precomputation for the filtering recursions, but seconds for our algorithm.
3. **Simulated large data -** a large data set with 300 000 observations, where it is not possible to use filtering recursions due to the presence of numerical error.

For each of the datasets above, we compare our adaptive MCMC algorithm to the filtering recursions approach of Fearnhead [5]. We first take a long run of the filtering recursions at full precision and the posterior distribution from this long run is taken as the ground truth. Our adaptive MCMC algorithm is then compared to this ground truth by examining an approximate version of the Kullback-Leibler divergence in the posterior distribution of the number of changepoints over time. We define this measure of divergence as follows. Let Q be the posterior distribution for the number of changepoints based on the ground truth (filtering recursions) and P be the posterior distribution for the number of changepoints based on our adaptive MCMC algorithm. The divergence is defined as

$$\mathrm{D}_\delta(P|Q) = \sum_{k=0}^{n-1} \left[(1-\delta)P(i) + \delta\frac{1}{n}\right] \log \frac{(1-\delta)P(i) + \delta\frac{1}{n}}{(1-\delta)Q(i) + \delta\frac{1}{n}}. \tag{6.1}$$

The correction parameter δ is necessary to ensure that the support of P and Q overlap and is chosen small ($< 1 \times 10^{-10}$). Note that the Kullback-Leibler divergence results when $\delta = 0$.

6.1. Tuning of the results

We now give some guidelines about how to tune the input parameters of the adaptive algorithm. In general, the input parameter α_{target} depends on k, the number of inferred changepoints. The smaller k is relative to n then the lower α_{target} should be, since in this instance we would expect that many changepoints which are proposed would not be accepted. However in practice the performance of the algorithm is not very sensitive to the choice of α_{target} and we have found that $\alpha_{\text{target}} \in [0.01, 0.2]$ works well in practice. In terms of tuning α_{target}, our approach has been to run a short pilot run of the non-adaptive MCMC algorithm and to set α_{target} to the estimated acceptance rate. Finally, in terms of tuning h our approach has been to set $h = 1/n$ and experimentation has shown that this has yielded good performance. In fact, in further experimentation not reported here, we have found that the performance of the algorithm is relatively insensitive to the choice of h, provided h is not too large as to adapt too quickly. The choice of $h = 1/n$ reflects the structure of the MCMC algorithm, more specifically the length of the vector $\boldsymbol{z}$. The number of iterations required to update each element of $\boldsymbol{z}$ is at least $n - 1$ and thus the adaptive algorithm should respect this waiting time and tune the algorithm proportional to $1/n$.

6.2. Dataset 1 – Gaussian mean changepoint – Well Log Drilling Data

The problem of detecting changepoints in well log drilling data has been studied numerous times in the changepoint literature [5, 17]. The well log drilling dataset originates from Ruanaidh and Fitzgerald [17] and consists of 4050 probe measurements of the nuclear-magnetic response of underground rocks. The data was obtained by lowering the detection probe into a pilot drilled hole in the rock and recording the nuclear-magnetic response at discrete depth intervals. A changepoint is thought to occur when the rock type changes and such a change in signal is observed in the dataset. The data is shown in Figure 4 with outliers removed as in Fearnhead [5]. These data have previously been analysed using filtering recursions to compute the posterior distribution of the number and position of changepoints. We will show that our algorithm reaches the same stationary distribution as the filtering recursions approach in the same time. The approximate filtering recursions using a lower level of precision will be also compared to our algorithm.

6.2.1. Well log drilling – model

We follow the approach of Fearnhead [5] by considering a Geometric ($p = 0.013$) prior on the gap length between successive changepoints. The observations be-

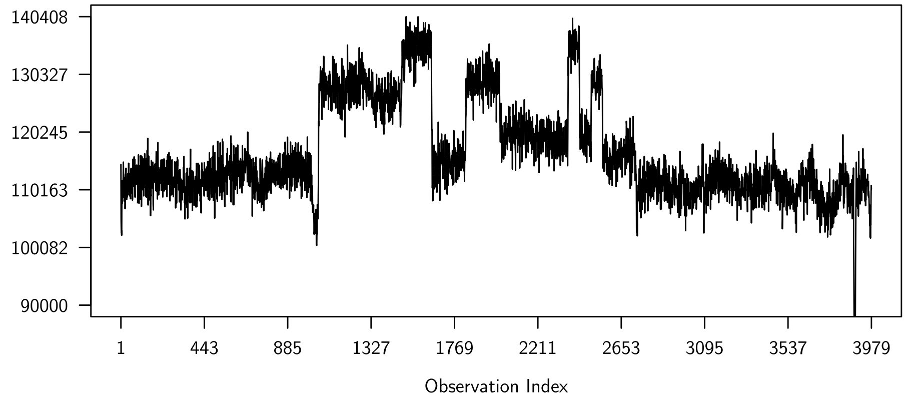

FIG 4. ***Well Log data -*** *The data consists of 3979 observations after outliers have been removed. Visually there are many changepoints in the data, prior to analysis.*

tween changepoints are modelled as $\mathcal{N}(\mu_i, \sigma^2)$, where μ_i is the mean parameter for the ith segment and σ is fixed to 2,500. Independent $\mathcal{N}(115{,}000, \tau^2\sigma^2)$ priors are placed on each μ_i with τ^2 set fixed to 16. Using the methods of Section 2.1 the segment marginal likelihood can be shown (Appendix B) to be

$$\begin{aligned}\mathrm{P}(a,b) &= (2\pi\sigma^2)^{-k/2}(k\tau^2+1)^{-1/2} \\ &\quad\times \exp\left(-\frac{1}{2\sigma^2}\left[\left(s_2 - \frac{s_1^2}{k}\right) + \frac{k}{k\tau^2+1}\left(m - \frac{s_1}{k}\right)^2\right]\right). \qquad (6.2)\end{aligned}$$

The quantities s_1 and s_2 are the sum and the sum of squares of the data $\{y_a, \ldots y_b\}$, respectively.

