BAYESIAN ANALYSIS ON MIXTURE MODELS, FOR UNDERSTANDING THE PROCESS OF MYOSIN BINDING TO THE THIN FILAMENT

by Madalina-Daniela Mihailescu

A thesis presented for the degree of

Doctor of Philosophy

Department of Mathematical Sciences,

Faculty of Science and Health,

University of Essex

ABSTRACT

Understanding how access is granted to myosin by the actin thin filament has not been fully understood yet. The process of thin filament activation is explored by developing a new variation of hidden Markov models to extract dynamic information from image data and to establish how many myosins are present in an activated region against time. Hidden Markov models supply an extension to mixture models in such a way that they allow for spatial data. The novelty lies in the model allowing for spatial information in the image to be encoded through contextual constraints of a neighbourhood structure based on three nearest neighbours. Furthermore, for the purpose of Bayesian inference about the unknown number of K components, the Metropolis-Hastings algorithm is employed.

The Bayesian analysis shows that, when compared to reversible jump Markov chain Monte Carlo, our proposed model provides a better alternative for the finite mixture model at capturing the behaviour of myosin binding to the thin filament. The estimated mean intensity values of fluorescence from both models are exemplified in separate kymographs, where the variation in light intensity gives us information about how the myosin binding phenomenon is clustered or varies over time.

ACKNOWLEDGEMENTS

Foremost, I would like to express my special thanks and the deepest appreciation to my supervisor, Dr. Hongsheng Dai, for his immense support during my PhD study years, for his patience, enthusiasm, extensive knowledge and for always pushing me to work harder. I could not have imagined having a better advisor and mentor for my PhD study.

I would also like to take this opportunity to express my heartfelt gratitude to my close family who has always supported my ventures and has believed in me.

Last but not least, I am also indebted to one of closest friends, Amalia Becherescu, because these past years would have been grim and difficult without her constant encouragement and guidance.

CONTENTS

CHAPTER 1

INTRODUCTION

The mechanism of muscle contraction at the molecular scale has not been fully understood yet, mainly because a model incorporating the full intricacy of the thin filament is yet subjected to experimental scrutiny. The data used in this paper is unique and extremely insightful. It has not been previously analyzed for the same purposes using the techniques presented here. Full understanding of the myosin binding process might allow, amongst others, novel treatments of genetic heart disease. It cannot yet be predicted how a genetic failure in muscle protein will lead to a physiological change in the heart. This is because yet there is no clear connection between muscle contraction and cell or organ (Spudich [64]). While further work is necessary, this decade will see an exponential expansion in our comprehension of the complex mechanism of muscle contraction.

In the past few decades, mathematical modeling has played a significant role in our understanding of the link between muscle contraction at the micro and macro scale (Niederer et al. [49]). Many theoretical concepts, such as the sliding filament theory,

have been implemented using mathematical models. These models have proven to be very accurate. For example, Walcott and Sun [70] performs a simple simulation to examine the interaction between myosin and actin. These two proteins are the main contributors to muscle contraction. Similarly, this paper measures the interactions between myosin molecules and the thin filament. It has previously been established that myosin molecules bind in clusters along the thin filament and their size depends on the solution conditions. These clusters represent the number of myosins in an active region and the regulatory system of the thin filament can accommodate at most 11 binders in an active area (Desai et al. [19]). Myosin binders in an area of activation attach, detach and catastrophically collapse. Based on studies of myosin binding to regulated actin, McKillop and Geeves [40] propose a model with three regulatory states: blocked, closed and open. The blocked state stops any myosin binding, the closed state allows weak myosin binding as observed at low concentrations of Calcium, and the open state allows both weak and strong myosin binding. Furthermore, both Mijailovich et al. [46] and Desai et al. [19] discuss cooperative activation of thin filament. These two articles argue that myosin facilitates its own binding by developing locally fully active regions.

The aim of this thesis is to examine the process of thin filament activation by developing novel mathematical methods to extract dynamic information from image data. To solve the problem that coexisting fluorophores cannot be optically resolved, we will use a novel mathematical variation of hidden Markov models to establish how many myosins are present in an activated region against time. This will offer unprecedented access into the mechanism of a complex system at the single molecule level. The implementation of a latent spatial process is being considered, in order to model the rate of myosin molecules binding on the thin filament, the rate of myosin molecules leaving the thin filament and how myosin molecules interact with each other. We are particularly keen on studying how bound myosin molecules lead to cooperative activation of the thin filament to generate further binders, and also on the catastrophic collapse of the thin filament. This catastrophic collapse involves full detachment of the bound myosin

molecules in an active region, in only one step. In other words, an active region on the thin filament is turned off in only one step.

Modeling problems in this thesis are addressed mainly from the computational viewpoint. The primary concern is how to define the objective function for the optimal solution for the given image analysis problem and how to find the optimal solution using Bayesian methods. The reason for defining the solution in an optimisation sense is due to numerous uncertainties in the image processes. Thus, we search for the optimal solution, using encoded constraints to fit the given data. Contextual constraints are indispensable in the correct interpretation of visual information because the aspects of context are important for the task of identifying the context, and determining how they constrain the process under consideration.

The research presented within this thesis will demonstrate how the reversible jump MCMC algorithm, within a Bayesian structure, can be used in the area of thin filament activation for parameter estimation and model selection. This approach deals with the issue of selecting the number of clusters and the validity of a given model is addressed in a principled and formal way. The performance of the reversible jump MCMC will then be compared to the performance of the hidden Markov model proposed in this thesis. Both models will be applied to similar datasets and evaluated to provide a better understanding of how the thin filament activates and also deactivates.

1.1 Contribution to knowledge

The thesis studied the activation of the thin filament and how myosin binds in clusters along the thin filament via finite mixture models in Chapter 3, using the reversible jump MCMC algorithm. This detailed examination using the reversible jump MCMC approach reveals a high probability of myosin binding in a more classical cooperative activation. It also reinforces that myosin spreads its own binding by creating locally fully active regions known as regulatory units.

To further explain such cooperative phenomenon, a new variation of hidden Markov models has been considered because these models supply an extension to mixture models in such a way that they allow for spatial data. The novelty lies in the model allowing for spatial information in the image to be encoded through contextual constraints of a neighbourhood structure based on three nearest neighbours. This neighbourhood process has been carefully chosen to capture the effects light intensity from neighbouring positions has on the average intensity of light of a pixel. The latent variable produces the number of bound myosins at a time point and location, and also deals with the transitions between different states of behaviour. This enabled an assumption-free model of the attachment and detachment probabilities for myosin to be determined. It is expected that myosin binds to actin stochastically and forms clusters. The highly elevated collapse probability suggests a concerted mechanism of deactivation (relaxation), and explains the ability of muscle to relax in conditions that would be expected to still permit myosin binding.

The performance of the proposed Markov chain Monte Carlo algorithm in simulating the myosin binding process is compared to the raw data and to the performance of the reversible jump Markov chain Monte Carlo. The purpose of such mixture analysis of our novel variation of HMMs and RJMCMC is inference about the unknown number of K components, component parameters and the proportion of each cluster. The analysis reveals that our proposed model has an improved performance when compared to RJMCMC because it gives a better simulation of the number of myosins found in each pixel. Thus, the hidden Markov model provides a better alternative for the finite mixture model at capturing the behaviour of myosin binding to the thin filament.

1.2 Biological background and data description

Nowadays, statistical modelling is more and more frequently used in research of the behaviour of biological systems, especially in the process of rehabilitation, in medicine and sports biomechanics. This is due to the need for identification and connection of the series of phenomena occurring in the organism. In the majority of cases, the direct measurements of data that describe those phenomena cause disturbances of organisms functions or permanent damage to its organs. For that reason the statistical modelling has been used during the identification of cause-and-effect relationships. The interaction of actin and myosin filaments is the basis for muscle contraction in all cases. So, full understanding of the thin filament activation process could help the identification of novel treatments of genetic heart disease.

In this section, a brief theoretical review on the mechanism of muscle contraction is presented in the first section. Then, the second part of this section is dedicated to introducing a dataset example and displaying some descriptive statistics in order to be made familiar with the datasets before commencing any analysis. This information will become more useful in later chapters, in order to comprehend the underlying theory and its applications.

1.2.1 Muscle contraction

The human body has three types of muscle tissue: skeletal, cardiac and smooth (Silverthorn et al. [62]). Skeletal muscle, which is attached to the bones of the skeleton, controls body movement. This type of muscle contracts only in response to a signal from a somatic motor neuron (a highly specialized cell in the nervous system which conducts impulses to skeletal muscles). Cardiac muscle is the muscle of the heart which contracts to squeeze blood out of the heart, and relaxes to fill the heart with blood. Skeletal and cardiac muscles are known as striated muscles because they have cross striations formed from the organisation of the sarcomere, which is formed of the different thick (myosin) and thin (actin) filaments within muscle cells. Smooth muscle is

the primary muscle of internal organs and tubes such as stomach and walls of blood vessels. Its dominant function is to impact the movement material into, out of, and within the body (Silverthorn et al. [62]). This paper focuses on striated muscle only.

Muscles have two main functions, which include the production of motion and of force. Force generation in striated muscle is a reaction of the cyclical interaction of myosin with actin, a process that is regulated by local Calcium ion concentration. The two principal muscle proteins implicated in contraction are myosin and actin. Myosin molecules join to create a thick filament with the heads assembled at each end and actin is the protein that forms the thin filament. These proteins are arranged as a repeating pattern of thick and thin filaments in the sarcomere. According to the sliding filament model which was firstly introduced by Huxley and Hanson [36], each myosin headpiece engages in a repetitive cycle of making and breaking crossbridges to an adjacent thin filament. Each actin molecule has a single myosin-binding site, and each myosin head has one actin binding site and one binding for adenosine triphosphate (ATP) (Tobacman [68]). A Calcium signal initiates the power stroke and repeats many times as a muscle fiber contracts. The myosin head reaches forward, binds to actin, contracts, releases actin molecule, and then reaches forward again to bind actin in a new cycle. Crossbridges form when myosin heads of thick filament bind to actin in the thin filament, which results in the actin filament being drawn along a certain distance, towards the centre of the sarcomere. This sliding force established between the filaments is hydrolysing of ATP by myosin at a rapid rate, thus releasing energy for muscle contraction. When activation of the muscle ends, interaction between myosin cross bridges and actin filaments stops. Therefore, no sliding energy is created and the muscle is relaxed once more(Stracher [67]).

McKillop and Geeves [40] have shown that without Calcium (Ca^{2+}), the position of tropomyosin partially covers actin's myosin-binding sites, preventing the myosin cross-bridges from binding to the thin filament. This process is presented in Figure (1.1) (OpenStax [50]). Weak binding can still occur but myosin is blocked from complet-

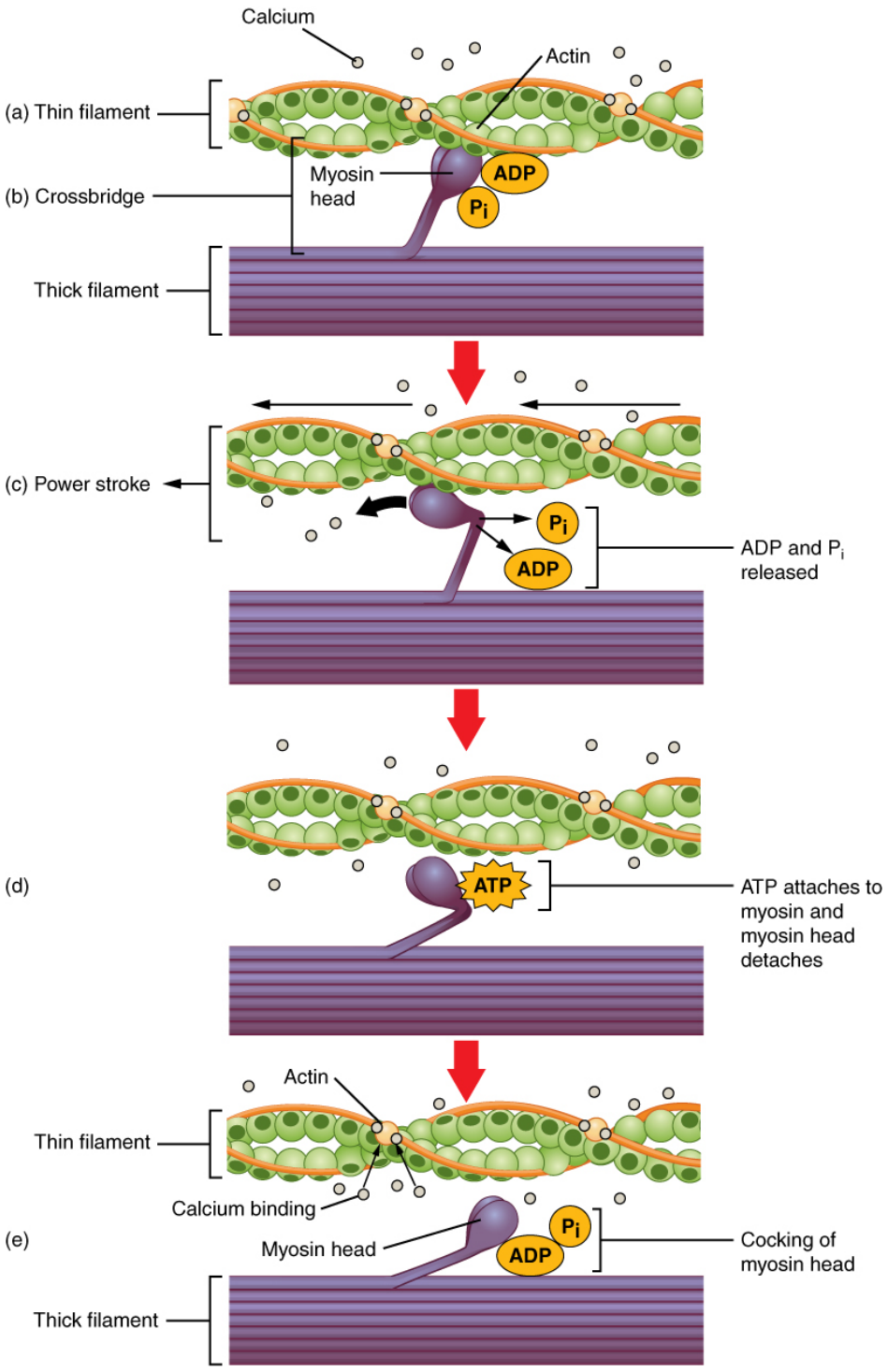


Figure 1.1: Thin Filament Activation

ing its power stroke. In order to trigger the contraction of the muscle, tropomyosin must shift its position to uncover the myosin binding sites. Calcium and ATP are cofactors necessary for the contraction of muscle cells. ATP is the supplier of energy, whereas calcium sends out signals to troponin which regulates the off-on positioning of tropomyosin. Specifically, troponin pulls tropomyosin completely away from actin's myosin binding sites. This "on" position and sufficient ATP allows the myosin heads to form strong crossbridges and complete their power strokes, moving the thin filament. In the absence of Calcium, this binding does not occur, so the presence of calcium is a valuable regulator of muscle contraction(Silverthorn et al. [62]). According to Desai et al. [19]: "Once bound, myosin is hypothesized to potentiate the binding of further myosins. We have found that 2 myosin heads are required to laterally activate a regulatory unit of thin filament. The regulatory unit is found to be capable of accommodating 11 additional myosins." This suggests that the binding of one myosin molecule in one position facilitates the same action in neighbouring binding sites through the activation of the thin filament. So, when the thin filament is fully active, myosin is capable of opening a region on the thin filament which permits up to 11 supplementary myosin heads to bind.

Furthermore, as explained in Kad et al. [37], subsequent studies have shown that myosin's interaction with the thin filament consist of three states: blocked, closed and open. As previously mentioned, when Ca^{2+} is not present tropomyosin occupies the binding site preventing myosin from binding to actin. At the time of Ca^{2+} binding to troponin, tropomyosin's equilibrium position moves toward the closed state, exposing some binding sites that allow myosin weak binding. Once bound, myosin's weak-to-strong binding transition shifts tropomyosin's equilibrium position further towards the open state, permitting cooperative binding of additional myosins by exposing neighbouring actin binding sites. Thus, the full activation of the thin filament occurs only when an open state exists.

1.2.2 Introduction to the Data

The datasets used in this thesis for the purposes of analysis have been collected by Desai et al. [19] during preliminary experiments. The experiments involved the development of an *in vitro* model of thin filament activation based on the use of single molecule imaging [1]. Single headed myosin II have been fluorescently tagged to behave as both an activator and a reporter of activation. Also, single reconstituted thin filaments suspended between silica beads, have been employed. These are known as "tightropes". This technique of single molecule imaging has presented new insights into how motion and force is being generated. In Figure (1.2), which illustrates a typical image dataset of the thin filament activation, one would be able to notice that myosins bind in clusters along the thin filament. These clusters differ in size depending on the solution conditions. A study across a range of Calcium and myosin concentrations is required to provide accurate modelling of the complex mechanism of thin filament activation. Such analysis will determine to what extent a catastrophic collapse is a stochastic process. This is because thin filament activation is inherently a stochastic process influenced by the concentration of Calcium and myosin. One would expect that with greater solution concentrations, the active regions will grow and unite to turn on the entire thin filament.

Figure 1.2: Kymograph at pCa=6 and myosin II=5 nM

As an example, Figure (1.2) represents a kymograph of myosin interactions with thin filaments. The horizontal axis (x-axis) exhibits time, where the time difference between two pixels is 300ms and the frame rate is approximately 3.3 frames per second (fps). This comes from the exposure time, and therefore fps is determined by the exposure

[1] The full extent of the experiment can be found in Desai et al. [19]

time of 300ms. The vertical y-axis represents the single thin actin filament and each pixel of the kymograph is 126.4nm of the thin filament. The conditions of this specific image are 15nM myosin at pCa=6 (calcium concentration) with 0.5 μM ATP. The fluorescent spots in the kymograph illustrate binders. As the intensity of a spot increases, one would expect a greater number of binders in that position.Each pixel has a value which represents the intensity of light in the pixel. Thus, the brighter the pixel, the larger the number of binders in that position at that point in time. Many fluorescent pixels appearing next to each other indicate clustered myosin binding.

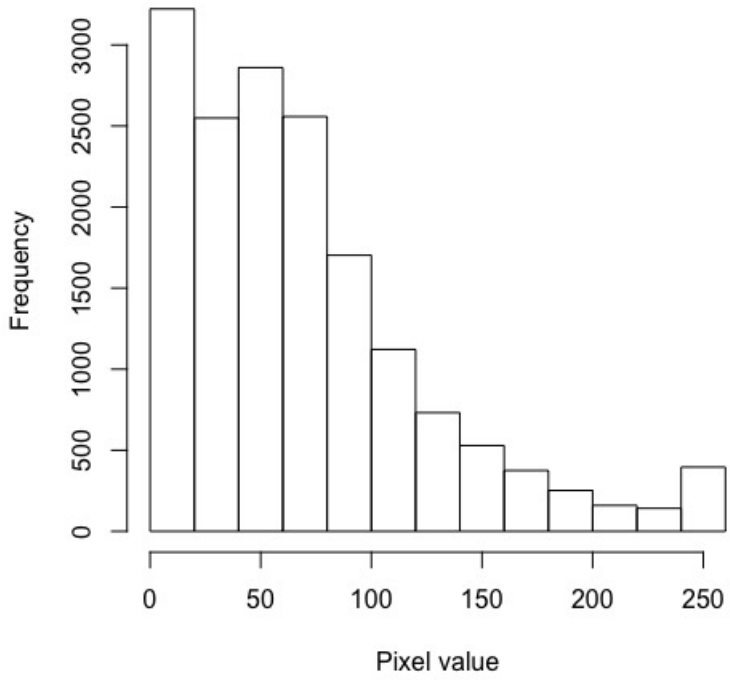


Figure 1.3: Histogram at pCa=6 and myosin II=5 nM

Table 1.1: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
0	28	58	69.41	94	255	57

To gain some initial insight into the datasets, descriptive statistics of Figure 1.2 are used to summarise and describe data. This kymograph is representative of the type of data to be analysed in later chapters. Firstly, Table 1.1 presents the summary statistics of the kymograph presented in Figure 1.2, where the mean value of a pixel is 69.41. Also, the standard deviation (SD) is 57, which measures the average distance between a single observation and the mean. Figure 1.3 displays the positively skewed distribution of data, where most of values are found in between 0 and 150. Moreover, by looking at this histogram, one cannot tell how many distinct subgroups are within this dataset, even if it is well known from literature that myosin binding occurs in clusters.

Having introduced the theoretical review on the mechanism of muscle contraction and some descriptive statistics of the datasets, the next chapter will provide a comprehensive review of Bayesian inference for mixture models. One of the main interests in this thesis is to determine the number of clusters in each dataset and then model the process of clustering, which is represented by the myosin binding to the thin filament.

CHAPTER 2

REVIEW OF BAYESIAN INFERENCE FOR MIXTURE MODELS

Because the exact theories of optimal estimates are challenging to apply, attention has deviated to approximate algorithmic methods, such as simulation. In this thesis, it is more appropriate to use a Bayesian approach when dealing with a stochastic process. Inference is made about an unknown parameter, say θ, from the data, which allows one to explore and learn about this parameter θ.

Bayesian analysis is a method of statistical inference (named after the English mathematician Thomas Bayes) that allows one to combine prior information about a population parameter with evidence from information contained in a data sample to guide the statistical model. Bayesian inference is an extremely powerful set of tools for modeling any random variable. It involves drawing conclusions, based on real data, about quantities that are not observed. Bayesian inference starts with the formulation of a model that we consider as an appropriate representation of a situation that holds

our interest. This is done by utilising practical methods for drawing conclusions from representative data, using probability models for both observed and unobserved quantities. In this type of analysis, uncertainty is expressed in terms of probability and common-sense interpretation of statistical conclusions is facilitated. For instance, a 95% credible interval for an unknown quantity is interpreted as having a 95% probability of containing the unknown quantity. Whereas, a 95% frequentist (confidence) interval is interpreted as a range of values so defined that there is a 95% probability that the unknown quantity lies within this range.

The four steps of a Bayesian analysis are

1. Specify a joint distribution for the outcome(s) and all the unknowns, which typically takes the form of a marginal prior distribution for the unknowns multiplied by a likelihood for the outcome(s) conditional on the unknowns. This joint distribution is proportional to a posterior distribution of the unknowns conditional on the observed data.

   In practice, two major challenges confront the practical implementation of Bayesian analysis - the specification of the prior distributions and the calculation of the posterior distribution.

2. Draw from posterior distribution using Markov Chain Monte Carlo (MCMC) methods.

   Recent advances in computing technology coupled with developments in numerical and Monte Carlo methods, most notably Markov Chain Monte Carlo (MCMC), have opened up new and promising directions for addressing this challenge. The basic idea behind MCMC here is the construction of a sampler which simulates a Markov chain that is converging to the posterior distribution.

3. Evaluate how well the model fits the data and possibly revise the model.

4. Draw from the posterior predictive distribution of the outcome(s) given interest-

ing values of the predictors in order to visualize how a manipulation of a predictor affects (a function of) the outcome(s).

The fundamental feature of Bayesian methods is their specific use uncertainty by a probability distribution over hypotheses. Ones ability to make inferences depends on ones degree of confidence in the chosen prior, and the robustness of the findings to alternate prior distributions may be relevant and important. Whereas, the frequentist method only uses conditional distributions of data given specific hypotheses. The presumption is that some hypothesis (parameter specifying the conditional distribution of the data) is true and that the observed data is sampled from that distribution.

This chapter aims to introduce the reader to the construction, prior elicitation, estimation and evaluation of mixture distributions in a Bayesian setting. It will show that mixture models provide a flexible framework for statistical modelling and analysis. The focus lies on methods, given that the practical aspects will be presented in later chapters. In Section 2.1 the basis of Bayesian inference is being introduced, followed by basic properties of mixtures in Section 2.2. Then, Section 2.3 talks about the use of prior distributions and their importance. Most importantly, Section 2.4 describes the powerful tool of MCMC methods and algorithms that can be used for the approximation to the posterior distribution on mixture parameters.

2.1 Bayesian inference

The basis for Bayesian inference is derived from Bayes' theorem. This theorem provides an expression for the conditional probability of A given B, which is equal to

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Replacing B with observations $\mathbf{y}$, A with the set of parameters θ and probabilities P with densities π results in

$$\pi(\theta|\mathbf{y}) = \frac{f(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\rho(\mathbf{y})} \tag{2.2}$$

where

$$\begin{aligned} \rho(y) &= \int f(y|\boldsymbol{\theta})\pi(\boldsymbol{\theta})dy. \quad \text{for continuous data} \\ \text{and} & \\ \rho(y) &= \sum_{i=1}^{n} f(y|\theta_i)\pi(\theta_i) \quad \text{for discrete data} \end{aligned}$$

According to Gelman et al. [28], in Bayesian statistics the parameter θ is taken as being random. The statistical model $f(y, \boldsymbol{\theta})$ represents a family of distributions, each of which has assigned a unique parameter θ_i, for $i = 1, 2, \cdots, n$. Gelman et al. [28] also explains that in oder to draw probability conclusions about $\boldsymbol{\theta}$ given y, one must combine the likelihood, $f(y|\boldsymbol{\theta})$, and the prior distribution $\pi(\boldsymbol{\theta})$ in the posterior distribution,$\pi(\theta|\mathbf{y})$. $\pi(\boldsymbol{\theta})$ is the set of prior distributions for the set of parameters before $\mathbf{y}$, the data, is observed.The likelihood function, also denoted as $L(\boldsymbol{\theta}|y)$, is thought of as the information brought in by the data.

