# GPseudoRank: MCMC for sampling from posterior distributions of pseudo-orderings using Gaussian processes

Magdalena E. Strauß, John E. Reid, and Lorenz Wernisch

30th October 2017

**Abstract**

A number of previous approaches to pseudotime estimation have provided point estimates of the ordering of cells for scRNA-seq data, while more recently, Gaussian process latent variable models and MCMC methods have been applied to understand the uncertainty associated with the pseudotemporal ordering. We present a new type of Gaussian process latent variable model for pseudotemporal ordering, which samples a distribution on the probability space of the orderings, that is on the group of permutations, rather than on the hugely high-dimensional vector space of possible pseudotimes, as done by previous models.
We determine the best proposal distribution for our Metropolis-Hastings sampler for different types of data in an extensive simulation study, and show on a microarray data set that it is both able to capture complicated posterior distributions with modes close to pseudotime estimates found by state-of-the-art methods for point estimation of pseudotime orderings, and identify a global maximum of the distribution close to the true order. Finally, in an application to scRNA-seq data we demonstrate the particular potential of our method to identify phases of lower and higher pseudotime uncertainty during a biological process.
Software in the form of Matlab code, together with sample input data sets, is available on request from the first author.

## 1 Introduction

In order to obtain information about the changes of gene activity during a process such as a response to an infection, or the transition between cellular states, the study of time trajectories of mRNA expression levels helps us infer which genes regulate the process at various stages. For instance, we might be interested whether up- or down-regulation of specific genes in mRNA expression occurs early or late in the process.

Single cell RNA sequencing (scRNA-seq) provides mRNA expression levels of genes for individual cells. For a description of the technical aspects of scRNA-seq, see [18]. scRNA-seq has shown that gene expression across cells is heterogeneous in many situations and contexts, part of which is attributable to technical noise and part of which is genuine cell hetereogeneity. See, for instance, [6, 40].

As the cells are destroyed as a result of the measurement process during scRNA-seq, the method only provides a single measurement per cell [37]. Therefore, it is not possible to obtain time series data following the development of one single cell. Typically, we obtain measurements for large numbers of cells at the same time point, or at a few different capture times. However, individual cells progress through changes at different time scales [39], and it is possible to obtain a form of time series data even from cross-sectional data by statistical means, an approach referred to as pseudotemporal ordering. Previous approaches to pseudotemporal ordering are described in Section 2.

Figure 1 is a one-dimensional illustration of the concept of pseudotime ordering. The given data are cross-sectional, with a number of cells measured at a few given capture times. The cells measured at a specific capture time are at different stages in their respective biological development. Pseudotemporal ordering estimates the underlying biological ordering of the cells statistically.

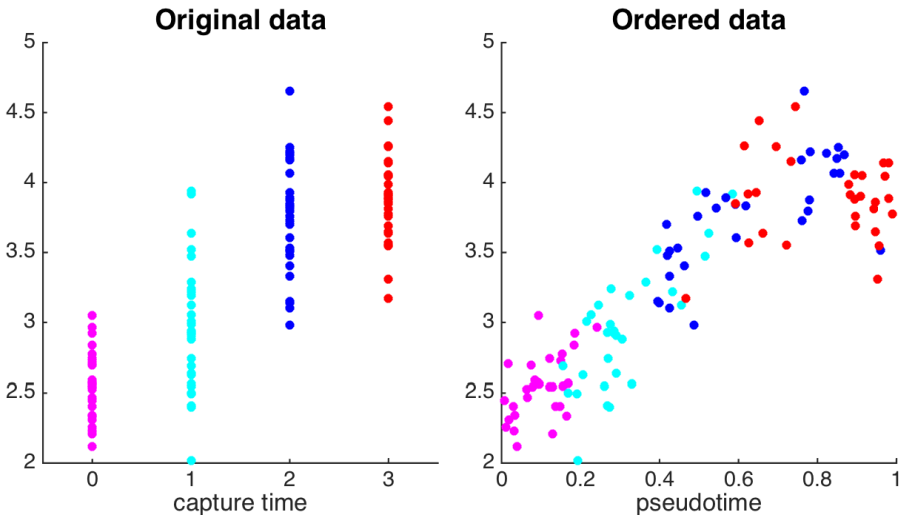


Figure 1: Comparing unordered and ordered data

There are two types of pseudotime algorithms, the first estimating the ordering of the cells in terms of a permutation of the set $\{1, \dots T\}$, where $T$ is the number of cells, the second providing estimates of the developmental stage of each individual cell in terms of a continuous pseudotime parameter. The proposed

method, GPseudoRank is the first fully Bayesian pseudotime method to sample from a posterior distribution of permutations rather than continuous pseudotimes, capturing uncertainty and even bi- and multi-modality of the orderings in terms of permutations.

## 2 Background

There are a number of previous approaches to pseudotemporal ordering. Most of them are based on considering cells as elements in the space $\mathbb{R}^{n_g}$, where $n_g$ is the number of genes. The gene expression measurements for each cell are represented as a vector in $\mathbb{R}^{n_g}$. For review papers on existing pseudotemporal methods see [3, 9].

Wanderlust [4] first computes a $k$ nearest neighbours graph for the high-dimensional data, which connects cells with short distances in $\mathbb{R}^{n_g}$, that is cells with similar gene expression profiles. Then it repeatedly applies a randomised shortest path algorithm that orders the cells, in order to obtain an average pseudotime for each cell.

Wishbone [32], based on Wanderlust, uses diffusion maps [11] to reduce the dimensionality of the data, reduce noise and avoid the problem of short circuits, that is cells being near each other in the ordering which are far from each other in terms of their developmental stages. Unlike Wanderlust, Wishbone is able to identify a branching point as well. It uses randomly sampled cells to improve an initial trajectory based on shorted paths distances. Diffusion maps have also been used for pseudotemporal ordering in other publications [2, 16, 17].

SLICER [41] also uses a $k$ nearest neighbours graph. This algorithm first applies LLE (local linear embedding) [31] to reduce the dimensionality of the space. Several methods for pseudotime estimation are based on minimal spanning trees (MST). Waterfall [34] first uses PCA to reduce the dimensionality of the gene-space from $n_g$ to 2. It then performs k-means clustering in the space of dimension 2 to group and computing an MST over the cluster centroids to find a pseudotime path. The pseudotimes of the individual cells are positioned based on their distance from the path found by computing the MST over the centroids.

CellTree [14] uses latent Dirichlet allocation (LDA) [5] in a Bayesian frame- work to construct point estimates of distance matrices between cells. LDA identifies underlying 'topics'. Different mixtures of topics explain the gene expression levels of different cells. CellTree uses Gibbs sampling to obtain histograms of the topics of the different cells, and compares these histograms to obtain the matrix of pairwise differences required for the computation of the MST, rather than propagating the uncertainty obtained through the Gibbs sampling throughout the entire algorithm. Therefore, the output of the algorithm is a point estimate.

There are several other methods using MST and clustering methods to obtain a point estimate of a pseudotemporal order. TSCAN [22] is based on the construction of an MST between centroids of clusters, and, like Waterfall, uses PCA for dimensionality reduction before performing the clustering and the ordering. Unlike Waterfall, it does not use k-means clustering but finite mixtures of Normal distributions. Mpath [10] uses MST and hierarchical clustering to find pseudotemporal orderings, constructing a network of neighbouring 'landmark' clusters, clusters of cells of a certain minimum size and purity. Another well-known method using MST and clustering is Monocle 2 [28], which applies a recent machine learning algorithm called graph structure learning [26].

All the algorithms discussed so far provide point estimates for the orders, and do not sample from a posterior distribution to quantify the uncertainty of the ordering. There are two existing methods for pseudotime estimation which use MCMC to sample from a posterior distribution [8, 30]. They use Gaussian processes (GPs, see Section 3.1) to model the data. However, the algorithms presented in [8, 30] sample from posterior distributions of pseudotime vectors $\mathbb{R}^n$, rather than sampling the ordering as a permutation.

To our knowledge there is no previous algorithm sampling directly from a posterior distribution of pseudo-orders. In addition, our approach is the first to provide a sampling strategy specifically tailored to the problem of pseudotemporal ordering, which is shown to perform well with noisy data and little information on capture times. Unlike for other discrete sampling problems with applications in computational biology, like networks or phylogenetic trees, there have not been any specific strategies developed to sample from complicated posterior distributions on a sample space of orderings to our knowledge. Moreover, our approach jointly models individual gene trajectories shares information across them, while approaches using MST or other dimensionality reduction methods are not able to borrow strength across different gene trajectories.

In summary, there are a number of existing approaches to estimating pseudotimes, but very few of them provide an estimate of the uncertainty of the the pseudotimes. The approaches that do so [8, 30] provide posterior distributions of the pseudotimes of the individual cells, and not directly of the ordering of the cells. Moreover, our model is one of very few [30] for pseudotime ordering that also models not only the uncertainty of the ordering, but also the uncertainty of the actual trajectory of the process given noisy observations. Further, our approach provides the most detailed and stringent convergence analysis.

# 3 Methods

## 3.1 Single-cell trajectories as stochastic processes

Our approach does not only order the data, but it also models the variation of the individual mRNA expression levels conditional on each ordering. In fact, our model has two components of uncertainty: first, the uncertainty of the pseudotime ordering, and second the uncertainty from the stochasticity of the process. We assume that there is an underlying stochastic process which all of the cells follow. Figure 2 illustrates the mean and standard deviation of this process. Each sample from this process is a trajectory. Figure 2 illustrates three possible trajectories from the same process. The actual measurements are then noisy observations of this trajectory. The lines represent sample trajectories of the GP, and the points in the corresponding colours are the corresponding noisy measurements.

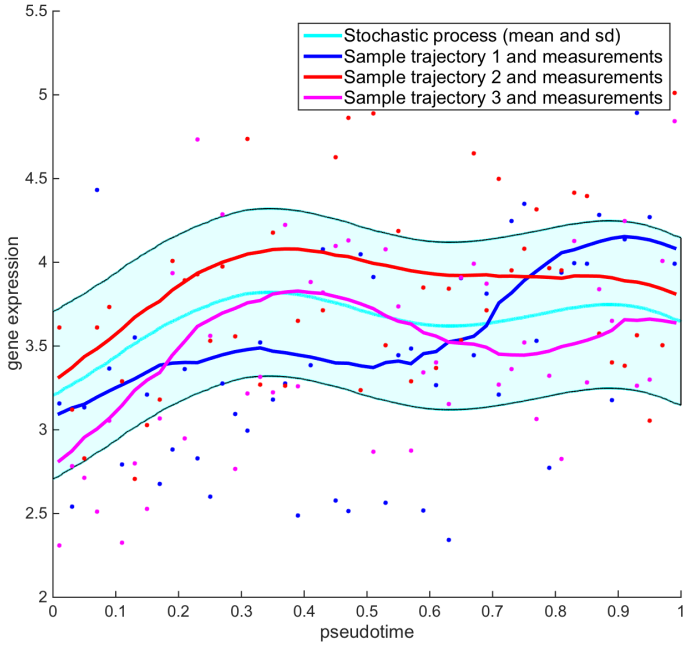


Figure 2: Single cell trajectories (lines) and noisy measurements (points)

The stochastic process we use to model the gene expression trajectories are Gaussian processes (GPs) [29]. GPs have been widely used to model time series data for biological systems in areas other than pseudotemporal ordering, in particular for the modelling of microarray time series data. For instance, there are applications to differential expression analysis for time series microarray

data [36, 23], for the modelling of gene expression profilebs in an ODE based model to identify possible targets of transcription factors [21], for the inference of gene regulatory networks [1], or to model the parametric components for time-course data in mixture models [12, 19, 20, 24].

The proposed approach uses Gaussian processes (GPs) to model the dynamics of each individual RNA expression profile, conditional on the ordering. GPs are stochastic processes defining families of functions in terms of a mean function and a covariance function. More precisely, $f \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if for any vector of input points $\boldsymbol{\tau} = \begin{pmatrix} \tau_1 & \cdots & \tau_T \end{pmatrix}$ we have $\begin{pmatrix} f(\tau_1) & \cdots & f(\tau_T) \end{pmatrix} = \mathcal{N}_T(\boldsymbol{\mu}(\boldsymbol{\tau}), \boldsymbol{\Sigma}(\boldsymbol{\tau}, \boldsymbol{\tau}))$. We use radially symmetric functions for $\boldsymbol{\Sigma}$. That is, the covariance between $f(t)$ and $f(\tilde{t})$ for any $t$ and $\tilde{t}$ depends only on a distance function $d(t, \tilde{t})$ of the input points.

We use the squared exponential covariance function, that is

$$[\boldsymbol{\Sigma}(\boldsymbol{\tau})]_{i,j} = \sigma_w^2 \exp(-\frac{d(\tau_i, \tau_j)}{2l^2}) \tag{1}$$

were $d$ refers to the Euclidean distance. We refer to $\sigma_w^2$ as the scale parameter, and to $l$ as the length scale.

We model pseudo-ordered data in terms of GPs conditional on the ordering of the cells. Let $\mathbf{y}(\mathbf{o}) = \begin{pmatrix} y_1(\mathbf{o}) \\ \vdots \\ y_n(\mathbf{o}) \end{pmatrix}$ be the $n_g$-dimensional vector of pseudo-ordered trajectories of $n_g$ genes, that is gene trajectories depending on their ordering $\mathbf{o}$. Then, conditional on the ordering, we model each of the trajectories as a GP, that is

$$y_j|\mathbf{o} \sim GP(\mathbf{o}, \boldsymbol{\Sigma}) \qquad j = 1, \ldots, n, \tag{2}$$

where, unless $\boldsymbol{\Sigma}$ is known a priori, it will depend on the ordering $\mathbf{o}$. In practice, to assume a zero-mean GP we subtract the overall mean across all genes and cells from our given matrix of gene expression levels (see Sections 3.5.5 and 3.5.6).

## 3.2 Accounting for irregular pseudotimes: pseudotime versus rank time

If we may assume pseudotime points to be approximately uniformly distributed, then the input points for the GP modelling may be chosen equidistant. We refer to this approach as rank time. However, if the rate of development during a biological process changes substantially, then the GP might not be stationary with respect to rank time, and we need to apply the concept of pseudotime as a latent variable measuring biological development [8, 30, 42]. By estimating or sampling pseudotime vectors as vectors in $\mathbb{R}^n$, existing GP methods for pseudotime estimation [8, 30, 42] provide a latent pseudotime parameter, which

measures biological development. We have extended our model to estimate both pseudotime and rank time simultaneously, for the case of pseudotime differing significantly from rank time. Thus, our approach provides both pseudotimes and rank times automatically, and if, for instance we want to compare genetic and epigenetic data from different cells undergoing the same biological process, we do not need to introduce additional steps to convert the different pseudotimes to a common timescale as in [42].