6.2.2. Well Log Drilling – results & algorithm comparison

The results for the Well Log Drilling data run across adaptive MCMC, non-adaptive MCMC and filtering recursions are shown in Figure 5. The filtering recursions were run at 3 different levels of precision (full precision, 1e-6, 1e-4) to give 5 sets of results. The adaptive MCMC changepoint sampler was run for 5 seconds (16 000 000 iterations) with adaptive parameter $h = 0.00119$ and a target acceptance rate of 15%. The non-adaptive MCMC sampler was run for 5 seconds (20 000 000 iterations). The results in the right panel of Figure 5 illustrate that the modal number of changepoints is estimated as 51 for all algorithms. All algorithms capture the same posterior distribution of the number of changepoints and changepoint positions. Additionally, the left panel of Figure 5 displays the posterior position of changepoints from the adaptive MCMC run which (although not presented here) was very similar to the non-adaptive MCMC and filtering recursion algorithms. The acceptance rates for the adaptive and non-adaptive MCMC algorithms were 15.31% and 15.10%, respectively and both the adaptive and non-adaptive MCMC algorithms were started from the

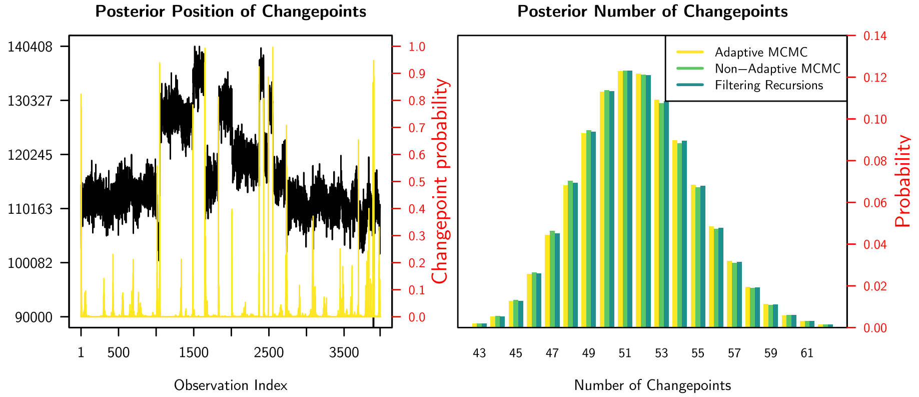

Fig 5. ***Well log data -*** *The left panel presents the estimated posterior probability of a changepoint at each observation based on the adaptive MCMC algorithm. The right panel presents the estimated posterior probability of the number of changepoints for each of the adaptive MCMC, non-adaptive MCMC and filtering recursions algorithms. This illustrates that each algorithm converges to the same stationary distribution.*

same changepoint configuration, 40 changepoints randomly distributed throughout the data.

To compare the results of the adaptive MCMC changepoint sampler against filtering recursions, we compare the divergence of the adaptive and non-adaptive MCMC changepoint samplers to the output of filtering recursions run at full precision for 100 million chains (≈ 1 hour) from Carpenter's algorithm. For the 2 lower levels of precision (1e-6, 1e-4) in Figure 6, the filtering recursions algorithm fails to target the correct posterior once the precision level of the recursions equals 1×10^{-4}. The adaptive MCMC changepoint sampler appears to converge marginally quicker to the target distribution, in the sense of reaching a low divergence, than the non-adaptive and filtering recursions algorithms. However we would overall recommend the use of filtering recursions for datasets of this size and smaller since the computational time is reasonable in these instances. The adaptive algorithm is marginally faster than the non-adaptive version and with a higher acceptance rate (15.31%). This outlines that the Adaptive MCMC is competitive not only to the filtering recursions but also to the non-adaptive algorithm.

6.3. Dataset 2 – Gaussian precision changepoint – channel noise data

Variations in a signal can be detected by considering the change in variance around a fixed mean. For example a web server may exhibit rapid variations in traffic across a period of time or a failing component of a machine may lose its precision as it fails. If we assume a constant mean for each of the segments and allow there to be a change in precision $\lambda = \frac{1}{\sigma^2}$ at a changepoint we can model a process such as shown in Figure 7.

