



# StatAlign: Towards the second major release

H. Eiriksson, V. Jonsson, D. Wood, Á. Novák, R. Lyngsø and J. Hein<sup>‡</sup>

Department of Statistics, University of Oxford, 1 South Parks Road, OX1 3TG, United Kingdom

<sup>‡</sup> hein@stats.ox.ac.uk

## Motivation

Improvements were needed to the phylogenetic analysis program StatAlign as part of progress towards a new release. It lacked both consensus tree and consensus network calculating and drawing algorithms to summarise the phylogenetic trees it produced. In addition the program could be speeded up by parallelising the main Markov Chain Monte Carlo calculations using a (MC)<sup>3</sup> technique.

## Background

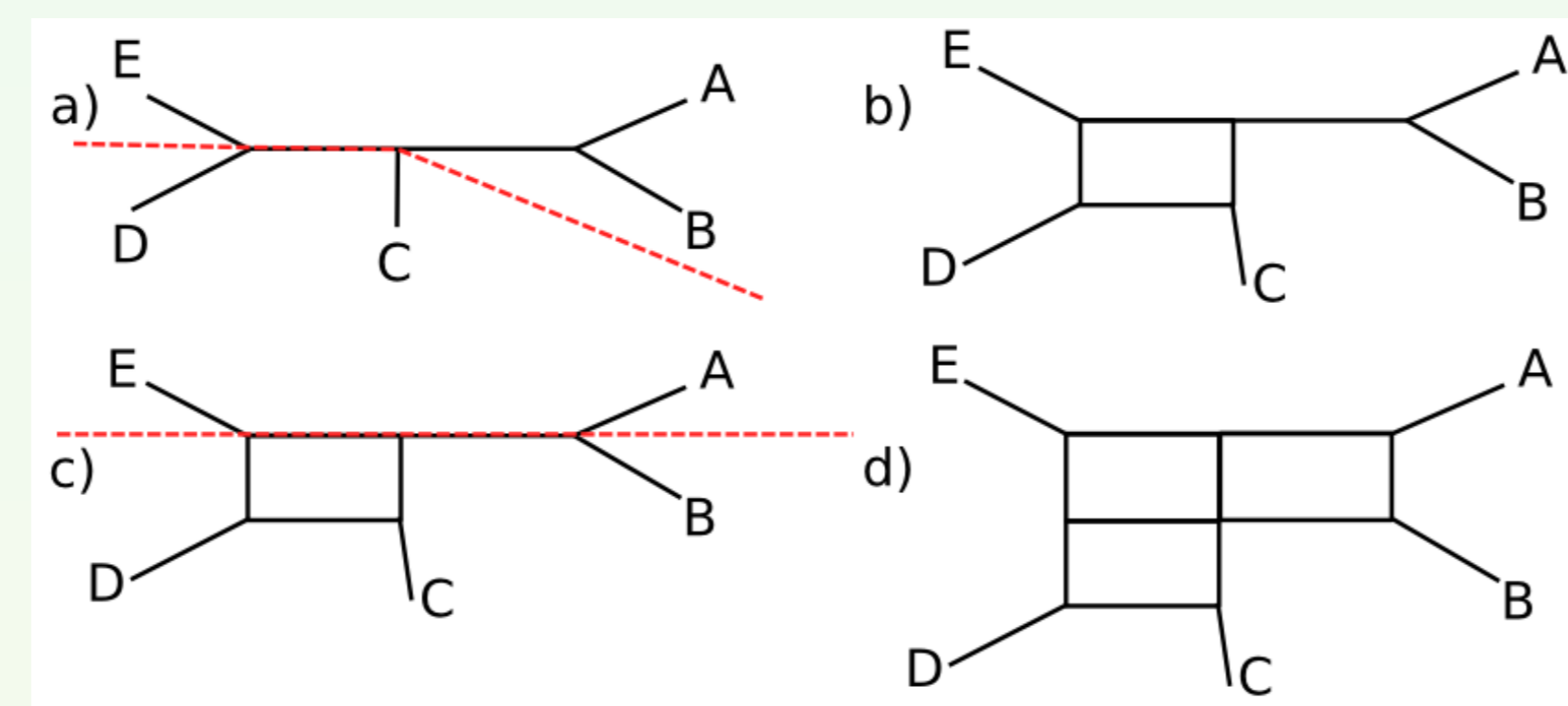
StatAlign is a java-based software package first released in 2008 that simultaneously performs sequence alignment and phylogeny reconstruction [1]. Beginning from an initial alignment by score-based methods and then improving this with Markov Chain Monte Carlo (MCMC) techniques it samples the joint posterior distribution of alignments after a user specified burn-in time. A range of substitution models are available to the user and the TKF92 model is used for insertions and deletions to calculate the MCMC probabilities. What sets StatAlign apart from most programs with similar aims is joint sampling of tree and alignment space (as opposed to fixing the alignment before sampling from tree space).