The marginal likelihood $\rho(y)$ is an integral over all the values of $\boldsymbol{\theta}$ of the product $f(y|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ and is viewed as a normalising constant to guarantee that $\pi(\boldsymbol{\theta}|y)$ is a proper density. The Bayesian approach often results in integration problems, due to the difficulty in calculating the normalising constant, $\rho(y)$, for large n. Given this, the posterior distribution can now be expressed as

$$\pi(\boldsymbol{\theta}|y) \propto f(y|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) \tag{2.3}$$

which is

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.4}$$

The posterior distribution, $\pi(\boldsymbol{\theta}|y)$,as discussed above, represents the current updated beliefs about the parameter $\boldsymbol{\theta}$, after observing the data sample, y, and the prior information about $\boldsymbol{\theta}$. Without the normalising constant, Bayesian inference will not be available analytically. In order to solve this issue, numerical approximation methods are applied by using advanced computational tools. For example, if the number of dimensions is too large, calculations by hand are beyond the bounds of possibility and statistical softwares could be used instead.

.Simulation can be used to approximate $\mathbb{E}[f(y|\boldsymbol{\theta})]$. Monte Carlo simulation takes independent samples $\{\boldsymbol{\theta}_i : i = 1, 2, \cdots, m\}$ from $\pi(\boldsymbol{\theta})$ and uses the approximation

$$\mathbb{E}[f(y|\boldsymbol{\theta})] \approx \frac{1}{m} \sum_{i=1}^{n} f(y|\boldsymbol{\theta}_i) \tag{2.5}$$

which holds by the strong Law of Large Numbers (Leonard E. Baum [38]) when m goes to $+\infty$. The $\{\boldsymbol{\theta}_i\}$ need not be independent as long as they have the correct distribution and so a Markov chain with stationary distribution $\pi(\boldsymbol{\theta})$ may be used to draw correlated iterations from $\pi(\boldsymbol{\theta})$. This is called Markov chain Monte Carlo and it will be discussed in Section 2.4.

The components of Bayesian inference are

1. $\pi(\boldsymbol{\theta})$ represents the set of prior distributions for the parameter space $\boldsymbol{\theta}$ and uses probabilities to quantify uncertainty about the parameters before observing the data.

2. $f(\mathbf{y}|\boldsymbol{\theta})$ is known as the likelihood function, which is a function of the unknown set of parameters, $\boldsymbol{\theta}$, which indexes the distribution from which y_i is generated.

3. $\pi(\boldsymbol{\theta}|\mathbf{y})$ is the posterior distribution and it communicates the uncertainty about the set of parameters after taking into account both the prior distributions and the data. One can also consider looking at a single parameter of interest by constructing the marginal posterior distribution, $\pi(\theta_i|\mathbf{y})$

2.2 Finite mixture model estimation

A finite mixture model (FMM) is a statistical model that assumes the presence of unobserved groups, called latent classes, within an overall population. One can compare models with differing numbers of latent classes and different sets of constraints on parameters to determine the best fitting model.

Over the years, a variety of methods have been used to estimate mixture distributions. The main reason for the vast literature on the methodology of mixture estimation is that explicit formulas for parameter estimates are yet not available. Practitioners are increasingly turning to Bayesian methods for the analysis of complicated statistical models. This is due to the arrival of the high speed computers and the accelerated development in posterior simulation techniques such as MCMC methods for enabling Bayesian estimation to be perfomed.

In the past decade the extent and the potential of the applications of mixture models have widened substantially. The flexibility of mixture models allows them to be more and more exploited as a convenient, semiparametric way in which to model unknown distributional structures. This is in addition to their applications to group-structured data, also known as cluster analysis.

Finite mixture models (FMMs) are a flexibile and powerful probabilistic modeling tool for both univariate and multivariate data. The Bayesian approach to such models has attracted increasing attention amongst researchers from both theoretical and practical points of view. This is primarily because of the emergence of Markov chain Monte Carlo methods. The main concept in finite mixture modeling is that the observed data comes from distinct, but unobserved, subpopulations. The versatility of mixture models in any area which involves statistical modeling of data (such as pattern recognition, image analysis, computer vision etc) has been widely acknowledged in recent years. Because of their flexibility, finite mixture models have been used to adjust for clustering, and to model unobserved heterogeneity. More generally, FMMs allow mixtures of

linear and generalized linear regression models, including models for binary, ordinal, nominal, and count responses, and allow the inclusion of covariates. Inferences can also be made about each sobpopulation and individual observations can be classified into a subpopulation. Thus, these models enable the use of Bayesian methods to make statistical inferences about the properties of the sub-populations given only observations on the pooled population, without sub-population identity information.

The initial application to a mixture model-based approach was firstly introduced by Pearson [51]. This paper suggested that two subspecies were present and it fitted a mixture of two univariate normal components to some crab measurements provided by his colleague Weldon [71],Weldon [72]. Given the amount of work involved in this early approach, many have tried to simplify it and in the early 1900s work continued on the use of the method of moments for this mixture problem. Maximum Likelihood (ML) estimation of the parameters in a mixture distribution was firstly implemented by Rao [55], using Fisher's approach of scoring for a mixture of two univariate distributions with equal variances. Later on, Dempster et al. [18] presents an expectation - maximization (EM) algorithm as an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, when some of the data is missing. This iterative scheme was formalized in a general context through the EM algorithm that the convergence properties of the ML solution for the mixture problem were established on a theoretical basis. Dempster et al. [18] has turned out to be a great stimulant for further research into the applications of finite mixture models. This is confirmed by the subsequent flow of papers on finite mixtures in the literature (McLachlan [42], McLachlan [43],McLachlan and Basford [44],Aitkin [1]).

There are two primary classes of estimation methods for mixture models, and these are the EM algorithm, explained above, and the Bayesian methods. The use of Bayesian methods for estimation of mixture distributions has been limited until the arrival of Gelfand and Smith [26]. This paper brought into focus the great power of the Gibbs

sampler algorithm in a wide range of statistical problems. This is because they have realised that many of the Bayesian computations could be implemented using the Gibbs sampler. Gibbs sampling is applicable when the joint distribution is not known explicitly or is difficult to sample from directly, but the conditional distribution of each parameter is known and is easier to sample from. Thus, based on the two classes mentioned above, other approaches have further been developed to fit these mixture models. For example, Stephens et al. [66] presents an MCMC method which views the parameters of the model as a (marked) point process and creates a Markov birth-death process with an appropriate stationary distribution. Similarly, the book of Frühwirth-Schnatter [24] presents an inclusive summary of the Bayesian analysis for finite mixture models and Markov switching models. Bayesian techniques yield a greater amount of information about the unknown parameters, but they can also be computationally expensive.

While MCMC provides an appropriate way to draw inference from complicated statistical models, there are many problems associated with the MCMC analysis of mixtures. One problem is caused by the non-identifiability of the components under symmetric priors, which is now known as the label-switching problem in the MCMC output. This means that during MCMC simulation, the components' weight and parameters switch, making it difficult to determine whether the chain has reached the limiting distribution. Thus, ergodic averages of component specific quantities will be identical and meaningless for inference.

Another key issue in mixture modeling is the selection of the number of components. The usual trade off in model order selection problems arises: When too many components are present, the mixture may over-fit the data, while a mixture with too few components may not be flexible enough to approximate the true underlying data structure. As demonstrated by Chen [11], when the number of components is unknown, the optimal convergence rate of the estimate of a finite mixture model is slower than the optimal convergence rate when it is known. The use of Wasserstein distance was sug-

gested by Nguyen [48] to investigate the indentifiability issue and the optimal rates of convergence for the parameters of multiple types in finite mixtures. Bayesian analysis of mixture models for an unknown number of components has been made possible using techniques such as reversible jump MCMC (Richardson and Green [56]) and birth and death MCMC (Stephens [65]).

A stable approach to model selection is to combine the complexity of the model with the performance of the model into a score, then select the model that minimizes the score. This approach is referred to as statistical or probabilistic model selection as the scoring method uses a probabilistic framework. A straightforward solution to the problem of evaluating several candidate models is to select the model that gives the most accurate description of the data. However, this selection process is not simple by the fact that a model with many free parameters is more flexible than a model with few parameters. But the complex model is not always assumed to be the best. According to Wagenmakers and Farrell [69], the generally accepted view is that the best model is the one that provides an acceptable account of the data. Since increasing complexity is accompanied by a better fit, models are compared by trading off the measure of fit (typically a deviance statistic) and complexity (i.e. the number of free parameters in the model); so following early work of deLeeuw [16], proposals are often based on minimising a measure of expected loss on a future replicate data set. The Akaike information criterion (AIC) calculates for each model the Kullback-Leibler discrepancy, which is a measure of distance between the probability density generated by the model and reality, and does not assume that any of the candidate models is undoubtedly true.

A popular alternative model selection criterion is the Bayesian information criterion or BIC. Wagenmakers and Farrell [69] also explains that a formal comparison in terms of performance between AIC and BIC is very difficult, particularly because AIC and BIC address different questions. Burnham and Anderson [8] agrees and discusses that the fundamental difference between AIC and BIC model selection is their different philosophies, including the exact nature of their target models and the conditions

under which one outperforms the other for performance measures such as predictive mean square error. Thus, selection for use of these two criteria must be based on comparing measures of their performance under conditions realistic of applications.

Another popular model selection criterion is the deviance information criterion (DIC), which was introduced by Spiegelhalter et al. [63]. DIC was constructed to compare the relative fit of a test of Bayesian hierarchical models. It is similar to AIC in combining a measure of goodness-of-fit and measure of complexity, both based on the deviance. As the number of independent parameters in a Bayesian hierarchical model is not clearly defined, DIC estimates the effective number of parameters by the difference of the posterior mean of the deviance and the deviance at the posterior mean. This coincides with the number of independent parameters in fixed effect models with flat priors. DIC can be viewed as a Bayesian analogue of AIC, with a similar justification but wider applicability. It is also applicable to any class of model, involves negligible additional analytic work or Monte Carlo sampling and appears to perform reasonably across a range of examples. The DIC has been used largely in many disciplines and works well for exponential family models but due to its dependence on the parameterisation and focus of a model, its application to mixture models is questionable. In conclusion, a limitation of probabilistic model selection methods is that the same general statistic cannot be calculated across a range of different types of models. Instead, the metric must be carefully derived for each model.

2.2.1 Basic definition

We let $\mathbf{y} = (y_1, \cdots, y_n)$ denote an observed random sample of size n, where y_i is an observed value corresponding to the ith recording of some features on the phenomenon under study. Note that we are using $\mathbf{y}$ to denote the entire sample. We can view $f(y_i)$ as a density of y_i and can be written in the form

$$f(y_i|\boldsymbol{\Theta}) = \sum_{j=1}^{K} w_j f_j(y_i, \boldsymbol{\theta}_j), \quad \text{with} \quad w_j \geq 0, \quad \sum_{j=1}^{K} w_j = 1 \tag{2.6}$$

where the $f_j(y_i, \boldsymbol{\theta}_j)$ are densities of independent and identically distributed (i.i.d) observed data, with $K > 1$, w_j being the mixing proportions of each component and $\boldsymbol{\theta}_j$ denoting the unknown parameters for the jth component in the mixture. We shall refer to the density (2.6) as a K-component finite mixture distribution. Thus, the heterogeneous data is made up of K subgroups, where each group has a different mixing proportion. Due to heterogeneity, $\mathbf{y}$ has a different probability distribution in each group , usually assumed to arise from the same parametric family $f(y|\boldsymbol{\Theta})$, however with each parameter θ_j differing across the groups (McLachlan and Peel [41]). In practice, the components are often taken to belong to the normal family, leading to normal mixtures and this will be pursued further in Section 3.2.1 .

In this particular interpretation of the mixture model, the number of clusters is assumed to be fixed. However, in many applications, the value of K is unknown and has to be inferred from the data, together with the mixing proportions and the parameters for the component densities. When the number of components K is unknown, the parameter space is simultaneously ill-defined and of infinite dimension. This prevents the use of classical testing procedures and priors. The usual approach therefore is to fit the mixture model for fixed K and then to consider the choice of K according to some information criterion that typically penalizes the log likelihood for the complexity of the adopted model, possibly adjusted for the sample size. Thus, Richardson and Green [56] presents a fully Bayesian approach with K taken to be an unknown parameter. Their MCMC methods allow jumps to be made for variable dimension parameters and thus can handle K unspecified and it is also used in Section 3. This method is known as the reversible jump MCMC. In some applications, the clustering of the data is the primary aim of the analysis. In such cases, the mixture model is used purely as a device for exposing any grouping that may underlie the data.

Bayesian inference from data modeled by a mixture distribution can feasibly be performed via Monte Carlo simulation. This approach presents the true Bayesian predictive distribution, implicitly integrating over the whole underlying parameter space.

An infinite number of mixture components can be sustained without any problems, using a prior distribution for mixing proportions that selects a feasible subset of components to explain any finite training set. Thus, the necessity to decide on a correct number of components is in this case avoided. The empirical results exhibited in Neal [47] show that modeling data as an infinite mixture also performs well when there are only a small finite number of components in the real mixture. Thus, this option of infinite mixture models is an attractive option whenever dealing with an unknown number of components, because it avoids the issue of selecting between models with various number of components. Except for the simplest nonhierarchical models, the posterior distributions of Bayesian mixture models can be complicated and analytically intractable, needing simulation-based inferential strategies such as MCMC.

2.3 The prior distribution

Adopting Bayesian analysis, therefore, provides the flexibiluty of incorporating external information as prior beliefs about the parameters. A prior probability distribution of a parameter expresses one's beliefs about this parameter before the data is taken into account. The relative influence of the prior distribution and data on updated beliefs depends on the weight given to the prior and the ability of the data. The more weight given to a prior, the more informative it becomes. This is a sensitive practical problem which must be applied very carefully in order to enclose suitable beliefs in the statistical model. As stated in Gelman et al. [28, p.37]:"A very general feature of Bayesian inference: the posterior distribution is centered at a point that represents a compromise between the prior information and the data, and the compromise is controlled to a greater extent by the data as the sample size increases".

Priors could be created using various methods, depending on whether they are informative or non-informative. Informative prior distributions can be based on pure judgement, a mixture of data and judgement, or data alone. Informative priors include Conjugate priors and Jeffreys priors. The use of informative priors clearly sets

out that the analysis is based on more than just the data and also contains a given amount of judgement concerning plausible values of the parameters based on external information. For example, a prior can be established based on previous experiments or according to some specific principle such as symmetry. It would be more sensible to construct a prior distribution on a scale on which one has a good perception of magnitude, rather than one which may be convenient for mathematical purposes but is fairly complex to understand. The important element to consider is not necessarily to prevent an influential prior, but to be aware of the extent of its influence on the values of the parameters (Congdon [15]).

A prior is said to be a conjugate prior for a family of distributions if the prior $\pi(\boldsymbol{\theta})$and posterior distribution $\pi(|\boldsymbol{\theta}|y)$ are from the same family, which means that the form of the posterior has the same distributional form as the prior distribution. For example, if the likelihood is binomial, $y \sim B(n, p)$, a conjugate prior on p is the beta distribution; it follows that the posterior distribution of is also a beta distribution. Other commonly used conjugate prior/likelihood combinations include the gamma/Poisson, gamma/gamma, gamma/beta and normal/normal cases. The development of conjugate priors was partially driven by a desire for computational convenienceconjugacy provides a practical way to obtain the posterior distributions. Bayes estimators for mixture models are well defined so long as the prior distributions are proper. Provided that suitable (conjugate) priors are used, the posterior density will be proper, thereby allowing the application of MCMC methods such as the Gibbs sampler to provide an accurate approximation to the Bayes solution.

Congdon [15] explains that non-informative or flat priors are appropriate if one is attempting to undertake a more objective approach for analysis or if one has little information about the parameter. This type of priors do not support particular values of the parameter over others, which creates a balance among outcomes by assigning equal probabilities to all values of the parameter. If a Bayesian analysis with non-informative priors is being implemented, this may lead to procedures with attractive

frequentist properties, as many classical procedures correspond to Bayesian analysis with improper priors. One main hindrance is that improper priors yield improper posterior distributions. To determine whether a posterior distribution is proper, you need to make sure that the normalizing constant is finite for all y. An alternative is to use a "partially proper prior". By this is meant a prior that does not require subjective input for the component parameters, yet the posterior is proper (Roeder and Wasserman [59]).

2.4 Prediction by Markov chain Monte Carlo

Markov chain simulation (Metropolis et al. [45]), also known as MCMC, is a powerful tool for calculating probabilities or expectations that are unmanageable by analytical methods or other numerical approaches. It is a general method for the simulation of stochastic processes having probability densities known up to a constant of proportionality. Its advantage over the classical Monte Carlo methods is that it does not require the precise construction of an importance function, while taking into account the characteristics of $\pi(\boldsymbol{\Theta}|y)$. Markov chain Monte Carlo has extensive applicability, even though its performance changes widely, depending on the complexity of the problem. A detailed discussion on MCMC theory and application could be found in Robert and Casella [57].

The basic idea is very simple. If one is unable to find a way to simulate independent realizations of some complicated stochastic process, it is almost as useful to be able to simulate dependent realizations $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \cdots$ forming an irreducible Markov chain having the distribution of interest $\pi(\boldsymbol{\Theta}|y)$ as its stationary distribution. Such methods allow the construction of an ergodic Markov chain with stationary distribution equal to the posterior distribution of the parameter of interest.

The implementation of an MCMC general algorithm starts with arbitrary values chosen for the parameter space $(\theta_1^0, \cdots, \theta_n^0)$ in the first step. Then, in iteration t, new values

$(\theta_1^t, \cdots, \theta_n^t)$ are stochastically chosen in turn, with θ_i^t being picked at position i with probability

$$P(A_h = \theta_h^t | A_i = \theta_i^t, A_j = \theta_j^{t-1} : 1 \leq i < h, h < j \leq n) \tag{2.7}$$

where $(A_1, \cdots, A_n)$ is a set of random variable. The samples for $\boldsymbol{\theta}$ are produced from the joint distribution of these random variables. New values are chosen at each position from the conditional distribution for that position given the latest values at all other positions. Hence, the draws form a Markov chain. Then $\pi(\theta_i^0, \theta_i^1, \cdots, \theta_i^T)$ is a Markov chain if and only if

$$\pi(\theta_i^T | \theta_i^0, \theta_i^1, \cdots, \theta_i^{T-1}) = \pi(\theta_i^T | \theta_i^{T-1}), 1 \leq i \leq n \tag{2.8}$$

so that each iteration is dependent only on the preceding iteration. Then, the iterations are used to calculate the quantities of interest from the posterior distribution. The chain, which is a bundle of draws from the parameters, wanders around the parameter space remembering only where it has been in the last period. For some suitably large T, this method is guaranteed to converge to the equilibrium distribution and the simulated values $(\theta_1^T, \cdots, \theta_n^T)$ can be treated as coming from the desired distribution for $(A_1, \cdots, A_n)$. Such methods allow the construction of an ergodic Markov chain (MC), meaning that the stationary distribution, $\pi(\boldsymbol{\theta})$, is equal to the posterior distribution of the parameters, which is demonstrated in Gelfand and Smith [26]. The stationary distribution can be written as

$$\pi(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|y) \tag{2.9}$$

where $\pi(\boldsymbol{\theta})$ is some distribution Π.

The number or draws required for this method to give an accurate approximation can be difficult to determine. Thus, empirical tests of the use of this method in any

application are necessary.

Except for the simplest nonhierarchical models, the posterior distributions of Bayesian mixture models can be complicated and analytically intractable, necessitating simulation-based inferential strategies such as MCMC. Bayesian estimators for mixture models are well defined so long as the prior distributions are proper. Given that conjugate priors are used, the posterior distribution will be proper, allowing the employment of MCMC methods such as the Gibbs sampler to determine an accurate solution. Diebolt and Robert [20] introduced the Gibbs sampling algorithm for finite mixture models. The Gibbs sampler for hidden Markov mixture models was developed by Albert and Chib [2]. Also, Robert and Titterington [58] synthesizes a general approach to the estimation of the parameters of a hidden Markov model in the cases of normal and Poisson distributions, with the use of a Gibbs sampling algorithm.

Two of the mostly used MCMC methods, Metropolis Hastings algorithm and Gibbs sampler, will be briefly described. However, before introducing the Metropolis-Hastings algorithm and the Gibbs sampler, both discrete time and continuous time Markov chains will be discussed in the next sections.

2.4.1 Discrete time and discrete state space Markov chains

Let $\{A_t\}$ denote the value of a random variable at time t, and let the discrete state space refer to the range of possible A values. the random variable is a Markov process if the transition probabilities between different values in the state space depend only on the random variable's current state, i.e.,

$$P\{A_{t+1} = s_j | A_0 = s_k, \cdots, A_t = s_i, \text{ for } z \leq t\} = P\{A_{t+1} = s_j | A_t = s_i\} \qquad (2.10)$$

This property shows that the probability of any future evolution of the process depends only on its current position, and is not affected by past behaviour. Thus, $\{A_{t+1}\}$ is determined by the previous step $\{A_t\}$ of the process and all of the past steps are

forgotten. A Markov chain refers to a sequence of random variables $(A_0, \cdots, A_n)$ generated by a Markov process. A chain is defined by its transition probabilities (or the transition kernel), which represent the probability that a process at state space s_i moves to state s_j in one step,

$$P(i,j) = P(A_{t+1} = s_j | A_t = s_i) \tag{2.11}$$

Thus, the aim is to estimate the transition rates and probabilities of moving from a state s_i to another state s_j. If, further, the process is independent of time, then the continuous-time Markov chain is said to have stationary or homogeneous transition probabilities. All Markov chains considered in this thesis will be assumed to have stationary transition probabilities.

Let

$$\pi_j(t) = P(A_t = s_j) \tag{2.12}$$

denote the probability that the chain is in state j at time t, and let $\boldsymbol{\pi}(t)$ represent the row vector of the state space probabilities at step t. The chain gets started by defining a starting vector $\pi(0)$. As the chain advances, the probability values get spread out over the state space.

The probability that the chain has state value s_i at step $t+1$ is given by the Chapman Kolmogorov equation, which sums over the probability in being in a particular state at the current step and the transition probability from that state into s_i,

$$\begin{aligned} \pi_i(t+1) &= P(A_{t+1} = s_i) \\ &= \sum_k P(k,i)\pi_k(t) \end{aligned}$$

Then, define the probability transition matrix $\mathbf{P}$ as the matrix whose i,jth element is $P(i,j)$, the probability of moving from state i to state j. The Chapman Kolmogorov

equation in matrix form now becomes

$$\boldsymbol{\pi}(t+1) = \pi(t)\mathbf{P}$$

Now, it becomes obvious how to iterate the Chapman Kolmogorov equation as

$$\boldsymbol{\pi}(t) = \pi(t-1)\mathbf{P} = \pi(t-2)\mathbf{P}^2 = \cdots = \pi(0)\mathbf{P}^t$$

By defining the n-step transition probability p_{ij}^n as the probability that the process is in state j given that n steps ago it was in state i (i.e. $P(A_{t+n} = s_j | A_t = s_i)$, it follows that p_{ij}^n is just the ij-th element of $\mathbf{P}^n$.

A Markov chain is said to be irreducible if there exists a positive integer such that $p_{ij}^{n_{ij}} > 0$ for all i, j. That is, all states communicate with each other, as the chain can always go from any state to any other state (not necessarily in one move). By their nature, all Markov chain produced by MCMC algorithms are irreducible. Then these chains are positive recurrent with stationary distribution $\pi(\boldsymbol{\theta}|y)$. These Markov chains are also ergodic, which means that the distribution of $\boldsymbol{\theta}^T$ converges to $\pi(\cdot|y)$ for almost every starting value $\boldsymbol{\theta}^0$, so the influence of the starting value disappears. Likewise, a state is said to be aperiodic when the number of steps required to move between two states is not required to be mulpitle of some integer. In other words, the chain is not forced into some fixed length cycle between certain states.

Therefore, for k large enough, the resulting $\boldsymbol{\theta}^k$ is approximately distributed from $\pi(\boldsymbol{\theta}|y)$, no matter what the starting value $\boldsymbol{\theta}^0$ is. The problem in practice is then to determine what a "large" k means, since it governs the number of simulations to run. The speed of convergence, that is, the type of decrease in the distance between the distribution of $\boldsymbol{\theta}^k$ and its limit, brings an answer to this problem., but so far it has been mainly studied from a theoretical point of view. Moreover, this rate of convergence often depends on the starting point and a given k does not provide the same quality of approximation

for different values of $\boldsymbol{\theta}^0$. There are thus practical hindrances in the use of Markov chains for simulation since we often ignore whether the chain has been run long enough. But as detailed in Robert and Casella [57], there now are diagnistic tests that provide different indicators on the stationarity of the chain, and thus reduce this difficulty.

2.4.2 Metropolis Hastings algorithm

Once the principle of using a Markov chain with stationary distribution π - instead of i.i.d variables exactly distributed from π -to approximate the parameters is accepted, the implementation of this principle requires the construction of a generation mechanism to produce such Markov chains. An almost universal algorithm satisfying this constraint does exist and it has been constructed by Metropolis et al. [45]. It actually applies to a wide variety of problems, since its main restriction is that the distribution of interest be known up to a constant. A great advantage of this algorithm is the limitless number of proposal distributions that produce a Markov chain that converges to the distribution of interest.

Consider the target distribution, $\pi(\boldsymbol{\Theta}|y)$, over a multidimensional continuous parameter space from which we would like to generate representative values. We must be able to compute the value of $\pi(\boldsymbol{\Theta}|y)$ for any candidate value of $\boldsymbol{\theta}$. The distribution $\pi(\boldsymbol{\Theta}|y)$ does not have to be normalised, however. In typical applications, $\pi(\boldsymbol{\Theta})$ is the unnormalized posterior distribution on $\boldsymbol{\theta}$ (the product of the likelihood and the prior).