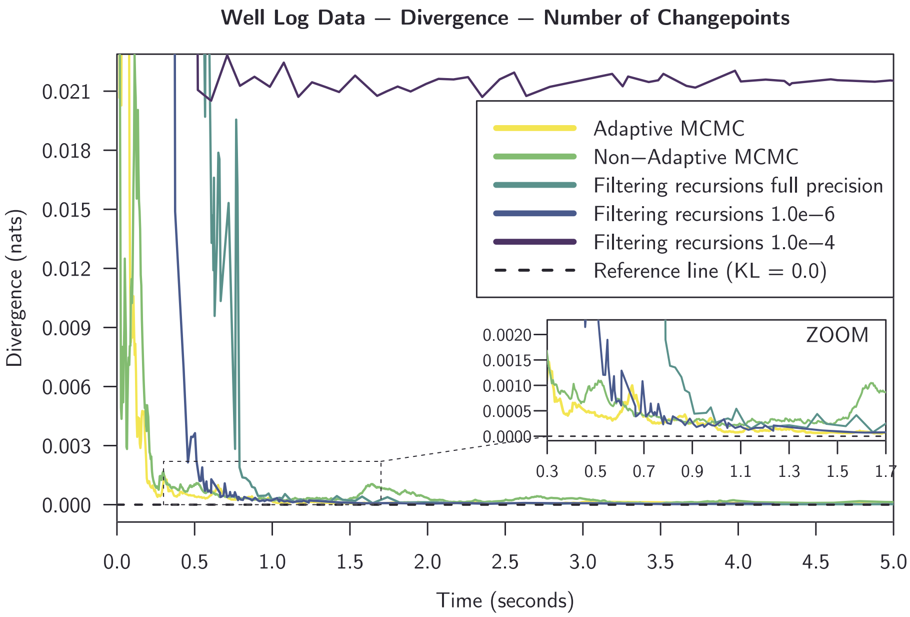

Fig 6. ***Well log data -*** *Divergence, D_δ, between a precise estimate of the posterior distribution of the number of change points based on a long run of the filtering recursion algorithm to the adaptive and non-adaptive MCMC algorithm. This plot shows the convergence to the ground truth. All chains converge to the ground truth except for the low precision recursions. The adaptive algorithm is the most competitive of the MCMC algorithms.*

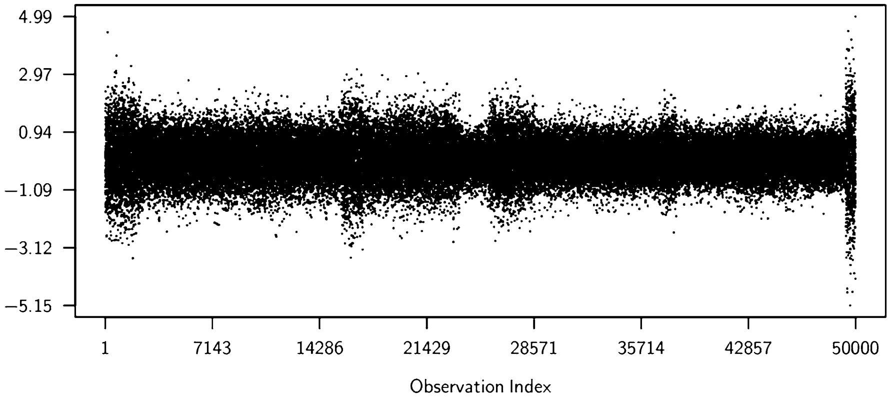

Fig 7. ***Channel Noise data -*** *A simulated dataset of 50,000 observations where the variance is assumed to change over time around a fixed mean. The data was simulated with 25 changepoints.*

The likelihood for each observation y_i is $\mathcal{N}(\mu, \lambda^{-1})$ for a fixed μ. Assuming a prior on the precision, $\lambda \sim \text{Gamma}(\alpha_0, \beta_0)$, allows one to integrate over $0 < \lambda < \infty$, leaving a marginal likelihood

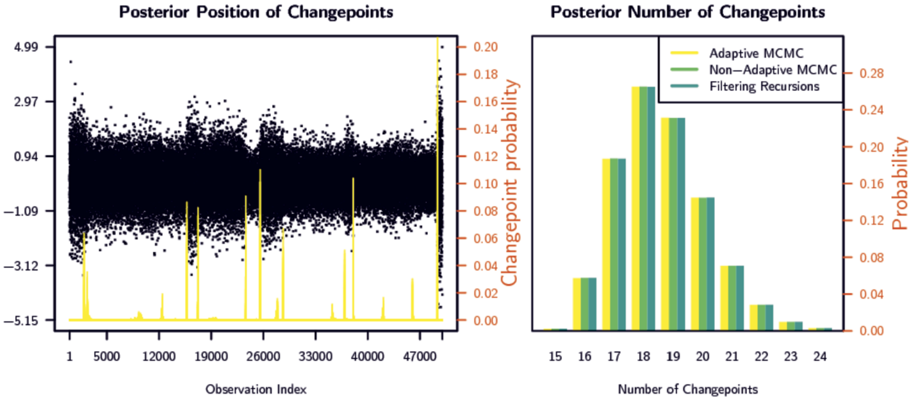

Fig 8. ***Channel Noise data -*** *The estimated posterior probability of a changepoint at each observation based on the adaptive MCMC algorithm is shown (left panel). The right panel presents the estimated posterior probability of the number of changepoints for each of the adaptive MCMC, non-adaptive MCMC and filtering recursions algorithms. This illustrates that each algorithm converges to the same stationary distribution.*

$$\mathrm{P}(a,b) = \frac{(2\pi)^{-n/2}\Gamma(k/2+\alpha_0)}{\left(\beta_0 + \sum_{i=a}^{b}(x_i-\mu)^2/2\right)^{\alpha_0+k/2}}, \qquad \text{where } k = b-a+1. \tag{6.3}$$

For this data the hyperparameters were set to $\alpha_0 = 12.0, \beta_0 = 4.8$ and $\mu = 0$. The parameter μ can be set to 0 prior to analysis provided the data is shifted using its known mean. A geometric gap prior was placed on $\boldsymbol{z}$ with $p = 0.0006$. The results for the channel noise data are shown in Figure 8 for filtering recursions, the adaptive MCMC changepoint sampler and the non-adaptive MCMC changepoint sampler. The adaptive MCMC changepoint sampler was run with a target acceptance rate of 10.5% and with h set to 0.00008. All 3 algorithms give a modal 18 number of changepoints in the data and each algorithm captures the full posterior distribution for both the positions and number of changepoints. The adaptive MCMC and non-adaptive MCMC algorithms were each run for 300 seconds, 60 000 000 iterations and 67 000 000 iterations respectively. The filtering recursions were run for the necessary precomputation time of 572 seconds and then a further 8500 seconds using Carpenter's algorithm (100 000 000 chains). Extra runs of the filtering recursions were run at precision levels (1e-12, 1e-10 and 1e-8) for comparison.