Our proposed method samples orderings directly, and we obtain a rank time $t_j \in (0,1)$ for each cell as follows: if $c_j$ is the position of cell $j$ in the ordering, then the corresponding rank time is $t_j = \frac{c_j - \frac{1}{2}}{T}$, where $T$ is the number of cells. Thus rank time is approximately uniformly distributed and therefore an approximation to linear time, rather than pseudotime. The advantage of rank time as opposed to pseudotime is the fact that it allows us to compare directly genetic versus epigenetic measurements on different cells of the same population, process, and cell type. Pseudotime measures the biological development, that is pseudotime intervals are shorter than rank time intervals during periods of slower development, whereas pseudotime intervals are longer than the corresponding rank time ones during periods of faster development. Rank time is similar to the concept of master time developed in [42], where the authors map pseudotime to corresponding uniform quantiles and then use linear interpolation or Gaussian process regression to obtain approximations to genetic and epigenetic measurements of different cells corresponding to the same uniformly distributed master time.

Our model provides pseudotime values corresponding to given orderings as follows: if $\mathbf{o} = \begin{pmatrix} o_1 \cdots o_T \end{pmatrix}$ is a given ordering, and

$$\mathbf{y}(\mathbf{o}) = \begin{pmatrix} y_1(o_1) & y_1(o_2) & \cdots & y_1(o_T) \\ \ddots & \ddots & \ddots & \ddots \\ y_n(o_1) & y_1(o_2) & \cdots & y_n(o_T) \end{pmatrix}$$

the $n \times T$ dimensional vector of ordered RNA expression levels, then we set $\tilde{\boldsymbol{\tau}} = \begin{pmatrix} \tilde{\tau}_1 \cdots \tilde{\tau}_T \end{pmatrix}$, with $\tilde{\tau}_1 = 0$ and $\tilde{\tau}_{j+1} = \tilde{\tau}_j + \|\mathbf{y}(o_{j+1}), \mathbf{y}(o_j)\|_2$, where $\mathbf{y}(o_j) = \begin{pmatrix} y_1(o_j) \\ \vdots \\ y_n(o_j) \end{pmatrix}$ and $\|.\|_2$ refers to the Euclidean norm in $\mathbb{R}^n$. Then we set $\boldsymbol{\tau} = \frac{\tilde{\boldsymbol{\tau}}}{\max(\tilde{\boldsymbol{\tau}})}$, to obtain pseudotimes in $[0,1]$.

This constructs pseudotimes as sums of Euclidean distances between neighbouring cells. This approximates geodesic distances on the underlying manifold [38].

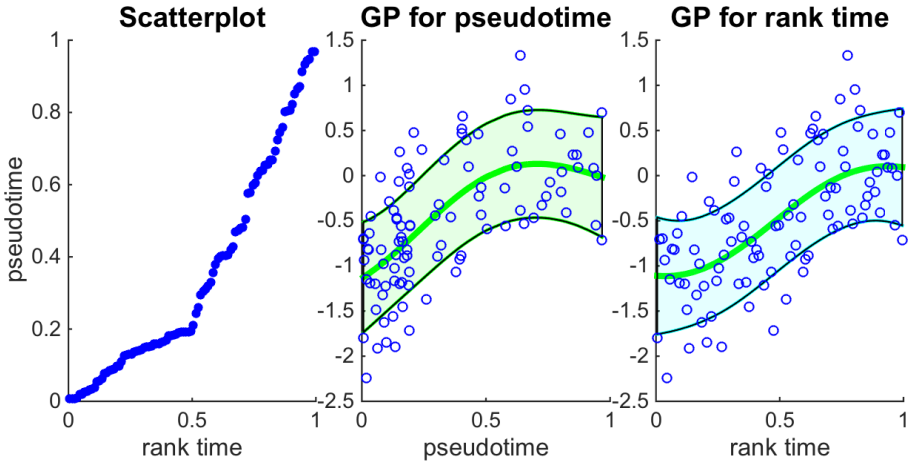


Figure 3: Rank time versus pseudotime

Figure 3 illustrates the difference between rank time and pseudotime. We use simulated data corresponding to a situation where there is slower development during the first part of the process. We see from the GP interpolation that for the rank time approximation the curve is flatter at the beginning and then becomes steeper as the second phase of faster development sets in. This effect is not there if we use pseudotime instead of rank time, as pseudotime intervals stretch and shrink to compensate for different developmental speeds.

### 3.3 Extending the model: GP parameters

For real datasets the parameters of the GPs underlying the gene expression trajectories are not known. As integrating them out would computationally be all but unfeasible, we sample them. A distribution over both orders and GP parameters is, however, almost guaranteed to be multi-modal, in particular with local modes where less likely orders are compensated by a very noisy or short length scale GP. In order to avoid this, our method uses informative priors on the GP parameters reflecting our knowledge that likely orders are associated with a higher signal-to-noise ratio than less likely orders. In addition, to avoid getting trapped in local modes with both a high scale parameter $\sigma_w^2$ and a high noise parameter $\sigma_\epsilon^2$, we sample only $\sigma_w^2$ and set $\sigma_\epsilon^2 = V - \sigma_w^2$, where $V$ is the sample variance taken across the entire $n_g \times T$ matrix of gene expression levels of $T$ cells for $n_g$ genes.

We use log-normal prior distributions for the scale parameter $\sigma_w^2$ and the length

scale $l$, and a uniform prior on the orders. Formally, our model is as follows:

$$\sigma_w^2 \sim \log\mathcal{N}(.,.) \qquad l \sim \log\mathcal{N}(.,.) \qquad \mathbf{o} \sim \text{uniform} \tag{3}$$

$$\mathbf{y}_j|\mathbf{o}, \sigma_w^2, l, \sim GP(0, \kappa(\sigma_w^2, l, V - \sigma_w^2)) \tag{4}$$

where $\kappa$ is a squared exponential covariance function for a GP (see equation 1).

Let $V$ be the overall variance of the $n_g \times T$ matrix of gene expression data. We use the following prior distributions for the GP parameters: $\log(\sqrt{\sigma_w^2}) \sim \mathcal{N}(\log(\sqrt{(0.9 \cdot V)}), 0.01)$, $\log(l) \sim \mathcal{N}(\log(\frac{1}{2}), v)$, where $v = 0.1$ for the microarray data set considered in [43] (see Section 3.5.5) and $v = 0.01$ for the scRNA-seq data set [33] (see Section 3.5.6). The stronger prior for the scRNA-seq data set was chosen to account for the high noise levels and the fact that the prior with high signal and low noise may otherwise lead to getting trapped in local modes with very short length scales and orders that would be unlikely given longer length scales.

## 3.4 Fully Bayesian approaches and MCMC

We use a Bayesian approach for inference, as we believe that a full understanding of the uncertainty of the ordering of the cells is desirable, as different orderings will lead to different gene trajectories, which in turn will lead to different inferences about differential expression and gene regulatory networks. We use a Bayesian approach and an inference method which samples from the posterior distribution of the model. MCMC is probably the most widely used method for this.

## 3.5 Data sets

### 3.5.1 Simulation studies

The following simulation studies illustrate the efficacy of the individual moves and of combinations of different moves for different types of data. We simulate 50 genes with 90 cells. For each of the simulation studies we generate 16 data sets. On each of these data sets we run MCMC chains using all the possible combinations of our 4 proposed moves. For the chains with combinations of more than one moves, we apply each of them with equal probability.

We perform 3 different simulations to illustrate the strengths of the individual moves, as different moves are particularly powerful in different situations. The first simulation is relatively informative data, with detailed information on capture times and a substantial, though not very high, noise level. The second simulation has fewer capture times, and the third is more noisy data.

To focus on the convergence of the orders and on their proposal distribution, we use the known true parameters in the model, and do not sample them. We

use regular spaced pseudotimes, as we simulated them from a uniform $U(0,1)$ distribution.

We run all the chains without adaptation of the proposal distribution, which, provided we have found reasonably good fixed parameters for the proposal moves, has not shown to accelerate convergence in general.

#### 3.5.2 Simulation 1

We generate 16 data sets as follows: For the GP parameters, we draw 16 samples each from the following distributions:

$$\log(\sigma_w) \sim \mathcal{N}(0, 0.1) \qquad \log(L) \sim \mathcal{N}(\log(0.4), 0.1)$$
$$\log(\sigma_\epsilon) \sim \mathcal{N}(\log(\frac{1}{\sqrt{2}}, 0.1).$$

Using the parameters sampled above, we simulate 50 orderings with 90 input points each from the 16 zero-mean GP with the squared exponential covariance matrix specified by the 16 different sets of GP parameters. The input points are drawn from a uniform distribution $U(0,1)$. For this simulation, we assume three capture times, with the first 30 cells drawn from the first capture time, the second 30 cells from the second capture times, and the remaining 30 cells from the last capture time.

#### 3.5.3 Fewer capture times

To gain better understanding of how the moves perform in different situations relevant to the analysis of single-cell data, we next look at how the different combinations of moves perform when there are fewer capture times. The setup is identical to simulation 1, however instead of three capture times with 30 cells each, we only have two capture times, with the first 30 cells belonging to the first, and the remaining 60 belonging to the second capture time.

#### 3.5.4 Higher noise

Finally, as single-cell data tends to be quite noisy, we also explore the performance of the individual moves in the presence of higher noise levels. We use the same number of capture times, cells and genes as in simulation 1. We sample $L$ and $\sigma_w$ for the generation of simulated data from the same distribution as with simulation 1, but this time we sample $\sigma_\epsilon$ from the same distribution as $\sigma_w$, which means that on average we have a signal-to-noise ratio of 1.

#### 3.5.5 Microarray data

To validate the proposed algorithm, we apply it to a data set with known true order, the whole leaf Arabidopsis thaliana microarray data set [43]. [43] studied the response of Arabidopsis thaliana to infection by the fungal pathogen Botrytis cinerea, generating microarray time series data over 48 h, with measurements at intervals of 2h. We compare our distribution of posterior orders to the true order and to estimates produced by two established pseudotime methods, TSCAN [22] and SLICER [41]. We subtract from the data its sample mean across all cells and genes. For the analysis with TSCAN and SLICER we use the standard settings implemented for the respective algorithm, and we use the 150 genes mentioned in the paper by Windram et al. [43]. With SLICER, we tried several different values for the number of edges of the nearest neighbours graph in the low dimensional space, and found that the values leading to the order closest to the true one are 4 and 5. Like Reid and Wernisch [30], we group the known 24 capture times into 4 groups of 6, assuming that we do not have any further information, in order to test the algorithm.

#### 3.5.6 scRNA-seq data

Shalek et al. [33] examined the response of primary mouse bone-marrow-derived dendritic cells in three different conditions using single-cell RNA- seq. We apply our model to the data on the lipopolysaccharide stimulated (LPS) condition. Shalek et al. [33] identified four modules of genes. For the ordering, we use a total of 74 genes from the four modules with the highest temporal variance relative to their noise levels [30]. The number of cells is 307, with 49 unstimulated cells, 75 captured after 1h, 65 after 2h, 60 after 4h, and 58 after 6h. We subtract from the data its sample mean across all cells and genes.

### 3.6 The proposal distribution for the orderings

We apply the following five moves, each with probability $p_j$, $j = 1, \ldots, 5$:

1. Move 1- iterated swaps of neighbouring cells: we draw a number $r_1$ uniformly from $\mathcal{U}(1, n_0)$, where $T$ is the number of cells, and $n_0$ is a constant. Then we draw $r_1$ cell positions $P_1, \ldots, P_r$ from $\mathcal{U}(1, T-1)$ with replacement, and iterate the following for $j = 1, \ldots, r_1$: We swap the cell currently at position $P_j$ with its current right-hand neighbour. For instance, if $P_k = 1, 2, 3, 4, 5, 6, 7, 8$ is the current ordering, then with $r_1 = 2$, the next state could be $P_{k+1} = 1, 3, 2, 4, 5, 7, 6, 8$. However, there could also be an overlap, for instance $P_{k+1} = 1, 3, 4, 2, 5, 6, 7, 8$.

   By default, we set $n_0 = \lfloor \frac{T}{4} \rfloor$, and we use this default setting for the simulation studies, but we often use a different value to optimise the acceptance rate.

Adaptive version: During a short period at the beginning, we adjust $n_0$ to achieve an optimal acceptance rate between 0.2 and 0.3, but requiring $\lfloor \frac{T}{3} \rfloor \leq n_0 \leq \lfloor \frac{T}{30} \rfloor$. For many data sets the acceptance rate for this move will therefore be above 0.3.

2. Move 2-swaps of cells with short $L^1$-distances: we compute a distribution over the pairs of cells proportional to a squared exponential of the $L^1$-distance between the cells, that is $p_{ij} \propto \exp(-\frac{d(c_i,c_j)^2}{\gamma_1})$, where $d$ refers to the $L_1$ distances to cells as elements in $\mathbb{R}^{n_g}$, where $n_g$ is the number of genes. We then sample one pair of cells from $p_{ij}$, to swap them. If $P_k = 1,2,3,4,5,6,7,8$ is the current ordering, and cells 2 and 5 are close in terms of their $L^1$-distance, then the proposed ordering could be $P_{k+1} = 1,5,3,4,2,6,7,8$.

   Adaptive version: During a short period of the beginning, we temper or anneal the distribution $f$ from which we sample the pairs of cells by sampling from $f^a$ instead of $f$, in case of very high or low acceptance rates.

3. Move 3 - reversals: again we compute a distribution over the pairs of cells proportional to a squared exponential of the $L^1$-distance between the cells, that is $p_{ij} \propto \exp(-\frac{d(c_i,c_j)^2}{\gamma_2})$. Now the proposed move is to reverse the ordering between these two sampled cells, where the reversal includes the pair of cells sampled. If $P_k = 1,2,3,4,5,6,7,8$ is the current ordering, and cells 2 and 5 are close in terms of their $L^1$-distance, then the proposed ordering could be $P_{k+1} = 1,5,4,3,2,6,7,8$.

   Adaptive version: During a short period of the beginning, we temper or anneal the distribution from which we sample the pairs of cells in case of very high or low acceptance rates.

4. Move 4-short permutations: for a fixed $n_3$ we draw a number $r_2$ uniformly from $\mathcal{U}(1,n_3)$. For each $j = 1,\ldots,r_2$, we draw a number $r_{3,j}$ uniformly from $\mathcal{U}(3,\max(n_{3a},3))$ for a fixed $n_{3a}$, and a cell position $k$ uniformly from $\mathcal{U}(1,T-r_3)$. We then randomly permute the cells at positions $k,\ldots,k+r_3$. If $P_k = 1,2,3,4,5,6,7,8$ is the current ordering, $r_2 = 2$, $r_{3,1} = 3$, $r_{3,2} = 4$, then the proposed ordering could be $P_{k+1} = 3,1,2,4,7,8,5,6$, where we randomly permuted $1,2,3$ and $5,6,7,8$. There could also be an overlap of the two or more permutations, for instance $P_{k+1} = 3,7,4,2,1,8,5,6$.

   By default, we set $n_3 = \lfloor \frac{T}{20} \rfloor$, and $n_{3a} = \lfloor \frac{T}{12} \rfloor$.

   Adaptive version: At the beginning, we adapt $n_3$ and $n_{3a}$ in case of acceptance rates above 0.3 or below 0.1, but we require $\lfloor \frac{T}{50} \rfloor \leq n_3 \leq \lfloor \frac{T}{10} \rfloor$ and $4 \leq n_{3a} \leq \lfloor \frac{T}{10} \rfloor$.