The Metropolis-Hastings algorithm can be described as follows. Given a density $\pi(\boldsymbol{\Theta}|y)$, known up to a normalising factor, and a proposal density $q(\boldsymbol{\theta}^{'}|\boldsymbol{\theta})$, the algorithm generates the chain $(\boldsymbol{\theta}^m)_n$ by

1. Choose the initial values $\boldsymbol{\theta}^0 = (\theta_1^{(0)}, \theta_2^{(0)}, \cdots, \theta_n^{(0)})$.

2. Repeat the following $t = 1, 2, \cdots$, until convergence;

2.1. Update θ_1^{t-1} to θ_1^t, according to the conditional density $\pi(\theta_1|\theta_2^{t-1}, \cdots, \theta_n^{t-1})$; simulate θ_1^* from the proposal distribution $q(\cdot|\theta_2^{t-1}, \cdots, \theta_n^{t-1})$ such that

$$\theta_1^{(t)} = \left\{ \begin{array}{cc} \theta_1^* & \text{with probability } \alpha(\theta_1^{(t-1)}, \theta_1^*|\theta_2^{(t-1)}, \cdots, \theta_n^{(t-1)}), \\ \theta_1^{(t-1)} & \text{with probability } 1 - \alpha(\theta_1^{(t-1)}, \theta_1^*|\theta_2^{(t-1)}, \cdots, \theta_n^{(t-1)}), \end{array} \right\}.$$

where

$$\begin{aligned} & \alpha(\theta_1^{(t-1)}, \theta_1^*|\theta_2^{(t-1)}, \cdots, \theta_n^{(t-1)}) \\ = & \min\left\{ \frac{\pi(\theta_1^*|\theta_2^{t-1}, \cdots \theta_n^{t-1})}{\pi(\theta_1^{t-1}|\theta_2^{t-1}, \cdots, \theta_n^{t-1})} \frac{q(\theta_1^{t-1}|\theta_2^{t-1}, \cdots, \theta_n^{t-1})}{q(\theta_1^*|\theta_2^{t-1}, \cdots, \theta_n^{t-1})}, 1 \right\} \end{aligned}$$

$\vdots$

2.n. Update θ_n^{t-1} to θ_n^t, according to the conditional density $\pi(\theta_n|\theta_1^t, \theta_2^t, \cdots, \theta_{n-1}^t)$; simulate θ_n^* from the proposal distribution $q(\cdot|\theta_1^t, \theta_2^t, \cdots, \theta_{n-1}^t)$ such that

$$\theta_n^{(t)} = \left\{ \begin{array}{cc} \theta_n^* & \text{with probability } \alpha(\theta_n^{(t-1)}, \theta_n^*|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)}), \\ \theta_n^{(t-1)} & \text{with probability } 1 - \alpha(\theta_n^{(t-1)}, \theta_n^*|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)}), \end{array} \right\}.$$

where

$$\begin{aligned} & \alpha(\theta_n^{(t-1)}, \theta_n^*|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)}) \\ = & \min\left\{ \frac{\pi(\theta_n^*|\theta_1^{(t)}, \theta_2^{(t)}, \cdots \theta_{n-1}^{(t)})}{\pi(\theta_n^{(t-1)}|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)})} \frac{q(\theta_n^{(t-1)}|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)})}{q(\theta_n^*|\theta_1^{(t)}, \theta_2^{(t)}, \cdots, \theta_{n-1}^{(t)})}, 1 \right\} \end{aligned}$$

Algorithm 1: The Metropolis-Hastings Algorithm

And the M-H algorithm is interpreted as follows: the walk starts at some arbitrary point

specified by the user. The random walk progresses at each time step by proposing a move to a new position in parameter space and then deciding whether or not to accept the proposed move, based on the acceptance probability. If the present state of the Markov process θ^{t-1} is s_i, generate the next state according to the proposal probability $q_{ij} = P(\theta^t = s_j|\theta^{t-1} = s_i)$ and accept a proposal sample $s_j(\neq s_i)$ with probability α_{ij}. If this is rejected (which happens with probability $1 - \alpha_{ij}$) the chain remains at s_i. It means that for any two states i and j, the stationary rate of moving from i to j is equal to the stationary rate j to i. The parameter α_{ij} is called the 'acceptance ratio'. In order to accept the proposed variate $\theta^t = s_j$ with the probability α_{ij} so that the detailed balance equations can be met, we use a uniform random variable $U(0, 1)$.

Proposal distributions can take on many different forms, with the goal being to use a proposal distribution that efficiently explores the regions of the parameter space where $\pi(\boldsymbol{\theta}|y)$ has most of its mass. Of course, we must use a proposal distribution for which we have a quick way to generate random values!

Convergence will be slow and mixing properties will be poor if the proposed transitions are mostly between nearby states in the state space. However, if we choose a proposal distribution with a wide support aiming at distant transitions, it may result in a lower acceptance ratio, which leads to slow convergence and poor mixing. Thus, the proposal distribution should be chosen in such a way as to allow both distant transitions and high acceptance ratio. One way to achieve this is to alternate different proposal distributions in light of sampled elements.

A class of proposals, more related to standard Monte Carlo methods, are the independent proposals, where we choose $q(\cdot|\theta)$ that do not depend on θ,

$$q(\theta^*|\theta) = h(\theta^*).$$

The most common choice for q, starting with Hastings [34], is the random-walk proposal, where $q(\theta^* - \theta)$. This approach takes into account the previously simulated value

to generate the next value. This is to consider a local exploration of the neighbourhood of the current value and then see if the new value θ^* is likely for the target distribution. Here, efficiency is a trade-off between small step size with high probability of acceptance and large step sizes with low probability of acceptance. The Markov chain will thus stay longer in a given point θ^* if the corresponding posterior value $\pi(\theta^*)$ is higher and, conversely, will never visit points θ^* such that $\pi(\theta^*) = 0$. Standard choices for the proposal q are normal, uniform or Cauchy distributions.

2.4.3 The Gibbs sampler

The Metropolis-Hastings algorithm presented in the previous section is very attractive for its universality. The main advantage of Metropolis-Hastings over Gibbs Sampling is that we do not have to derive the conditional distributions analytically. We just need to know the joint distribution. But, in contrast, the lack of connection between the proposal q and the target distribution π can be detrimental to the convergence properties of the approach (if the probability of approaching far away parts of the target distribution is too small). Using a different outlook, the Gibbs sampling method is actually based on the target distribution π.

Gibbs sampling (Geman and Geman [29]) is a Monte Carlo technique for generating random variables from a conditional distribution. At each iteration in the cycle, we are drawing a proposal for a new value of a particular parameter, where the proposal distribution is the conditional posterior probability of that parameter. This means that the proposal move is always accepted. Hence, if we can draw samples from the conditional distributions, Gibbs sampling can be much more efficient than regular Metropolis-Hastings. Such conditional distributions are far easier to simulate than complex joint distributions and usually have simple forms (often being normals, inverse χ^2, or other common prior distributions). Thus, one simulates n random variables sequentially from the n univariate conditionals rather than generating a single n-dimensional vector in a single pass using the full joint distribution.

The idea in Gibbs sampling, as described in Casella and George [9] is to generate posterior samples by sweeping through each variable (or block of variables) to sample from its conditional distribution with the remaining variables fixed to their current values. For instance, consider the random variables $A_1, A_2, \cdots, A_n$. We start by setting these variables to their initial values $\theta_1^{(0)}, \cdots, \theta_n^{(0)}$. At iteration t, one samples $\theta_1^{(t)} \sim f(A_1 = \theta_1 | A_2 = \theta_2^{(t-1)}, \cdots A_n = \theta_n^{(t-1)}), \theta_2^{(t)} \sim f(A_2 = \theta_2 | A_1 = \theta_1^{(t)}, A_3 = \theta_3^{(t-1)}, \cdots A_n = \theta_n^{(t-1)}), \cdots$ and $\theta_n^{(t)} \sim f(A_n = \theta_n | A_1 = \theta_1^{(t)}, A_2 = \theta_2^{(t)}, \cdots A_{n-1} = \theta_{n-1}^{(t)})$. This process continues until convergence (the sampled values have the same distribution as if they were sampled from the true posterior joint distribution). Algorithm 2 presents a generic Gibbs sampler.

1. Initialize $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \cdots, \theta_n^{(0)})$;
2. Repeat the following, $i = 1, 2, \cdots$, until convergence;
   2.1.Simulate $\theta_1^{(i)}$ from the conditional density $f(\theta_1 | \theta_2^{(i-1)}, \cdots, \theta_m^{(i-1)})$;
   2.2.Simulate $\theta_2^{(i)}$ from the conditional density $f(\theta_2 | \theta_1^{(i)}, \theta_3^{(i-1)}, \cdots, \theta_n^{(i-1)})$;
   $\vdots$
   2.n. Simulate $\theta_n^{(i)}$ from the conditional density $f(\theta_n | \theta_1^{(i)}, \cdots, \theta_{n-1}^{(i)})$.
3. The stationary distribution $\boldsymbol{\theta}^{(M)} = (\theta_1^{(M)}, \cdots, \theta_n^{(M)})$, for M large enough, is the true posterior distribution of $f(\theta_1, \cdots, \theta_n | y)$.

Algorithm 2: The Gibbs sampler

the theory of MCMC guarantees that the stationary distribution of the samples generated under Algorithm 2 is the target joint posterior that we are interested in. For comprehensive discussions and implementation on the Gibbs sampling procedure refer to Gelman et al. [27] and Casella and George [9].

The implementation of Gibbs sampler requires the availability of full conditional probability density functions (pdfs) of all the parameters of interest of a problem. For some problems, however, some of the full conditional pdfs have a known form, but some of them cannot be analytically determined. In such cases one can use a hybrid algorithm

where the Metropolis-Hastings and Gibbs sampling procedures can be implemented in a single algorithm, and is known as the Metropolis-within-Gibbs algorithm (Gamerman and Lopes [25]). This method uses the acceptance/rejection sampling approach of the Metropolis-Hastings algorithm, so that some of the component conditional distributions are sampled via Metropolis-Hastings.

In this chapter, the focus has been on introducing the theoretical background for the construction, prior elicitation, estimation and evaluation of mixture distributions using Bayesian inference. It has been demonstrated that mixture models provide a flexible framework for statistical modelling and analysis.

Another extension of the Metropolis-Hastings algorithm presented in this chapter is the reversible jump sampler, where the dimension of the parameter vector varies and is more challenging theoretically; however the resulting algorithm is surprisingly simple to follow. An outline of the samplers theoretical underpinnings is introduced in the following chapter, together with the discussion on the analysis of sampler output. The performance of the reversible jump MCMC is assessed through application to real data sets.

CHAPTER 3

APPLICATION TO FINITE MIXTURE MODELS USING REVERSIBLE JUMP MCMC

In the previous chapter, the Metropolis-Hastings algorithm was introduced. Green [32] applies it to a varying-dimension problem and proposes a new framework for the construction of reversible jump Markov chain algorithm that allows simulation of the posterior distribution on spaces of differing dimensionality. This novel method overcomes computational restraints involving an extensive number of covariates. This is because it is impossible to compute all possible models when the number of covariates is large. Thus, the simulation is possible even if the number of parameters in the model is unknown, which is flexible and fully constructive. This significantly extends the scope of Metropolis-Hastings methods.

In order to appropriately model the datasets used in this thesis, a full Bayesian analysis

of finite mixtures of univariate normals with an unknown number of clusters is presented. It has already been established in Chapter 1.2 that myosin molecules bind in clusters along the thin filament and the size of each cluster depends on the solution conditions. These clusters represent the number of myosin molecules in an active region. Each observation in the dataset is assumed to have emerged from one of K clusters. Thus, the purpose of the mixture analysis is inference about the unknowns: the number K of components, component parameters and the proportion of each cluster. Markov chain Monte Carlo methods are adopted to determine the number of clusters in an active region, via Reversible jump Markov chain Monte Carlo (RJMCMC) modelling.

In this chapter, RJMCMC is presented and discussed. In the first section, the basic assumptions of the algorithm is introduced. Section 2 focuses on the general idea of mixture models with an unknown number of components. In Section 3 a comprehensive analysis of the metastable myosin binding data is used to determine the association rate constant, in order to derive transition matrices. Using this analysis, it is possible to calculate the expected probability for any number of myosins binding to any existing active region. In Section 4, the performance of the methodology is assessed through application to three real data sets. Lastly, in Sections 5 sensitivity and MCMC performance issues are considered.

3.1 RJMCMC algorithm - multiple move types and the model choice problem

Suppose that there exists a countable collection of candidate models $\{\mathfrak{R}_m, m \in \mathfrak{R}\}$, where model $\mathfrak{R}_m$ has a vector $\boldsymbol{\theta}^{(m)}$ of unknown parameters with dimension ρ_m, which may vary from model to model. Under model $\mathfrak{R}_m$, the posterior distribution of $\boldsymbol{\theta}^{(m)}$ takes the form

$$\pi(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m) \propto \pi^*(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m) = L(y|\boldsymbol{\theta}^{(m)}, \mathfrak{R}_m)\pi(\boldsymbol{\theta}^{(m)}|\mathfrak{R}_m) \tag{3.1}$$

where $L(y|\boldsymbol{\theta}^{(m)}, \mathfrak{R}_m)$ is the likelihood function, y is the data, $\pi(\boldsymbol{\theta}^{(m)}|\mathfrak{R}_m)$ is the prior distribution and $\pi^*(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m)$ represents the unnormalised posterior density. Then the joint distribution of $(m, \boldsymbol{\theta}^{(m)})$ given the data takes the form

$$\pi(m, \boldsymbol{\theta}^{(m)}|y) \propto \rho_m \pi^*(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m) \tag{3.2}$$

Reversible jump MCMC is a random movement Metropolis Hastings approach (Metropolis et al. [45]) adjusted for general state spaces. This algorithm has been discussed by several authors, including Richardson and Green [56], Dellaportas and Papageorgiou [17] and Bouguila and Elguebaly [6]. This sampling strategy generates samples from the joint distribution $\pi(m, \boldsymbol{\theta}^{(m)}|y)$ given in (3.2). Just as in ordinary MCMC, although each move is a transition kernel reversible with respect to π, multiple types of moves are required to cross through the whole space $\mathfrak{R}$. The scanning through the available moves is done according to various deterministic or random schedules. Attention is restricted to Markov chains in which the detailed balance is satisfied within each move type.

When the current state is m, a move of type s is proposed, that would take the state to another state m^*, with probability $\alpha_s(m, m^*)$. As is the custom with Metropolis-Hastings algorithms, the proposed value is accepted based on the acceptance probability, where the probability of each move type depends only on the current state. These moves are proposed to have a high acceptance probability. Indexing the move types by s in a countable set $\mathfrak{S}$, a move type s consists of both the forwards move from $(m, \boldsymbol{\theta}^{(m)})$ to $(m^*, \boldsymbol{\theta}^{(m^*)})$ and the reverse, taking $(m^*, \boldsymbol{\theta}^{(m^*)})$ to $(m, \boldsymbol{\theta}^{(m)})$, for a specific pair (m, m^*).

The algorithm is based on producing a Markov chain which can "jump" between models with parameter spaces of different dimensions, whilst satisfying the detailed balance that guarantees the correct limiting distribution. The RJMCMC method outlined in Chen et al. [12], which involves sampling from $\pi(m, \boldsymbol{\theta}^{(m)}|y)$, is given by Algorithm 3.

Assuming that the current state of the chain is $(m, \boldsymbol{\theta}^{(m)})$, proceed as follows

Step 1. Propose a new model $\mathfrak{R}_m$ with probability $j_s(m^*, m)$.

Step 2. Generate $\mathbf{u}$ from a specified proposal density $q_s(\mathbf{u}|\boldsymbol{\theta}^{(m)}, m, m^*)$.

Step 3. Set $(\boldsymbol{\theta}^{(m^*)}, \mathbf{u}^*) = g_s(\boldsymbol{\theta}^{(m)}, \mathbf{u})$ where g_s is a bijection between $(\boldsymbol{\theta}^{(m)}, \mathbf{u})$ and $(\boldsymbol{\theta}^{(m^*)}, \mathbf{u}^*)$ and the lengths of $\mathbf{u}$ and $\mathbf{u}^*$ must satisfy $\rho_m + dim(\mathbf{u}) = \rho_{m^*} + dim(\mathbf{u}^*)$.

Step 4. Accept the proposed move to $(m^*, \boldsymbol{\theta}^{(m^*)})$ with probability

$$\alpha_s(m, m^*) = \min\left\{1, \frac{\rho_{m^*}\pi^*(\boldsymbol{\theta}^{(m^*)}|y, \mathfrak{R}_{m_*})j_s(m|m^*)q_s(\mathbf{u}^*|\boldsymbol{\theta}^{(m^*)}, m^*, m)}{\rho_m\pi^*(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m)j_s(m^*|m)q_s(\mathbf{u}|\boldsymbol{\theta}^{(m)}, m, m^*)} \times \left|\frac{\partial g_s(\boldsymbol{\theta}^{(m)}, \mathbf{u})}{\partial(\boldsymbol{\theta}^{(m)}, \mathbf{u})}\right|\right\}$$

where $\pi^*(\boldsymbol{\theta}^{(m)}|y, \mathfrak{R}_m)$ is given by equation (3.1). Here, the Jacobian factor is from the transformation from $(\boldsymbol{\theta}^{(m)}, \mathbf{u})$ to $(\boldsymbol{\theta}^{(m^*)}, \mathbf{u}^*)$, and is dependent on the move type s.

Algorithm 3: Reversible jump MCMC

To summarise, reversible jump MCMC is just a Metropolis-Hastings algorithm, defined to allow for sampling from a distribution on a group of spaces of various dimensions, and enabling state-dependent choice of move type. This method provides great flexibility to the algorithm designer to profit from the structure of the problem at hand.

3.2 Mixtures Analysis with an unknown number of components

As mentioned in the previous section, RJMCMC allows the Markov chain to move between parameter subspaces corresponding to statistical models with different dimensions. In this chapter, it is assumed that K, which represents the number of binders

on the thin filament, is a random variable and $K \leq K_{max}$ for a given value of K_{max}. The RJMCMC algorithm is constructed in a manner similar to the one-dimensional approach of Richardson and Green [56], so the required reversible jump transformation requires split, merge and birth-death moves. The approach produces a good mixing of the chains and is tested with real data.

The results from this model are used to compute the transitions from a state/sub-population i to another state j, for $i, j = 1, \cdots, K$ and $i \neq j$. First, we model the number of components and the mixture component parameters jointly and base inference about these quantities on their posterior probabilities. Using MCMC enables the simultaneous exploration of the parameters and model space by treating the number of myosin binders as being random and it is automatically adapted at each step. The RJMCMC regularly proposes a move to a different dimension and rejects this proposal with appropriate probability to ensure the chain crosses the stationary distribution.

3.2.1 Univariate normals with an unknown number of components

Here it is assumed that each data point y arises from a mixture of normal distributions and is modeled as follows:

$$\rho(\mathbf{y}|\boldsymbol{\delta}, \mathbf{w}, K) = \sum_{j=1}^{K} w_j \mathcal{N}(\mathbf{y}, \delta_j), \quad \text{with} \quad w_j \geq 0, \quad \sum_{j=1}^{K} w_j = 1 \tag{3.3}$$

The above equation is an association of K normal distributions $\mathcal{N}(\boldsymbol{.})$ of independent and identically distributed (i.i.d) observed data, $\mathbf{y} = y_1, \cdots, y_n$, with $K \in \{0, 1, \cdots, 11\}$ and w_j being the proportion of each component. Thus, the purpose of the mixture analysis is inference about the unknowns: the number K of myosin binders, group parameters $\boldsymbol{\delta}$ and the proportions $\mathbf{w}$. The parameter $\boldsymbol{\delta}$ is a vector of combinations (μ_j, σ_j^2) for $j = 1, 2, \cdots, K$.

The above model implies a heterogeneous population consisting of components $j =$

$1, \cdots, K$ of sizes proportional to w_j from which the random sample is selected. The label of the component from which each sample is drawn is unidentified. Thus, we represent the mixture components generating each observation via latent allocation variables z_i, for $i = 1, \cdots, n$. Each z_i is an integer denoting the unknown component from which each observation y_i is drawn. These different realizations z_i of the unobserved vector $\mathbf{Z} = (z_1, \cdots, z_n)$ are drawn independently from the distributions

$$\rho(z_i = j) = w_j \quad \text{for} \quad j = 1, 2, \cdots, K \tag{3.4}$$

and conditional on $\mathbf{Z}$, the realizations y_i are selected from their respective normal densities:

$$\rho(y_i | z_i = j, \mathbf{w}, \boldsymbol{\delta}) = \mathcal{N}(\mathbf{y_i}; \delta_j) \quad \text{for} \quad i = 1, \cdots, n. \tag{3.5}$$

Integrating out the missing data $\mathbf{Z}$ then returns the model in equation (2.6).

Hierarchical Model and Priors

In this section, a general hierarchical model for mixtures is presented, which has been proved to be weakly informative by Richardson and Green [56]. This case is more appropriate for our datasets because an objective prior is preferred. The inference should be done mostly based on the data available, as the prior information is not very solid at this stage.

In Bayesian analysis, the unknown parameters $K, \mathbf{w}$, and $\boldsymbol{\delta}$ are viewed as random variables and are selected from suitable prior distributions. The joint probability distribution of all these variables, in a general format, can be written as follows:

$$\rho(\mathbf{K}, \mathbf{w}, \mathbf{z}, \boldsymbol{\delta}, y) = \rho(K)\rho(\mathbf{w}|K)\rho(\mathbf{Z}|\mathbf{w}, \mathbf{K})\rho(\boldsymbol{\delta}|\mathbf{Z}, \mathbf{w}, K)\rho(y|\boldsymbol{\delta}, \mathbf{Z}, \mathbf{w}, K). \tag{3.6}$$

A common approach is to introduce the conditional independecies $\rho(\boldsymbol{\delta}|\mathbf{Z}, \mathbf{w}, K) = \rho(\boldsymbol{\delta}|K)$ and $\rho(y|\boldsymbol{\delta}, \mathbf{Z}, \mathbf{w}, K) = \rho(y|\boldsymbol{\delta}, \mathbf{Z})$, such that the joint probability distribution is

now simplified into

$$\rho(\mathbf{w}, \mathbf{Z}, K, \boldsymbol{\delta}, y) = \rho(K)\rho(\mathbf{w}|K)\rho(\mathbf{Z}|\mathbf{w}, K)\rho(\boldsymbol{\delta}|K)\rho(y|\boldsymbol{\delta}, \mathbf{Z}).$$

For full adaptability, an extra tier is introduced to the hierarchy to allow the priors for $(K, \mathbf{w}, \boldsymbol{\delta})$ to depend on hyperparameters λ, ϵ and η respectively. These hyperparameters are selected from independent hyperpriors. As a result of introducing the extra tier to the hierarchy, $\rho(K) = \rho(K|\lambda)$ because K is now dependent on λ. Also, $\rho(\mathbf{w}|K) = \rho(\mathbf{w}|K, \epsilon)$ as $\mathbf{w}$ becomes dependent on ϵ and $\rho(\boldsymbol{\delta}|K) = \rho(\boldsymbol{\delta}|K, \eta)$ because $\boldsymbol{\delta}$ depends on η. Then, the joint probability distribution can be written as

$$\rho(\lambda, \epsilon, \eta, K, \mathbf{w}, \mathbf{z}, \boldsymbol{\delta}, y) = \rho(\lambda)\rho(\epsilon)\rho(\eta)\rho(K|\lambda)\rho(\mathbf{w}|K, \epsilon)\rho(\mathbf{Z}|\mathbf{w}, K)\rho(\boldsymbol{\delta}|K, \eta)\rho(y|\boldsymbol{\delta}, \mathbf{Z}). \tag{3.7}$$

where $\rho(\lambda)$ represents the prior distribution for λ, $\rho(\epsilon)$ is the prior distribution for ϵ and $\rho(\eta)$ is the prior distribution for the hyperparameter η.

The specification of the prior distributions should be done with great care. Even in the absence of strong prior information, prior specification should be done at the appropriate scale of biological interest. Being fully non-informative and achieving a proper posterior distribution is unattainable in a mixture context. This is because there is always a chance that no observations are assigned to one or more components and thus the data are uninformative about these components. Hence the hyperprior structure and the default hyperparameter choices are presented, which correspond to making only the slightest assumptions on the data.

In specifying the priors, the approach suggested by Richardson and Green [56] is still being followed (also employed by Stephens [65]). The parameter $\boldsymbol{\delta}$ is a vector of combinations (μ_j, σ_j^2) for $j = 1, 2, \cdots, K$ such that

$$f(y|\delta_j) = \mathcal{N}(y|\mu_j, \sigma_j^2),$$

with μ_j and σ_j^{-2} drawn independently from a normal and a gamma prior distribution respectively

$$\mu_j \sim \mathcal{N}(\zeta, \kappa^{-1}) \quad \text{and} \quad \sigma_j^{-2} \sim \Gamma(\alpha, \beta). \tag{3.8}$$

Another hierarchical level is included by allowing β to follow $\Gamma(g, h)$, where $\alpha > 1 > g$ is taken to suggest that the σ_j^2 (variances) are similar, without imposing any information about their size. Note that η has now become $(\zeta, \kappa, \alpha, \beta)$. The scale parameter h is set as $10/R^2$, where R is the range of data y.

A key assumption that has to be made is related to the labelling of components. To allow for detectability of each component, an unique labelling system has to be used. That is where the μ_j are in increasing numerical order; hence the joint prior distribution of the model parameters is $K!$ times the product of the individual normal and gamma distributions, limited to the set $\mu_1 < \mu_2 < \mu_3 \cdots < \mu_K$.

The prior on $\mathbf{w}$ is in all cases selected as symmetric Dirichlet

$$\mathbf{w} \sim D(\epsilon, \epsilon, \cdots, \epsilon) \tag{3.9}$$

and a proper prior must be adopted for K. It is chosen such that

$$p(K) \sim Poisson(\lambda). \tag{3.10}$$

Lastly, in this thesis, ϵ and λ are kept fixed.

The hierarchical model with fixed α and random β applied for the variance distribution allows a low degree of information to be passed on to the results of the analysis. Accordingly, we have chosen $\alpha = 2, g = 0.2, \text{and } h = 10/R^2$ in agreement with Richardson and Green [56].