For this example, it is possible to compare the convergence properties of the adaptive MCMC, non-adaptive MCMC and filtering recursions. Due to the extended precomputation time required for the filtering recursions for datasets of this size, we can only compare the two algorithms after the precomputation has completed. We compare the algorithms by examining the time to converge to a certain level of divergence, D_δ. In Figure 9 we mark an approximate convergence point at 53 seconds for the adaptive changepoint sampler with a divergence of

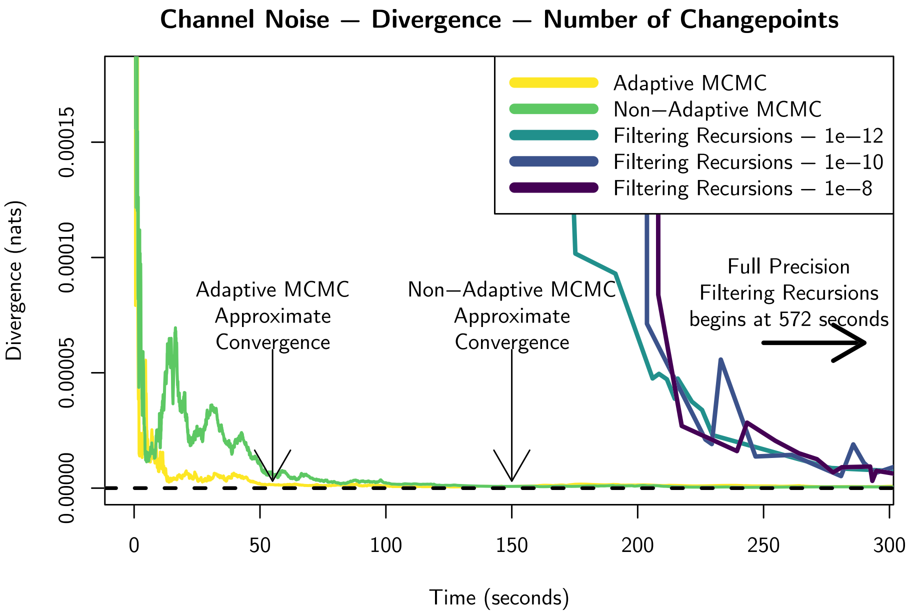

FIG 9. ***Channel Noise data -*** *Divergence, D_δ, between a precise estimate of the posterior distribution of the number of change points based on a long run of the filtering recursion algorithm to the adaptive and non-adaptive MCMC algorithm. The adaptive MCMC algorithm outperforms non-adaptive MCMC and filtering recursions for all precision levels. We mark a convergence point for the adaptive MCMC of 53 seconds with divergence 1.43×10^{-6} nats in the posterior distribution of the number of changepoints. The non-adaptive MCMC algorithm takes 150 seconds to reach this level of divergence. Note that (although not shown on the plot) the full-precision filtering takes* 1738 *seconds to reach this same level of divergence.*

1.43×10^{-6} nats. The filtering recursions is then run at full precision until it reaches this level of divergence or below, which takes 1738 seconds. The results of this analysis are show in Figure 9 and Table 1, with Table 1 showing the relative speed of each algorithm. There is a vast in improvement using the adaptive MCMC algorithm.

TABLE 1
Channel Noise data - *Relative speed and acceptance rates of each (MCMC) algorithm are shown.*

	Adaptive	**Non-Adaptive**	**Filtering Recursions**
Relative Speed	1.0 (53 seconds)	2.83 (150 seconds)	32.8 (1738.64 seconds)
Acceptance Rate	12.1%	10.9%	-

6.4. Dataset 3 – Gaussian mean changepoint – large data example

For large datasets it is much slower to use filtering recursions due to the quadratic complexity of computing the recursions. For significantly large datasets, the quadratic complexity becomes prohibitive and numerical error in computing the recursions is also an issue. The large data we analyse consists of 300 000 observations and is displayed in Figure 10. The data was simulated from a Gaussian distribution with 40 changepoints assumed on the mean as in the Well log data of Section 6.2.2. We assume a Geometric prior for the changepoint positions with parameter $p = 40/299999 \approx 1.33 \times 10^{-4}$. The likelihood is taken as $\mathcal{N}(\mu_j, \sigma^2)$ and the prior for μ_j is $\mathcal{N}(115{,}000, \tau^2\sigma^2)$ with $\sigma = 2{,}500$ and $\tau = 4$.

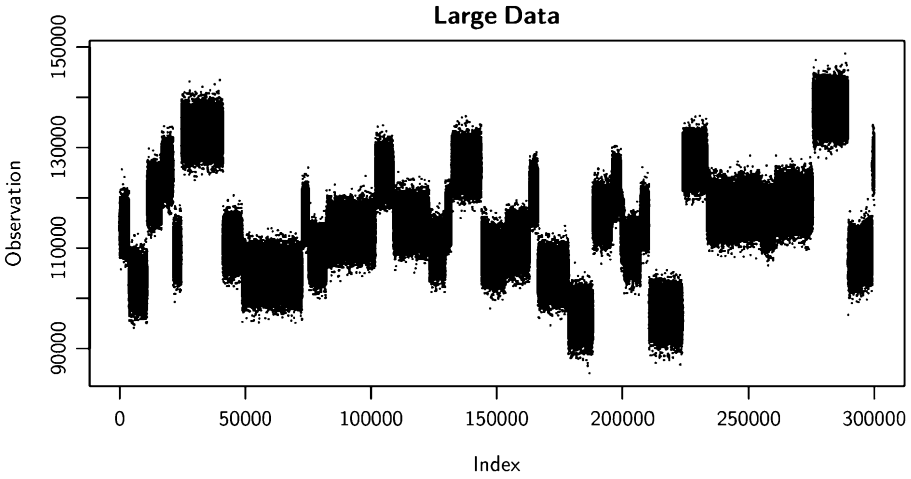

FIG 10. ***Large dataset -*** 300 000 *observations simulated from a Gaussian distribution with a changepoint on the mean. The data was simulated with 40 changepoints. The parameters for the mean were the same as in the Well log example of Section 6.2.2.*