5. Move 5-reversing the entire ordering: with a low probability $p_5$ we reverse the entire ordering, which is accepted with probability 1. This move allows us to exploit the symmetry of our posterior distribution for convergence

assessment. If $P_k = 1, 2, 3, 4, 5, 6, 7, 8$ is the current ordering, then the proposed ordering would be $P_{k+1} = 8, 7, 6, 5, 4, 3, 2, 1$.

For the simulation studies we apply all possible combinations of moves 1-4. For the microarray data we apply one move 3, as it was shown to be the best sampling strategy for multi-modal distributions. For the scRNA-seq data set, we use all the moves. For the microarray data set we use $\gamma_2 = 1000$ and an additional tempering factor $a = 0.1$. For the scRNA-seq data set we set $\gamma_1 = \gamma_2 = 4000$, without any tempering factor for moves 2 or 3.

### 3.7 Order symmetry and MCMC parameters

As our posterior distribution is a symmetric function of the order, each order and its reverse will be sampled with equal probability from the posterior distribution. However, we are only interested in those orders that correlate positively with the capture times of the cells. Therefore, we reverse the orders negatively correlated with the capture times.

For the simulated and the microarray data sets we run chains for 100,000 iterations and apply a thinning factor of 10. For the scRNA-seq data we use the same thinning factor but 500,00 iterations. For each of the data sets, we run 12 chains from random starting orders and with random GP parameters sampled from the prior distribution. For the starting orders we restrict the randomness to permuting the data randomly within capture times, but not across capture times.

### 3.8 Assessment of MCMC convergence

MCMC convergence needs to be checked rigorously for the following two reasons. First, we need to check that our proposal distribution effectively reaches all the regions of high density of the posterior distribution, and does not get trapped around one single mode. Second, we need to make sure that we have run our chain long enough for it to have converged to its invariant distribution. The Gelman-Rubin $\hat{R}$-statistic is an established method to assess convergence. It was first proposed in [15], and later corrected to account for sampling variability [7]. The $\hat{R}$-statistic requires several MCMC chains with over-dispersed initial states, and estimates the potential scale reduction factor, the factor by which the pooled variance across all the chains is larger than the within-sample variance. For convergent chains, $\hat{R}$ approaches 1 as the number of samples tends to infinity. According to [7], we may assume that convergence has been reached if $\hat{R} < 1.2$ for all model parameters. We use the more stringent recommendation to run MCMC-chains until a value of 1.1 has been reached for the $\hat{R}$-statistic [35].

Apart from determining the length of the burn-in and checking whether a set of MCMC chains have converged, we may also use the $\hat{R}$-statistic to assess the

speed of convergence for different proposal distributions. More previously, for different values of $\epsilon > 0$, we check whether the $\hat{R}$ reaches below $1+\epsilon$ for a given number of samples. A different way of assessing the speed of convergence is to measure the number of samples required for the chain to reach below $1 + \epsilon$.

#### 3.8.1 Comparison of move efficiency

To compare the efficacy of the individual moves described in Section 3.6 in the simulation setting described in Section 3.5.1, we construct for each of the 16 simulated data sets 5 different starting orderings (one for each simulated data set) by permuting the cells within, but not across, the three capture times. For each combination of moves, and each of the 16 simulated data sets, we run 5 chains starting from the 5 starting orderings for 100,000 iterations, with a thinning factor of 10. We compute for each combination of moves and for each of the 16 sets of 5 chains the Gelman-Rubin $\hat{R}$-statistic as in [7] on the log-likelihood as follows: For each batch for which we compute the $\hat{R}$-statistic on only the second half, discarding the first 50% as burn-in. We first compute the statistic after 50 thinned samples, which would correspond to 500 samples without the thinning. Then we recompute the statistic after each additional 20 thinned samples. That is, we obtain values for the statistic at the following thinned sample numbers: 51, 71, 91, 101, ... . To compute the Gelman-Rubin statistic as in [7], we use the R-package coda citecoda.

For each ordering in each chain we compute the corresponding positions of the cells, that is the inverse permutation of the given ordering. Then we compute for each sample the $L^1$-distance of the corresponding cell positions to the reference positions $1 : 100$. For each combination of moves and each of the 16 sets of 5 chains we compute the $\hat{R}$-statistic on these $L^1$-distances. Again we compute the $\hat{R}$-statistic with the same intervals as above and discarding the first 50% of each batch as burn-in. Thereby, for each combination of moves and for each of the 16 simulations, we obtain two curves of $\hat{R}$-statistics, one for the log-likelihood and one for the $L^1$-distances.

We assess the moves according to the following criteria: Whether they have converged before 10,000 thinned samples according to the $\hat{R}$-statistic, at different levels. That is, we check whether for all the 16 simulations for a given combination of moves the chains have converged at the 1.1, 1.07, 1.05, and 1.02 levels. For the moves for which there is convergence at a given level we compute the average number of thinned samples to convergence, where the average is based on computations of the $\hat{R}$-statistic at intervals of 20. We also compute the maximum number of thinned samples to convergence, again based on computations of the $\hat{R}$-statistic at intervals of 20.

# 4 Results

## 4.1 Results of the simulation studies

### 4.1.1 Simulation study 1

For the following combinations of moves the chains on all simulated data sets have an $\hat{R} < 1.02$ before the last sample, both for the log-likelihood and the $L^1$-distances: 4; 1,4; 2,4; 1,3,4; 1,2,3,4; Simulation study 1 therefore indicates that these combinations of moves are the most suitable ones for sampling from data sets with medium, but not very high, noise levels, and with relatively good information on capture times. The worst performing combinations of moves with these sets of simulated data is clearly move 3, for which some of the 16 data sets have $\hat{R}$-statistics which remain above the 1.07 threshold. Among the best performing moves, it is less clear-cut which one is the combination of moves leading the fastest convergence, as the sample size of our quite extensive simulation study is still limited. However, assigning ranks to a number of criteria, we see that move 4 is ranked first in seven of these criteria. For a more detailed analysis, see in the supplementary materials. Figure 4 compares the best and the worst performing move in the set-up of simulation 1. We plotted the Gelman-Rubin statistics for the log-likelihoods for all the 16 simulated data sets.

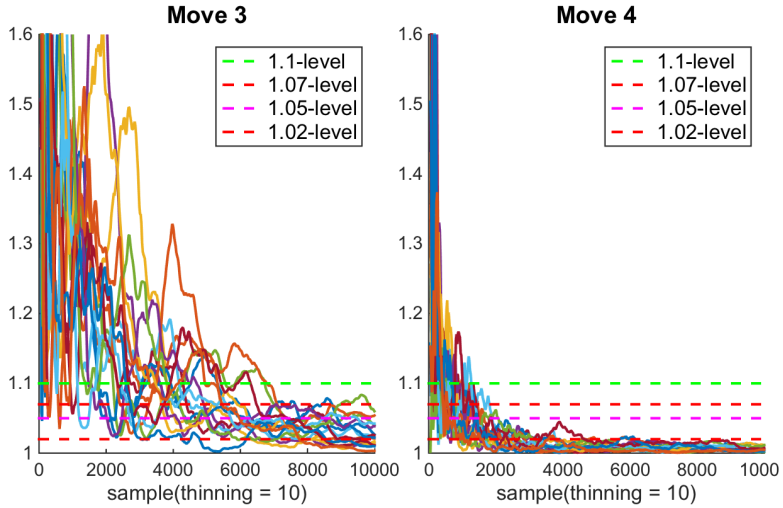


Figure 4: The worst and the best performing move in simulation 1, $\hat{R}$ for log-likelihood

#### 4.1.2 Simulation 2: fewer capture times

The performance of the different combinations of different moves is very different to what we saw in simulation 1. In fact, as illustrated by Figure 5, when comparing the performance of single moves, we now see that move 3 performs better than any other single move.

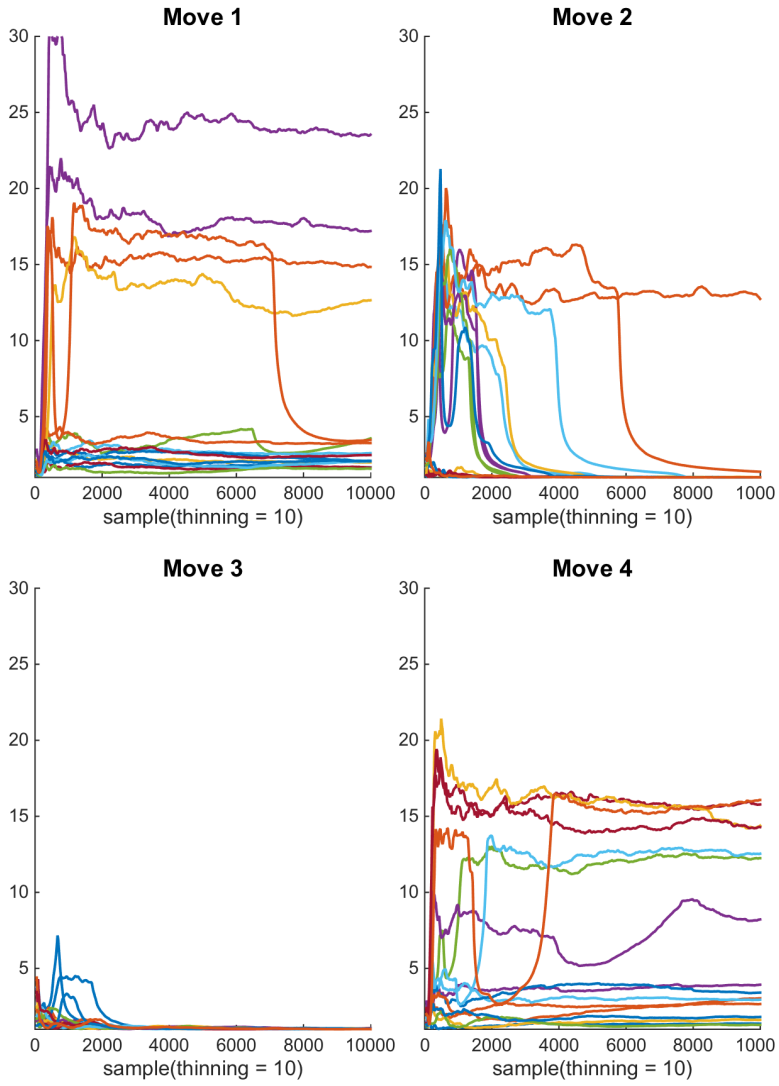


Figure 5: Performance of individual moves with fewer capture times, $\hat{R}$ for log-likelihood

In fact, move 3 is the only move for which all $\hat{R}$-statistics go below 1.1 within the first 10,000 thinned samples. With move 1 and move 4, the $\hat{R}$-statistic does not go below 1.1 for any single simulated data set. Fortunately, we can improve convergence in this case by combining at least three moves, as long as they do not include both move 1 and move 4. In fact, the only combination of moves for which not for a single one of the simulated data sets the $\hat{R}$-statistics gets below 1.1 is the combination of moves 1 and 4, which performed well in simulation 1, when we had more capture times. Fortunately, very reasonable levels of convergence are still attained with combinations of moves more suited to the challenge of fewer capture times. In fact, for the following combinations of moves, all the $\hat{R}$-statistics are below 1.05 by the last sample: 1,3; 2,3; 1,2,3; 2,3,4; 1,2,3,4

The combination of moves that is ranked first according to the largest number of criteria is the combination 1,2,3,4 of all the moves. See table 2 in Section A.1 of the supplementary materials for details on the performance of each individual combination of moves. See Figures 15 and 18 for a visual comparison of all the combinations of moves in terms of the $\hat{R}$ statistic for both the log-likelihood and the $L^1$-differences.

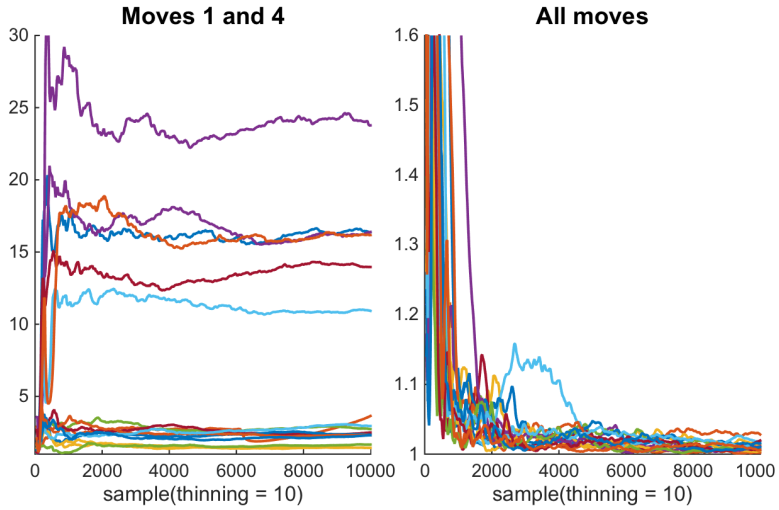


Figure 6: Fewer capture times: worst and best combination of moves

Figures comparing the performance of all the 15 possible combinations of the four moves can be found in Section A.1 in the supplementary materials, where we also provide a table (table 1) listing the performance of the combinations of moves on our criteria outlined in this Section and provide plots not only of the $\hat{R}$-statistics for the log-likelihood, but also for the $L^1$-differences from the

reference permutation.

#### 4.1.3 Simulation 3: higher noise

Our moves still perform well in this situation, with the exception of move 3, which does not even reach the 1.1 level for the Gelman-Rubin statistic for all the simulated data sets. The moves which are ranked the highest in the largest number of our categories are move 4, and the combination 1,4 (see table 3 in Section A.1 of the supplementary materials for details). The following other combinations of moves also provide convergence for all the simulated data sets at the 1.05 - level: 2; 1,2; 2,4; 3,4 and all combinations of at least three moves. Again, we refer the reader to the supplementary materials for detailed tables and Figures.

#### 4.1.4 Adaptive moves

For this simulation study we used suitable parameters for the definition of the moves, and did not adapt them during a fixed number of iterations at the beginning. We repeated all the simulations using adaptations of the moves at the beginning as described in Section 3.6. However, our results indicate that convergence tends to be better without the adjustment, provided that parameters for the moves with good acceptance rates can be found, for instance by running the sampler for a short number of iterations with different settings. A detailed comparison without the non-adaptive and the adaptive version can be found in table 4 in Section A.1 of the supplementary materials.

### 4.2 Validation on microarray data

We first apply GPseudoRank with irregular pseudotimes. The irregular pseudotimes reflect the different speed of biological development during different phases of the process, even if we know that the real time points are equidistant with 2h each between them. We again use the Gelman-Rubin statistic to assess convergence (see Figure 7).