Normal mixtures

Normal mixtures are considered, rather than mixtures of other distributions, because the mixture components are normally distributed. Following 3.7, for the hierarchical normal mixture model there are 6 move types:

- a) updating the weights;
- b) updating parameters (μ, σ);
- c) updating the allocation z;
- d) updating the hyperparameter β;
- e) splitting one component into two, or combining two into one;
- f) birth or death of an empty component.

The only randomness in the scanning is the random choice in moves e) and f), where the algorithm has to choose between splitting and combining or to choose between birth and death. Moves e) and f) involve changing K by 1 and making essential corresponding changes to (μ, σ, w, z). A sweep is represented by one complete pass over all 6 moves, which is a basic time step for the RJMCMC algorithm.

Move types a), b), c) and d) are relatively simple to define, since the conjugate nature of the priors leads to relatively simple forms for the full conditional distribution of the desired parameter. Thus the first 4 moves are Gibbs sampling moves and they largely follow Diebolt and Robert [20].

Through conjugacy, the full conditional distribution for the weights w takes the form

$$\rho(w_j|\epsilon, n_j) \sim D(\epsilon + n_1, \cdots, \epsilon + n_k), \tag{3.11}$$

where $n_j = \#\{i : z_i = j\}$, meaning that n_k is the number of observations allocated to component K. Thus w can be updated by a Gibbs move, sampling from the full conditional distribution by drawing independent gamma random variables.

The full conditionals for μ_j are

$$\rho(\mu_j|y,\sigma_j^{-2},\zeta,\kappa) \sim \mathcal{N}\left\{ \frac{\sigma_j^{-2}\sum_{i:z_i=j} y_i + \kappa\zeta}{\sigma_j^{-2}n_j + \kappa}, (\sigma_j^{-2}n_j + \kappa)^{-1} \right\} \tag{3.12}$$

The full conditionals for σ_j^2 are

$$\rho(\sigma_j^{-2}|y,\mu_j,n_j,\alpha,\beta) \sim \Gamma\big\{\alpha + \frac{1}{2}n_j, \beta + \frac{1}{2}\sum_{i:z_i=j}(y_i-\mu_j)^2\big\}, \tag{3.13}$$

and for the latent variables we have

$$\rho(z_i = j|y,w_j,\mu_j,\sigma_j^2) \propto \frac{w_j}{\sigma_j}\exp\left\{ -\frac{(y_i-\mu_j)^2}{2\sigma_j^2} \right\} \tag{3.14}$$

.

Also, the full conditional distribution for β, the hyperparameter which is not fixed, is a gamma distribution

$$\rho(\beta|g,\kappa,\alpha,h,\sigma_j^{-2}) \sim \Gamma(g+\kappa\alpha, h+\sum_j \sigma_j^{-2}). \tag{3.15}$$

For the split/combine move e), the reversible jump mechanism is required. The main criteria which must be met when designing these moves are that they are irreducible, aperiodic, form a reversible pair and satisfy the detailed balance. This move takes the form of a Metropolis-Hastings step where a move from state $\boldsymbol{\theta}^{(m)}$ to state $\boldsymbol{\theta}^{(m')}$ is proposed, with $\pi(\boldsymbol{\theta}^{(m)})$ the target distribution and $q_s(\boldsymbol{\theta}^{(m)}, \boldsymbol{\theta}^{(m')})$ the proposal distribution for the move s. The proposed transition is then accepted with probability α_s

$$\alpha_s = min\left\{ 1, \frac{\pi(\boldsymbol{\theta}^{(m')})q_s(\boldsymbol{\theta}^{(m')}, \boldsymbol{\theta}^{(m)})}{\pi(\boldsymbol{\theta}^{(m)})q_s(\boldsymbol{\theta}^{(m)}, \boldsymbol{\theta}^{(m')})} \right\} \tag{3.16}$$

For the case where a move from state $\boldsymbol{\theta}^{(m)}$ to state $\boldsymbol{\theta}^{(m')}$ lies in a higher dimensional space, the move can be completed by drawing a vector of continuous random variables u, independent of $\boldsymbol{\theta}^{(m)}$. The new state $\boldsymbol{\theta}^{(m')}$ is decided upon using an invertible deterministic function of $\boldsymbol{\theta}$ and u. Green [33] shows that the acceptance probability of transitioning to a higher dimensional space is given by

$$\alpha_s = min\left\{1, \frac{\pi(\boldsymbol{\theta}^{(m')})r_s(\boldsymbol{\theta}^{(m')})}{\pi(\boldsymbol{\theta}^{(m)})r_s(\boldsymbol{\theta}^{(m)})q(u)}\left|\frac{\partial\boldsymbol{\theta}^{(m)}}{\partial(\boldsymbol{\theta},u)}\right|\right\} \tag{3.17}$$

where $r_s(\boldsymbol{\theta}^{(m)})$ is the probability of choosing move type s when in state $\boldsymbol{\theta}^{(m)}$ and $q(u)$ represents the density function of u. The final term in the ratio is a Jacobian emerging from the change of variable from $(\boldsymbol{\theta}^{(m)}, u)$ to $\boldsymbol{\theta}^{(m')}$.

In move e), a random choice is made between attempting to split or combine with one of its neighbours with probability ρ_{s_k} and $\rho_{c_k} = 1 - \rho_{s_k}$, depending on k. Also, $\rho_{c_1} = 1$ and $\rho_{s_K} = 0$, where K is the maximum number of components; otherwise $\rho_{s_k} = \rho_{c_k} = 0.5$, for $k = 2, 3, \cdots, K-1$. If the choice is to combine the component, then a pair of components (j_1, j_2) that are adjacent in terms of their means is chosen at random. These two components are merged, reducing k by 1 and the new component is labelled j^c. Values for $(w_{j^c}, \mu_{j^c}, \sigma_{j^c})$ have to be created, so the parameters for j^c are calculated from

$$\begin{aligned} w_{j^c} &= w_{j_1} + w_{j_2}; \\ w_{j^c}\mu_{j^c} &= w_{j_1}\mu_{j_1} + w_{j_2}\mu_{j_2}; \\ w_{j^c}(\mu_{j^c}^2 + \sigma_{j^c}^2) &= w_{j_1}(\mu_{j_1}^2 + \sigma_{j_1}^2) + w_{j_2}(\mu_{j_2}^2 + \sigma_{j_2}^2). \end{aligned} \tag{3.18}$$

This proposal to combine components is deterministic once the choices for j_1 and j_2 have been made, so the acceptance probability is given by equation (3.16).

If the decision is to split, a component j^s is chosen at random and split into two components, labelled j_1 and j_2, with parameters conforming to equations (3.18). In making this transformation there are 3 degrees of freedom, so a three-dimensional random vector u has to be generated to enable the specification of the new component weights and parameters. Beta distribution is used to generate the random vector u because u is a continuous random variable whose range is between 0 and 1. This random generation is done as follows

$$u_1 \sim \text{Beta}(2,2), \quad u_2 \sim \text{Beta}(2,2), \quad u_3 \sim \text{Beta}(2,2).$$

The split transformation, as proposed by Richardson and Green [56], is then defined by:

$$\begin{aligned} w_{j_1} = w_{j^s}\mu_1, \qquad w_{j_2} &= w_{j^s}(1-\mu_1), \\ \mu_{j_1} &= \mu_{j^s} - u_2\sigma_{j^s}\sqrt{\left(\frac{w_{j_2}}{w_{j_1}}\right)}, \\ \mu_{j_2} &= \mu_{j^s} + u_2\sigma_{j^s}\sqrt{\left(\frac{w_{j_1}}{w_{j_2}}\right)}, \\ \sigma^2_{j_1} &= u_3(1-u_2^2)\sigma^2_{j^s}\frac{w_{j^s}}{w_{j_1}}, \\ \sigma^2_{j_2} &= (1-u_3)(1-u_2^2)\sigma^2_{j^s}\frac{w_{j^s}}{w_{j_2}}, \end{aligned} \tag{3.19}$$

which provide the needed weights and parameters, satisfying equations (3.18)

In the birth and death move f), a random choice between birth and death is made first, using the same probabilities ρ_{s_k} and ρ_{c_k} as above. For a birth, a weight and parameters for the proposed new component are selected using

$$w_{j^s} \sim Beta(1,k) \qquad \mu_{j^s} \sim \mathcal{N}(\zeta, \kappa^{-1}) \qquad \sigma^{-2}_{j^s} \sim \Gamma(\alpha, \beta).$$

In order to allow for a new component, the existing weights are rescaled, so that all

weights sum up to 1. For a death, a random choice is made between any existing empty components. The chosen component is deleted and the remaining weights are rescaled to sum up to 1. Detailed balance holds for this move, given that births and deaths are accepted according to equation 3.17, where u is substituted by $({w_j}^s, {\mu_j}^s, {\sigma_j}^{s2})$.

At this point one has a better understanding of the concepts needed to comprehend the RJMCMC algorithm and the mathematical tools for executing it with either real or simulated data. So the next section covers the application of the RJMCMC algorithm on our data.

3.3 Performance of reversible jump MCMC

Examples of the results obtained from real datasets are displayed in this section. First to be presented, is the description of model performance with default settings for the hyperparameters. Six real data sets are used throughout the thesis, as a basis for our comparison. An introduction to this data has been made in Section (1.2.2). Three of these datasets have been collected under the same conditions, and the following three have been collected in slightly different conditions. However, only one of each conditions will be presented in this section and the rest of the results are to be found in Appendix B and C.

The implementation of the RJMCMC algorithm has been done by employing the existing **miscF** package (Feng [22]) for R. The built-in function **uvnm.rjmcmc** was used to estimate the parameters of an univariate normal mixture model including the number of components using the RJMCMC method. Similarly, the **coda** package (Plummer et al. [53]) contains several graphics functions for visualising MCMC output, which have been used successfully to check for Markov chain convergence. The trace plots, density plots and Gelman plots were created using graphics functions from the **coda** package.

The analysis has been carried out with the hierarchical normal random β mixture model defined in Section (3.2.1). Also, for each of the six datasets, we report results corresponding to 30000 draws and a burn-in period of 5000 draws. We consider that these numbers go beyond what is needed to obtain reliable results. Moreover, for each data set four chains were run in parallel with different starting points. At each sweep of the RJMCMC algorithm, the chain has the ability to move between different values of K.

When making an inference from an MCMC analysis, one has to ensure that an equilibrium distribution, also known as convergence, has indeed been reached by the Markov chain. An MCMC creates a sample from the posterior distribution, and the question

is whether this sample is sufficiently close to the posterior to be used for analysis. For each parameter, the initial value of the chain is started at an arbitrary point. Because consecutive draws are dependent on the previous value of each parameter, the actual values chosen for the starting points are observable for a while before the chain becomes independent of the initial values. Nevertheless, a bad starting value can lead to slow convergence. This can be diagnosed from one run and corrected by changing the starting value. These first draws are to be discarded at the burn-in stage as they are unrepresentative of the equilibrium distribution of the Markov chain.

From Markov chains theory, the chains are expected to eventually converge to the stationary distribution. Chains should be run out long enough so that all the potential scale reduction factors are small enough. However, there is no guarantee that the chains will converge after M draws. There are several ways to check for convergence, both visual and statistical.

The simplest method, which is also employed in this chapter, is just to inspect plots of the chains visually: they should look like nice oscillograms around a horizontal line without any trend. This method shows how well the chain is mixing, or moving around the parameter space. If the chain is taking a long time to move around the parameter space, then it will take longer to converge. Only the draws obtained after the chain has converged should be included in the analysis, for accurate and relevant results.

A statistical convergence diagnostics is also applied after the visual examination, in order to guarantee the efficiency of the model. The method developed by Andrew Gelman [3] is employed, as this is perhaps one of the most popular diagnostics. This Gelman-Rubin diagnostic measures whether there is a significant difference between the variance within several chains and the variance between several chains by a value that is called scale reduction factors. To do this, at least two chains would have to be simulated in parallel, each with different starting points which are overdispersed with respect to the target distribution.

A factor of 1 means that between variance and within chain variance are equal, larger values mean that there is still a notable difference between chains. Any value significantly above 1 would suggest lack of convergence. Thus, this statistic measures the potential advancement, in terms of the estimate of the variance in the variable, which could be achieved by running the chains to infinity. When a small amount of advancement could be gained, the chains are taken as being mixed. The Gelman diagnostic plot is also a nice tool to see roughly where this point is, that is, from which point on the chains seem roughly converged. The Gelman plot shows the development of the scale-reduction over time (chain steps), which is useful to see whether a low chain reduction is also stable. This is done by calculating the shrink factor of all the parallel Markov chains at various points in time.

Below, Figure 3.1 illustrates the distributions of all 6 datasets to be analysed in this chapter. It is very clear from these histograms that the actin data does not have a consistent mean light intensity in all 3 datasets. Dataset 1, which is the grey distribution in the left histogram, does not overlap at all with the other two datasets with the same conditions. Whereas the other 2 actin datasets have a high degree of overlap, with the most frequent mean intensity of around 1000. Then , on the right hand side, there is the histogram of calcium datasets with an approximate mean intensity value of 3200. One can clearly observe that these calcium datasets have a high degree of overlap, showing consistency in the data.

After this initial view of the data distributions, in the analysis it is expected to see an increased number of clusters for higher light intensities, thus for pCa 6 datasets. Desai et al. [19] argues that increased concentrations of myosin and calcium favor myosin association, leading to more active regions on the thin filament. Thus, thin filament activation responds to myosin and calcium.

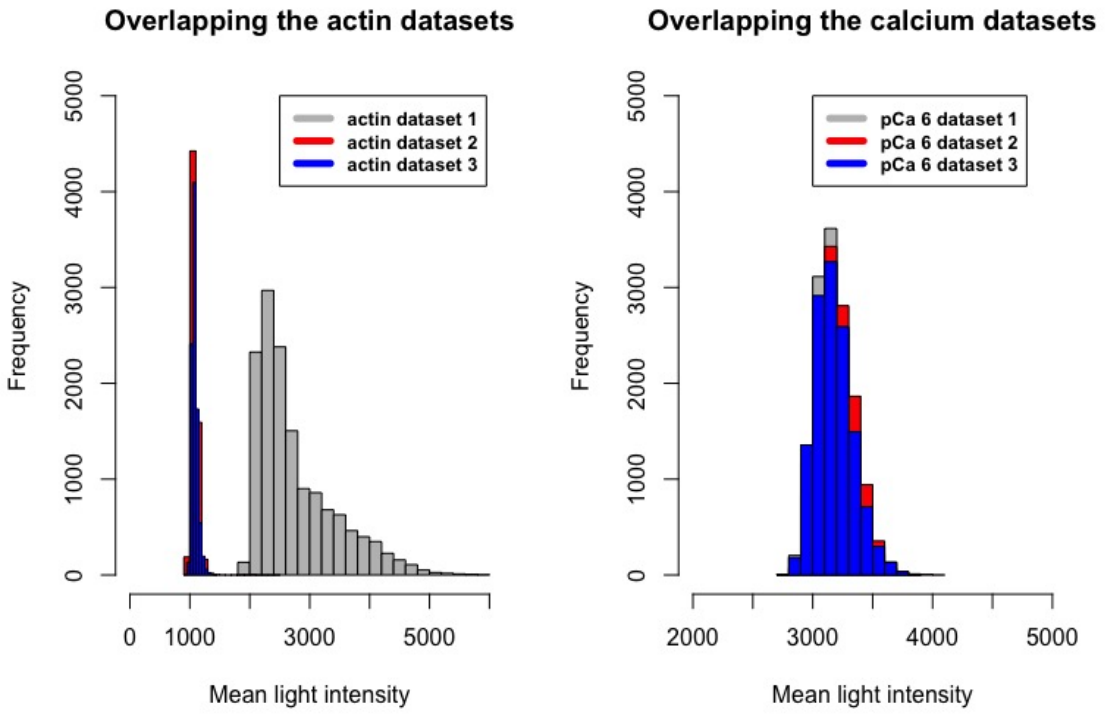


Figure 3.1: Overlapping histograms of all 6 datasets with conditions 10 nM of myosin at pCa 6 on the left hand side histogram and 5 nM actin on the right hand side histogram. The peaks correspond to the most common number of light intensities, and the histograms become skewed to higher intensities as more myosins bind to the thin filament.

3.3.1 Dataset 1 with conditions 5 nM actin

As a first step, we examined the interaction of 5nM actin data. Movies of these interactions were taken, and slices along thin filaments were projected through time to generate kymographs. In this section, we present the results of one kymograph under the conditions mentioned above because datasets with the same conditions show similar analysis results. This can be seen in Appendix B where the outcomes from two further kymographs with the same conditions could be found.

Before doing any analysis, we looked at the descriptive statistics for the raw data presented in Table 3.1. The table shows that the minimum value of a pixel is 1877 and

the mean value is 2760. Also, the standard deviation (SD) is 657. Figure 3.2 displays the positively skewed distribution of data, where 75% of values are found in between 1877 and 3098. So by looking at the distribution of data, it is not clear how many clusters there are in this dataset.

Table 3.1: Summary statistics

Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**
1877	2270	2532	2760	3098	5944	657

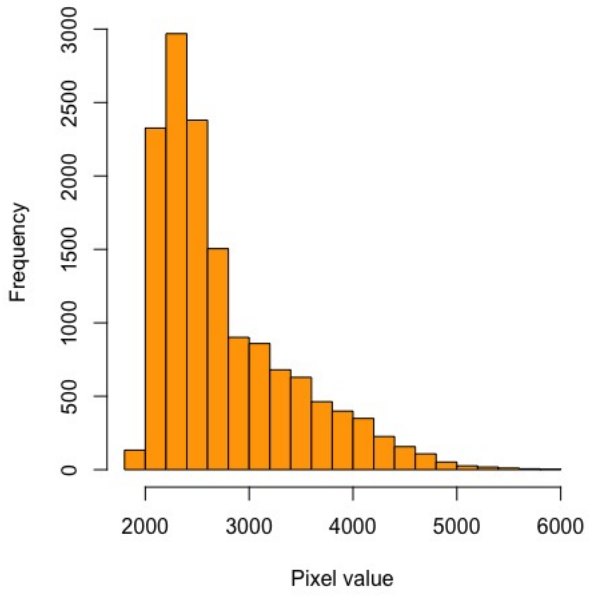


Figure 3.2: Histogram of dataset with conditions actin only and myosin II = 5nM

The implementation of the RJMCMC algorithm will now be presented, corresponding to 30000 iterations and a burn in period of 5000 draws. Given that the number of clusters is unknown, the starting points for the Markov chains were chosen based on the summary statistics and expert knowledge.

To understand the behaviour of myosin within active regions we applied a Markov Chain statistical approach. This allows the binding and release events within a dataset

to be analysed. For one kymograph/dataset we generated four independent Markov chains using the RJMCMC algorithm and calculated the average mean and variance value for each population of binders. The first two chains were obtained by conditioning on $K = 1$, whereas the other 2 chains were conditioned on $K = 6$. The starting values can be found in Table 3.2 and these were chosen to resemble the descriptive statistics.

Table 3.2: Starting values used for the RjMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.15	0.15	0.2	0.2	0.18	0.12
Mean	2600	2000	2300	2760	3000	3400	3800
Variance	657	550	600	657	600	595	665

Markov chain diagnostics is a critical issue. This is because these tools are used to check whether the quality of a sample generated with an MCMC algorithm is sufficient to provide an accurate approximation of the target distribution. As a result, we start with the visual inspection of the trace and density plots for parameter K for all the chains, where K represents the number of myosins bound to the thin filament.

Figure (3.3) illustrates the mixing over K for each chain, which shows the values taken by K during the run time of the chain. A good sign of convergence is that there are no breaks in the chains to suggest poor mixing. Also the chains seem to move from one state to another without getting stuck for too long in one state. Healthy chains jump up and down frequently. It appears that most sampled values of all three chains are between 7 and 9 components after reaching equilibrium. This can also be seen in the density plots in Figure (3.4), where the distribution of the values for parameter K is presented. Thus, from the visual inspection we can say that the MCMC method has captured the true population distribution. However, to establish that convergence is achieved, we also examine the Gelman plot.

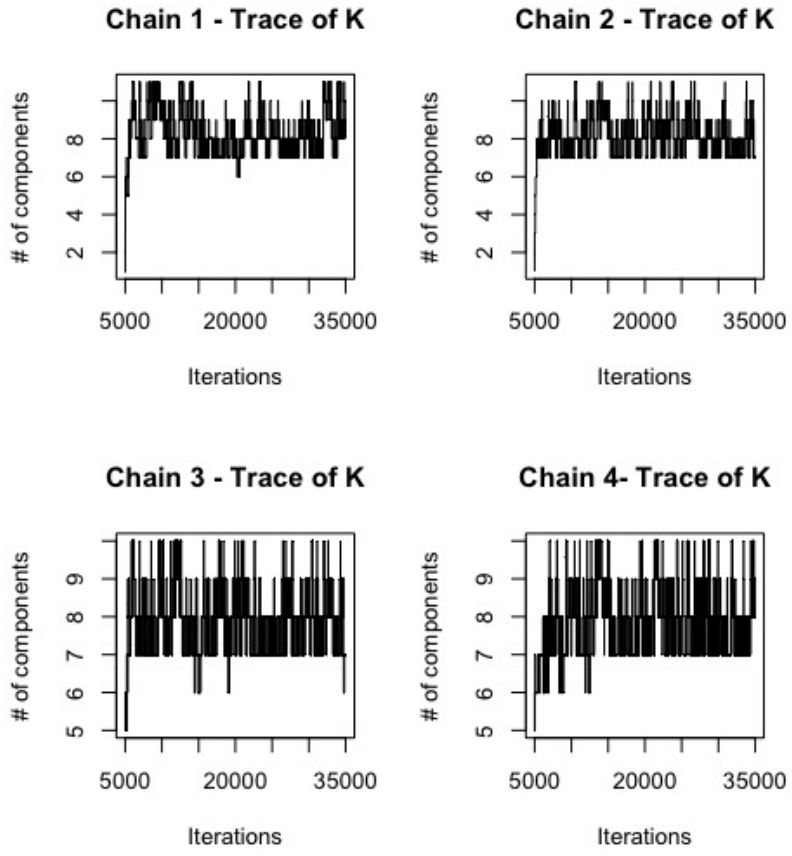


Figure 3.3: Traces for mixing over K for a concentration of actin only and myosin II = 5nM, over 30000 sweeps

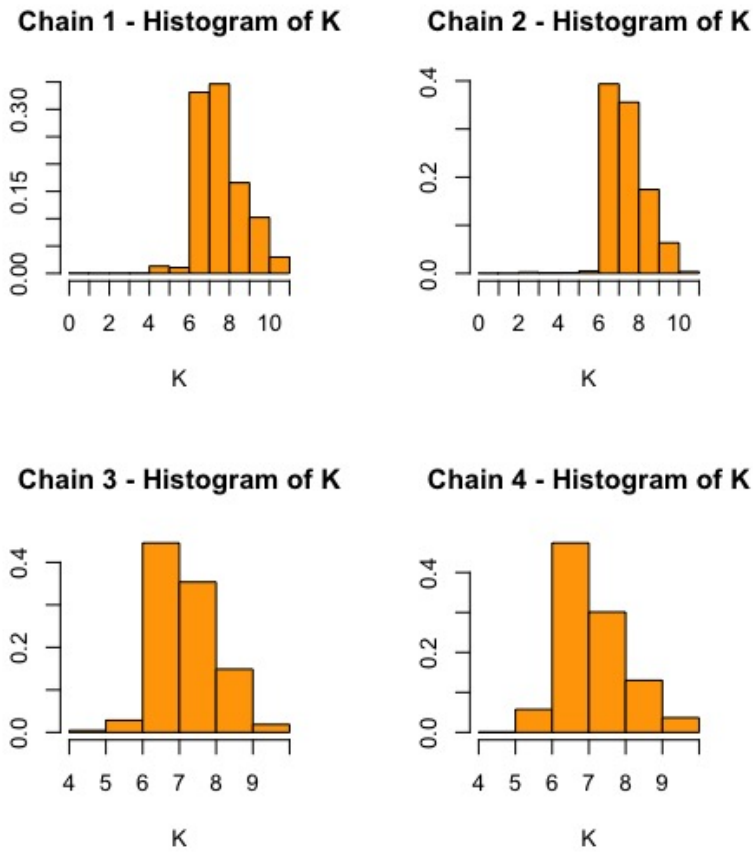


Figure 3.4: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

Figure 3.5 illustrates the Gelman plot, which shows the development of the scale reduction factor over time (chain steps). It is helpful to see whether a low chain reduction is stable, meaning that it does not go down and then up again. This is because the bias that arises from the starting values until convergence, has to be discarded. So when the shrink factor gets close to 1 (perhaps not greater than 1.1 or 1.2) that is roughly the point when the chains reach convergence. Only iterations that are on the converged part of the chain must be used for statistical analysis.