## Consensus network drawing

The program extends the equal angle method proposed by Meacham [3] for drawing trees to draw networks. The method begins at a node in the tree (1 in the figure) with a full revolution of angle space assigned to it (360°) from which directions of adjoining edges can take. This angle space is then divided up among adjoining edges proportionally to the number of taxa that need to be drawn from it. Boxes cause complications which constrain certain edges to be adjacent. Also angle space such as  $\alpha$  and  $\beta$  in the figure occur for edges that must be drawn in a direction outside of it as they belong to a bipartition which has already had an edge drawn. Finally, the branch lengths used are simply a mean average from the edges in the sample trees. StatAlign implements calculation and drawing of zoomable majority consensus trees made up of bipartitions that occur in over half the sample trees and 2 dimensional networks with bipartitions that occur in over a third of the trees.

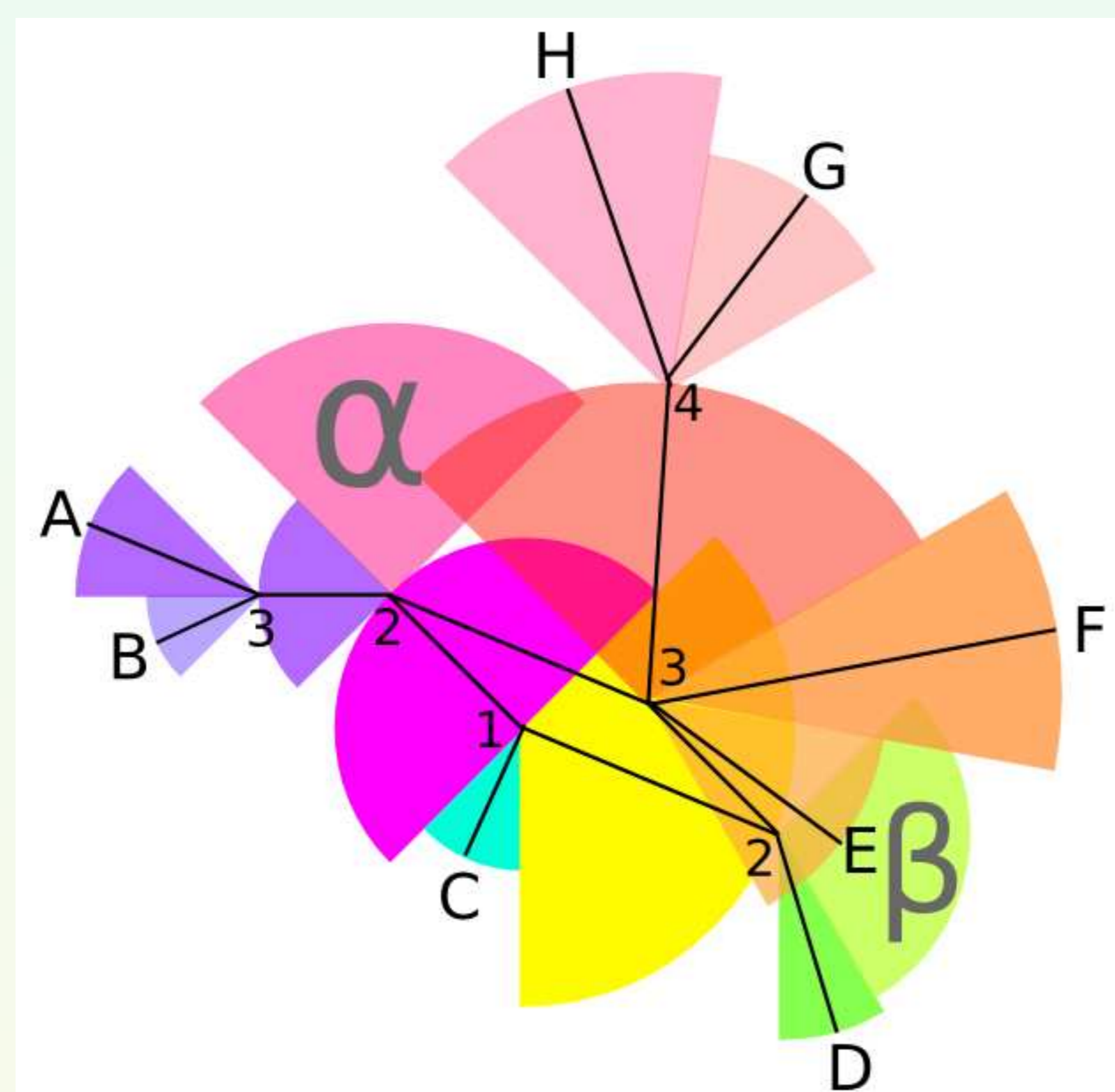
## Consensus network calculation

Removing an edge from a tree splits the tree into two sets of leaves: a bipartition. Frequently two or more bipartitions occur with high frequencies but are incompatible with one another, i.e. they can't both be present in a tree and are represented as boxes in network diagrams. We implement a modified version of the Sul and Tiffani algorithm [2] to store bipartition occurrences in a sample of rooted trees in a hash table. We then create a consensus tree as they suggest and add further bipartitions to this to create the network.