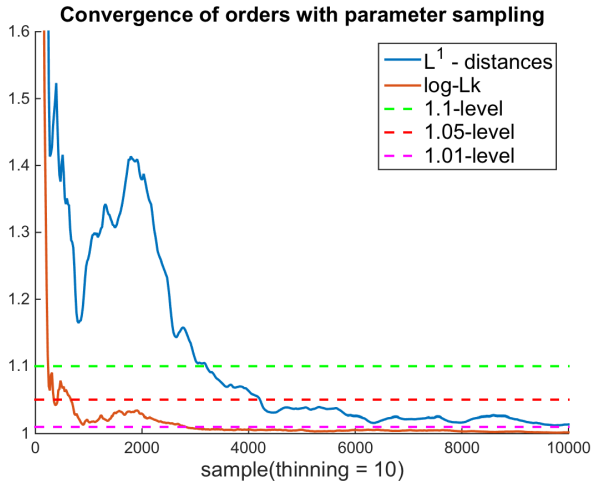


Figure 7: Convergence analysis for GPseudoRank: Windram data

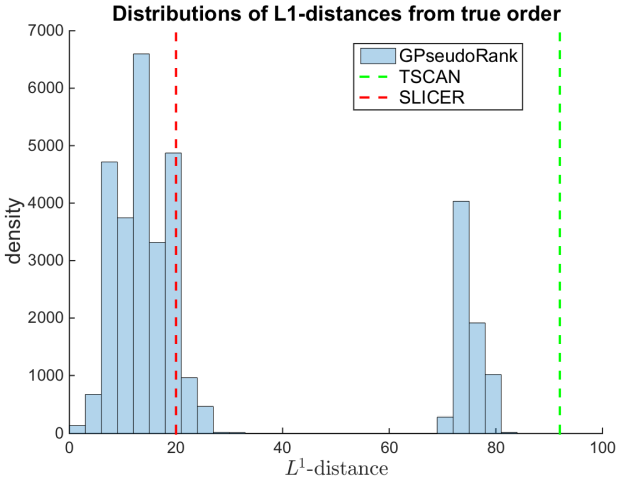


Figure 8: $L^1$-distances from the real order: Windram data

We plotted the Gelman-Rubin statistic for both the log-likelihood and the $L^1$-distances from the true order. From Figure 7 we see that it would be easily sufficient to run the chain for 36,000 samples, that is 3,600 thinned samples, as by that number both of the $\hat{R}$ statistics have reached below 1.1. Discarding half of the 3,600 thinned samples as burn-in as recommended in [35], we discard a burn-in of 1,800 samples.

We also plotted the $L^1$-distances from the real order for GPseudoRank, as well

as for two algorithms providing point estimates. As illustrated by Figure 8 the distribution of GPseudoRank is bi-modal, while the estimates provided by SLICER [41] and TSCAN [22] are both close to one of the modes of the distribution of GPseudoRank. This illustrates how important is it to sample from the distribution of the orderings rather than just obtaining a point estimate. In practice, when we do not know the true order and two ordering algorithms provide very different point estimates, then we do not know which one is the closer to the true underlying biological order. Therefore, it is preferable to quantify the uncertainty, rather than only providing point values.

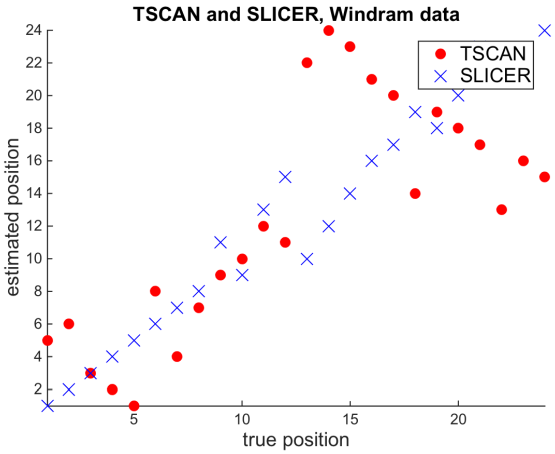


Figure 9: TSCAN and SLICER: Windram data

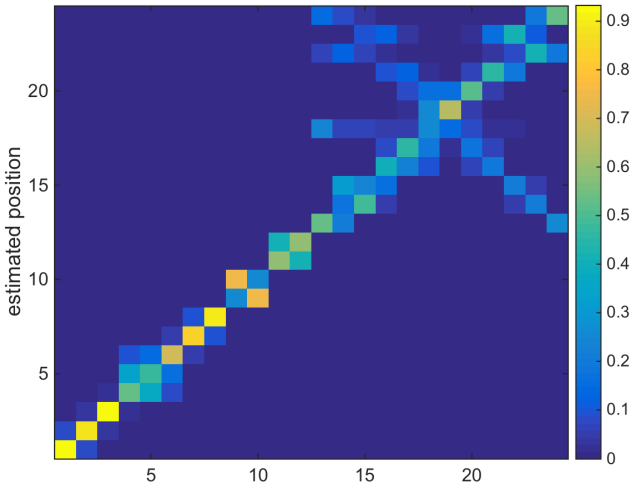


Figure 10: Distribution of cell positions: Windram data

Figures 9 and 10 also illustrate the differences between point estimates for the order and the sampling from a distribution. The Figures compare the true order of the cell along the x-axis with its positions (TSCAN, SLICER) or the distribution of its position (GPseudoRank) estimated by the models.

The run time of GPseudo rank with irregular pseudotimes for 100,000 iterations for the Windram data is about 7min 20s on an Intel Xeon X5 2.0GHz CPU.

## 4.3 Pseudotemporal uncertainty varies during a response to an infection

We again check convergence carefully using the $\hat{R}$-statistics on both the log-likelihood and the $L^1$-distances from a reference order (figure 11). This time the true order is unknown. We could take any fixed reference order, so we use $1:307$.

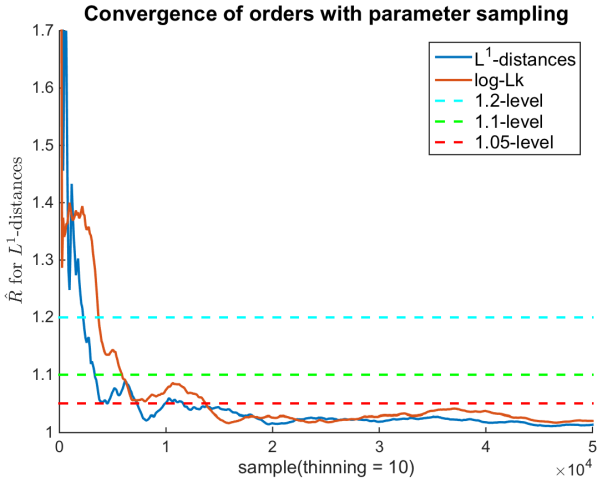


Figure 11: Convergence analysis for GPseudoRank: Shalek data

The run time of GPseudo rank for 500,000 iterations for the Shalek data is about 3 hours on an Intel Xeon X5 2.0GHz CPU. It should be noted that because of the focus of this paper on showing the good convergence properties of our method we run the chain for long to show that high levels of convergence can be achieve. However, according to the recommendation to run chains until an $\hat{R} < 1.2$ has been reached ([35]), 10,000 thinned samples, that is a total of 100,000 samples would easily be sufficient even with a stricter threshold of 1.1. We discard a burn-in of 5,000 thinned samples at the beginning of each chain.

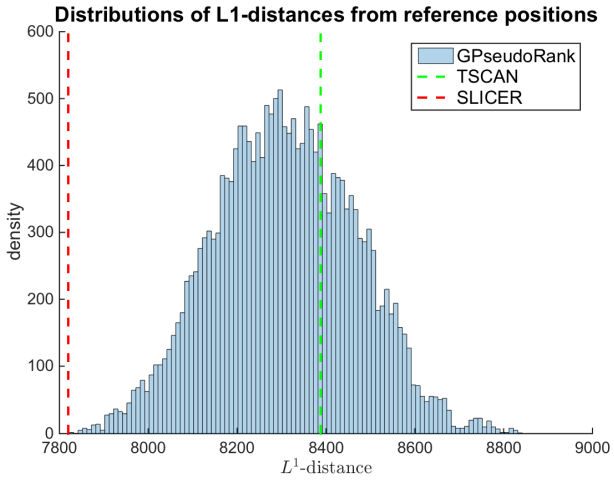


Figure 12: $L^1$-distances from the real order: Shalek data

Figure 12 underlines again that it is important to provide a posterior distribution over the orders, rather than a point estimate, as again the two methods for point estimation give rather different results, and not knowing the uncertainty will lead to over-confidence in the results. Of course, now with the true order not known, it is not clear which of the two point estimates is closer to the truth.

In addition to providing the uncertainty of the orders, our approach also shows how this uncertainty varies during the course of the process.

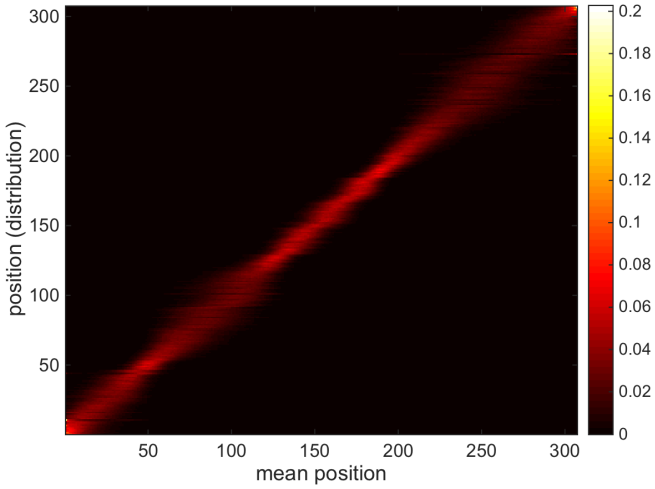


Figure 13: Uncertainty of the ordering as a function of pseudotime: Shalek data

Figure 13 illustrates a particular insight we gain using GPseudoRank. The Figure shows how the distribution of the cell position changes as a function of the mean position. While at the beginning and end of the process the standard deviation of the marginal distributions of the individual cell positions is high, it decreases a lot during the middle of the process, when the response that takes place in reaction to the infection has set in but is not yet completed.

# 5 Discussion

## 5.1 Possible alternatives to MCMC

There are, of course, alternatives to MCMC for sampling from posterior distributions, in particular importance sampling and extensions to it such as annealed importance sampling [27], or annealed SMC sampling [13]. However, the geometry of our posterior distribution makes it all but impossible to implement importance sampling. While annealed SMC sampling could be an interesting alternative for future work, we have implemented a proposal distribution for MCMC that is particularly suitable to our problem and has good convergence properties even if the data are very noise or little is known about their capture times.

## 5.2 Two-step approaches to branching processes

One drawback of our model is that, like the other pseudotime models using GPs, it is not able to model branching processes in a one-step process. How- ever, it should be noted that while models using GPs do not directly infer the branching structure of a branching process, they have been used for variational inference in a two-step process that first infers the pseudotimes and then identifies the branches [25]. Along the same lines, our model could be used to provide a full posterior distribution of the pseudotimes, which could be used as a basis to infer branches depending on the pseudotimes, although this would be computationally more expensive than the variational approach. Alternatively, we could first use a well-established existing method that provides a point estimate of the branching structure, such as TSCAN or SLICER, and then use the method proposed in this paper to provide full inference of the posterior distribution of the pseudotimes of each branch, an approach that is likely to be more efficient computationally.

# 6 Conclusions

We presented a new type of Gaussian process latent variable model for pseudotemporal ordering, which samples a distribution on the probability space of the orderings, that is on the group of permutations, rather than on the hugely high-dimensional vector space of possible pseudotimes, as done by previous models. We have implemented a Metropolis-Hastings sampler on our sample space, which is still very large and difficult to sample from. We found it necessary to develop novel moves in order to explore the posterior effectively. Our proposal distribution allows the sampler to make long distance moves in this space with a good acceptance rate.

An extensive simulation study examined how different combinations of moves perform in different situations relevant to the analysis of single-cell data. Combinations of all the four moves proposed perform well in all the situations considered. We therefore generally recommend to use a combination of all four moves. Combinations of three moves also perform well, except for those containing both moves 1 and 4, which do not perform well with fewer capture times. If we have several capture times with not very many cells each, then move 4, on its own or combined with move 1, will probably provide the fastest convergence. However, even if there are several capture times, there tends to be an overlap between them with real data, as opposed to the situation we simulated. In fact, towards the end of a biological process, the last two or so capture times might overlap considerably (see Section 4.3). This means that de facto we have one capture time less.

Important insights are also gained from the validation of our algorithm on microarray data. As illustrated by Figure 8, point estimates of cell orderings, even if computed using state-of-the-art pseudotime estimation methods, only provide estimates near one mode of the distribution of the orderings. Even if for this data set we know the real order, orders near the second mode are still likely orders, and with a data set without a known true order it would be important to sample the full distribution, in particular for a data set where the two modes are so far apart. This illustrates the particular importance of methods accounting for uncertainty and possible multi-modality in pseudotime ordering.

Finally, an application to an scRNA-seq data set illustrates the good convergence properties of our algorithm, and a comparison to point estimation methods of orderings illustrates once more the need for a method of estimating orderings that fully captures the uncertainty in a Bayesian way. One particular interesting aspect of the uncertainty of the ordering captured by our algorithm is the fact that the uncertainty is lowest during the middle of the process, where the heterogeneity of cells with regard to their progress through the response to the infection is highest.

# References

[1] Tarmo Äijö and Harri Lähdesmäki. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22):2937–2944, 2009. doi: 10.1093/bioinformatics/btp511.

[2] P Angerer, L Haghverdi, M Büttner, FJ Theis, C Marr, and D Buettner. destiny: diffusion maps for large-scale single-cell data in R. *Bioinformatics*, 32(8):1241–1243, 2016. doi: 10.1093/bioinformatics/btv715.

[3] AC Babtie, TE Chan, and MPH Stumpf. Learning regulatory models for cell development from single cell transcriptomic data. *Current Opinion in Systems Biology*, 5(Supplement C):72–81, 2017. ISSN 2452-3100. doi: https://doi.org/10.1016/j.coisb.2017.07.013. URL `http://www.sciencedirect.com/science/article/pii/S2452310017301257`.

[4] SC Bendall, KL Davis, el-AD Amir, MD Tadmor, EF Simonds, TJ Chen, DK Shenfeld, GP Nolan, and D Pe'er. Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development. *Cell*, 157(3):714–725, 2014. doi: 10.1016/j.cell.2014.04.005.

[5] DM Blei, AY Ng, and MI Jordan. Latent Dirichlet allocation. *J Mach Learn Res*, 3, 2003.

[6] P Brennecke, S Anders, JK Kim, AA Kolodziejczyk, X Zhang, V Proserpio, et al. Accounting for technical noise in single-cell RNA-seq experiments. *Nat Meth*, 10 (11):1093–1095, 2013. URL `http://dx.doi.org/10.1038/nmeth.2645`.

[7] SP Brooks and A Gelman. General methods for monitoring convergence of iterative simulations. *J Comput Graph Stat*, 7(4):434–455, 1998. doi: 10.1080/10618600.1998.10474787.

[8] KR Campbell and C Yau. Order under uncertainty: Robust differential expression analysis using probabilistic models for pseudotime inference. *PLOS Comput Biol*, 12(11):1–20, 11 2016. doi: 10.1371/journal.pcbi.1005212. URL `https://doi.org/10.1371/journal.pcbi.1005212`.

[9] R Cannoodt, W Saelens, and Y Saeys. Computational methods for trajectory inference from single-cell transcriptomics. *Eur J Epidemiol*, 46(11): 2496–2506, 2016. ISSN 1521-4141. doi: 10.1002/eji.201646347. URL `http://dx.doi.org/10.1002/eji.201646347`.