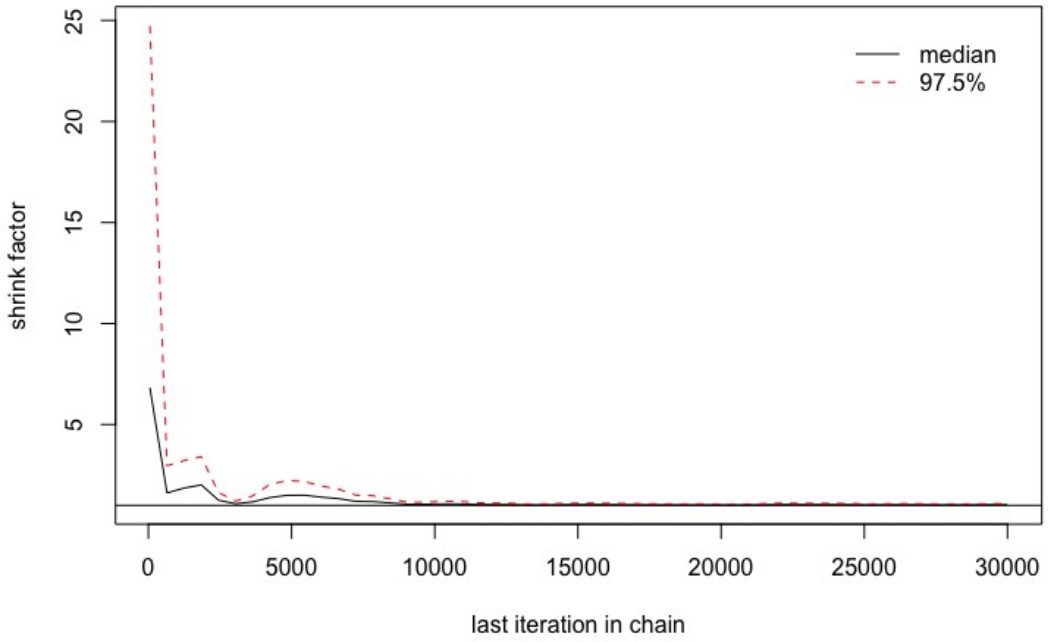


Figure 3.5: Gelman plot for all four chains with different starting points

The Gelman plot is a useful tool to see roughly where the convergence point is, in order to appropriately decide on the burn in stage. Figure 3.5 shows that the chains start getting close to 1 after 10000 iterations, meaning that the stationary distribution has being reached at that stage. So, as a result the burn in stage is set at 10000 draws, being left with 20000 draws for analysis. Similarly, Table 3.3 presents the potential scale reduction factor (psrf) for the Gelman diagnostic of all chains, where both values are almost 1. This provides further assurance that the chains have indeed attained stationarity.

Table 3.3: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.034217	1.08674

The RJMCMC algorithm calculates the average mean, variance value and weight for

each population of binders. The mean corresponds to the idealized pixel intensity values for the stated number of binders. In order to decide which number of components would interest us most, we have looked at the spread of data for each chain given by Table 3.4. It appears that all four chains have jumped to 7 and 8 components the most, meaning that the RJMCMC algorithm considers that the dataset is most likely to have 7 or 8 clusters. Prior information says that light intensity increments are linear with the number of binders. Thus, we will examine both 7 and 8 clusters to find out which of the two have a more linear relationship.

Table 3.4: The spread of data resulting from the RjMCMC algorithm

# of components	**6**	**7**	**8**	**9**	**10**	**11**
Chain 1	278	8187	7421	2501	1372	241
Chain 2	0	8053	7615	3222	1048	62
Chain 3	494	9343	7578	2386	199	0
Chain 4	0	10831	6178	2389	602	0

Table 3.5 presents the mean intensities, variance and weights for each binder corresponding to a total of either 7 components (equivalent to background plus 6 binders) or 8 components (equivalent to background plus 8 binders). For both 7 and 8 components, there are clear linear increments between binders, as seen in Figure 3.6. The linear plot was constructed using the mean values from Table 3.5. Given that one of our assumptions is linear relationship, it seems that the 8 component simulation provides a more saturated Gaussian mixtures model and will be used as the basis for further analysis. Further confirmation of the choice of component number derives from the weighting or relative abundance of these binders. For 8 components the predominant population is between 0 and 4 molecules per bound cluster but there is still a 29% chance for 5 to 7 binders (8% weighting for 5 binders, 8% for 6 binders and 13% for 7 binders).

A simple approach for choosing the cut-off pixel intensity value of each population of binders is to use the midpoint between successive means for the 8 components. For

Table 3.5: Summary results including the mean, variance and weight of having a total of either 6 or 7 myosin binders

# of	**Total of 6 binders (7 components)**			**Total of 7 binders (8 components)**		
binders	Mean = μ_6	Variance= σ_6	Weight=w_6	Mean = μ_7	Variance= σ_7	Weight=w_7
0	2150	88	0.2	2139	88	0.19
1	2343	122	0.22	2329	119	0.2
2	2583	165	0.24	2530	156	0.19
3	3012	161	0.09	2807	159	0.11
4	3353	230	0.08	3126	187	0.08
5	3823	500	0.16	3473	289	0.08
6	5046	384	0.01	3935	477	0.13
7				5071	360	0.01

example, the cut-off values for pixel intensity between binders 1 to 2 (values found in Table 3.5: 2329 to 2530) is 2429.5 and from baseline to binder 1 (values in Table 3.5: 2139 to 2329) is 2234. Therefore any peaks in the kymograph within this intensity range would be assigned as a singly bound. The mean (3.5) and cut-off values (3.6) are used for further analysis to generate the rates of transition from a cluster of bound myosins to another cluster of bound myosins and to understand the mechanism of myosin binding to the thin filament.

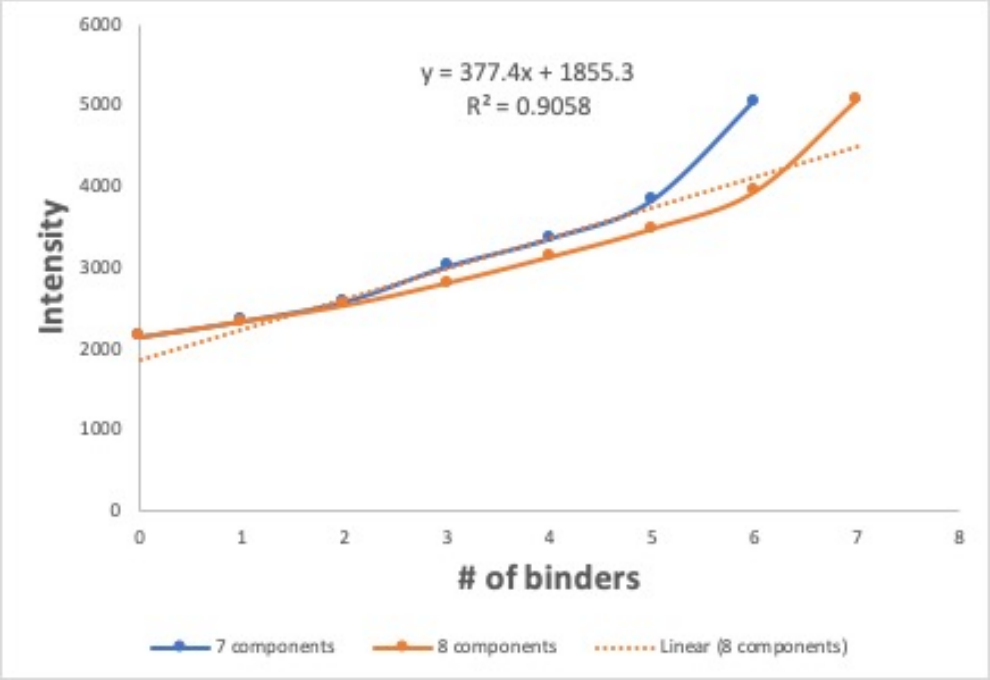


Figure 3.6: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 7 or 8 total components for linear plots of the mean intensity vs. the number of binders.

Table 3.6: Range of values for up to 7 or 8 components, which will be used to transform the raw data and then calculate the rates of transition from a component to another component

Range of	**7 components**		**8 components**	
values for	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**
0 binders	1877	2246.5	1877	2234
1 binder	2246.6	2463	2234.1	2429.5
2 binders	2463.1	2797.5	2429.6	2668.5
3 binders	2797.6	3182.5	2668.6	2966.5
4 binders	3182.6	3588	2966.6	3299.5
5 binders	3588.1	4434.5	3299.6	3704
6 binders	4434.6	5944	3704.1	4503
7 binders			4503.1	5944

Examining Appendix B, which displays the analysis for two more kymographs with conditions 5 nM of actin, it is very clear that the analysis output is not consistent.

This is because the simulation performed by the RJMCMC algorithm indicates that there are 7 or 8 components in Dataset 1 (i.e. kymograph presented in this section), in comparison to Datasets 2 and 3 where the mostly proposed number of clusters is 4. This could be due to the fact that the distribution of Dataset 1 does not overlap at all with Dataset 2 and 3. This is illustrated in Figure 3.1 and it could mean that there might have been data collection variations.

3.3.2 Dataset 1 with conditions 10 nM myosin at pCa 6

A kymograph with conditions 10 nM of myosin at pCa 6 should allow a clearer view of how the active regions collide and collapse catastrophically, according to Desai et al. [19]. Cluster formation is boosted by increased myosin and increased calcium levels. Thus, an image with 10 nM of myosin at pCa 6 was used to determine how many myosins were bound per cluster. Extra analysis results for two more image with the same conditions could be found in Appendix C.

The descriptive statistics presented in Table 3.7 shows that the minimum value of a pixel is 2767 and the mean value is 3169, with a standard deviation of 146. Figure 3.7 shows that skewness is close to zero, where 75 % of data is found in between 2767 and 3259. This histogram does not reveal clearly how many subgroups could be found in this kymograph.

Table 3.7: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
2767	3066	3154	3169	3259	3869	146

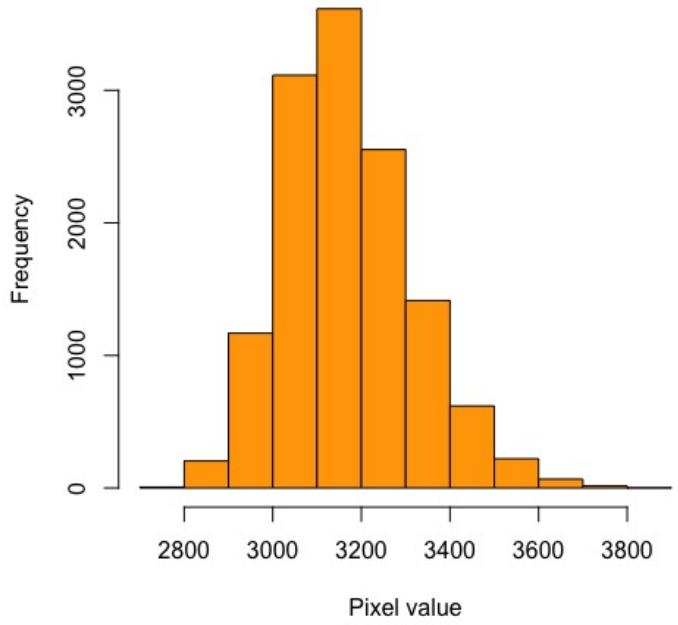


Figure 3.7: Histogram of dataset with conditions pCa6 and S1=10nM

A simulation of 30000 draws and a burn in stage of 5000 iterations is implemented using the RJMCMC algorithm. The starting values of the Markov chains have been chosen based on expert knowledge and the descriptive statistics described above. The same statistical approach as in Section 3.3.1 was followed; where four different chains conditioned on either K=1 or K=6 were generated. The starting values are presented in Table 3.8.

Table 3.8: Starting values used for the RJMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.19	0.2	0.15	0.16	0.2	0.1
Mean	3169	2900	3000	3160	3300	3450	3600
Variance	146	105	100	140	133	170	110

The quality of the simulation is shown in Figure 3.8, where the trace plots of the Markov chains are illustrated. All four chains seem to be moving around the parameter space,

jumping up and down frequently. It appears that the chains reach stationarity after approximately 5000 draws and the most sampled values are 2-3. This means that the RJMCMC algorithm reads that there exist 2-3 subgroups in this dataset. The same information is presented in Figure 3.9 by the distribution of parameter K in each Markov chain.

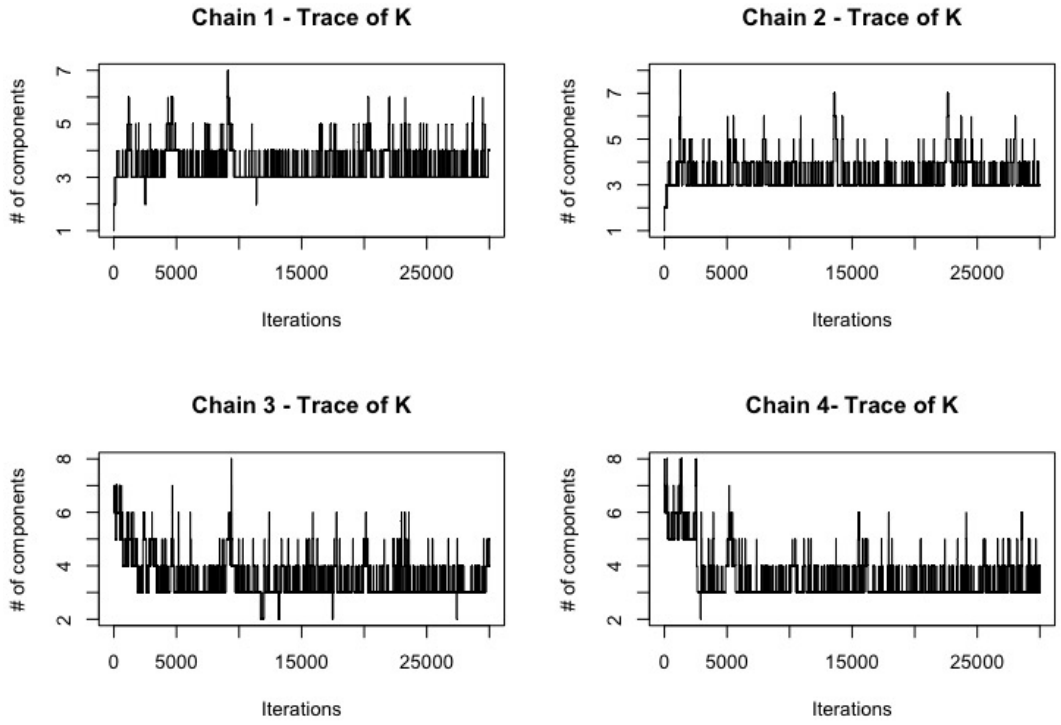


Figure 3.8: Traces for mixing over K for 10nM myosin at pCa 6, over 30000 sweeps

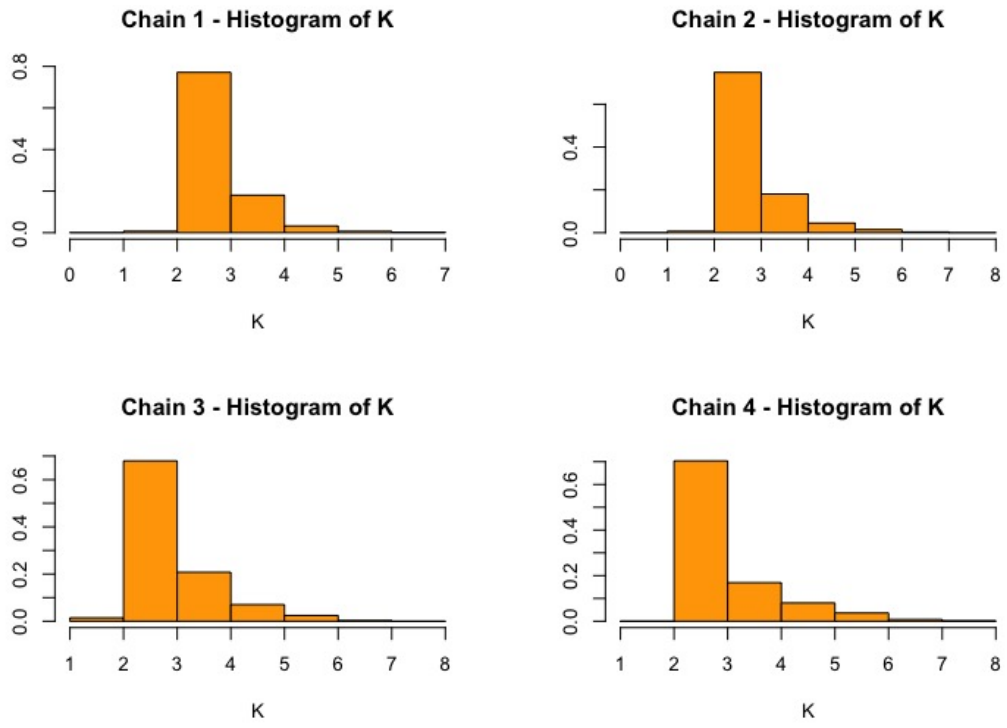


Figure 3.9: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

Further chain diagnostics were carried out via the Gelman plot and the Gelman diagnostic. Figure 3.10 illustrates that the chains reach stationarity fairly rapidly around 5000 iterations, which is our burn in stage. Convergence is also confirmed by the potential scale reduction factor, presented in Table 3.9, which is very close to 1.

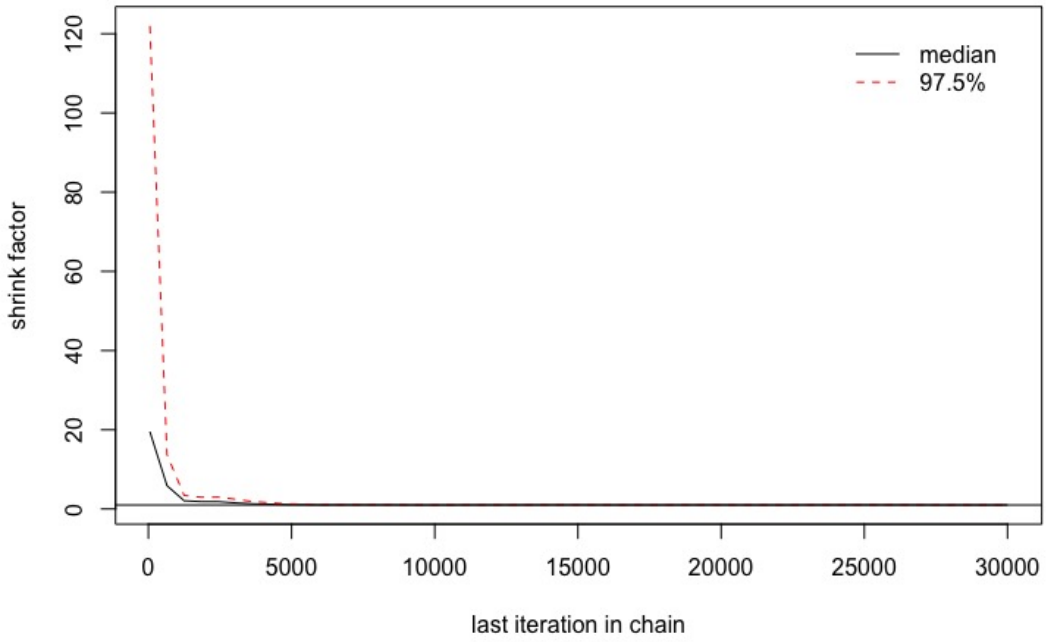


Figure 3.10: Gelman plot for all four chains with different starting points

Table 3.9: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.005338	1.009267

Given that the diagnostics have confirmed the convergence of the Markov, the next step is to decide which number of components would be best to use for further analysis. Thus, we have looked at the spread of data for each chain given by Table 3.10. It appears that all four chains have jumped to 3 and 4 components the most, meaning that the RJMCMC algorithm considers that the dataset is most likely to have 3 or 4 clusters. Prior information says that light intensity increments are linear with the number of binders. Thus, we will examine both 3 and 4 clusters to find out which of the two have a more linear relationship.

Table 3.11 displays the mean intensities, variance and weights for each myosin binder

Table 3.10: The spread of data resulting from the RJMCMC algorithm

# of components	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Chain 1	35	19979	4262	563	124	37	0
Chain 2	0	18679	4742	1096	394	89	0
Chain 3	433	18742	4622	1005	186	3	9
Chain 4	0	19310	4618	933	138	1	0

equivalent to a total of either 3 components (corresponding to background plus 2 bound myosins) or 4 components (corresponding to background plus 3 myosin binders). Then Figure 3.11 exemplifies the mean intensities of each subpopulation, which results in a linear plot with a slope corresponding to the intensity change per additional bound myosin. So, this linear regression fit to the data indicates an intensity change per myosin binder of 143.5 (based on a trendline fit for 3 components). The slopes for both 3 and 4 components are similar. So, it seems that the 4 component simulation supports a better fit for the mixture model in this dataset because the other two datasets (please see Appendix C) show very similar results, which further validates our choice. Moreover, for 4 components the predominant population is between 0 and 1 bound myosins per cluster, with a 71% chance. Clusters of 2 myosin binders also have a good chance of 20%, whereas 3 binders are very unlikely to happen, with a 9% probability.

Table 3.11: Summary results including the mean, variance and weight of having a total of either 2 or 3 myosin binders

# of	**Total of 2 binders (3 components)**			**Total of 3 binders (4 components)**		
binders	Mean = μ_2	Variance= σ_2	Weight=w_2	Mean =μ_3	Variance=σ_3	Weight=w_3
0	3097	100	0.59	3069	96	0.41
1	3252	118	0.31	3174	104	0.3
2	3384	145	0.1	3288	117	0.2
3				3396	141	0.09

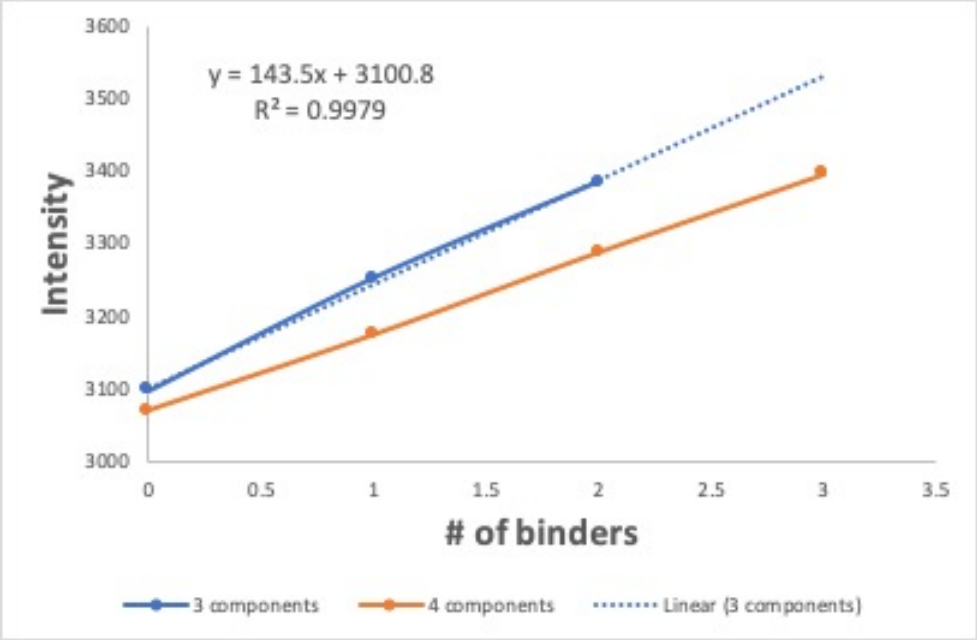


Figure 3.11: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 3 or 4 total components for linear plots of the mean intensity vs. the number of binders.

It's been decided that the fitting number of myosin binders is 4, for data collected under the same conditions. This means that there are 4 clusters in the kymographs with 10nM of myosin at pca 6 , according to the RJMCMC algorithm. As a result, the cut-off pixel intensity value of each subpopulation of bound myosins is established using the midpoint between consecutive means for the 4 components. As an example, the cut-off value between 1 to 2 bound myosins (values found in Table 3.11: 3174 to 3288) is 3231. The range of values for all 4 subpopulations is found in Table 3.12. These values are used for further analysis. So then the raw data is translated into 4 subpopulations using these ranges, in order to generate the rates of making a transition from a group of myosin binders to a different group of binders.

The other two kymographs presented in Appendix C are of the same nature as the kymograph introduced in this section, having been collected under the same experimental conditions. We have seen in Figure 3.1 that the distributions of the 3 calcium datasets are the same. This is because there is a great amount of overlap in the data

Table 3.12: Range of values for up to 3 or 4 components, which will be used to transform the raw data and then calculate the rates of transition from a component to another component

Range of	**3 components**		**4 components**	
values for	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**
0 binders	2767	3174.5	2767	3121.5
1 binder	3174.6	3318	3121.6	3231
2 binders	3318.1	3869	3231.1	3342
3 binders			3342.1	3869

points, which means that the analysis results would be expected to be very similar. Knowing that the outcomes are very much alike gives us a boost of confidence in the performance of the algorithm and in our overall analysis.

3.4 Description of transition rates and probabilities

Activation of the thin filament occurs through the initial association of myosin to open the thin filament for subsequent myosins to bind Desai et al. [19]. These bind in a collision limited process and, although stochastic, the process of activation is predictable. To determine the association rate constant we analysed our myosin binding data using RJMCMC to derive transition matrices.

A quantitative comparison between two datasets with different conditions is presented in this section. These are Dataset 1 with 5 nM actin and Dataset 1 with 5 nM myosin at pCa 6. The results were obtained using the thresholds introduced in Sections 3.3.1 and 3.3.2 to rescale the data and calculate the rates and probabilities of transition between myosin binders. Knowing the number of myosins in each cluster enabled us to calculate transition matrices, which provide information on how myosin molecules attach to or detach from the thin filament. To generate transition matrices we examined the fate of each cluster by measuring the frequency of transitions from one cluster size to another within a single frame (vertical slice of the kymograph). From these measurements we

generated matrices of probabilities and rates.

The diagonal rates for remaining in the same state, for an exponentially distributed length of time, are found in Figures 3.12 and 3.13 for the 5 nM actin dataset and 10 nM of myosin at pCa 6 respectively. From the previous section we have already established that the actin dataset has up to 7 clusters, which is why there are 8 binding states in total (this includes state 0, where there is no binding) and that the Calcium dataset accommodates up to 3 clusters, so 4 binding states.

By closely examining these matrices, we notice that all states have similar rates of remaining in the current state, apart from State 0 (S(0)) and State 6 (S(6)) in the actin dataset, and S(0) in the Calcium dataset. This means that if the binding process is found in State 0, it stays in State 0 for an exponentially distributed length of time, with a rate of 0.65 or 0.74, and then moves on to a different state. So, the rate of remaining in the current state is lowest when there are 0 bound myosins and highest in states S(1), S(2) and S(3).