6.4.1. Difficulty with filtering recursions for large data

For a dataset of this size the numerical stability of the filtering recursions can cause problems. In the calculation of the transition probabilities (3.1), which are needed to sample from the recursions using the Carpenter's algorithm [2], there is potential to encounter numerical errors arising from building the forward proposal distribution of the next changepoint. For this dataset, considering every possible changepoint location $j \in \{1, \dots n-1\}$, there will be a maximum $\binom{300{,}000}{2} \approx 4.50 \times 10^{10}$ transition probabilities $\mathrm{P}(\tau_j|\tau_{j-1})$ to calculate, see equation (3.1). Many of the $\mathrm{P}(\tau_r|\tau_{j-1})$ terms will be very small with values less than subnormal machine precision even when computed on the log scale. Numerically, transition probabilities close to 0 are regarded as having negligible contribution to proposing changepoints and will not be sampled. However for datasets where changepoints are far apart i.e. $\tau_r \gg \tau_{j-1}$, the calculation of the cumulative distribution which is needed to propose the next changepoint,

$$\mathrm{P}(\tau \le \tau_r|\tau_{j-1}) = \log\left(e^{\mathrm{P}(\tau_r|\tau_{j-1})} + e^{\sum_{i=1}^{r-1}\mathrm{P}(\tau_i|\tau_{j-1})}\right), \tag{6.4}$$

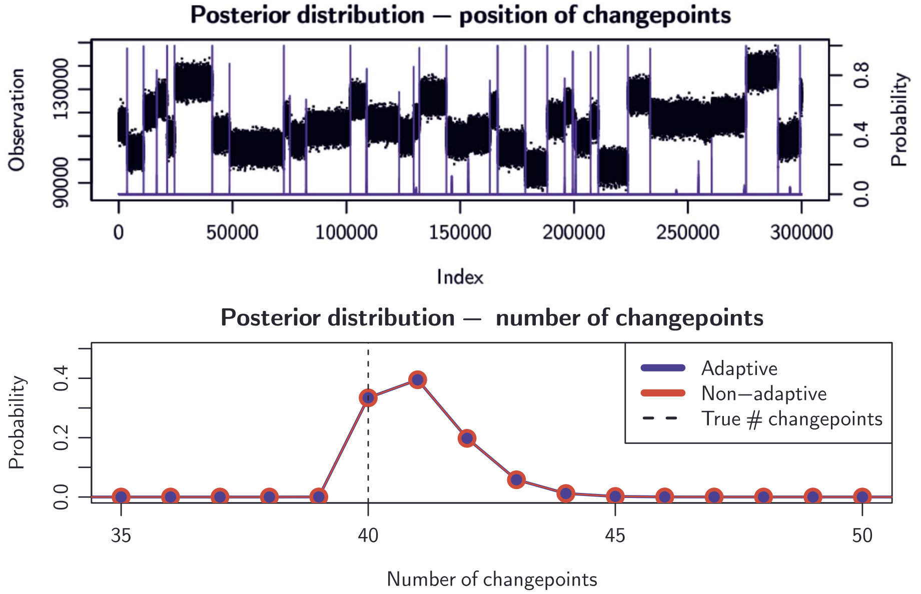

Fig 11. ***Large data -*** *The top figure shows the posterior distribution of changepoint positions captured by the adaptive algorithm. The bottom figure shows the posterior distribution of the number of changepoints for the adaptive algorithm and the non-adaptive algorithm with a mode of 41 changepoints. The true number of changepoints is also shown.*

will not correctly accumulate all of these small probabilities. Since the number of small probabilities is significantly large, this leads to those probabilities greater than subnormal machine probabilities to be artificially inflated relative to the magnitude they would normally appear had the small probabilities being accumulated to infinite precision. The effect of this is that these points will be chosen as changepoints more frequently than they should be and points with small probabilities never to be chosen even though taken together they consume non-negligible mass of the transition distribution. This agrees empirically with our analysis for this dataset and other even larger datasets.

6.4.2. Results for the adaptive algorithm

The adaptive algorithm was run for 1 000 000 000 iterations with 100 000 iterations removed by burn-in. The adaptive parameter h was set to 4×10^{-6}, while a target acceptance rate of 2.0% was chosen to tune the adaptive scheme. The acceptance rate is quite low, however the number of simualted changepoints (40) relative to the size of the dataset indicates that many moves may be rejected. The algorithm took 400 seconds on an Intel i7 3.40GHz and the achieved acceptance rate was 2.19%. The adaptive changepoint sampler and the non-adaptive sampler both detect a mode of 41 changepoints.