[10] J Chen, A Schlitzer, S Chakarov, F Ginhoux, and M Poidinger. Mpath maps multi-branching single-cell trajectories revealing progenitor cell progression during development. *Nat Commun*, 7:11988, 2016. doi: 10.1038/ncomms11988.

[11] RR Coifman, S Lafon, AB Lee, M Maggioni, B Nadler, F Warner, and SW Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *P Natl Acad Sci USA*, 102(21):7426–7431, 2005. doi: 10.1073/pnas.0500334102.

[12] EJ Cooke, RS Savage, PDW Kirk, R Darkins, and DL Wild. Bayesian hierarchical clustering for microarray time series data with replicates and outlier measurements. *BMC Bioinformatics*, 12(1):399, Oct 2011. doi: 10.1186/1471-2105-12-399.

[13] P Del Moral, A Doucet, and A Jasra. Sequential Monte Carlo samplers. *J R Stat Soc B*, 68(3):411–436, 2006. doi: 10.1111/j.1467-9868.2006.00553.x.

[14] DA duVerle, S Yotsukura, S Nomura, H Aburatani, and K Tsuda. Celltree: an R/bioconductor package to infer the hierarchical structure of cell populations from single-cell rna-seq data. *BMC Bioinformatics*, 17(1):363, Sep 2016. doi: 10.1186/s12859-016-1175-6.

[15] A Gelman and DB Rubin. Inference from iterative simulation using multiple sequences. *Stat Sci*, 7(4):457–472, 1992. URL `http://www.jstor.org/stable/2246093`.

[16] L Haghverdi, F Buettner, and FJ Theis. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, 31(18):2989–2998, 2015.

[17] L Haghverdi, M Buttner, FA Wolf, F Buettner, and FJ Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nat Meth*, 13(10):845–848, 2016. doi: 10.1038/nmeth.3971.

[18] D Hebenstreit. Methods, challenges and potentials of single cell RNA-seq. *Biology*, 1(3):658–667, 2012. doi: 10.3390/biology1030658.

[19] J. Hensman, ND Lawrence, and M Rattray. Hierarchical bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BMC Bioinformatics*, 14(1):252, Aug 2013. doi: 10.1186/1471-2105-14-252. URL `https://doi.org/10.1186/1471-2105-14-252`.

[20] J Hensman, M Rattray, and ND Lawrence. Fast nonparametric clustering of structured time-series. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99), 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2318711.

[21] A Honkela, C Girardot, EH Gustafson, Y-H Liu, EEM. Furlong, ND Lawrence, and M Rattray. Model-based method for transcription factor target identification with limited data. *P Natl Acad Sci USA*, 107(17):7793–7798, 2010. doi: 10.1073/pnas.0914285107.

[22] Z Ji and H Ji. TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. *Nucleic Acids Res*, 44(13):e117–e117, 2016.

[23] AA Kalaitzis and ND. Lawrence. A simple approach to ranking differentially expressed gene expression time courses through gaussian process regression. *BMC Bioinformatics*, 12(1):180, May 2011. doi: 10.1186/1471-2105-12-180.

[24] PDW Kirk, JE Griffin, RS Savage, Z Ghahramani, and DL Wild. Bayesian correlated clustering to integrate multiple datasets. *Bioinformatics*, 28(24):3290–3297, 2012. doi: 10.1093/bioinformatics/bts595.

[25] T. Lönnberg et al. Single-cell RNA-seq and computational analysis using temporal mixture modeling resolves TH1/TFH fate bifurcation in malaria. 2(9), 2017. doi: 10.1126/sciimmunol.aal2192.

[26] Q Mao, S Wang, Liand Goodison, and Y Sun. Dimensionality reduction via graph structure learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 765–774, New York, NY, USA, 2015. ACM. doi: 10.1145/2783258.2783309.

[27] RM Neal. Annealed importance sampling. *Stat Comput*, 11(2):125–139, 2001. doi: 10.1023/A:1008923215028.

[28] X Qiu, Q Mao, Y Tang, L Wang, R Chawla, HA Pliner, and C Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nat Meth*, advance online publication:–, 2017. URL `http://dx.doi.org/10.1038/nmeth.4402`.

[29] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.

[30] JE Reid and L Wernisch. Pseudotime estimation: deconfounding single cell time series. *Bioinformatics*, 32(19):2973–2980, 2016. doi: 10.1093/bioinformatics/btw372.

[31] ST Roweis and LK Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323.

[32] M Setty, MD Tadmor, S Reich-Zeliger, O Angel, TM Salame, P Kathail, et al. Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nat Biotech*, 34(6):637–645, 2016.

[33] AK Shalek et al. Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature*, 498(7453):236–240, 2013. doi: 10.1038/nature12172.

[34] J Shin, DA Berg, Y Zhu, JY Shin, J Song, MA Bonaguidi, et al. Single-cell RNA-seq with Waterfall reveals molecular cascades underlying adult neurogenesis. *Cell Stem Cell*, 17:360–372, 2015. doi: 10.1016/j.stem.2015.07.013.

[35] K Shirley. *Inference from Simulations and Monitoring Convergence*, chapter 6, pages 163–174. Chapman and Hall/CRC, 2011. doi: doi:10.1201/b10905-7.

[36] O Stegle, Denby KJ, EJ Cooke, DL Wild, Z Ghahramani, and KM Borgwardt. A robust Bayesian two-sample test for detecting intervals of differential gene expression in microarray time series. *J Comput Biol*, 17(3):355–367, 2010. doi: doi: 10.1089/cmb.2009.0175.

[37] O Stegle, SA Teichmann, and JC Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nat Rev Genet*, 16(3):133–145, 2015. doi: 10.1038/nrg3833.

[38] JB Tenenbaum, V de Silva, and JC Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[39] C Trapnell et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nat Biotechnol*, 32(4):381–386, 2014.

[40] CA Vallejos, John C Marioni, and S Richardson. BASiCS: Bayesian analysis of single-cell sequencing data. *PLOS Comput Biol*, 11(6):1–18, 2015. doi: 10.1371/journal.pcbi.1004333. URL `https://doi.org/10.1371/journal.pcbi.1004333`.

[41] JD Welch, AJ Hartemink, and JF Prins. SLICER: inferring branched, nonlinear cellular trajectories from single cell rna-seq data, May 2016.

[42] JD Welch, AJ Hartemink, and JF Prins. MATCHER: manifold alignment reveals correspondence between single cell transcriptome and epigenome dynamics. *Genome Biol*, 18(1):138, 2017. doi: 10.1186/s13059-017-1269-0.

[43] O Windram et al. Arabidopsis defense against botrytis cinerea: Chronology and regulation deciphered by high-resolution temporal transcriptomic analysis. *The Plant Cell Online*, 24(9):3530–3557, 2012. doi: 10.1105/tpc.112.102046. URL `http://www.plantcell.org/content/24/9/3530`.

# A Supplementary material

## A.1 Detailed results of the simulation studies

We assessed the 15 possible combinations of our four moves according to the following:

- The proportion out of the 16 Gelman-Rubin statistics for the log-likelihood and the $L^1$-distances from a reference permutation
- Whether or not all the chains have converged by the 10,000th thinned sample at the 1.1, 1.07, 1.05, and 1.02 levels in terms of both the log-likelihood and the distance function
- The average number of samples till convergence (note that we compute the $\hat{R}$ statistics in intervals of 20 for the thinned samples)
- The maximum number of samples till convergence for the 16 simulated data sets (note again that the $\hat{R}$ statistics are in intervals of 20)

Tables 1, 2, and 3 list the values of these criteria for each combination of moves for the three different scenarios explored in our simulation studies. For each criterion, the best performing combination is marked in magenta.

Figures 14, 15 and 16 illustrate the performance of the different combinations of moves in terms of the $\hat{R}$ statistic of the log-likelihood. Each line in the plots represents the $\hat{R}$- statistic that belongs to one of the 16 simulated data sets. Figures 17, 18 and 19 do the same for the $\hat{R}$ for the distance function.

Finally, table 4 compares the non-adaptive simulations with adaptive ones, where we adapt the parameters for the proposal moves at every 50th iteration during the first 5,000 samples.

Table 1: Simulation 1, GR statistics on log-likelihood and $L^1$-distances from reference positions

| **moves** | **1** | **2** | **3** | **4** | **1,2** | **1,3** | **1,4** | **2,3** | **2,4** | **3,4** | **1,2,3** | **1,2,4** | **1,3,4** | **2,3,4** | **1,2,3,4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GR<1.01 at end, logLk | 0.75 | 0.31 | 0.13 | 0.81 | 0.75 | 0.56 | 0.94 | 0.38 | 0.81 | 0.69 | 0.56 | 0.81 | 0.94 | 0.44 | 0.63 |
| GR<1.01 at end, L1-dist | 0.63 | 0.38 | 0 | 0.88 | 0.5 | 0.56 | 0.94 | 0.19 | 0.88 | 0.69 | 0.44 | 0.69 | 0.88 | 0.69 | 0.81 |
| GR<1.02 at end, logLk | 0.94 | 0.75 | 0.38 | 1 | 1 | 0.81 | 1 | 0.75 | 1 | 0.94 | 0.94 | 0.94 | 1 | 0.94 | 1 |
| av. # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | 1314 | 2014 | NaN | 1379 | NaN | 1705 | NaN | NaN | NaN | 1561 | NaN | 1619 |
| max # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | 2571 | 3651 | NaN | 7391 | NaN | 3511 | NaN | NaN | NaN | 3231 | NaN | 3051 |
| GR<1.02 at end, L1-dist | 0.94 | 0.81 | 0.31 | 1 | 0.94 | 1 | 1 | 0.56 | 1 | 0.94 | 0.94 | 1 | 1 | 0.94 | 1 |
| av. # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | 1240 | NaN | 3025 | 1759 | NaN | 2056 | NaN | NaN | 2051 | 1855 | NaN | 2230 |
| max # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | 2811 | NaN | 5731 | 4571 | NaN | 6051 | NaN | NaN | 5751 | 3631 | NaN | 5331 |
| GR<1.05 at end, logLk | 1 | 1 | 0.88 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.05,logLk | 676 | 1630 | NaN | 539 | 881 | 834 | 435 | 2011 | 837 | 772 | 1039 | 766 | 710 | 1349 | 1026 |
| max # thin. samp. to GR<1.05,logLk | 1131 | 4071 | NaN | 1571 | 1491 | 1691 | 991 | 5351 | 1591 | 1491 | 3511 | 1671 | 1491 | 3691 | 2771 |
| GR<1.05 at end, L1-dist | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.05,L1-dist | 1212 | 1149 | NaN | 539 | 1056 | 1271 | 729 | 1312 | 889 | 1114 | 1080 | 906 | 817 | 777 | 1104 |
| max # thin. samp. to GR<1.05,L1-dist | 2891 | 2611 | NaN | 1611 | 2931 | 3011 | 1891 | 2191 | 2031 | 2451 | 1971 | 1991 | 1771 | 2191 | 2791 |
| GR<1.07 at end, logLk | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.07,logLk | 557 | 585 | 2456 | 402 | 610 | 597 | 382 | 1369 | 482 | 660 | 820 | 610 | 574 | 946 | 565 |
| max # thin. samp. to GR<1.07,logLk | 1051 | 1311 | 5431 | 771 | 1111 | 1651 | 831 | 3331 | 851 | 1291 | 2211 | 1331 | 1291 | 2031 | 1491 |
| GR<1.07 at end, L1-dist | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.07,L1-dist | 884 | 875 | NaN | 485 | 770 | 801 | 585 | 1056 | 625 | 639 | 804 | 486 | 605 | 632 | 716 |
| max # thin. samp. to GR<1.07,L1-dist | 2131 | 1831 | NaN | 1511 | 2211 | 1991 | 1351 | 1991 | 1391 | 1471 | 1771 | 1431 | 1071 | 2051 | 2631 |
| GR<1.1 at end, logLk | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,logLk | 435 | 502 | 1864 | 246 | 377 | 376 | 287 | 812 | 330 | 500 | 521 | 475 | 449 | 605 | 396 |
| max # thin. samp. to GR<1.1,logLk | 851 | 1191 | 4171 | 471 | 911 | 931 | 611 | 2131 | 791 | 1051 | 1291 | 991 | 791 | 1231 | 1071 |
| GR<1.1 at end, L1-dist | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,L1-dist | 571 | 620 | 1536 | 427 | 376 | 517 | 390 | 837 | 494 | 441 | 564 | 355 | 436 | 437 | 500 |
| max # thin. samp. to GR<1.1,L1-dist | 1851 | 1131 | 3451 | 1371 | 1031 | 951 | 1131 | 1591 | 911 | 1191 | 1311 | 1011 | 951 | 831 | 1051 |
| av. AR | 0.58 | 0.25 | 0.21 | 0.43 | 0.41 | 0.4 | 0.51 | 0.23 | 0.34 | 0.32 | 0.36 | 0.42 | 0.42 | 0.29 | 0.37 |
| min. AR | 0.52 | 0.22 | 0.19 | 0.38 | 0.39 | 0.37 | 0.45 | 0.21 | 0.32 | 0.3 | 0.34 | 0.39 | 0.39 | 0.27 | 0.35 |
| max. AR | 0.66 | 0.28 | 0.23 | 0.52 | 0.46 | 0.44 | 0.59 | 0.26 | 0.4 | 0.37 | 0.39 | 0.48 | 0.48 | 0.32 | 0.42 |