a) Original Network of splits DE|ABC and CDE|AB. b) Added split DC|ABE (incompatible with DE|ABC). c) Finding the path to split over when adding BCD|AE, being incompatible with DE|ABC and CDE|AB but satisfying condition of being compatible with DC|ABE. d) Final consensus network with split BCD|AE now added.

GUI improvements were made and a new GUI plugin system written. We plan to carry out further work to validate the new features with real data.



## Parallelisation

When the posterior space is large and contains several regions of high probability separated by regions of low probability the samples needed to reach convergence can grow unmanageably large. We addressed this with Metropolis Coupled MCMC (or (MC)<sup>3</sup>) which adds heated chains to regular MCMC that run in parallel to the main cold chain. Heat in this case is a parameter in the range [0 1] and is used to adjust the acceptance probability:

$$R_i = \min \left[ 1, \left( \frac{f(X | \psi'_i)}{f(X | \psi_i)} \frac{q(\psi_i)}{q(\psi'_i)} \right)^{\beta_i} \right] \quad (1)$$

Here  $\psi_i$  is the current state of the Markov chain  $i$  and  $\psi'_i$  represents the proposed state.  $f(X | \psi'_i)$  is the likelihood of the new state and  $f(\psi'_i)$  is the prior probability of the new state.  $q(\psi_i)$  and  $q(\psi'_i)$  represent the backward and forward proposal probabilities, respectively.  $\beta$  is the heat of chain  $i$ . The new state,  $\psi'_i$ , is accepted with a probability of  $R_i$ . By taking the power of a number smaller than one the heated chains sample from flattened distributions which