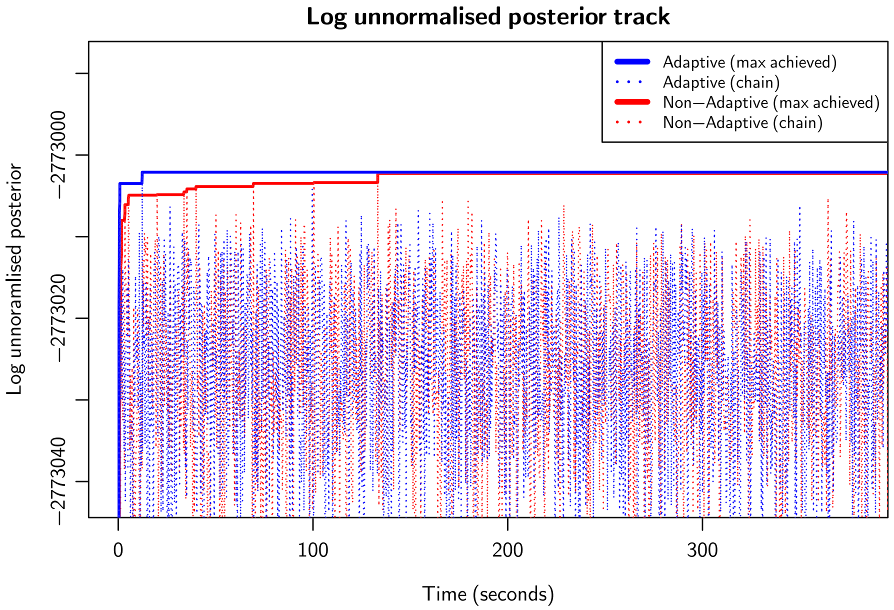

Fig 12. ***Large data -*** *Comparison between the trajectory of the log unnnormalised posterior for the adaptive and non-adaptive algorithms. The adaptive algorithm climbs to an area of high posterior probability many times faster than the non-adaptive algorithm. The adaptive algorithm locates a maximum posterior region after 12.3 seconds and the non-adaptive after 133.26 seconds.*

6.4.3. Algorithm comparison - adaptive and non-adaptive MCMC

It is not possible to run the filtering recursions for this data due to issues discussed in Section 6.4.1. In particular, and in contrast with the two previous examples, it is not possible to assess how well each algorithm converges to the target posterior distribution. However we can still provide a good indication of the convergence of each of the adaptive and non-adaptive MCMC algorithms by exploring the trajectory of the state of each chain towards the maximum *a posteriori* of the target distribution. We performed a run of both the adaptive and non-adaptive algorithms for over 1 000 000 000 iterations and monitored the maximum unnormalised *a posteriori* (MAP) estimate achieved. The results are shown in Figure 12. In Figure 12, the adaptive algorithm reaches a high area of the posterior after about 1 second and continues to find a higher area of posterior mass until 12.3 seconds. In constrast the non-adaptive version is much slower to climb to this area taking 133.26 seconds to achieve the same level as the adaptive algorithm. Figure 12 also shows the mixing of the chain for both algorithms indicating that both algorithms are mixing well. The maximum unnormalised posterior gives some indication that the adaptive MCMC algorithm is better able to reach the high-posterior density regions than the non-adaptive MCMC algorithm. We therefore conclude for datasets of this size, that the adaptive algorithm is many times more competitive than the non-adaptive algorithm and due to filtering recursion being unavailable is an ideal algorithm for big data changepoint problems.

7. Conclusion & discussion

This paper introduces an adaptive changepoint sampling algorithm for multiple changepoint problems. We have described how our algorithm is be designed to learn *on-the-fly* where changepoints are likely to be located in a dataset. We prove that the adaptive MCMC scheme which we develop leaves the posterior distribution ergodic. Moreover the adaptive MCMC algorithm scales to large datasets in contrast to the filtering recursions of Fearnhead [5] which is unreliable and prone to numerical instability in this case. Three datasets increasing in size from 4000 observations to 300 000 observations have been illustrated in this paper. The latter and largest dataset is unable to be analysed using filtering recursions and we show that our algorithm works well here to detect the number and location of changepoints in a reasonable computational time. We recommend using the filtering recursions for smaller datasets (e.g. up to size 100 000 observations) and where computational time is not an issue. However for datasets with more than 100 000 observations we advocate using the adaptive changepoint sampler.

Further work will involve extending this adaptive MCMC approach to other posterior distributions on discrete state spaces. For example, the likelihood of the data in this paper assumes independent observations within a segment between two changepoints. This could be replaced with a dependence within segment likelihood as in the work of Wyse et al. [22] where the marginal segment likelihood is replaced with integrated nested Laplace approximations.

The diminishing adaptation condition we have proved in this paper is just one method of automatically tuning adaptive proposals. Our adaptation condition takes the form of a stochastic approximation algorithm but more involved adaptation schemes may be designed using the theory developed in this paper and this is a focus of future work. To conclude, we feel that there is much wider scope for the implementation of adaptive MCMC in practice and we hope that this article will encourage more work in this direction.

Acknowledgements

he Insight Centre for Data Analytics is supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289. Alan Benson and Nial Friel's research was also supported by a Science Foundation Ireland grant: 12/IP/1424.

Appendices

Appendix B: Normal marginal likelihood calculation

The marginal likelihood for a changepoint in the mean parameter for normally distributed data with known variance (σ^2) and with a $\mathcal{N}(\mu, \tau^2\sigma^2)$ prior on μ

can be expressed with $k = b - a + 1$ as the integral of the product of two normal densities

$$\mathrm{P}(a,b) = \int_0^{\infty} \frac{(2\pi\sigma^2)^{-(k+1)/2}}{\tau} \times \prod_{i=a}^{b} \exp\left(-\frac{1}{2\sigma^2}\left[\left(k+\frac{1}{\tau^2}\right)\mu^2 - 2\left(s_1 + \frac{m}{\tau^2}\right)\mu + \left(s_2 + \frac{\mu^2}{\tau^2}\right)\right]\right) d\mu \tag{B.1}$$

where $s_1 = \sum_{i=a}^{b} y_i$ and $s_2 = \sum_{i=a}^{b} y_i^2$. Completing the square and rearranging

$$= (2\pi\sigma^2)^{-k/2}(k\tau^2+1)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}\left[\left(s_2 + \frac{\mu^2}{\tau^2}\right) - \frac{\tau^2}{k\tau^2+1}\left(s_1 + \frac{\mu}{\tau^2}\right)^2\right]\right) \tag{B.2}$$

completing the square again with the term inside the square brackets gives

$$= (2\pi\sigma^2)^{-k/2}(k\tau^2+1)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}\left[\left(s_2 - \frac{s_1^2}{k}\right) + \frac{k}{k\tau^2+1}\left(m - \frac{s_1}{k}\right)^2\right]\right) \tag{B.3}$$

This is a more numerically stable version than (B.2) as $s_2 - \frac{s_1^2}{k}$ is the sum of squared deviations from the segment sample mean which can be calculated recursively and $m - \frac{s_1}{k}$ is the distance of the segment sample mean from the prior which will cause no numerical issues.