Table 2: Simulation with fewer capture times

| **moves** | **1** | **2** | **3** | **4** | **1,2** | **1,3** | **1,4** | **2,3** | **2,4** | **3,4** | **1,2,3** | **1,2,4** | **1,3,4** | **2,3,4** | **1,2,3,4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GR<1.01 at end, logLk | 0 | 0.44 | 0.06 | 0 | 0.56 | 0.56 | 0 | 0.25 | 0.25 | 0.63 | 0.69 | 0.31 | 0.63 | 0.44 | 0.56 |
| GR<1.01 at end, L1-dist | 0 | 0.63 | 0.13 | 0 | 0.44 | 0.63 | 0 | 0.31 | 0.56 | 0.75 | 0.75 | 0.44 | 0.5 | 0.81 | 0.56 |
| GR<1.02 at end, logLk | 0 | 0.75 | 0.25 | 0 | 0.75 | 0.88 | 0 | 0.75 | 0.69 | 0.94 | 0.88 | 0.56 | 0.69 | 0.88 | 0.94 |
| av. # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| max # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| GR<1.02 at end, L1-dist | 0 | 0.81 | 0.44 | 0 | 0.75 | 0.94 | 0 | 0.69 | 0.63 | 0.88 | 0.94 | 0.56 | 0.69 | 1 | 0.88 |
| av. # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2515 | NaN |
| max # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 4831 | NaN |
| GR<1.05 at end, logLk | 0 | 0.88 | 0.88 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.05,logLk | NaN | NaN | NaN | NaN | NaN | 1784 | NaN | 1667 | NaN | NaN | 1267 | NaN | NaN | 992 | 1002 |
| max # thin. samp. to GR<1.05,logLk | NaN | NaN | NaN | NaN | NaN | 7291 | NaN | 3591 | NaN | NaN | 2391 | NaN | NaN | 2771 | 1871 |
| GR<1.05 at end, L1-dist | 0 | 0.88 | 0.88 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.05,L1-dist | NaN | NaN | NaN | NaN | NaN | 1721 | NaN | 1856 | NaN | NaN | 1142 | NaN | NaN | 1224 | 1054 |
| max # thin. samp. to GR<1.05,L1-dist | NaN | NaN | NaN | NaN | NaN | 6071 | NaN | 3671 | NaN | NaN | 4331 | NaN | NaN | 2451 | 1971 |
| GR<1.07 at end, logLk | 0 | 0.88 | 0.94 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.07,logLk | NaN | NaN | NaN | NaN | NaN | 1549 | NaN | 1069 | NaN | NaN | 955 | NaN | NaN | 866 | 836 |
| max # thin. samp. to GR<1.07,logLk | NaN | NaN | NaN | NaN | NaN | 7251 | NaN | 3351 | NaN | NaN | 1551 | NaN | NaN | 1571 | 1751 |
| GR<1.07 at end, L1-dist | 0 | 0.88 | 0.94 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.07,L1-dist | NaN | NaN | NaN | NaN | NaN | 1594 | NaN | 1335 | NaN | NaN | 787 | NaN | NaN | 932 | 827 |
| max # thin. samp. to GR<1.07,L1-dist | NaN | NaN | NaN | NaN | NaN | 5431 | NaN | 3591 | NaN | NaN | 1531 | NaN | NaN | 2311 | 1311 |
| GR<1.1 at end, logLk | 0 | 0.88 | 1 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.1,logLk | NaN | NaN | 1637 | NaN | NaN | 1319 | NaN | 759 | NaN | NaN | 759 | NaN | NaN | 657 | 595 |
| max # thin. samp. to GR<1.1,logLk | NaN | NaN | 3211 | NaN | NaN | 7171 | NaN | 2491 | NaN | NaN | 1371 | NaN | NaN | 1291 | 1571 |
| GR<1.1 at end, L1-dist | 0 | 0.88 | 1 | 0 | 0.75 | 1 | 0 | 1 | 0.69 | 0.94 | 1 | 0.56 | 0.69 | 1 | 1 |
| av. # thin. samp. to GR<1.1,L1-dist | NaN | NaN | 1641 | NaN | NaN | 1157 | NaN | 871 | NaN | NaN | 729 | NaN | NaN | 767 | 679 |
| max # thin. samp. to GR<1.1,L1-dist | NaN | NaN | 3211 | NaN | NaN | 4871 | NaN | 1771 | NaN | NaN | 1331 | NaN | NaN | 1871 | 1171 |
| av. AR | 0.49 | 0.25 | 0.21 | 0.37 | 0.41 | 0.4 | 0.43 | 0.23 | 0.34 | 0.32 | 0.36 | 0.41 | 0.41 | 0.29 | 0.37 |
| min. AR | 0.38 | 0.22 | 0.19 | 0.3 | 0.34 | 0.37 | 0.34 | 0.21 | 0.3 | 0.3 | 0.34 | 0.35 | 0.31 | 0.27 | 0.35 |
| max. AR | 0.6 | 0.28 | 0.23 | 0.47 | 0.46 | 0.44 | 0.59 | 0.26 | 0.39 | 0.37 | 0.39 | 0.48 | 0.48 | 0.32 | 0.41 |

Table 3: Simulation with higher noise levels

| **moves** | **1** | **2** | **3** | **4** | **1,2** | **1,3** | **1,4** | **2,3** | **2,4** | **3,4** | **1,2,3** | **1,2,4** | **1,3,4** | **2,3,4** | **1,2,3,4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GR<1.01 at end, logLk | 0.63 | 0.38 | 0 | 0.88 | 0.38 | 0.31 | 0.69 | 0.25 | 0.56 | 0.75 | 0.44 | 0.56 | 0.5 | 0.5 | 0.44 |
| GR<1.01 at end, L1-dist | 0.5 | 0.5 | 0 | 0.88 | 0.63 | 0.25 | 0.69 | 0.44 | 0.5 | 0.56 | 0.56 | 0.44 | 0.38 | 0.5 | 0.5 |
| GR<1.02 at end, logLk | 0.69 | 0.69 | 0.19 | 0.94 | 0.88 | 0.69 | 1 | 0.56 | 1 | 0.94 | 0.94 | 1 | 0.94 | 0.94 | 0.75 |
| av. # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | 1586 | NaN | 1816 | NaN | NaN | 2024 | NaN | NaN | NaN |
| max # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | 3191 | NaN | 5011 | NaN | NaN | 7351 | NaN | NaN | NaN |
| GR<1.02 at end, L1-dist | 0.88 | 0.88 | 0.13 | 0.94 | 0.81 | 0.63 | 0.94 | 0.81 | 0.94 | 1 | 0.75 | 0.94 | 0.94 | 0.88 | 0.88 |
| av. # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2596 | NaN | NaN | NaN | NaN | NaN |
| max # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 5091 | NaN | NaN | NaN | NaN | NaN |
| GR<1.05 at end, logLk | 0.94 | 1 | 0.75 | 1 | 1 | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.05,logLk | NaN | 2549 | NaN | 769 | 1150 | 1640 | 824 | NaN | 970 | 1340 | 1346 | 977 | 989 | 1572 | 1012 |
| max # thin. samp. to GR<1.05,logLk | NaN | 6651 | NaN | 1691 | 2031 | 9591 | 1571 | NaN | 2611 | 2791 | 2991 | 2191 | 2331 | 4631 | 2351 |
| GR<1.05 at end, L1-dist | 1 | 1 | 0.69 | 1 | 1 | 0.88 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.05,L1-dist | 1357 | 1780 | NaN | 875 | 1146 | NaN | 824 | 1759 | 974 | 1065 | 1216 | 1249 | 949 | 1430 | 1482 |
| max # thin. samp. to GR<1.05,L1-dist | 3271 | 3751 | NaN | 1791 | 2131 | NaN | 1931 | 3911 | 2131 | 2531 | 2091 | 3331 | 3651 | 3071 | 3011 |
| GR<1.07 at end, logLk | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.07,logLk | 594 | 1266 | NaN | 579 | 856 | 1444 | 730 | 1515 | 525 | 791 | 892 | 700 | 811 | 1085 | 647 |
| max # thin. samp. to GR<1.07,logLk | 1391 | 3791 | NaN | 1511 | 1431 | 9291 | 1431 | 3111 | 1251 | 1691 | 1851 | 1691 | 1791 | 2091 | 1391 |
| GR<1.07 at end, L1-dist | 1 | 1 | 0.81 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.07,L1-dist | 1029 | 1289 | NaN | 661 | 901 | 1610 | 601 | 1135 | 637 | 835 | 971 | 844 | 582 | 1156 | 1161 |
| max # thin. samp. to GR<1.07,L1-dist | 2491 | 2951 | NaN | 1691 | 1631 | 8671 | 1731 | 2571 | 1511 | 1571 | 1591 | 1691 | 1091 | 2251 | 2431 |
| GR<1.1 at end, logLk | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,logLk | 419 | 810 | NaN | 380 | 564 | 1154 | 531 | 1077 | 427 | 592 | 785 | 615 | 586 | 739 | 527 |
| max # thin. samp. to GR<1.1,logLk | 831 | 1571 | NaN | 691 | 1111 | 8591 | 1391 | 1851 | 1211 | 1291 | 1611 | 1631 | 1631 | 1331 | 1331 |
| GR<1.1 at end, L1-dist | 1 | 1 | 0.94 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,L1-dist | 689 | 857 | NaN | 512 | 676 | 1346 | 435 | 977 | 505 | 567 | 731 | 622 | 480 | 729 | 844 |
| max # thin. samp. to GR<1.1,L1-dist | 2311 | 1711 | NaN | 1011 | 1151 | 8051 | 891 | 2351 | 1451 | 971 | 1311 | 1511 | 891 | 1371 | 2291 |
| av. AR | 0.69 | 0.46 | 0.41 | 0.56 | 0.57 | 0.55 | 0.63 | 0.44 | 0.51 | 0.49 | 0.53 | 0.57 | 0.57 | 0.47 | 0.53 |
| min. AR | 0.66 | 0.42 | 0.37 | 0.54 | 0.54 | 0.52 | 0.6 | 0.39 | 0.48 | 0.46 | 0.49 | 0.54 | 0.54 | 0.44 | 0.5 |
| max. AR | 0.74 | 0.49 | 0.45 | 0.63 | 0.6 | 0.57 | 0.69 | 0.47 | 0.54 | 0.52 | 0.55 | 0.61 | 0.61 | 0.5 | 0.56 |