## Parallelisation Results

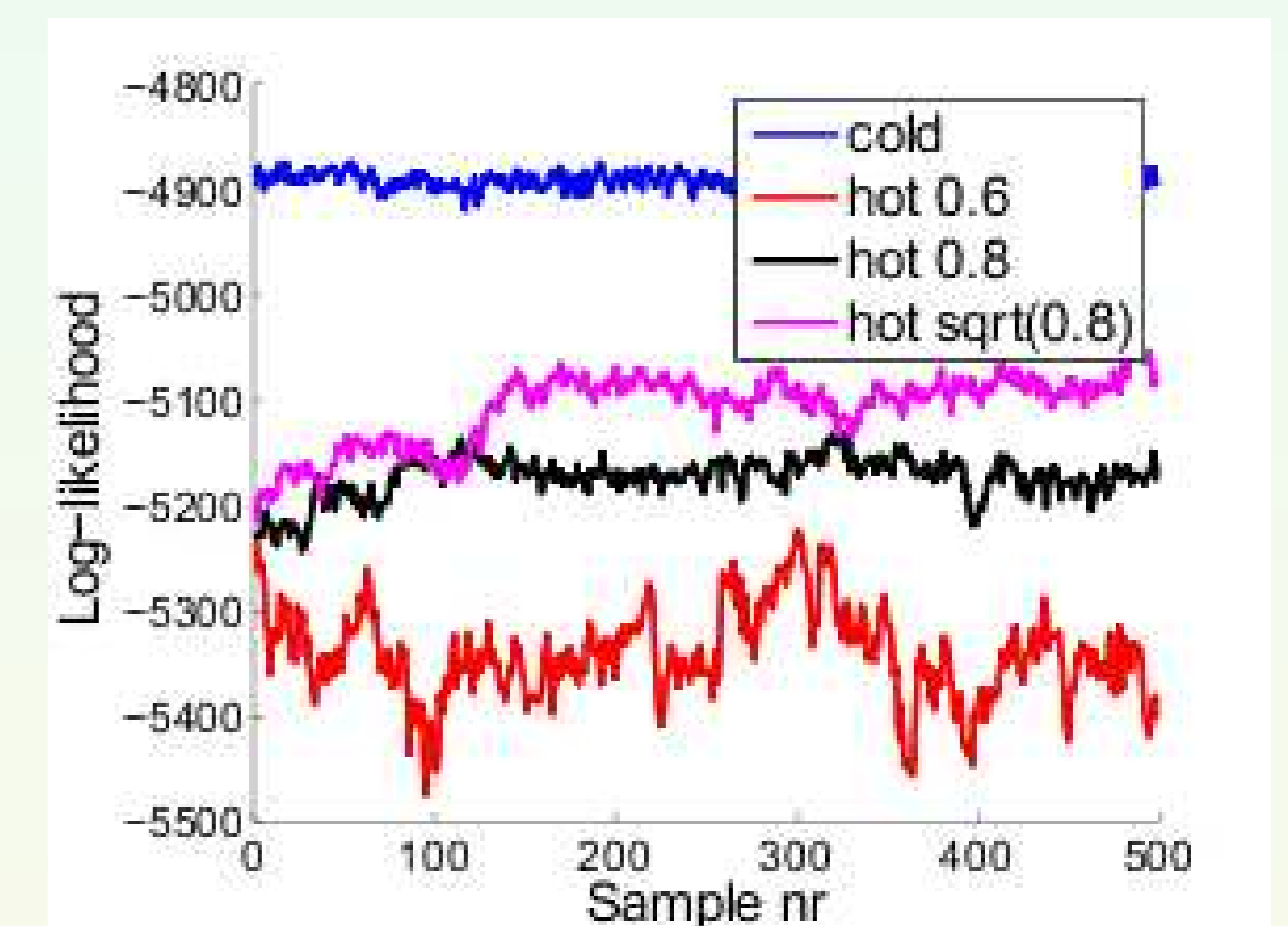
The figure shows a simple comparison of log likelihood levels for a set of chains with different heating for a well behaved data set where the heated chains have lower but stable likelihoods. A heated chain would be expected to reach similar log likelihood levels as the cold chain but exhibit larger dips in likelihood. Due to a problem with the proposal density, acceptance rates for transitions drop as temperature increases, contrary to expectations. They worked well with the cold chain but need to be flattened out for the heated chains just as the target distribution for the heated chains was. All chains implemented the rescaling of the transition matrices for the proposal HMMs in an effort to flatten out

allows the chains to move across the sample space more rapidly. Samples are not taken from the heated chains as these represent a skewed distribution. Swaps between chains facilitate longer jumps in posterior space for the cold chain and thus improves mixing. The acceptance ratio for swaps between two chains  $k$  and  $j$  is defined as follows:

$$R = \min \left[ 1, \frac{f(\psi_k | X)^{\beta_j} f(\psi_j | X)^{\beta_k}}{f(\psi_j | X)^{\beta_j} f(\psi_k | X)^{\beta_k}} \right] \quad (2)$$

The (MC)<sup>3</sup> method allows for a straightforward way of parallelising the MCMC process. We examined several different Java frameworks for both approaches. We quickly ruled out the shared memory approach since we would need to ship distributed shared memory software with our program. MPJ Express [4] was chosen as it uses the well developed MPI standard where processes have their own allocated memory and communication is through send and receive buffers. MPI adds extra overheads for communication but gains in thread safety and is also easier to implement on clusters.

proposal densities. A better way of flattening the proposal density is needed to make the heated chains usable and then further testing will be required.



## References and Acknowledgements