Figure 3.12: Rates of remaining in the current state for 5 nM actin

	$S(0)$	$S(1)$	$S(2)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$
S(0)	0.65	0	0	0	0	0	0	0
S(1)	0	1.34	0	0	0	0	0	0
S(2)	0	0	1.21	0	0	0	0	0
S(3)	0	0	0	1.35	0	0	0	0
S(4)	0	0	0	0	1.11	0	0	0
S(5)	0	0	0	0	0	0.98	0	0
S(6)	0	0	0	0	0	0	0.61	0
S(7)	0	0	0	0	0	0	0	0.82

Figure 3.13: Rates of remaining in the current state for 10 nM of myosin at pCa 6

$$
\begin{array}{c} \\ \mathrm{S}(0) \\ \mathrm{S}(1) \\ \mathrm{S}(2) \\ \mathrm{S}(3) \end{array}
\begin{array}{c}
\begin{array}{cccc} S(0) & S(1) & S(2) & S(3) \end{array} \\
\begin{bmatrix} 0.74 & 0 & 0 & 0 \\ 0 & 1.66 & 0 & 0 \\ 0 & 0 & 2.03 & 0 \\ 0 & 0 & 0 & 1.3 \end{bmatrix}
\end{array}
$$

Moving on to the rates of making a transition to a different state, the central diagonal is zero since we are measuring movement away from the current cluster size. All numbers to the right of the diagonal are binding events and to the left detachments. These matrices with the rates of leaving the current binding state are found in Figures 3.14 and 3.15.

In Figure 3.14, which is a matrix based on the actin dataset, it can be seen that for all states there can be at most a binding of 2 myosins on top of the current number of bound myosins. For example, if the binding process is in state S(1), which means that there is already 1 bound myosin at that position, then the rate of making a transition to state S(2) (2 binders) is 0.63 and the rate of going to S(3) is 0.03. The situation is very similar with all the other states because at any state, the rate of one myosin binder attaching is a lot greater than the rate of two binders attaching. This shows that myosins become attached to the thin filament one by one and rarely two at the same time. Similarly, the rates of detachment have similar values to the rates of attachment. There are up two myosins which could detach in a single frame, but they usually detach one at a time. So the binding process is most likely to go up in 1 myosin head or down in 1.

This process of myosin heads attaching to/detaching from the thin filament is not as clear in Figure 3.15. It is because the number of states in the Calcium dataset is a lot smaller. However, it is still visible that the rates of release/binding are highest

for single myosin heads. In neither of the two matrices corresponding to the rates of transition, there is no indication of complete detachment of all molecules in a single time frame. This communicates that the process of muscle relaxation occurs in stages and there is no sudden deactivation of the thin filament.

Figure 3.14: Rates of making transitions to different states for 5 nM actin

	$S(0)$	$S(1)$	$S(2)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$
S(0)	0	0.63	0.02	0	0	0	0	0
S(1)	0.68	0	0.63	0.03	0	0	0	0
S(2)	0.02	0.7	0	0.49	0.01	0	0	0
S(3)	0	0.01	0.81	0	0.53	0	0	0
S(4)	0	0	0.01	0.6	0	0.49	0.01	0
S(5)	0	0	0	0.01	0.55	0	0.42	0
S(6)	0	0	0	0	0	0.43	0	0.18
S(7)	0	0	0	0	0	0	0.82	0

Figure 3.15: Rates of making a transition to a different state for 10 nM of myosin at pCa 6

	$S(0)$	$S(1)$	$S(2)$	$S(3)$
S(0)	0	0.65	0.09	0
S(1)	0.81	0	0.71	0.14
S(2)	0.13	1.1	0	0.79
S(3)	0.01	0.31	0.98	0

The matrices which illustrate the transition probabilities, Figures 3.16 and 3.17, are just another way of displaying the results from the analysis. All rows sum to one and there are no vertical transitions. It can be seen that near the central diagonal there is an increased probability of cluster size corresponding to the release/binding of single

myosins. Binding does not appear concerted, since there is very little probability of forming complexes in the top right of the diagram.

Figure 3.16: Transition probabilities for 5 nM actin

	$S(0)$	$S(1)$	$S(2)$	$S(3)$	$S(4)$	$S(5)$	$S(6)$	$S(7)$
S(0)	0	0.97	0.03	0	0	0	0	0
S(1)	0.5	0	0.47	0.03	0	0	0	0
S(2)	0.02	0.57	0	0.4	0.01	0	0	0
S(3)	0	0.01	0.6	0	0.39	0	0	0
S(4)	0	0	0.01	0.54	0	0.44	0.01	0
S(5)	0	0	0	0.01	0.56	0	0.44	0
S(6)	0	0	0	0	0	0.7	0	0.3
S(7)	0	0	0	0	0	0	1	0

Figure 3.17: Probabilities of making a transition for 10 nM of myosin at pCa 6

	$S(0)$	$S(1)$	$S(2)$	$S(3)$
S(0)	0	0.88	0.12	0
S(1)	0.49	0	0.43	0.08
S(2)	0.06	0.54	0	0.39
S(3)	0.01	0.24	0.75	0

We have directly observed single molecules of fluorescently tagged myosins with different conditions, interacting with the thin filament, forming clustered regions of activation. This detailed examination using the reversible jump MCMC approach reveals a high probability of myosin binding in a more classical cooperative activation. It is clear that myosin binding occurs in clusters in such partially active conditions. This confirms that myosin spreads its own binding by creating local active regions known as regulatory units. Also, these results show that the process of muscle relaxation occurs

in stages and the catastrophic collapse of active regions is not so much present.

The results from the performance of RJMCMC assessed in this chapter through the application on real datasets reinforce that myosin facilitates its own binding by developing local active regions, just as it is presented in the literature. Using the transition matrices illustrated here, it is possible to calculate the expected probability for any number of myosins binding to any existing active region.

To further explain the phenomenon of cooperative activation and the catastrophic collapse of the thin filament, a new variation of hidden Markov models is considered in the following chapter. That is because hidden Markov models are an extension to mixture models in such a way that they allow for spatial data. So the model allows for spatial information in the image to be encoded through contextual constraints of a neighbourhood structure. Then the performance of this novel MCMC algorithm is compared to the performance of the RJMCMC algorithm analysed in this current chapter.

CHAPTER 4

HIDDEN MARKOV MODEL ON TWO DIMENSIONS AND ITS APPLICATION

The hidden Markov model (HMM) can be considered a generalisation of a mixture model where the hidden states, which control the mixture components corresponding to each observation, are related through a Markov process rather than being independent of each other. They have been applied to model various types of data: discrete, continuous, univariate, multivariate, mixed and mixture data (Zucchini et al. [73]. Consequently, they have been used in numerous applications in computational molecular biology, pattern recognition and computer vision such as image sequence modelling and object tracking. A brief description of both frequentist and bayesian approaches to HMMs is provided by McLachlan and Peel [41].

Hidden Markov models have been employed in this application, because it suitably provides a formulation for an extension of a mixture model, to allow for spatial data. HMM treats the unobserved latent variable, N as a sequence, which has a behaviour

of a Markov chain. The latent variable generates the observations (number of binders) n at a time point and location, and also models transitions between different states of behaviour. Thus the observed binding and release of individual myosins was modelled using Hidden Markov models. This enabled an assumption-free model of the attachment and detachment probabilities for myosin to be determined. It is expected that myosin binds to actin stochastically and forms clusters. This highly elevated collapse probability suggests a concerted mechanism of deactivation (relaxation), and explains the ability of muscle to relax in conditions that would be expected to still permit myosin binding.

HMMs require that $y_j(t_s)$ be drawn independently from a distribution conditional on the correspondent latent state $n_j(t)$ and are defined by three properties, as stated by Ghahramani [30]. The first property is that the observation y_i is generated by an unobserved process whose latent variable is hidden. In our case, we know that the observations are driven by the attachment/detachment process, without knowing the states of the contraction. The second property states that the hidden process must satisfy the Markov property. The last property requires the latent variable to have discreete states. Thus, our model satistfies all three requirements of the HMMs.

The first part of this chapter considers a new hidden Markov model. It illustrates this model on a biology application to data introduced in Chapter 2, on the contraction of the muscle. The novelty lies in the model allowing for spatial information in the image to be encoded through contextual constraints of a neighbourhood structure based on three nearest neighbours. This neighbourhood process has been carefully chosen to capture the effects light intensity from neighbouring positions has on the average intensity of light of a pixel.

The second part of this chapter looks at the evaluation of the MCMC algorithm performance. We discuss several methods to evaluate the convergence and mixing of the MCMC algorithm. It is important to note that there is no definitive way of assessing convergence and mixing for problems that involve analytically intractable densities.

Hence a combination of methods should be employed to satisfy the researcher that convergence has been reached. After the chain diagnostics, the next step is to draw inference about the parameters.

4.1 Hidden Markov models

Hidden Markov models (HMMs) are models in which the distribution that generates an observation depends on the state of an underlying and unobserved Markov process. These models have been used for at least three decades in signal-processing applications, especially in the context of automatic speech recognition (Pietrzykowski and Sałabun [52]), but interest in their theory and application has expanded to other fields, such as:

- all kinds of recognition: face. gesture, handwriting, signature (Pietrzykowski and Sałabun [52]);
- bioinformatics: biological sequence analysis (Eddy [21]);
- finance: series of daily returns (Bulla [7]).

Other terms used to describe Hidden Markov models are state space models (SSMs) and latent process models, in which the distribution of a sequence of observations is generated by the unobserved state variables. The main aim in most applications is to reconstruct the state variable based on a given set of observations, which can be derived as a recursive form of Bayes' rule (Chen et al. [13]). SMM provides a general framework for analysing deterministic and stochastic dynamical systems that are measured or observed through a stochastic process. The most well studied SSM is the Kalman filter, which defines an optimal algorithm for inferring linear Gaussian systems; see Leroux and Puterman [39]. Linear Gaussian state space models are used extensively in all areas of control and signal processing.

A hidden Markov model, as described by Ghahramani [31], is a tool for representing probability distributions over sequences of observations. The model assumes three

defining properties. First, the observations are generated by some process, $\mathbf{N}$, whose states are hidden from the observer. We use our binding process to illustrate $\boldsymbol{N}$. Denote $\boldsymbol{n}_j = \{n_j(t), t \in [0, \tau]\}, \boldsymbol{n}_t = (n_1(t_s), \cdots, n_J(t_s))$ and $\boldsymbol{N} = \{\boldsymbol{n}_j, j = 1, \cdots, J\}$, where $n_j(t)$ is the number of myosin binders at position j and at time t. Second, the states of this hidden process satisfy the first-order [1] Markov property. Thus, HMM treats the latent variable as a sequence, which has the behaviour of a Markov chain. A third assumption is usually that the hidden state variable is discreet. Taking these Markov properties together, it means that the distribution of the latent process can be determined as follows:

$$P(\mathbf{n}_t) = P(n_1(t)) \prod_{j=2}^{J} P(n_j(t)|n_{j-1}(t)) \tag{4.1}$$

Such HMM, $\mathbf{n}_t$, is not directly observed. Instead, we observe the independent data $y_i(t)$ at time points $t_1, t_2, \cdots, t_S$, where data are recorded from time $t_1 = 0$ to time $t_S = \tau$. Denote $\boldsymbol{Y}_j = (y_j(t_1), \cdots, y_j(t_S)), \vec{Y}_s = (y_1(t_s), \cdots, y_J(t_s))^T$ and $\mathbf{Y} = (\vec{Y}_1, \cdots, \vec{Y}_S) = (\boldsymbol{Y}_1^T, \cdots, \boldsymbol{Y}_J^T)^T$. The posterior density for the HMM is given by

$$p(\mathbf{Y_t}, \mathbf{n}_t, \boldsymbol{\theta}) = \{\prod_{j}^{J} f(y_j(t)|n_j(t); \boldsymbol{\theta})\} p(\boldsymbol{n}_t|\boldsymbol{\theta}) \pi(\boldsymbol{\theta}) \tag{4.2}$$

where $\pi(\boldsymbol{\theta})$ denotes the prior density for the parameter vector $\boldsymbol{\theta}$. This factorisation of the joint probability is also known as a Bayesian network or a probabilistic graphical model in some areas such as molecular biology (Friedman et al. [23]). Bayesian networks specify conditional independence relations for a hidden Markov model. Gibbs sampling can be used to generate draws of the parameter vector, with $\boldsymbol{\theta}$ augmented by the hidden component-indicator $\boldsymbol{n}_t$. For this to be attainable, simulation from all the conditional distributions has to be simple. HMMs are increasingly being employed in applications, since it extends the mixture model by allowing for weakly dependent heterogenous

[1] A first-order Markov property is one in which the current state $\boldsymbol{n}_t$ is independent of all previous states. An n-th order Markov process is when $\boldsymbol{n}_t$ given $\boldsymbol{n}_{t-1}, \cdots, \boldsymbol{n}_{t-n}$ is independent of $\boldsymbol{n}_\tau$ for $\tau < t - n$.

phenomena (Christian P. Robert [14]).

In general, a stationary Markovian model is defined for the distribution of the hidden vectors $\boldsymbol{n}_1, \cdots, \boldsymbol{n}_J$. In one dimension, this model is a Markov chain, and in higher dimensions it is a Markov random field (Besag [5]). MRFs are a type of stochastic processes that form a natural generalisation of Markov processes in which a time index is replaced by a space index. A more detailed discussion on MRFs is to be found in the next subsection.

In order to highlight the difference between a hidden Markov chain and a hidden MRF, let $n_{\delta j}(t)$ contain the neighbours of the hidden variable, $n_j(t)$. In the case of hidden Markov chain, let $n_{\delta j}(t) = \{n_{j+1}(t), n_{j-1}(t)\}$ for $j = \{2, \cdots, J-1\}, n_{\delta 1}(t) = n_2(t), n_{\delta J}(t) = n_{J-1}(t)$; and in the case of a hidden Markov random field, let $n_{\delta j}(t)$ be dependent on a neighbourhood structure of the underlying Markov random field. The hidden component indicator vector $\boldsymbol{n}_t$ can be tackled by generating each $n_j(t)$ from the conditional distribution

$$p(n_j(t)|n_{\delta j}(t), \mathbf{Y}_t, \boldsymbol{\theta}) \propto p(n_j(t)|n_{\delta j}(t); \boldsymbol{\theta}) \tag{4.3}$$

for $j = 1, \cdots, J$. Rydén and Titterington [61] have investigated the intractability of the full conditional distributions in the case of a MRF and have provided changes to this procedure to cope with this issue.

Parameter estimation in hidden Markov models usually relies on maximum likelihood or Bayesian methods, moments methods being intractable in this setting. In particular, calculation of maximum likelihood estimates (MLE) is nontrivial. The dependency structure can only aggravate the difficulties met in mixture estimation for i.i.d. data; see Archer and Titterington [4]. Using richer hidden representations invariably leads to computational intractability in the algorithms for inferring the hidden state from observations. According to McLachlan and Peel [41] Monte Carlo methods, such as Gibbs sampling and variational methods, are two ways of handling this intractability.

4.1.1 Hidden Markov random field

Hidden Markov random field (HMRF) models (McLachlan and Peel [41]) are multidimensional random processes which generalise the notion of a 1D Markov process. These models are widely used for image segmentation, as they appear naturally in problems where a spatially constrained clustering scheme is asked for. A HMRF model is a stochastic process generated by a Markov random field (MRF) whose state sequence cannot be observed directly, but can be indirectly observed through observations..The importance of the HMRF model derives from the way in which the spatial information in an image is encoded through the mutual influences of neighboring sites/pixels.

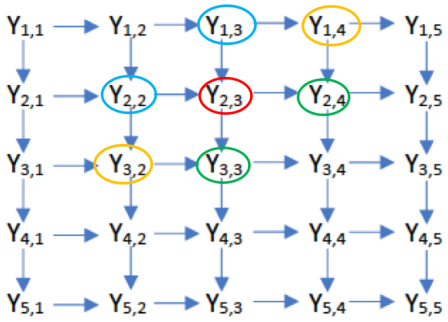
MRF, is also often defined on a discrete lattice, with a set of random variables described using an undirected graph (Chaudhary [10]) . MRF theory presents an appropriate and consistent way for modeling items that are context dependent, such as image pixels and correlated features. MRFs can be used to make inferences about the underlying image and scene structure to solve problems such as image reconstruction, image segmentation and object labelling.

Segmentation seeks to subdivide images into regions of similar attributes, playing a crucial role in image processing. A suitable neighbourhood structure has to be defined, with the constraint that being neighbours is symmetric. For example, the pixel values in an image usually depend most strongly on those in the immediate vicinity, and have only weak correlations with those further away. Hence, images must be divided into physical objects so that each region constitutes a semantically meaningful entity. Therefore, vision problems are well suited to the MRF optimization technique.

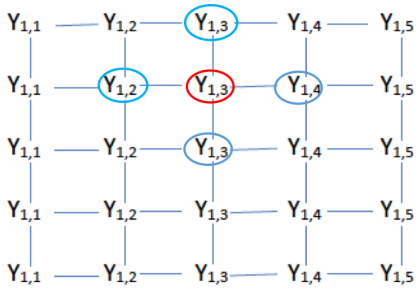
One approach is to employ a HMRF, using a simple neighbourhood structure based on 4-connected neighbours, also known as the nearest-neighbour process. A set of random variables which have some conditional independence properties is considered. The MRF contains a set of sites, which may be indexed with two values, j and t, to emphasise the 2D structure of the sites. Labels might take continuous values, but the

assumption will be to have a discrete set of labels. Every site will have a label and one can assign a site to every pixel in the image. So, in later sections, we might refer to the site $B_j(t)$ as the image intensity at position j and at time t, and its vertical neighbour $B_{j+1}(t)$ or its horizontal neighbour $B_j(t+1)$. Also, a site is not its own neighbour.

An undirected graphical model, also known as a Markov random field, can be used for image analysis or spatial statistics because it does not require the specification of edge orientations and seems much more natural for certain domains than directed graphical models (also known as Bayes nets). MRFs can be applied to a wider range of problems in which there is no natural directionality associated with variable dependency. These models have greater power than Bayes nets, but are more difficult to deal with computationally because many undirected models are intractable and require approximation techniques. A general rule of thumb is to employ Bayesian nets whenever possible, and only switch to MRFs if there is no logical way to model the problem with a directed graph. As an illustrative example, an undirected 2d lattice is shown in Figure (4.1b) and a directed acyclic graph (DAG) is presented in Figure (4.1a). In an MRF, node $Y_{2,3}$ is independent from the rest of the graph given its neighbours, also referred to as the Markov blanket of $Y_{2,3}$.



(a) Directed graphical model or Bayes net



(b) Markov random field or Markov network

Figure 4.1: (a) A 2D lattice represented as a directed graphical model. The red node $Y_{2,3}$ is independent of all other nodes (black) given its Markov blanket, which include its parents (blue), children (green) and co-parents (orange). (b) The same model represented as an unordered graphical model, also called a MRF or Markov network. The red node $Y_{2,3}$ is independent of the other black nodes given its nearest neighbours (blue nodes).

However, two dimensional MRFs are not suitable for the data described in this thesis. This is because the data has a space index and a time index, rather than two space indeces like MRFs; and an appropriate hidden Markov model which behaves like an undirected graphical model is proposed in this chapter. Hence, this naturally leads to the next section which describes Markov chains on a discrete state space and continuous time.

4.2 Continuous time and discrete state space Markov chains

The behaviour and properties of Markov chains in continuous time on a finite number of states is used to determine the rates and probabilities of transition between states, within the activation region of the thin filament. The myosin binding process is a Markov process because it is inherently a stochastic process influenced by the concentration of Calcium and myosin. This means that the future evolution of the binding

process is independent of previous steps. This section also introduces some general information on birth-death processes. These birth-death processes are continuous time Markov chains on the non-negative integers in which only jumps to adjacent states are allowed. Hence, the state transitions can be either *births* or *deaths*. This type of process will be used in a later section of this chapter to study the number of myosin binders attaching to / detaching from the thin filament.

4.2.1 Basics of continuous time Markov chains

Consider a continuous-time stochastic process A_t, where $t \in [0, \infty)$, defined on a discrete state space $\mathcal{S}$. As presented in Ross et al. [60], the process is a continuous-time Markov chain if for all $h, t \geq 0$, and nonnegative integers i, j,

$$f(h) = P(A_{t+h} = j | A_t = i, A_u = i_u \text{for } u \leq t) = P(A_{t+h} = j | A_t = i) \tag{4.4}$$

where there can be infinitely such h. So, $f(h)$ calculates the probability of being in a different state $j \in \mathcal{S}$ at time $t + h$, knowing that the chain is in state $i \in \mathcal{S}$ at time t. In other words, a continuous-time Markov chain is a stochastic process having the Markovian property that the conditional distribution of the future state at time $t + h$, given the present state at t and all past states, depends only on the present state and is independent of the past. In addition, the process is said to be time-homogenous if

$$P(A_{t+h} = j | A_t = i) = P(A_h = j | A_0 = i) = \rho_{ij}(h), \quad \text{for all } t \tag{4.5}$$

and the continuous-time Markov chain is said to have stationary or homogenous transition probabilities. This means that the probability of transitioning from a state i to a state j depends on the length of the time interval $(t + h) - t = h$. All Markov chains considered in this thesis are assumed to have stationary transition probabilities.

Holding times

Suppose that a continuous-time Markov chain enters state i at some time, say 0, and suppose that the process does not make a transition from state i to another state j during the next t time units. It is important to understand how long the chain remains in a given state, or what is the holding time in a specific state. By the Markovian property, the probability that the process remains in that state during the interval $[t, t+h]$ is just the unconditional probability that it remains in state i for at least h time units. That is, if T_i denotes the amount of time that the process stays in state i before making a transition, then

$$P(T_i > t+h | T_i > t) = P(T_i > h) \quad \forall \quad h, t \geq 0 \tag{4.6}$$

Hence, the random variable T_i satisfies the loss of memory property and is therefore exponentially distributed (since the exponential random variable is the only continuous random variable with this property).

Thus, the holding time in state i is T_i if

$$T_i = inf\{s \geq 0 : A_{t+h} \neq i | A_t = i\} \tag{4.7}$$

Consider the parameter of exponential holding time for state i as λ_i, where an useful observation could be made that

$$\mathbb{E}T_i = \frac{1}{\lambda_i};$$

which means that the higher the rate λ_i, which represents the rate of transitioning to a state $j, for j \neq i$, the smaller the expected time for the transition to occur. Even if the holding time parameter for state i is known, further information is required to figure out to which state the transition is made after leaving state i. So, the next step would be to study the transition probabilities associated with the process.

Transitions

The interest lies in small time steps, i.e. small values of $h > 0$. Surely, $f(0) = 0$. Under the assumption that f is differentiable at 0, one can obtain the derivative $f'(0)$ as follows

$$\lim_{h\to 0} \frac{P(A_{t+h} = j|A_t = i)}{h} = \lim_{h\to 0} \frac{f(h) - f(0)}{h - 0} = f'(0) = q_{ij}$$

and can be written as

$$P(A_{t+h} = j|A_t = i) = \rho_{ij}(h) = q_{ij}h + o(h). \tag{4.8}$$

This means that the chain moves to a new state $j \in \mathcal{S}$ with probability $q_{ij}h + o(h)$ in the small time interval $(t, t + h)$, given that the state was in state i at time t. Here, as $h \to 0$, the probability of two or more transitions in the time window $(t, t + h)$ is $o(h)$, as follows

$$\lim_{h\to 0} \frac{o(h)}{h} = 0$$

Similarly, no transition happens in the given time interval $(t, t + h)$ with probability

$$P(A_{t+h} = i|A_t = i) = \rho_{ii}(h) = q_{ii}h + o(h)$$

From equation (4.8) it is shown that, for every $i \in \mathcal{S}$ and for all $i \neq j$,

$$\rho_{ii}(h) = 1 - \sum_{j\in S} q_{ij}h + o(h) = 1 + q_{ii}h + o(h),$$

if $q_{ii} = -\sum_{j\in\mathcal{S}} q_{ij}$. As above $q_{ii} = f'(0)$ for $f(h) = P(A_{t+h} = i|A_t = i)$, but now $f(0) = 1$.

This information can be entered into a matrix, say $Q = (q_{ij} : i, j \in \mathcal{S})$, which contains all the information about the transitions in the given Markov chain. The properties of this matrix Q are

- all off-diagonal entries, q_{ij} are positive, where $i \neq j$;
- all diagonal entries q_{ii} are negative;
- the sum over the entries in each row is 0.

Still considering T_i as the amount of time the process remains in state i after entering state i, and also knowing that it is exponentially distributed with parameter λ_i, then for $j \neq i$

$$\rho_{ij} = P(A_{T_i} = j | A_0 = i)$$

is the probability that the process transitions to state j after leaving state i. It can be shown that the holding time T_i and the value of the new state, j, are independent random variables. This is because if the amount of time the chain stays in state i affects the transition probabilities, then the Markov property is not satisfied as it would be required to know both the current state and the amount of time taken by the chain to reach that state.

Then it is required to define

$$\lambda_{i,} = \lambda_i \rho_{ij} \tag{4.9}$$

Since T_i is exponential with parameter λ_i, then it follows that

$$P(T_i < h) = 1 - e^{-\lambda_i h} = \lambda_i h + o(h), \text{ as } h \to 0.$$

Thus, for $i \neq j$ and mild assumptions on function λ [2] ,

$$\begin{aligned} P(A_h = j | A_0 = i) &= P(T_i < h, A_{T_i} = j | A_0 = i) + o(h) \\ &= \lambda_i h \rho_{ij} + o(h) \\ &= \lambda_{i,j} h + o(h) \end{aligned} \tag{4.10}$$

as $h \to 0$. Therefore, $\lambda_{i,j}$ represents the local rate of transitioning from state i to state

[2]For example, λ_z cannot be equal to ∞ for any $z \in \mathcal{S}$

j. Also, it is worth emphasising that for $i \in \mathcal{S}$

$$\sum_{j \neq i} \lambda_{i,j} = \sum_{j \neq i} \lambda_i \rho_{ij} = \lambda_i$$

Similar to equation (4.10), the probability of remaining in state i, considering that the holding time is exponentially distributed, it follows that

$$\begin{aligned} P(A_h = i | A_0 = i) &= 1 - \sum_{j \neq i} P(A_h = j | A_0 = i) \\ &= 1 - \sum_{j \neq i} \lambda_{i,j} h + o(h) \\ &= 1 - \lambda_i h \sum_{j \neq i} \rho_{ij} + o(h) \\ &= 1 - \lambda_i h + o(h) \end{aligned} \tag{4.11}$$

It can be argued that any process satisfying equations (4.10) and (4.11) also satisfies the Markov property (4.4). This is because (4.10) and (4.11) utilise only the current state of the process and forget the entire past. This leads to a formal definition of a continuous time Markov chain that includes all the parameters of the model.