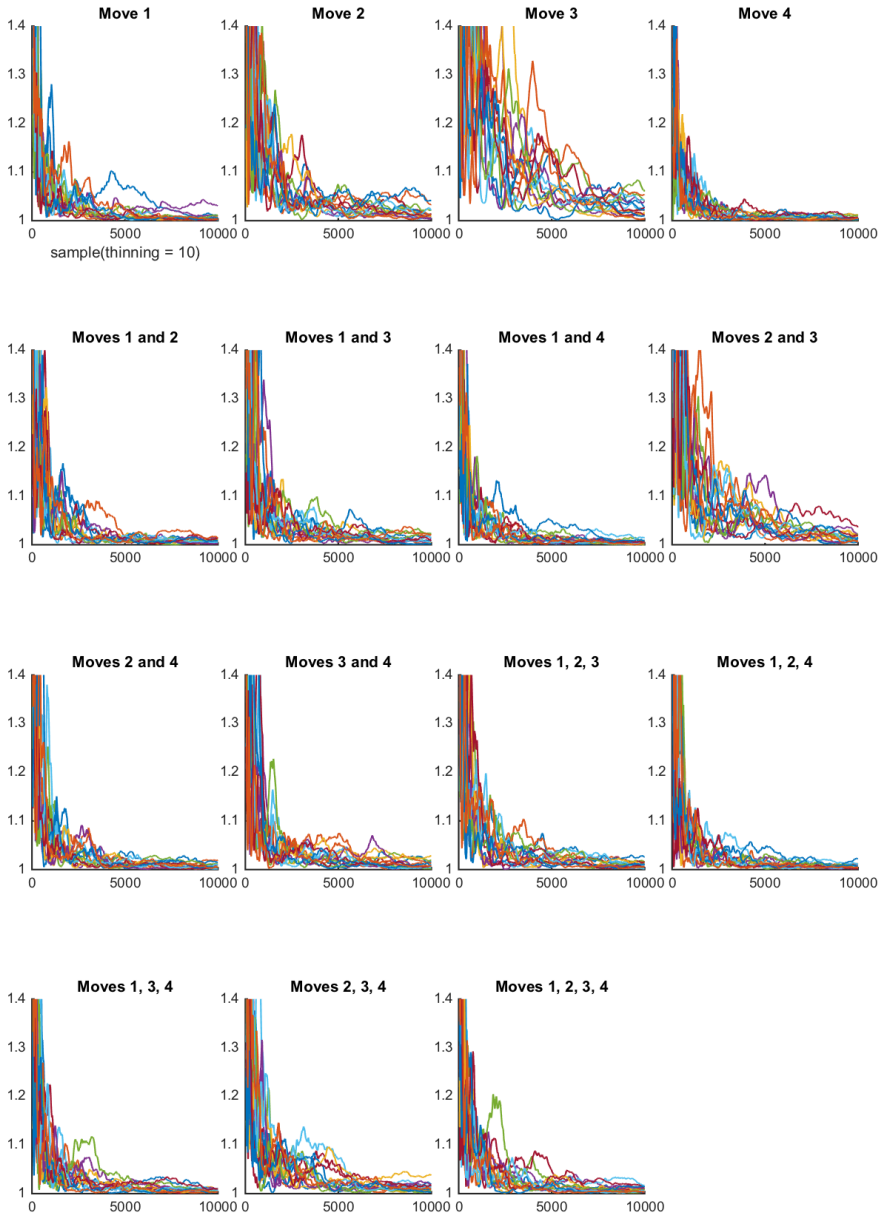


Figure 14: Simulation 1: GR statistic for log-likelihood

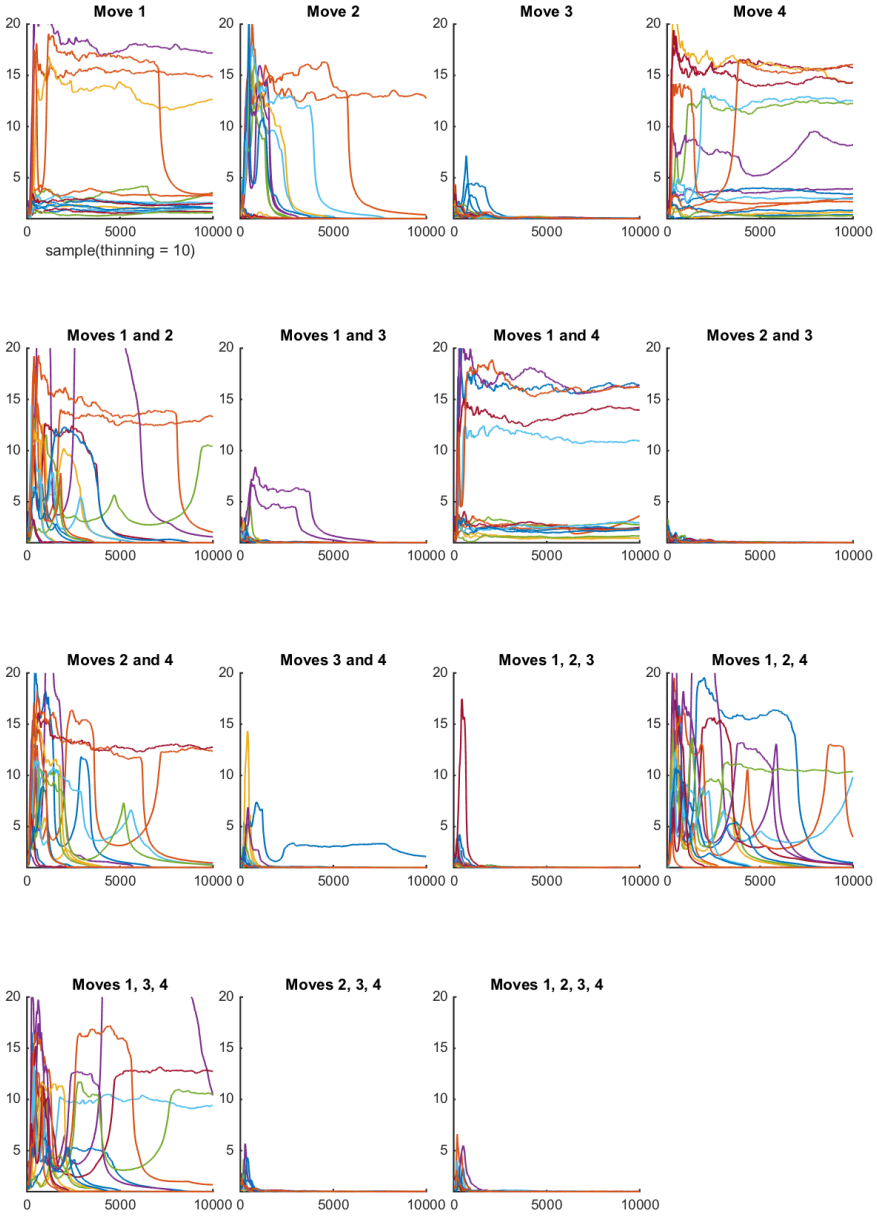


Figure 15: Simulation with fewer capture times: GR statistic for log-likelihood

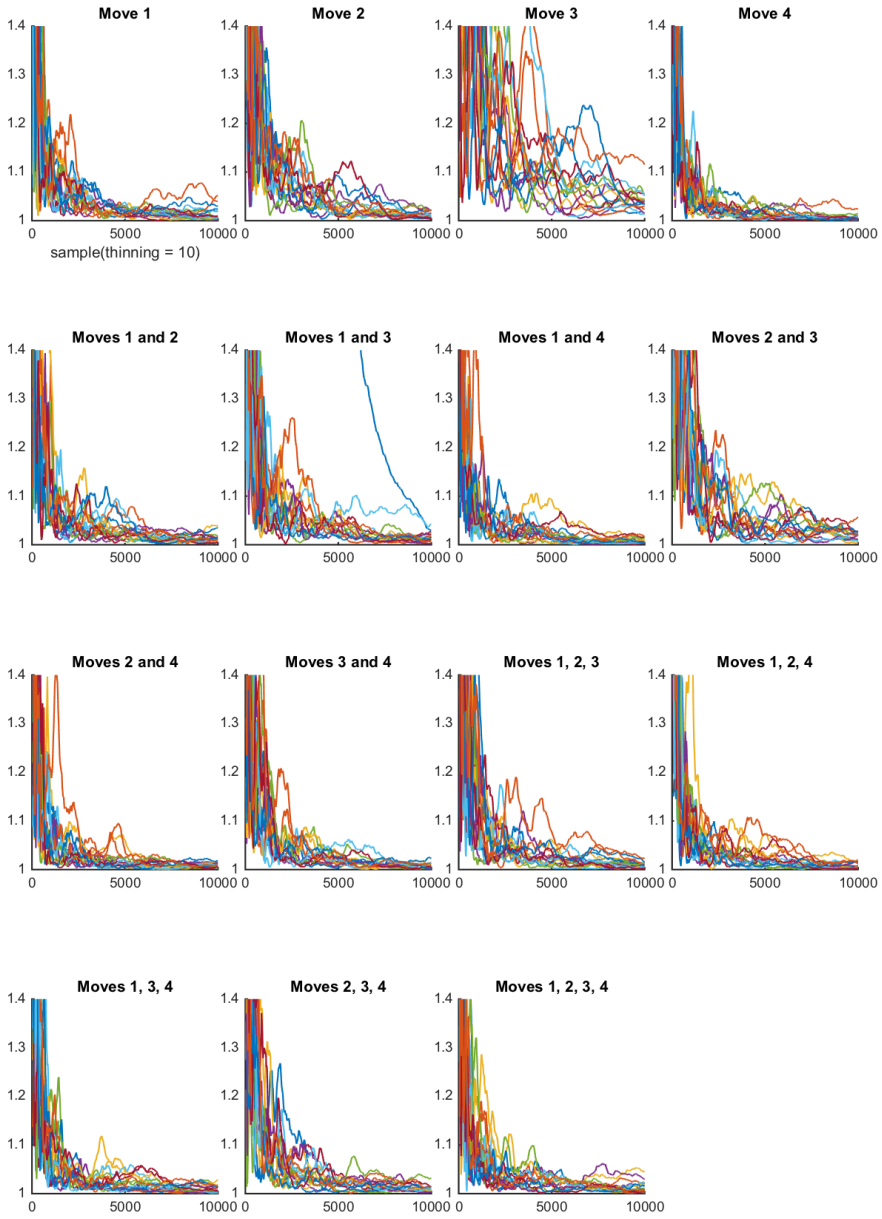


Figure 16: Simulation with more noise: GR statistic for log-likelihood

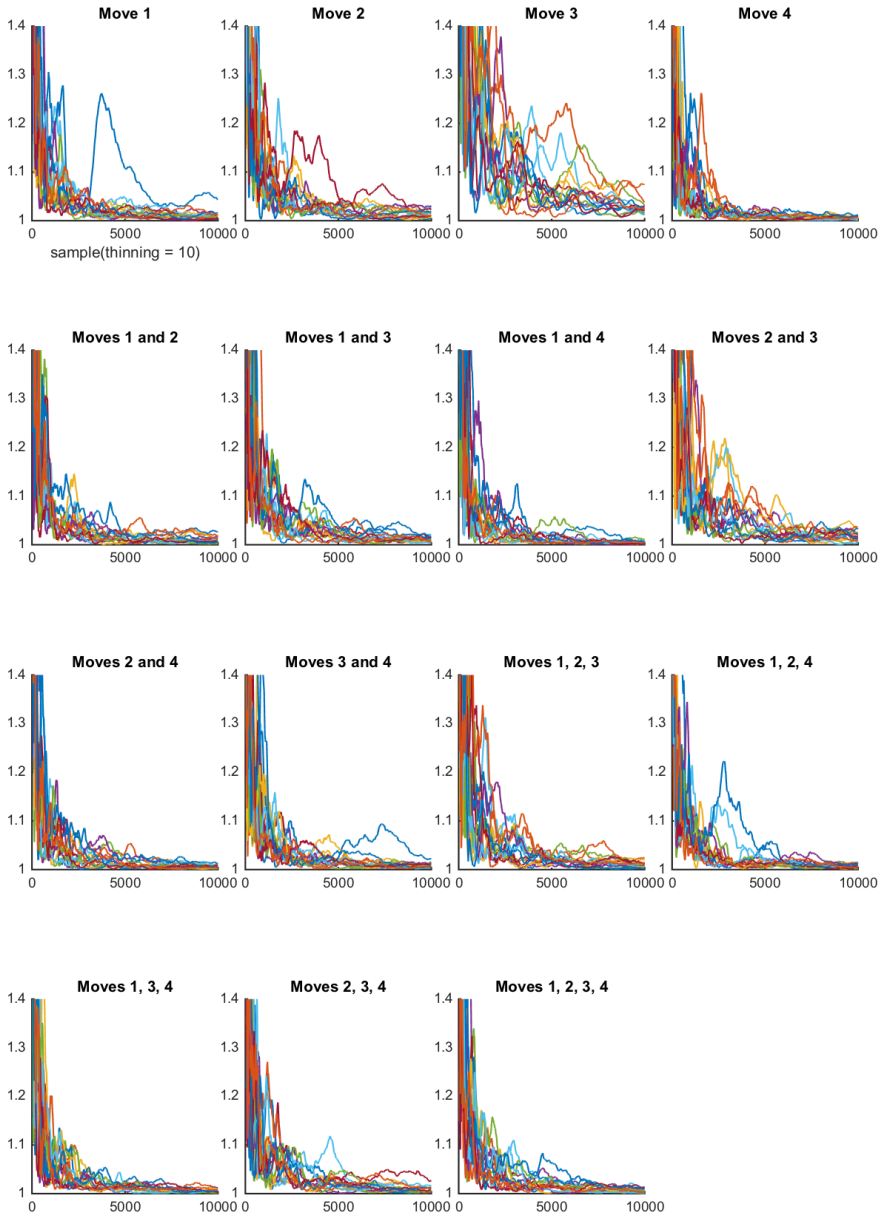


Figure 17: Simulation 1: GR statistic for $L^1$ - distance from ref. permutation

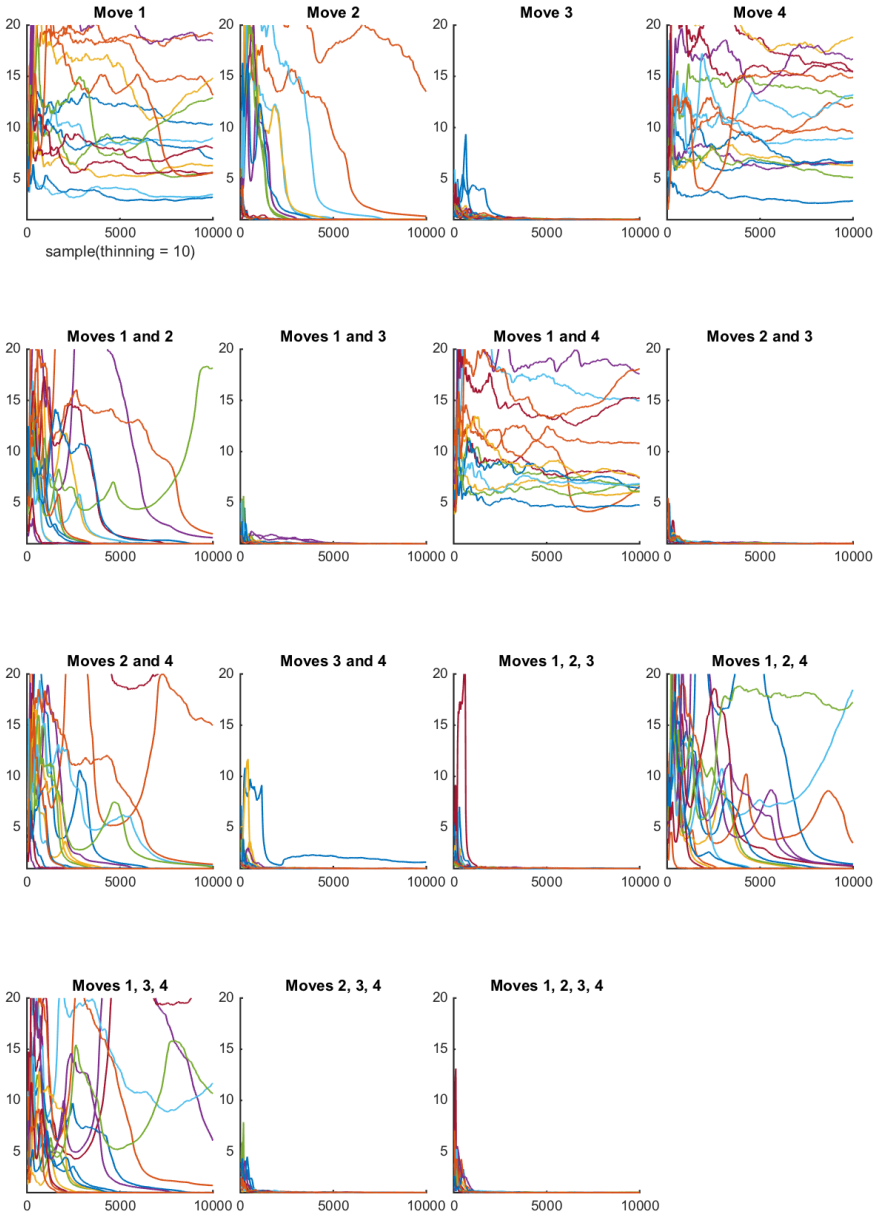


Figure 18: Simulation with fewer capture times: GR statistic for $L^1$ - distance from ref. permutation

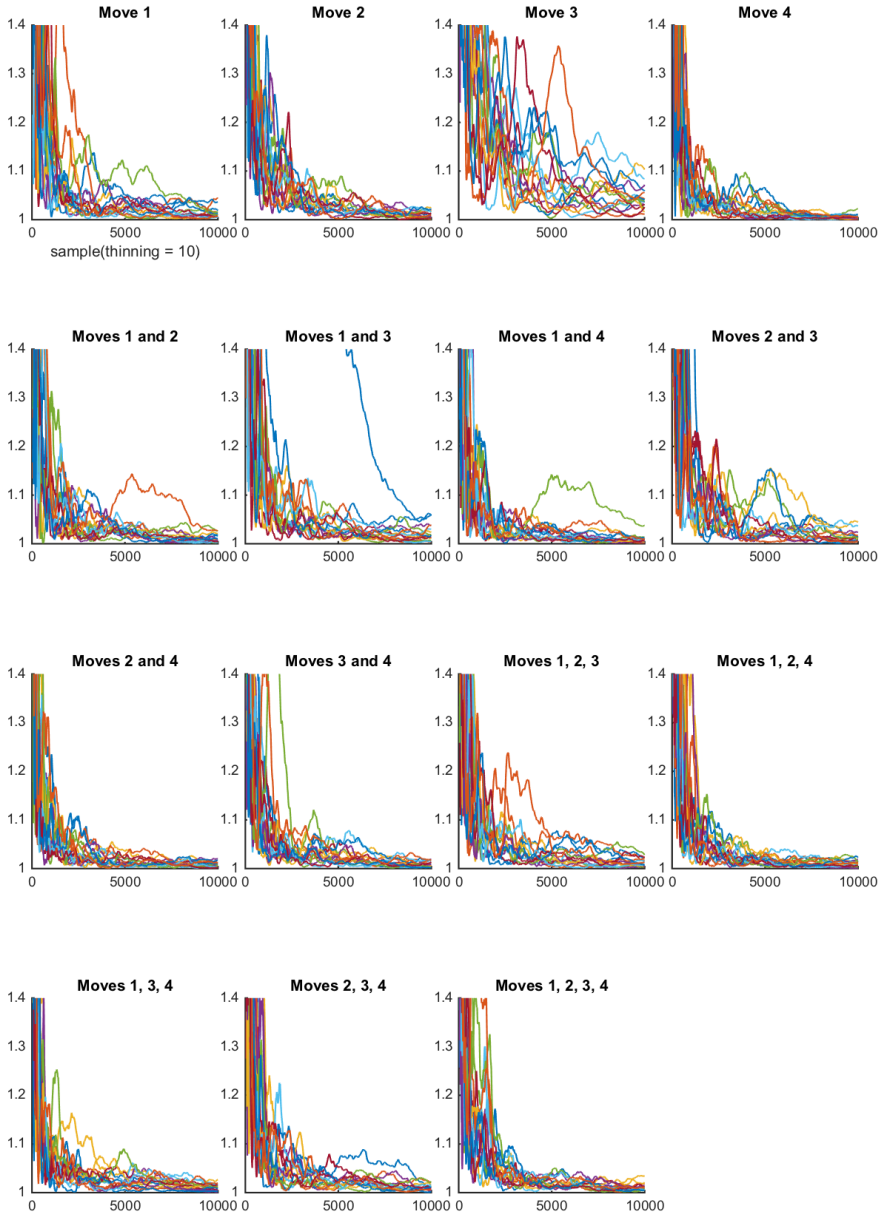


Figure 19: Simulation with more noise: GR statistic for $L^1$ - distance from ref. permutation

## A.2 Comparison between regular and irregular pseudotimes for the microarray data set

Figures 20 and 21 illustrate the improvement obtained through the use of irregular instead of regular pseudotimes. Each of the Figures contains Figures corresponding to 12 MCMC chains. The heatmaps illustrate the positions of each cell, with the true order corresponding to the diagonal of the matrix.