Appendix C: Walker's Alias Method with Vose's correction

The Alias Method is due to Walker [20] and the numerical safe approach to constructing Alias tables, which are needed for this method, is due to Vose [19]. The algorithm is a very simple approach to simulating from a general categorical distribution with k categories each having a (possibly unnormalised) weight w_k.

The weights are first normalised and then two tables are constructed, a probability table and an Alias table. Some of the normalised weights will be greater than the average probability $\frac{1}{k}$ and are known as Big Points, and some will be less than or equal to it, the Small Points. The method works by moving some of the probability mass from the Big Points to the Small Points. All Small Points will eventually be associated with at most one of the Big Points (its alias).

Once the Alias table has been constructed they can be sampled from in $\mathcal{O}(1)$ time. Simply select a Small Point uniformly at random and then use a biased coin flip to choose either that point or its Alias point. This method is extremely efficient and is currently the best of all methods for sampling from finite categorical distributions however if w_k changes for any k the entire tables must be reconstructed in $\mathcal{O}(n)$ more steps. Another method with a similar computational efficiency is the Gumbel Max Method [23]

Appendix D: Carpenter's Algorithm

Carpenter's Algorithm [2] is a method of sampling from a discrete probability distribution similar to the Alias method but without the need for precomputed probability tables. It works by exploiting the fact that the spacing in the uniform distribution on [0,1) is exponential with rate 1. To sample n values from $x = \{1, \dots M\}$ with $P(X = i) = p_i$

1. Simulate $e_1, \dots, e_{n+1} \sim \exp \lambda = 1$
2. Create the step function (CDF) $u_j = \frac{\sum_{i=1}^{j} e_i}{\sum_{i=1}^{n+1} e_i}$ for $j = 1, \dots n + 1$
3. Set $Q = 0$, $U = u_1$, $j = 1$, $i = 1$
4. If $U < Q + P(X = j)$ output j and set $U = u_{i+1}$ and $i = i + 1$. Otherwise set $Q = P(X = j)$ and $j = j + 1$. Repeat until $i = n + 1$.

Appendix E: Dual adaptation

Only one of the adaptive parameters for a point i ($\boldsymbol{a}_i$ / $\boldsymbol{d}_i$) are updated when either an add or delete move at this point has been accepted. Griffin et al. [7] has suggested that information can still be gained for both $\boldsymbol{a}_i$ and $\boldsymbol{d}_i$ regardless of which move has been performed.

Dual adaptation involves using the M-H ratio calculated for the current move, denoted $\alpha_F(\boldsymbol{z}, \boldsymbol{z}')$ for the *forward* move, and its *reverse* move, denoted $\alpha_R(\boldsymbol{z}', \boldsymbol{z})$. Calculation of α_R is trivial once α_F is available. Griffin et al. [7] shows how to modify the adaptation scheme so that it continues to target M. The average *a posteriori* mutation rate of the algorithm is

$$M = \int C(\boldsymbol{z}, \boldsymbol{z}')\alpha(\boldsymbol{z}, \boldsymbol{z}')q(\boldsymbol{z}, \boldsymbol{z}')\pi(\boldsymbol{z}|\boldsymbol{y})\, d\boldsymbol{z}\, d\boldsymbol{z}'$$

where $q(\boldsymbol{z}, \boldsymbol{z}')$ depends on the move (add / delete) and $C(\boldsymbol{z}, \boldsymbol{z}') = 0$ if $z_i = z_i' \ \forall i$.

If we wish to continue targeting this mutation rate under Dual adaptation we need to define a second chain to preserve detailed balance.

$$(\boldsymbol{\delta}, \boldsymbol{\delta}') = \begin{cases} (\boldsymbol{z}', \boldsymbol{z}), & \text{with probability } \alpha(\boldsymbol{z}, \boldsymbol{z}'), \\ (\boldsymbol{z}, \boldsymbol{z}'), & \text{with probability } 1 - \alpha(\boldsymbol{z}, \boldsymbol{z}'). \end{cases}$$

Now

$$\begin{aligned} M_{\boldsymbol{\delta}} &= \int C(\boldsymbol{\delta}, \boldsymbol{\delta}')\alpha(\boldsymbol{\delta}, \boldsymbol{\delta}')q(\boldsymbol{\delta}, \boldsymbol{\delta}')\pi(\boldsymbol{\delta}|\boldsymbol{y})\, d\boldsymbol{\delta}\, d\boldsymbol{\delta}' \\ &= \mathbf{E}[C(\boldsymbol{\delta}, \boldsymbol{\delta}')\alpha(\boldsymbol{\delta}, \boldsymbol{\delta}')] \\ &= \alpha(\boldsymbol{z}, \boldsymbol{z}')\mathbf{E}(C(\boldsymbol{z}', \boldsymbol{z})\alpha(\boldsymbol{z}', \boldsymbol{z})) + (1 - \alpha(\boldsymbol{z}, \boldsymbol{z}'))\mathbf{E}(C(\boldsymbol{z}, \boldsymbol{z}')\alpha(\boldsymbol{z}, \boldsymbol{z}')) \end{aligned}$$