4.2.2 Birth and death processes

A birth-death process refers to a Markov process with a discrete state space $\mathcal{S}$, which can be enumerated with index $i = 0, 1, 2, 3, \cdots$, such that state transitions can occur only between neighbouring states, $i \to i + 1$ or $i \to i - 1$. The state of the process is usually thought of as representing the size of some population, and when the population increases by 1, a birth occurs, and when the process decreases by one, a death occurs.

Let ϱ_{ij} be defined by

$$\varrho_{ij} = \upsilon_i \rho_{ij}, \quad \forall i \neq j$$

Since υ_i represents the rate at which the process leaves state i and ρ_{ij} is the probability that it then goes to state j, it follows that ϱ_{ij} is the rate when in state i that the process

makes a transition into state j; hence, ϱ_{ij} is the *transition rate* from i to j.

Then, let φ_i and ω_i be given by

$$\begin{aligned} \varphi_i &= \varrho_{i,i+1} \quad \text{for} \quad i \geq 0, \\ \omega_i &= \varrho_{i,i-1} \quad \text{for} \quad i \geq 1. \end{aligned}$$

The values φ_i and ω_i are called the birth rate and death rate respectively. Since $\sum_j \varrho_{ij} = \upsilon_i$, then

$$\begin{aligned} \upsilon_i &= \varphi_i + \omega_i, \\ \rho_{i,i+1} &= \frac{\varphi_i}{\varphi_i + \omega_i} = 1 - \rho_{i,i-1}. \end{aligned}$$

Consequently, a birth and death process could be thought of by supposing that whenever there are i myosin binders in the system, the time until the next attachment (birth) occurs is exponential with rate φ_i and is independent of the time until the next detachment (death), which is exponential with rate ω_i.

4.3 The latent myosin binding process

Consider the process $\boldsymbol{n}_t = (n_1(t), \cdots n_J(t))\,(t \in [0, \tau])$. Let $\mathbb{T}_j$ be a transformation, which only changes $\boldsymbol{N} = (\boldsymbol{n}_1, \cdots, \boldsymbol{n}_J)$ at the jth position from $\boldsymbol{n}_j$ to $\boldsymbol{n}_j^{'}$. We assume that in an instantaneous time point, at most one myosin binder can bind or leave the actin thin filament. Thus, $\boldsymbol{n}_t$ could be modeled as a spatial birth-death process. We have three events which could occur: ${n_j}'(t) = n_j(t_-) + 1$ (where a new myosin molecule binds to the thin filament), ${n_j}'(t) = n_j(t_-) - 1$ (where a myosin molecule detaches from the thin filament) and ${n_j}'(t) = n_j(t_-)$ (where there are no changes). The transition rate for the spatial process could be defined as follows

$$q(\boldsymbol{n}_t, \mathbb{T}_j(\boldsymbol{n}_t)) = r(\boldsymbol{n}_t, \mathbb{T}_j(\boldsymbol{n}_t)) \times \psi_{n_j'(t)} \psi_{n_j'(t)|n_{j-1}(t_-)} \psi_{n_{j+1}(t_-)|n_j'(t)} \tag{4.12}$$

$$r(\boldsymbol{n}_t, \mathbb{T}_j(\boldsymbol{n}_t)) = \begin{cases} \lambda & n_j{}'(t) = n_j(t_-) + 1 \\ \varsigma & n_j{}'(t) = n_j(t_-) - 1 \\ 0 & \text{otherwise} \end{cases}$$

In the above model, the values of λ and μ are proportional to the baseline rate of a new myosin binder arrives and that of a myosin binder leaves, respectively, if there is no interaction between myosin binders. We assume that $\lambda < \varsigma$ which is reasonable in our study as the myosin attaching rate is smaller than the detaching rate.

In reality, the myosin binders interact with each other. One myosin molecule binds on the actin thin filament and then it increases the chances of another myosin molecule binding next to it. Therefore, we also consider the interaction effects of myosin binders. The above model treats $\boldsymbol{n}_t = (n_1(t), \cdots n_J(t))$ at a particular time point t as a one dimensional hidden Markov model, where given $n_{j-1}(t)$ and $n_{j+1}(t)$ the random variable $n_j(t)$ is independent of the number of myosin binders at all other positions. Such a hidden Markov rmodel takes into account the following two effects: [1] the value $\psi_{n_j'(t)}$ describes how the myosin binders at position j affect other myosins at the same position at time t; [2] the values $\psi_{n_j'(t)|n_{j-1}(t_-)}\psi_{n_{j+1}(t_-)|n_j(t)'}$ describes how the myosin binders at position j affect other myosins in the neighbouring positions $j-1$ and $j+1$. If there is no cooperation between different myosin minders, then $\psi_{m|n}$ will be constant for all values of m and ψ_m will also be constant. In such case the transition rate $q(\cdot,\cdot)$ is just governed by a birth-death process with rates proportional to λ and ς. Therefore, the two-dimensional model described in equation (4.12) takes into account both the birth-death process on the spatial axis and the hidden Markov model on the time axis.

Let Υ be the maximum number of binders in a position. Here we assume that $\sum_{m=0}^{\Upsilon}\psi_{m|n} = 1$, which means that $\psi_{n_{j-1}|n_j}$ can be viewed as a transition probability. It could be interpreted as that if there are n_j myosin binders at position j, the probability of having n_{j-1} binders at position $j-1$, if there is no other effect. We also assume that $\sum_{n=0}^{\Upsilon}\psi_n = 0$. This is to standardize the parameters ψ_i and λ, ς are

identifiable.

We also define $n_0 = n_{J+1} = -1$ and $\psi_{-1|m} = 1$ for $m \geq 0$. Note that this defines the transition rates for the boundary points of the thin filament tight rope, i.e. $q(\boldsymbol{N}, \mathbb{T}_1(\boldsymbol{N})) = r(\boldsymbol{N}, \mathbb{T}_1(\boldsymbol{N})) \times \psi_{n_1'}\psi_{n_2|n_1'}$ and $q(\boldsymbol{N}, \mathbb{T}_J(\boldsymbol{N})) = r(\boldsymbol{N}, \mathbb{T}_J(\boldsymbol{N})) \times \psi_{n_{J-1}|n_J'}\psi_{n_J'}$.

Proposition 1. *Let $\boldsymbol{\eta} = (\lambda, \varsigma, \boldsymbol{\psi})$ be the unknown parameters , where $\boldsymbol{\psi}$ includes all conditional probabilities $\psi_{\cdot|\cdot}$ and all parameters $\psi_{\cdot\cdot}$. Then , the stationary distribution $\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})$ will satisfy the detail balance*

$$\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})q(\boldsymbol{N}, \mathbb{T}_j(\boldsymbol{N})) = \mathbb{P}(\mathbb{T}_j(\boldsymbol{N})|\boldsymbol{\eta})q(\mathbb{T}_j(\boldsymbol{N}), \boldsymbol{N}) \tag{4.13}$$

$$\textit{with} \quad \mathbb{P}(\boldsymbol{n}_t = \boldsymbol{N}|\boldsymbol{\eta}) = c(\boldsymbol{\eta})^{-1}\psi_{n_1}\prod_{j=2}^{J}\psi_{n_j|n_{j-1}}\prod_{j=1}^{J}\left(\frac{\lambda}{\varsigma}\right)^{n_j}$$

where $c(\boldsymbol{\eta})$ represents the normalising constant.

See Appendix A for proof of Proposition 1.

The detail balance condition is an important property of Markov chains because it is stronger that than required merely for a stationary distribution; that is, there are Markov processes with stationary distributions that do not have detailed balance. A Markov chain with stationary distribution $\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})$ is said to be reversible with respect to $\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})$ or to satisfy detailed balance with respect to $\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})$ if Proposition 1 holds. The detailed balance equations say the flow of probability is balanced locally: at each edge, the amount of probability that flows across in one direction in one step, equals the amount that flows in the opposite direction.

For convenience, we define $\mathbf{N}_1 = \overrightarrow{n}(0)$ and $\mathbf{N}_{1s} = \overrightarrow{n}(t), t_{s-1} < t \leq t_s, s = 2, \cdots, S$. If we define $\Delta n_j(t) = n_j(t) - n_j(t_-)$, the increment of myosin binders at position j and at time point t, the probability distributions for $\mathbf{N}$ is given by

$$
\begin{aligned}
\mathbb{P}(\mathbf{N}|\boldsymbol{\eta}) &= \left[c^{-1}\prod_{s=2}^{S}\mathbb{P}(\mathbf{N_s}|\boldsymbol{\eta},\mathbf{N}_{s-1})\right]\times\mathbb{P}(\mathbf{N}_1|\boldsymbol{\eta}) \\
\mathbb{P}\left(\mathbf{N}_s|\boldsymbol{\eta},\mathbf{N}_{s-1}\right) &= \prod_{j=2}^{J}\left[\Delta\lambda\psi_{n_j'(t)}\psi_{n_{j-1}(t_-)|n_j'(t)}\psi_{n_{j+1}(t_-)|n_j'(t)}\right]^{(n_k(t)-n_k(t_-))} \\
&\quad \left[\Delta\varsigma\psi_{n_j'(t)}\psi_{n_{j-1}(t_-)|n_j'(t)}\psi_{n_{j+1}(t_-)|n_j'(t)}\right]^{-(n_k(t)-n_k(t_-))} \\
&\quad \exp\left(-\Delta\lambda\psi_{n_j'(t)}\psi_{n_{j-1}(t_-)|n_j'(t)}\psi_{n_{j+1}(t_-)|n_j'(t)}\right) \\
&\quad \exp\left(-\Delta\varsigma\psi_{n_j'(t)}\psi_{n_{j-1}(t_-)|n_j'(t)}\psi_{n_{j+1}(t_-)|n_j'(t)}\right)
\end{aligned}
$$

where $n_k(t) - n_k(t_-) = \Delta n_k(t)$, which represents the latent variable.

4.4 Image intensity model given the latent process

The light emitted by the fluorophore on a myosin molecule b, located in pixel/position k of the thin filament at time t, will contribute to the image intensity of pixel k with a random value. We assume that the random values are independent and identically distributed for different values of k, b, t, following a normal distribution $\mathcal{N}(\mu, \sigma^2)$. Also, given that we do not distinguish between myosin molecules and fluorophores, we assume that the fluorophores are stable during the data collection and that the positions on the thin filament, where there exist myosin binders, do not have any effects on the light emitted by the fluorophores.

One would argue that larger image intensity at a pixel, would result in a greater number of myosin binders. However, this is not entirely correct. This is because the light upon light excitation emitted by the fluorophore attached on a myosin molecule can have effects within the neighbouring 379.2 nanometers (three pixel distance). Thus, the intensity of light in a pixel could be impacted by binders in neighbouring positions. For this particular reason the raw data must be modelled such that we arrive as close as possible to the real light intensity values.

Suppose that the thin filament is made up of J positions at each time point t, where each position corresponds to a pixel in the data image. If a fluorescently labelled myosin binder, b, is bound in pixel k at time t, then the image intensity in position j supplied by b is assumed to be equal to $d_{j,k}Z_{kb}(t)$, where $d_{j,k}$ is given as follows:

$$d_{j,k} = \begin{cases} 1 & j-k=0 \\ 0 & |j-k|>3 \\ \exp(-\delta|j-k|) & 1 \leq |j-k| \leq 3 \end{cases}$$

This means that the fluorophore of a myosin binder at position k will discharge the highest image intensity at the binding position and its effect on neighbouring positions will diminish via a factor $d_{j,k}$. As shown above, when $|j-k|>3$ we have $d_{j,k}=0$ because the impact is 0 for positions with distances greater than 3 pixels from the binding position. Similarly, when the distance between a position j and the fluorophore of a binder at k is equal to 0, the decay factor is equal to 1. This is because the position corresponds to the same pixel in such cases and its image intensity value must be true. The third scenario would occur when a position is within three pixels from a myosin binder. In such cases the decay factor,$d_{j,k}$, would be exponentially distributed.

4.4.1 The likelihood

Let $n_j(t)$ be the number of myosin binders at position j at time t. We assume that the noise intensity ϵ_{jt}, for position j at time t, follows the normal distribution $\mathcal{N}(\mu_0, \sigma_0^2)$. Denote $\boldsymbol{n}_j = \{n_j(t), t \in [0,\tau]\}$ and $\boldsymbol{N} = \{\boldsymbol{n}_j, j = 1, \cdots, J\}$. Let $B_{j,k}(t)$ be the image intensity on position j at time t, contributed by the fluorophores on the $n_k(t)$ binders in pixel k. In this study, we consider a simple model for the light emitted by the fluorophores, i.e.

$$B_{j,k}(t) = d_{k,j}Z_{n_k(t)}(t)$$

We define

$$B_j(t) = \sum_{k=1}^{J} B_{j,k}(t) + \epsilon_{jt}$$

where $B_j(t)$ is the intensity at position j and at time t. Then given $n_k(t)$ $(k = 1, \cdots, J)$ we have

$$B_j(t) \sim \mathcal{N}(\mu_{n_j(t)}, \sigma^2_{n_j(t)}) \quad \text{and} \quad \epsilon_{jt} \sim \mathcal{N}(\mu_0, \sigma_0^2)$$

with

$$\mu_{j(t)} = \sum_{k=1}^{J} E\left[B_{j,k}(t)\right] = \sum_{k=1}^{J} \mu_{n_k(t)} d_{k,j} + \mu_0$$

$$\sigma^2_{j(t)} = \sum_{k=1}^{J} \mathrm{Var}\left[B_{j,k}(t)\right] = \sum_{k=1}^{J} \sigma^2_{n_k(t)} d^2_{k,j} + \sigma_0^2.$$

Suppose that the data $Y_i(t)$ is observed at time points $t_1, t_2, \cdots, t_S$ and data are recorded from time $t_1 = 0$ to time $t_S = \tau$. Denote $\boldsymbol{Y}_j = (Y_j(t_1), \cdots, Y_j(t_S)), \vec{Y}_s = (Y_1(t_s), \cdots, Y_J(t_s))^T$ and $\mathbf{Y} = (\vec{Y}_1, \cdots, \vec{Y}_S) = (\boldsymbol{Y}_1^T, \cdots, \boldsymbol{Y}_J^T)^T$. Then given the latent process $\boldsymbol{N}$ and parameter $\boldsymbol{\theta} := (\mu, \sigma^2, \mu_0, \sigma_0^2)$, the probability density for the data $\mathbf{Y}$ is given by

$$\begin{aligned} & f(\mathbf{Y}|\boldsymbol{N}, \boldsymbol{\theta}) \qquad (4.14) \\ = & \prod_{s=1}^{S} \prod_{j=1}^{J} \left\{ \left[\sum_{k=1}^{J} \sigma^2_{n_k(t)} d^2_{k,j} + \sigma_0^2 \right]^{-1/2} \exp\left[-\frac{\left(y_j(t_s) - \sum_{k=1}^{J} \mu_{n_k(t)} d_{k,j} - \mu_0\right)^2}{2\sum_{k=1}^{J} \sigma^2_{n_k(t)} d^2_{k,j} + \sigma_0^2} \right] \right\} \end{aligned}$$

where μ_0 and σ_0^2 are the mean value and variance of the noise intensity. Then $\mu_{n_k(t)}$ and $\sigma^2_{n_k(t)}$ are the mean value and variance of $n_k(t)$ binders in pixel k. And $d_{k,j}$ represents the decay factor at position j.

4.4.2 The full posterior distribution

The full posterior distribution for $\boldsymbol{\theta}$, η and $\mathbf{N}$ is given by

$$
\begin{aligned}
\pi(\boldsymbol{\theta},\boldsymbol{\eta},\mathbf{N}|\mathbf{Y}) &\propto f(\mathbf{Y}|\mathbf{N},\boldsymbol{\theta}) \times \mathbb{P}(\mathbf{N}|\boldsymbol{\eta}) \times \pi_0(\boldsymbol{\theta},\boldsymbol{\eta}) \\
&\propto \prod_{s=2}^{S}\prod_{j=1}^{J}\Bigg\{\Bigg[\sum_{k=1}^{J}\sigma^2_{n_k(t)}d^2_{k,j}+\sigma_0^2\Bigg]^{-1/2} \\
&\exp\Bigg[-\frac{(y_j(t_s)-\sum_{k=1}^{J}\mu_{n_k(t)}d_{k,j}-\mu_0)^2}{2\sum_{k=1}^{J}\sigma^2_{n_k(t)}d^2_{k,j}+\sigma_0^2}\Bigg]\Bigg\} \\
&\prod_{t\in(t_{s-1},t_s]}\prod_{j=1}^{J}\Bigg\{\Big[\lambda\psi_{n'_j(t)}\psi_{n_{j-1}(t_-)|n'_j(t)}\psi_{n_{j+1}(t_-)|n'_j(t)}\Big]^{I[\Delta n_j(t)=1]} \\
&\Big[\varsigma\psi_{n'_j(t)}\psi_{n_{j-1}(t_-)|n'_j(t)}\psi_{n_{j+1}(t_-)|n'_j(t)}\Big]^{I[\Delta n_j(t)=-1]}\Bigg\} \\
&\exp\Bigg(-\int_{(t_{s-1},t_s]}\Bigg(\sum_{j=1}^{J}\lambda\psi_{n_j(t_-)+1}\psi_{n_{j-1}(t_-)|n_j(t_-)+1}\psi_{n_{j+1}(t_-)|n_j(t_-)+1} \\
&+\sum_{j=1}^{J}\varsigma\psi_{n_j(t_-)-1}\psi_{n_{j-1}(t_-)|n_j(t_-)-1}\psi_{n_{j+1}(t_-)|n_j(t_-)-1}\Bigg)dt\Bigg)\times\mathbb{P}(\mathbf{N}_1|\boldsymbol{\eta})
\end{aligned}
\tag{4.15}
$$

where π_0 is the prior distribution, $\boldsymbol{\theta} := (\boldsymbol{\mu}, \sigma^2, \mu_0, \sigma_0^2)$ and $\boldsymbol{\eta} = (\lambda, \varsigma, \zeta, \boldsymbol{\psi})$.

The full conditional distribution for $\boldsymbol{\psi}$ is as follows

$$\pi(\boldsymbol{\psi}|\boldsymbol{\theta},\varsigma,\lambda,\zeta,\mathbf{N}) \propto \mathbb{P}(\mathbf{N}|\boldsymbol{\eta}) \times \pi_0(\boldsymbol{\theta},\lambda,\varsigma,\zeta)$$

where $\pi_0(\lambda,\varsigma,\zeta) \propto 1$. So we have

$$\pi(\boldsymbol{\psi}|\boldsymbol{\theta},\varsigma,\lambda,\zeta,\mathbf{N}) \propto \mathbb{P}(\mathbf{N}|\boldsymbol{\eta}) \times \pi_0(\boldsymbol{\theta})$$

4.5 Numerical analysis

To investigate the behaviour and performance of the hidden Markov model proposed in this chapter, we applied this model to Dataset 1 using MCMC simulations. This dataset corresponds to the dataset used for analysis in Section 3.3.2 to estimate parameters using RJMCMC algorithm.

Dataset 1 has a Calcium concentration of 6nM and myosin 10nM. We have used an extract of this dataset, which represents the first 15 rows of its image and the first 50 columns. The entire dataset was not included due to heavy computational issues.

When determining the posterior distribution, Bayesian analysis incorporates prior distributions of the parameters of interest. The prior distributions are usually derived from prior knowledge about the statistics of the parameters. So the priors we have chosen for simulations are informative and different for each unknown parameter, in order to allow the incorporation of available expert information. The priors on λ, ς and ζ are selected as Gamma

$$\lambda \sim \Gamma(5.5, 2.3) \qquad \varsigma \sim \Gamma(5.7, 2.5) \qquad \zeta \sim \Gamma(6, 2) \tag{4.16}$$

because Gamma represents the conjugate prior of a normal distribution.

In specifying the priors, the combinations (μ_j, σ_j) for $j = 1, 2, \cdots, K$, we have μ_j and σ_j drawn independently from a normal and a Gamma distribution respectively

$$\mu_j \sim \mathcal{N}(\xi_j, \kappa_j) \quad \text{and} \quad \sigma_j \sim \Gamma(\alpha_j, \beta_j) \tag{4.17}$$

where ξ_j and κ_j are set as the previous step of the Metropolis-Hastings algorithm corresponding to mean and standard deviation. So the priors change according to the development of the Markov chains. Then, we also have α_j and β_j which are chosen in

a different manner. These are fixed such that

$$\boldsymbol{\alpha} = (100, 120, 120, 130, 100, 100, 110) \qquad \boldsymbol{\beta} = (3, 3, 2.5, 2.5, 2.7, 2.2, 2.5)$$

As proposal densities we have chosen truncated normal densities, $q(\cdot)$, for $\boldsymbol{\theta} := (\boldsymbol{\mu}, \boldsymbol{\sigma})$ and $\boldsymbol{\eta} := (\lambda, \varsigma, \zeta)$. This is because the original population is normally distributed with positive values; hence the densities have been restricted to sampling values which lie above 0. We have also restricted σ_1 to sample values below 750 because the variance parameters should not be too high. However, we have allowed σ_1 to be a lot higher than the variance for the other components because the first component incorporates noise.

In the construction of latent N, a Poisson proposal density was chosen for its simulation and was seen as fit for this proposed hidden Markov model. This is because this discrete distribution models the number of events occurring randomly in a given time interval, with its shape parameter (or mean rate) indicating the average number of occurrences in the given period. Hence this fits with determining the number of binders in each pixel. The mean rate used for this distribution is a changing rate so that each pixel (j, t) in the latent variable is generated from a position in a matrix matching to (j, t). For this corresponding matrix we use the transformed dataset from the reversible jump MCMC estimated in Section 3.3.2. As it has also been presented in Section 3.3.2, the range of values for all 4 subpopulations estimated using RJMCMC were used to translate the raw data into the transformed dataset. This means that the intensity values provided by the raw data were matched to the range of values for each subpopulation, which translates into a transformed dataset providing the estimated number of bound myosins at each position (j, t).

As MCMC practitioners we have to address two critical questions: where to start and when to stop the simulation. Although a great amount of research has gone into establishing convergence criteria and stopping rules with sound theoretical foundation,

in practice, MCMC users often decide convergence by applying empirical diagnostic tools. There are no general rules for how long one needs to run the chain, but generally it is a good idea to start the chain from different starting points and make sure they converge to the same density plots. Secondly, one can decide in advance on the number of samples one wants from the underlying distribution, and run the chain until the effective sample size reaches that number.

We have decided for the simulation study to be based on a Markov chain with 10,000 iterations. The first 1000 samples of each chain are discarded as burn-in samples when estimating the parameters. Furthermore, to achieve a better result the thinning method has also been applied, with a thinning factor $n = 2$. In order to check whether convergence has been achieved in our chains, we will assess convergence through chain diagnostics and acceptance rates.

4.5.1 Simulated binding process with 5 components

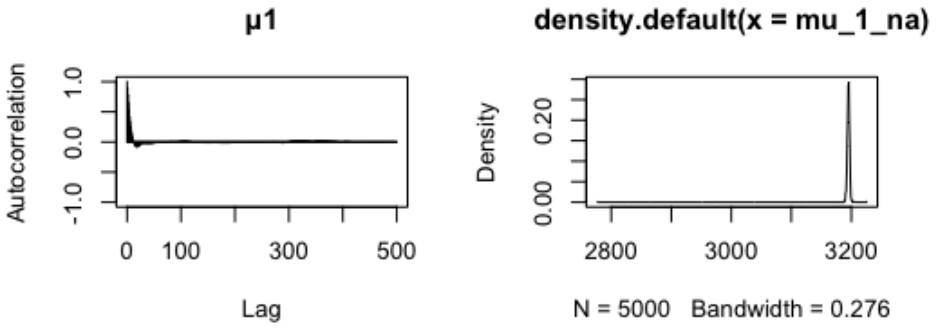
For the algorithm to be efficient, it is necessary to carefully choose the shape of the proposal density to that of the posterior density, and the scaling parameter of the proposal distribution (the variance of the truncated normal distributions in our case). Large values for the variance will generally favour jumps that are far away from the current state of the chain, often in regions where the target density is low. Consequently, proposed moves will usually be rejected and the chain will linger on some states for long periods of time. On the other hand, small values for variance will generate short jumps, resulting in a poor exploration of the state space. Thus, the jump size is determined by the variance or width of the proposal density. In this scenario, the global acceptance probability is about 47.8%, which is in the range suggested by the literature (30-50%).

In practice, there might be a concern on the quality of realisations generated by the Markov chains. Although there are algorithms developed to generate exact samples from the posterior based on MCMC, such as *Coupling from the past* [54], such ex-

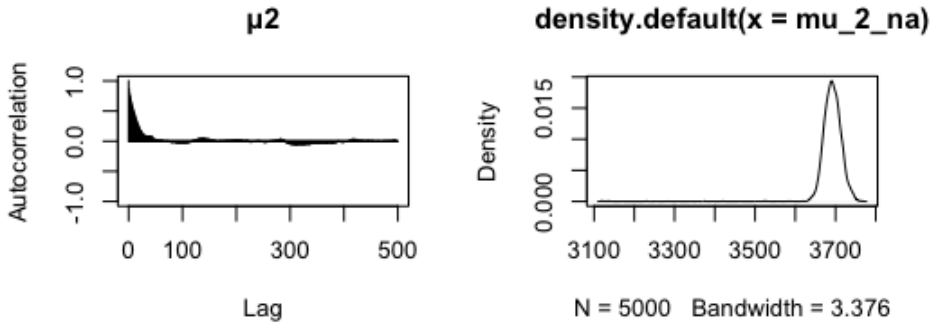
act methods only work for the simple toy examples. For very complicated Bayesian computational problems, there are no hard rules available to justify the convergence of MCMC chains and users usually make their decisions based on rules of thumb. Here, we consider using simple autocorrelation plots, trace plots and posterior density plots, to justify the burn in stage of the Markov chains. We also consider using the Markov chains from different starting points and use simulated realisations from parallel independent chains. Also, after seeing the outcome for μ_5 we have tried using different starting points, but the results were similar to the ones in Figure 4.3b. Given that the density found is multimodal, we believe that this last component includes information for other small modes μ_6, μ_7 ($K = 7$). But due to practical reasons and for the purpose of comparing the simulations in this chapter with the RJMCMC simulations, we chose to do our analysis with $K = 5$ components.