Table 4: Adaptive and non-adaptive moves in the simulations

| | Simulation 1 | | | | | | Simulation with fewer capture times | | | | | | Simulation with more noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **moves** | **1** | **1,ad.** | **2** | **2,ad.** | **3** | **3,ad.** | **4** | **4,ad.** | **1** | **1,ad.** | **2** | **2,ad.** | **3** | **3,ad.** | **4** | **4,ad.** | **1** | **1,ad.** | **2** | **2,ad.** | **3** | **3,ad.** | **4** | **4,ad.** |
| GR<1.01 at end, logLk | 0.75 | 0.75 | 0.31 | 0.31 | 0.13 | 0.13 | 0.81 | 0.56 | 0 | 0 | 0.44 | 0.19 | 0.06 | 0.13 | 0 | 0 | 0.63 | 0.44 | 0.38 | 0.19 | 0 | 0.06 | 0.88 | 0.31 |
| GR<1.01 at end, L1-dist | 0.63 | 0.81 | 0.38 | 0.25 | 0 | 0.06 | 0.88 | 0.56 | 0 | 0 | 0.63 | 0.38 | 0.13 | 0.38 | 0 | 0 | 0.5 | 0.38 | 0.5 | 0.5 | 0 | 0.13 | 0.88 | 0.5 |
| GR<1.02 at end, logLk | 0.94 | 0.94 | 0.75 | 0.69 | 0.38 | 0.31 | 1 | 0.94 | 0 | 0 | 0.75 | 0.5 | 0.25 | 0.38 | 0 | 0 | 0.69 | 0.88 | 0.69 | 0.63 | 0.19 | 0.25 | 0.94 | 0.88 |
| av. # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | 1314 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| max # thin. samp. to GR<1.02,logLk | NaN | NaN | NaN | NaN | NaN | NaN | 2571 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| GR<1.02 at end, L1-dist | 0.94 | 0.94 | 0.81 | 0.75 | 0.31 | 0.63 | 1 | 0.81 | 0 | 0 | 0.81 | 0.5 | 0.44 | 0.56 | 0 | 0 | 0.88 | 0.75 | 0.88 | 0.94 | 0.13 | 0.5 | 0.94 | 0.94 |
| av. # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | 1240 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| max # thin. samp. to GR<1.02,L1-dist | NaN | NaN | NaN | NaN | NaN | NaN | 2811 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| GR<1.05 at end, logLk | 1 | 1 | 1 | 1 | 0.88 | 0.81 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 0.88 | 0.94 | 0 | 0 | 0.94 | 1 | 1 | 1 | 0.75 | 0.94 | 1 | 1 |
| av. # thin. samp. to GR<1.05,logLk | 676 | 426 | 1630 | 1866 | NaN | NaN | 539 | 1264 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 955 | 2549 | 1629 | NaN | NaN | 769 | 1812 |
| max # thin. samp. to GR<1.05,logLk | 1131 | 951 | 4071 | 4891 | NaN | NaN | 1571 | 2171 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2271 | 6651 | 4111 | NaN | NaN | 1691 | 4851 |
| GR<1.05 at end, L1-dist | 1 | 1 | 1 | 1 | 0.94 | 0.94 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 0.88 | 0.94 | 0 | 0 | 1 | 1 | 1 | 1 | 0.69 | 0.94 | 1 | 1 |
| av. # thin. samp. to GR<1.05,L1-dist | 1212 | 1130 | 1149 | 1595 | NaN | NaN | 539 | 1391 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1357 | 1177 | 1780 | 1332 | NaN | NaN | 875 | 1140 |
| max # thin. samp. to GR<1.05,L1-dist | 2891 | 6731 | 2611 | 4431 | NaN | NaN | 1611 | 3451 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 3271 | 3731 | 3751 | 3711 | NaN | NaN | 1791 | 2031 |
| GR<1.07 at end, logLk | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 0.94 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0.94 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.07,logLk | 557 | 360 | 585 | 1145 | 2456 | 2469 | 402 | 900 | NaN | NaN | NaN | NaN | NaN | 3404 | NaN | NaN | 594 | 739 | 1266 | 1041 | NaN | 2691 | 579 | 1425 |
| max # thin. samp. to GR<1.07,logLk | 1051 | 651 | 1311 | 2551 | 5431 | 5391 | 771 | 2011 | NaN | NaN | NaN | NaN | NaN | 7251 | NaN | NaN | 1391 | 1431 | 3791 | 1791 | NaN | 5931 | 1511 | 3771 |
| GR<1.07 at end, L1-dist | 1 | 1 | 1 | 1 | 0.94 | 0.94 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 0.94 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0.81 | 0.94 | 1 | 1 |
| av. # thin. samp. to GR<1.07,L1-dist | 884 | 899 | 875 | 1131 | NaN | NaN | 485 | 1030 | NaN | NaN | NaN | NaN | NaN | 2385 | NaN | NaN | 1029 | 925 | 1289 | 967 | NaN | NaN | 661 | 864 |
| max # thin. samp. to GR<1.07,L1-dist | 2131 | 5671 | 1831 | 3271 | NaN | NaN | 1511 | 2171 | NaN | NaN | NaN | NaN | NaN | 4791 | NaN | NaN | 2491 | 3531 | 2951 | 1611 | NaN | NaN | 1691 | 1971 |
| GR<1.1 at end, logLk | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 1 | 1 | 0 | 0.06 | 1 | 1 | 1 | 1 | 0.94 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,logLk | 435 | 306 | 502 | 767 | 1864 | 1844 | 246 | 609 | NaN | NaN | NaN | NaN | 1637 | 2625 | NaN | NaN | 419 | 530 | 810 | 771 | NaN | 1879 | 380 | 1004 |
| max # thin. samp. to GR<1.1,logLk | 851 | 611 | 1191 | 2371 | 4171 | 4871 | 471 | 1931 | NaN | NaN | NaN | NaN | 3211 | 6791 | NaN | NaN | 831 | 1271 | 1571 | 1591 | NaN | 3811 | 691 | 2051 |
| GR<1.1 at end, L1-dist | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0.88 | 0.81 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0.94 | 1 | 1 | 1 |
| av. # thin. samp. to GR<1.1,L1-dist | 571 | 422 | 620 | 914 | 1536 | 1111 | 427 | 646 | NaN | NaN | NaN | NaN | 1641 | 1931 | NaN | NaN | 689 | 565 | 857 | 744 | NaN | 1420 | 512 | 544 |
| max # thin. samp. to GR<1.1,L1-dist | 1851 | 771 | 1131 | 3191 | 3451 | 1911 | 1371 | 1471 | NaN | NaN | NaN | NaN | 3211 | 4651 | NaN | NaN | 2311 | 1611 | 1711 | 1531 | NaN | 2931 | 1011 | 1071 |
| av. AR | 0.58 | 0.51 | 0.25 | 0.17 | 0.21 | 0.21 | 0.43 | 0.14 | 0.49 | 0.42 | 0.25 | 0.13 | 0.21 | 0.2 | 0.37 | 0.17 | 0.69 | 0.64 | 0.46 | 0.15 | 0.41 | 0.21 | 0.56 | 0.1 |
| min. AR | 0.52 | 0.46 | 0.22 | 0.13 | 0.19 | 0.19 | 0.38 | 0.07 | 0.38 | 0.32 | 0.22 | 0.11 | 0.19 | 0.18 | 0.3 | 0.06 | 0.66 | 0.61 | 0.42 | 0.14 | 0.37 | 0.19 | 0.54 | 0.07 |
| max. AR | 0.66 | 0.6 | 0.28 | 0.25 | 0.23 | 0.22 | 0.52 | 0.28 | 0.6 | 0.54 | 0.28 | 0.16 | 0.23 | 0.22 | 0.47 | 0.31 | 0.74 | 0.7 | 0.49 | 0.18 | 0.45 | 0.25 | 0.63 | 0.2 |

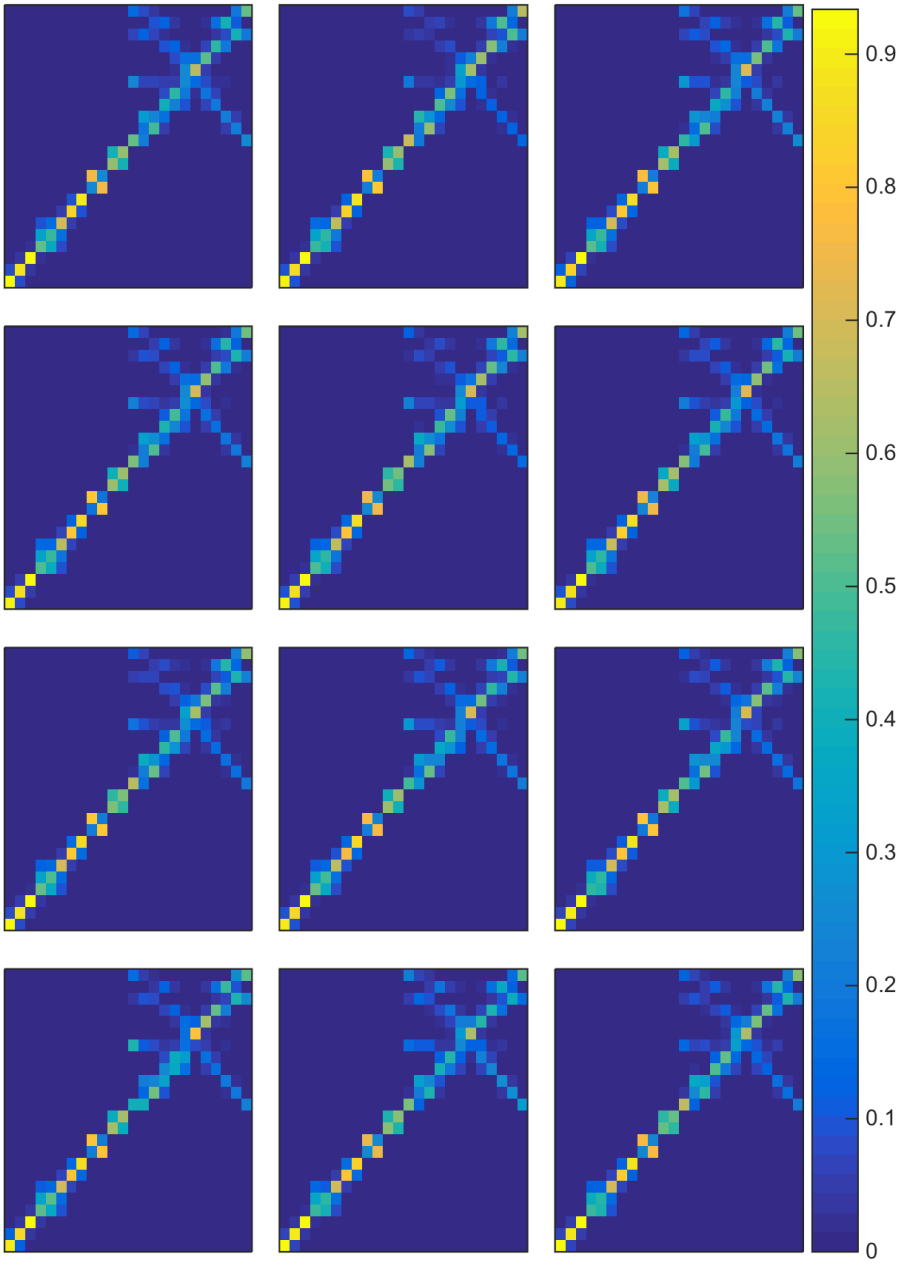


Figure 20: Distributions of cell positions with irregular pseudotimes: Windram data

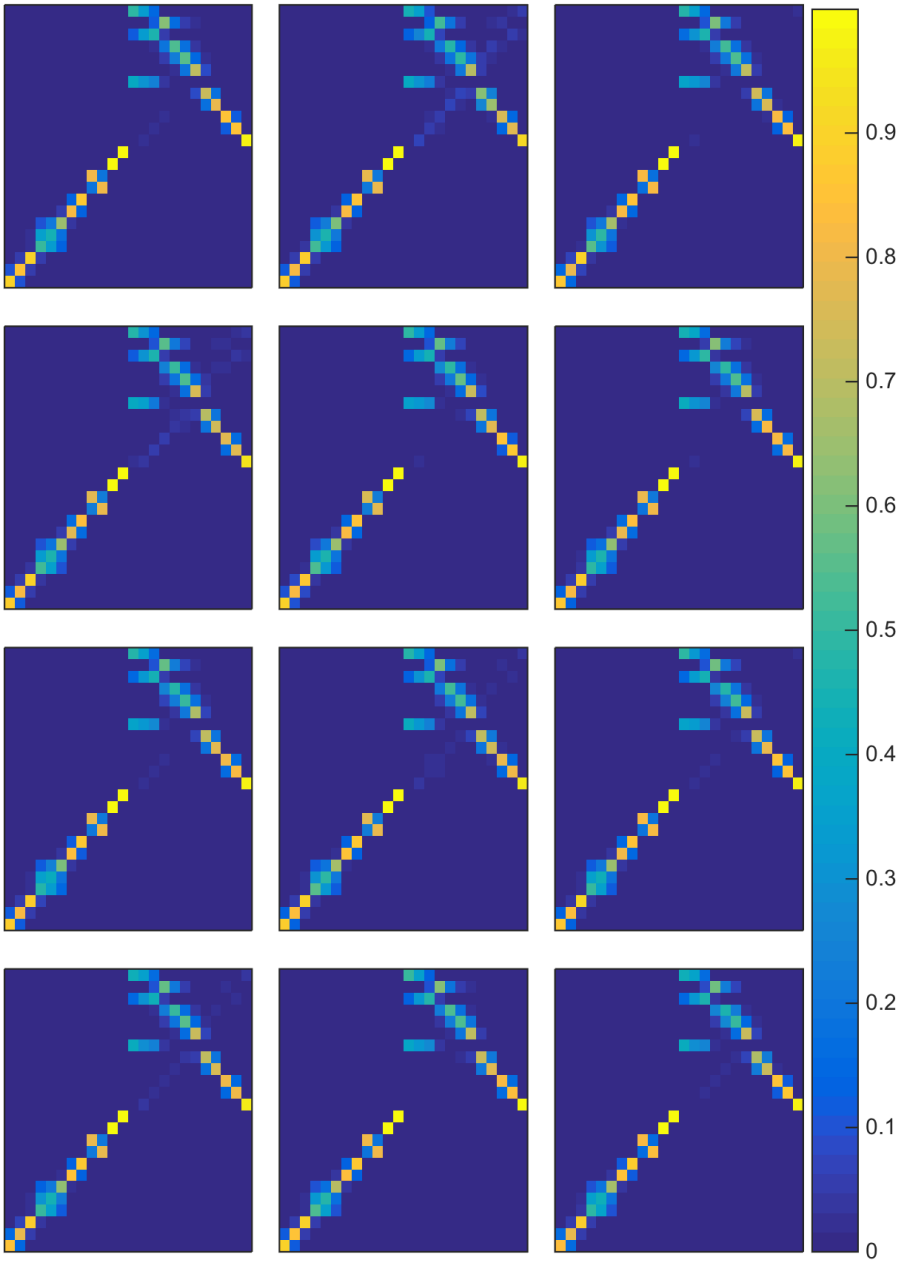


Figure 21: Distributions of cell positions with regular pseudotimes: Windram data