and weighting this with the original mutation rate we get

$$w\alpha(\boldsymbol{z}, \boldsymbol{z}')\mathbf{E}(C(\boldsymbol{z}', \boldsymbol{z})\alpha(\boldsymbol{z}', \boldsymbol{z})) + (1 - w\alpha(\boldsymbol{z}, \boldsymbol{z}'))\mathbf{E}(C(\boldsymbol{z}, \boldsymbol{z}')\alpha(\boldsymbol{z}, \boldsymbol{z}'))$$

note $C(\boldsymbol{z}', \boldsymbol{z}) = C(\boldsymbol{z}, \boldsymbol{z}')$

The new adaptive scheme becomes
If an add move has just been accepted:

$$\text{Update } \boldsymbol{a}_i \quad \log(\boldsymbol{a}_i^{(t+1)}) = \log(\boldsymbol{a}_i^{(t)}) + \left(\frac{h}{t/n}\right)(\alpha(\boldsymbol{z},\boldsymbol{z}') - \alpha_{\text{target}})\,(1 - w\alpha(\boldsymbol{z},\boldsymbol{z}')).$$

$$\text{Update } \boldsymbol{d}_i \quad \log(\boldsymbol{d}_i^{(t+1)}) = \log(\boldsymbol{d}_i^{(t)}) + \left(\frac{h}{t/n}\right)(\alpha(\boldsymbol{z}',\boldsymbol{z}) - \alpha_{\text{target}})\,\alpha(\boldsymbol{z},\boldsymbol{z}').$$

If a delete move has just been accepted:

$$\text{Update } \boldsymbol{a}_i \quad \log(\boldsymbol{a}_i^{(t+1)}) = \log(\boldsymbol{a}_i^{(t)}) + \left(\frac{h}{t/n}\right)(\alpha(\boldsymbol{z}',\boldsymbol{z}) - \alpha_{\text{target}})\,\alpha(\boldsymbol{z},\boldsymbol{z}').$$

$$\text{Update } \boldsymbol{d}_i \quad \log(\boldsymbol{d}_i^{(t+1)}) = \log(\boldsymbol{d}_i^{(t)}) + \left(\frac{h}{t/n}\right)(\alpha(\boldsymbol{z},\boldsymbol{z}') - \alpha_{\text{target}})\,(1 - w\alpha(\boldsymbol{z},\boldsymbol{z}')).$$

The choice of w is recommended as 0.5 by Griffin et al. [7].

References

[1] Barry, D. and J. A. Hartigan (1992, 03). Product partition models for change point problems. *Ann. Statist. 20*(1), 260–279. MR1150343

[2] Carpenter, J., P. Clifford, and P. Fearnhead (1999). Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation 146*(1), 2–7.

[3] Chen, J. and A. K. Gupta (2011). *Parametric statistical change point analysis: with applications to genetics, medicine, and finance*. Springer Science & Business Media. MR3025631

[4] Chib, S. (1998). Estimation and comparison of multiple change-point models. *Journal of econometrics 86*(2), 221–241. MR1649222

[5] Fearnhead, P. (2006). Exact and efficient Bayesian inference for multiple changepoint problems. *Statistics and Computing 16*(2), 203–213. MR2227396

[6] Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732. MR1380810

[7] Griffin, J., K. Latuszynski, and M. Steel (2014). Individual adaptation: an adaptive MCMC scheme for variable selection problems. *arXiv preprint arXiv:1412.6760v2*.

[8] Haario, H., E. Saksman, and J. Tamminen (2001, 04). An adaptive Metropolis algorithm. *Bernoulli 7*(2), 223–242. MR1828504

[9] Hocking, T. D., V. Boeva, G. Rigaill, G. Schleiermacher, I. Janoueix-Lerosey, O. Delattre, W. Richer, F. Bourdeaut, M. Suguro, M. Seto, et al. (2014). SegAnnDB: interactive web-based genomic segmentation. *Bioinformatics 30*(11), 1539–1546.

[10] Łatuszyński, K. and J. S. Rosenthal (2014). The containment condition and AdapFail algorithms. *Journal of Applied Probability 51*(04), 1189–1195. MR3301296

[11] Lavielle, M. and E. Lebarbier (2001). An application of MCMC methods for the multiple change-points problem. *Signal Processing 81*(1), 39–53.

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

[13] Matias, Y., J. S. Vitter, and W. Ni (1993). Dynamic generation of discrete random variates. In *SODA*, pp. 361–370. MR1213248

[14] Meyn, S. P. and R. L. Tweedie (2012). *Markov chains and stochastic stability*. Springer Science & Business Media. MR1287609

[15] Raftery, A. E. and V. E. Akman (1986). Bayesian analysis of a Poisson process with a change-point. *Biometrika 73*(1), 85–89. MR0836436

[16] Rosenthal, J. S. and G. O. Roberts (2007). Coupling and ergodicity of adaptive MCMC. *Journal of Applied Probablity 44*, 458–475. MR2394801

[17] Ruanaidh, J. and W. J. Fitzgerald (2012). *Numerical Bayesian methods applied to signal processing*. Springer Science & Business Media.

[18] Stephens, D. A. (1994). Bayesian retrospective multiple-changepoint identification. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 43*(1), 159–178.

[19] Vose, M. D. (1999). *The simple genetic algorithm: foundations and theory*, Volume 12. MIT press. MR1713436

[20] Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters 10*(8), 127–128.

[21] Wyse, J. and N. Friel (2010). Simulation-based Bayesian analysis for multiple changepoints. *arXiv preprint arXiv:1011.2932*.

[22] Wyse, J., N. Friel, et al. (2011). Approximate simulation-free Bayesian inference for multiple changepoint models with dependence within segments. *Bayesian Analysis 6*(4), 501–528. MR2869956

[23] Yellott, J. I. (1977). The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology 15*(2), 109–144. MR0449795