From the trace plots to be found in Appendix C.3, we see that the MCMC sampler seems to mix reasonably well for the mean parameters. The following Figures 4.2 and 4.3 illustrate the autocorrelations and density plots for the mean parameters of our model.

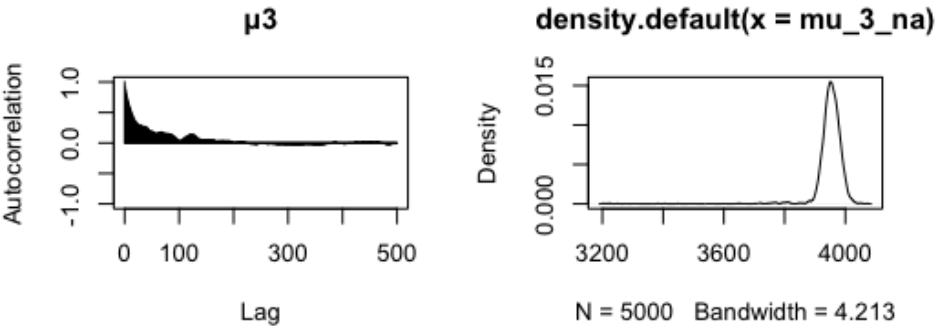
Some of the mean parameters suffer somewhat of higher auto-correlation at the beginning of the chain, which leads to a slow mixing of the chain. The worst parameter results are the autocorrelation plots for μ_5, but all other parameters μ have very good mixing properties (see in Figures 4.2 and 4.3). The autocorrelations for the other mean parameters show large positive correlations for several lags which quickly decay towards zero. Hence the iterations appear to not be linearly related to their past, for most mean parameters. Also, looking at the density plots it seems that apart from μ_5, which shows a multimodal distribution representing the possibility of some extra components may exist, all other component means have uni-mode posterior densities.



(a) Autocorrelation and density plot for μ_1 with mean parameter value $\mu_1 = 3195$

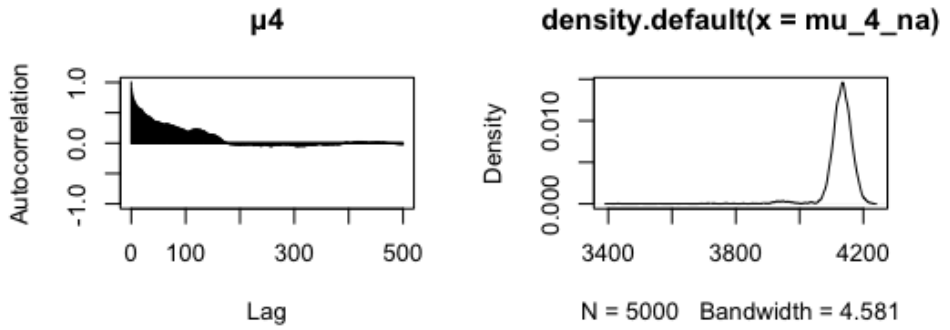


(b) Autocorrelation and density plot for μ_2 with mean parameter value $\mu_2 = 3693$

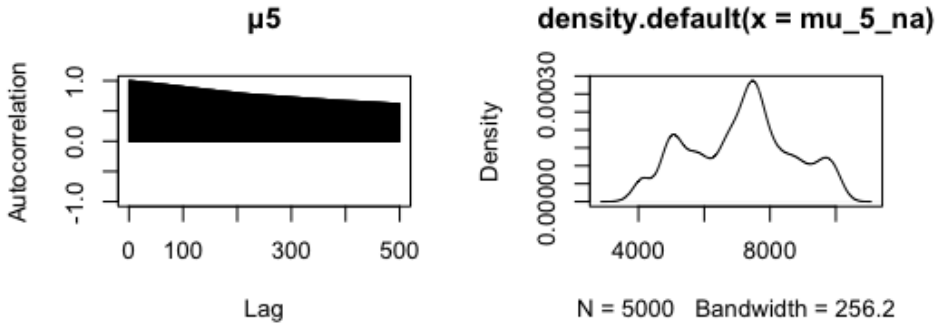


(c) Autocorrelation and density plot for μ_3 with mean parameter value $\mu_3 = 3956$

Figure 4.2



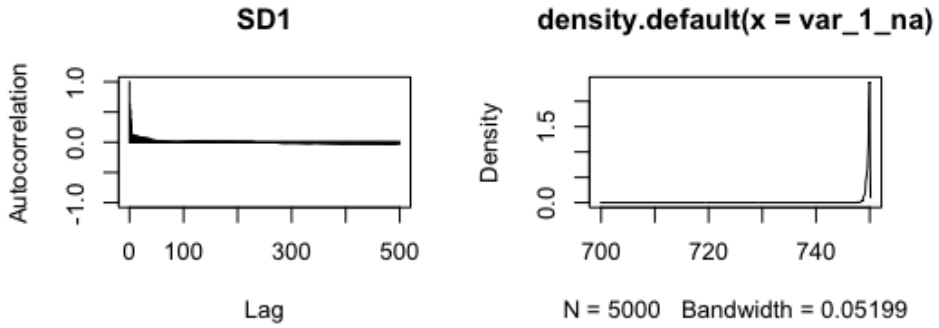
(a) Autocorrelation and density plot for μ_4 with mean parameter value $\mu_4 = 4135$



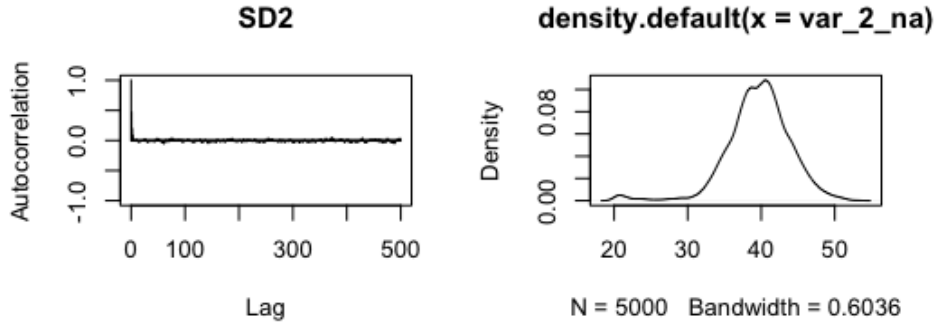
(b) Autocorrelation and density plot for μ_5 with mean parameter value $\mu_5 = 7398$

Figure 4.3

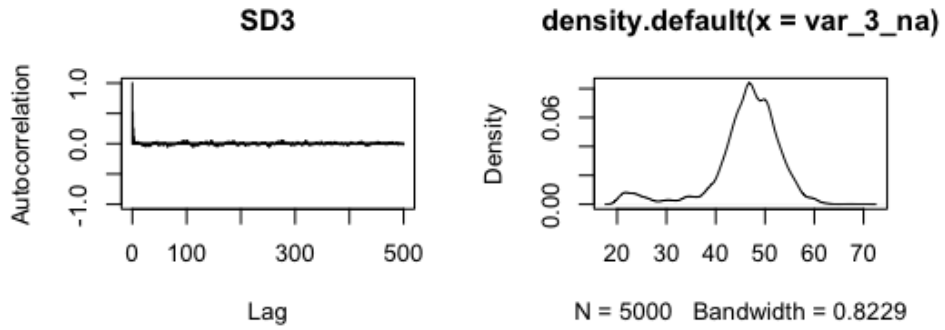
The following Figures 4.4 and 4.5 illustrate the chain diagnostics for the variance parameters $\sigma_1, \sigma_2, \sigma_3, \sigma_4$, and σ_5. A good sign of convergence for these variance parameters is that the autocorrelation plots show very weak correlation between the iterations of the Markov chain. This argument is further strengthened by the trace plots of σs in Appendix C.3. Note that we put a subjective prior distribution with a constraint being no larger than 750 for σ_1, the variance for noise component, to overcome possible overfitting problem.



(a) Autocorrelation and density plot for σ_1 with mean parameter value $\sigma_1 = 749.7$

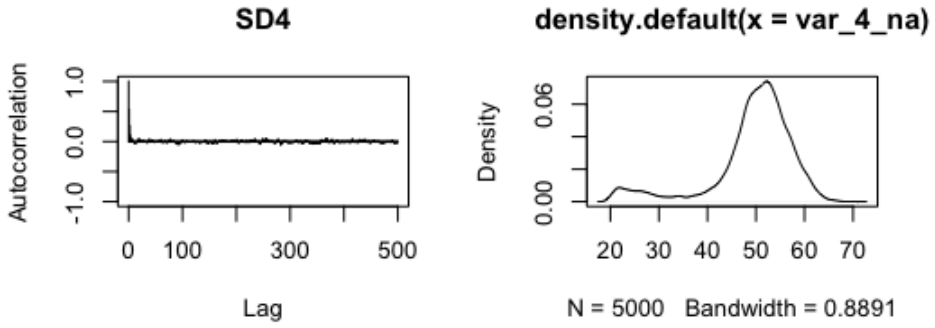


(b) Autocorrelation and density plot for σ_2 with mean parameter value $\sigma_2 = 39.6$

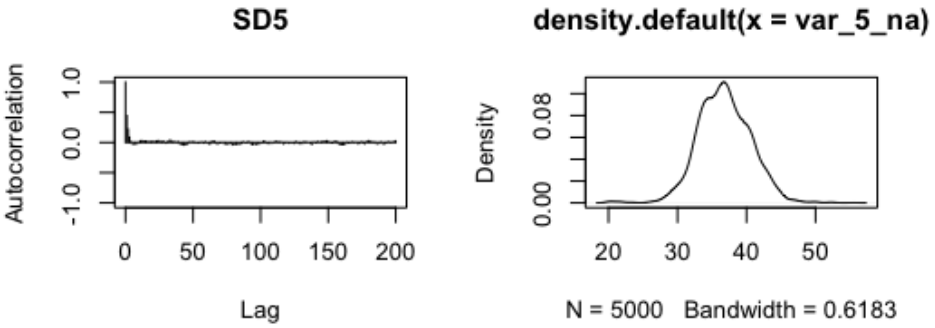


(c) Autocorrelation and density plot for σ_3 with mean parameter value $\sigma_3 = 46.3$

Figure 4.4



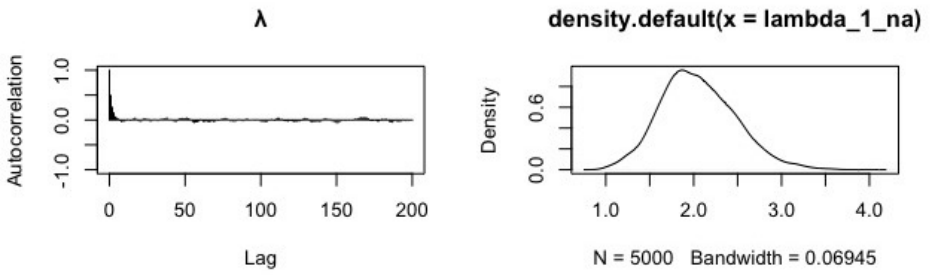
(a) Autocorrelation and density plot for σ_4 with mean parameter value $\sigma_4 = 49.5$



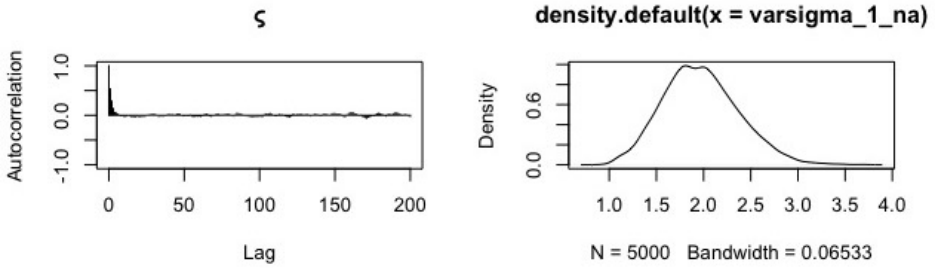
(b) Autocorrelation and density plot for σ_5 with mean parameter value $\sigma_5 = 36.8$

Figure 4.5

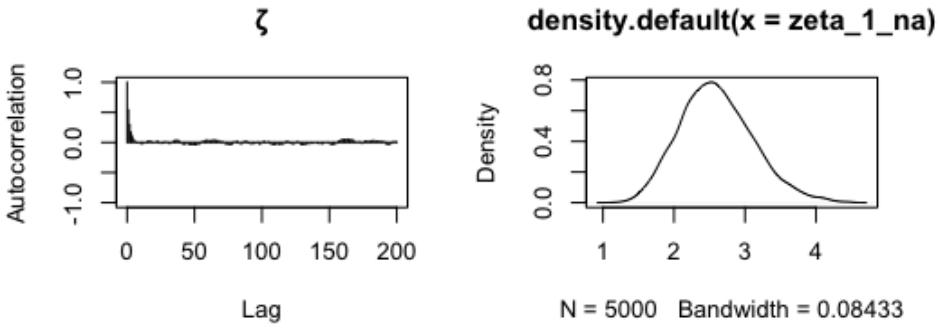
Autocorrelations and density plots for λ, ς and ζ are displayed in Figure 4.6. These plots reveal very low correlation in the Markov chains for these three parameters and display the posterior distributions. So this is a very good sign of chain convergence. Also, Appendix 4.5 presents very healthy trace plots of the Markov chains.



(a) Autocorrelation and density plot for λ with mean parameter value $\lambda = 2$



(b) Autocorrelation and density plot for ς with mean parameter value $\varsigma = 1.98$



(c) Autocorrelation and density plot for ζ with mean parameter value $\zeta = 2.6$

Figure 4.6

Credible intervals are an important concept in Bayesian statistics. Its core purpose is to describe and summarise the uncertainty related to the unknown parameters you are trying to estimate. So posterior medians and the 95% credible intervals are presented in Table 4.1. The interpretation of the Bayesian 95% credible interval is the following: there is a 95% probability that the true (unknown) parameter estimate would lie within the interval, given the evidence provided by the observed data (Hespanhol et al. [35]).

Parameter name	Estimated posterior median	95% Credible Interval
μ_1	3202	(3201, 3205)
μ_2	3570	(3557, 3602)
μ_3	3971	(3956, 4009)
μ_4	4288	(4265, 4342)
μ_5	6901	(5808, 8003)
σ_1	750	(750, 750)
σ_2	39.8	(37.2, 46.2)
σ_3	47.4	(44, 55)
σ_4	50.9	(46.8, 59.3)
σ_5	36.8	(34.2, 43.1)
λ	2.01	(1.74, 2.83)
ς	1.95	(1.7, 2.66)
ζ	2.56	(2.24, 3.5)

Table 4.1: Estimated posterior means and 95% credible intervals for each model parameter

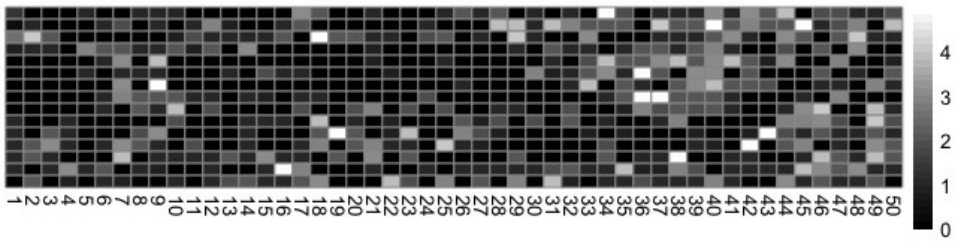
Inference results

By using direct imaging of single myosin S1 (simply termed myosin here) molecules interacting with suspended thin filaments, we are able to determine when and where the thin filaments are active. It is clear from the literature, that myosin binding occurs in clusters in such partially active conditions. To successfully extract useful information from these fluorescence images, which we have used as datasets, it has required us to compute meaningful image analysis. One way of approaching this analysis is to see it like a puzzle. The aim of our hidden Markov model in this chapter was to recreate the intensity kymograph (see Figure 4.7c), which shows the myosin binding process and to provide a quantitative description of the thin filament activation process. In the end, we have extracted some kind of quantitative measurements that are justified by the nature of the experiment and the facts of image formation. Each pixel contains all of the fluorescence intensity information for a single time point in the movie along the length of a thin filament. The pixel intensities have different scales in Figures 4.7a and 4.7b because of the models used for simulation have different levels of performance. These images illustrate both the simulated activation and deactivation of the thin filament,

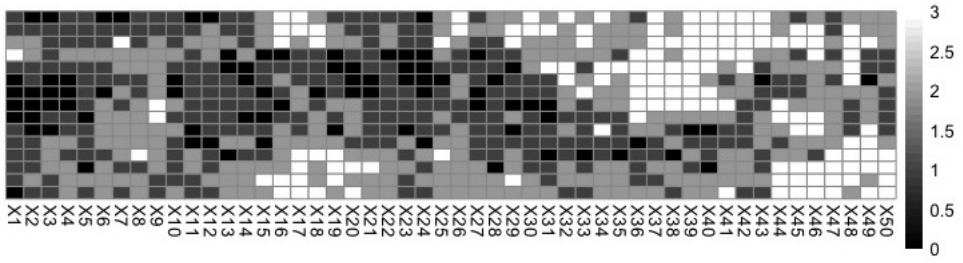
based on the raw data presented in Figure 4.7c.

The raw extract of the dataset has a dimension of 15 x 50, so the estimated latent variable, via reversible jump MCMC and also via the hidden Markov model, has the same dimension. This latent variable displayed as a kymograph, is the process of myosin binding and it is colour-coded to represent the number of binders. These are the estimates of the number of myosins bound to the thin filament for each location and each time point for Dataset 1. The scale is different for each of the three kymographs, but we are mostly interested in the formation of clusters, which are seen to move along the thin filament as myosins bind and then leave. At the lowest values on the scale, no binding occurs and the colour of the pixel is black. As fluorescence intensity increases, the thin filament partially activates, permitting the binding of myosins. Thus, the brighter the pixel, the larger the number of binders to be found in that position and at that point in time.

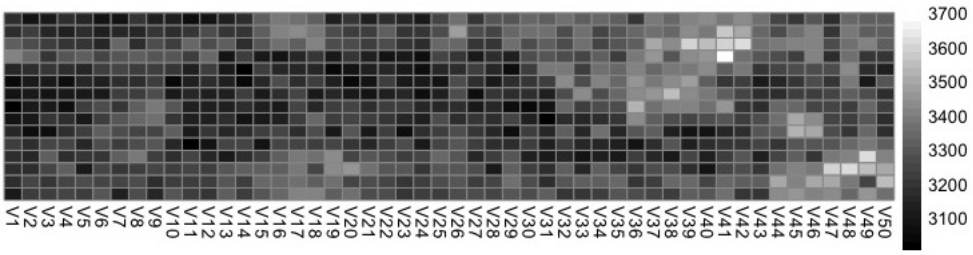
The heat maps in Figure 4.7 provide clear and accessible representations of the dynamic binding process, advancing our understanding of the data at hand. From this analysis, it is clear that myosin binding occurs in clusters even in partially active conditions and that myosin propagates its own binding. We are also able to show that active regions fuse to form larger units capable of moving along the thin filament. The top kymograph, which is the binding process simulated via our HMM where $K = 5$, seems to resemble the most the raw data, presented in the bottom kymograph, indicating good performance of the algorithm. Thus, the middle kymograph which displays the binding process simulated via RJMCMC with $K = 4$, did not perform as well as the top kymograph. A very important aspect of normal muscle function is its ability to relax. And in these results obtained using the proposed model we can observe the catastrophic collapse of the active regions. This suggests that the thin filament has the potential to be turned fully on or off in a binary fashion.



(a) The myosin binding process simulated using the hidden Markov model



(b) The myosin binding process estimated via reversible jump MCMC



(c) The raw image of the real myosin binding process

Figure 4.7: Mean intensity values of fluorescence are exemplified in each of the three kymographs. The variation in light intensity gives clues about how the myosin binding phenomenon is clustered or varies over space. Our Bayesian statistical simulations have produced the results in (a) and (b), which we have used to colour each pixel on the array white or black depending on the saturation and produce these heat maps. The raw dataset is displayed in (c) to compare it with our simulated data. The legends are the same for (a) and (b) because those numbers represent the number of binders captured in the simulations. The difference is that (a) captures up to 4 binders in its simulation and (b) captures up to 3 binders. In (c) the legend is different because it has a different scale as it represents the real values in the dataset. The x-axis is the same for all three kymographs, which represents time of exposure and the y-axis represents the positions on the single thin actin filament.

When looking at different regions in these kymographs, we can see that the RJMCMC algorithm overfits the data and tends to show a lot more binding than there is in many of the clusters. For example, looking at the cluster starting from column 34, row 1 and ending in column 46, row 9, the light intensity is very high for this whole region simulated by RJMCMC. And for this same region, the raw data shows that there is some clustering where myosin is rapidly binding and releasing until column 39 and row 3 where there is a brighter patch which lasts for 4 pixels. Our simulated results mirror the raw data a lot more than the RJMCMC results for the same region. We can see that the shape and the light intensity is very similar to that of the raw data, even if there is a small amount of data overfitting in a couple of pixels.

Similarly, the small cluster in the bottom right corner of the bottom kymograph is picked up in both simulations. The RJMCMC shows there is a lot of myosin binding in that area because of the maximum light intensity, whereas the hidden Markov model displays some less bright pixels which are closer to reality. Furthermore, another cluster area from column 16 and row 12 to column 20 and row 15, is found in all three kymographs; again the RJMCMC algorithm seems to fully overfit the number of binders to be found in this region and the HMM reproduces some of the light intensity found in the raw image. Thus, based on all of the above arguments, we are confident to argue that the hidden Markov model proposed in this chapter has outperformed the reversible jump MCMC.

The aim of our hidden Markov model in this chapter was to recreate the intensity kymograph, which shows the myosin binding process and to provide a quantitative description of the thin filament activation process. In the end, we extracted some kind of quantitative measurements that are justified by the nature of the experiment. The algorithm performance in simulating the myosin binding process is compared to the raw data and to the performance of the reversible jump Markov chain Monte Carlo. The purpose of such mixture analysis of our novel variation of HMMs and RJMCMC is inference about the unknown number of K components, component parameters and

the proportion of each cluster. The analysis reveals that our proposed model has an improved performance when compared to RJMCMC because it gives a better simulation of the number of myosins found in each pixel. To successfully extract useful information from the fluorescence images/kymographs, which we have used as datasets, it has required us to compute meaningful image analysis.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTION

This research work has developed a hidden Markov model to investigate the behaviour of myosin molecules binding to regulated thin filaments in a single molecule assay. This new method provides a formulation for an extension of a mixture model to allow for spatial data and is able to determine the number of binders found in an active region of the thin filament. This chapter summarises the contributions of the thesis and outlines future direction.

5.1 Hidden Markov model

At the molecular level, calcium modulates myosin's access to the thin filament. Once bound, myosin is assumed to potentiate the binding of further myosin molecules. The hidden Markov model proposed here, provides a theoretical framework for evaluating

thin filament regulation of actin – myosin interactions. Such Markov models appear naturally in problems where a spatially constrained clustering scheme is asked for. Thus, using this approach, the process of myosin binding along thin filaments has been quantified.

The analysis results obtained using our proposed model display evidence of cooperativity in the process of myosin binding to regulated actin filaments, as it had been anticipated. The stochastic nature of activation is strongly highlighted by the data, which was obtained in sub-optimal activation conditions, where the generation of activation waves and their catastrophic collapse can be observed. The data presented here provides evidence of a catastrophic collapse once the full-length myosin and calcium together cannot sustain the active state. This explains the ability of muscle to relax in conditions that would be expected to still permit myosin binding. Hence, using our developed model, we have been able to visualize cooperativity enabling the fundamental processes that underlie muscle relaxation to be studied.

Also, the performance of the MCMC method is compared against the performance of the reversible jump MCMC and against the raw data. The analysis discloses that our model provides a better simulation of the number of myosin molecules found in each pixel. Therefore, the hidden Markov model is a better substitute for the finite mixture model at capturing the interaction between myosin and actin.

5.2 Future direction

A possible extension of this methodology is to also estimate the model parameter, ψ, which is the parameter that considers the interaction effects of myosin binders. Due to its high dimension and also because of the high sensitivity of our model, it was difficult to simulate ψ. The current MCMC algorithm is not computationally efficient and simulating ψ would have decreased efficiency even more. This could be explored as further research.

Future work could also include doing a simulation study to evaluate how accurately the parameters can parameters can be estimated using the proposed hidden Markov model. A key strength of simulation studies is the ability to obtain empirical results about the performance of statistical methods in certain scenarios, as opposed to general analytic results, which may cover many scenarios. Thus, simulation studies can understand the behaviour of statistical methods because some parameters of interests are known from the process of generating the data.

Appendices

APPENDIX A

PROOF OF PROPOSITION 1 (SEE PAGE 94)

Proposition 1. *Let $\boldsymbol{\eta} = (\lambda, \varsigma, \boldsymbol{\psi})$ be the unknown parameters , where $\boldsymbol{\psi}$ includes all conditional probabilities $\psi_{\cdot|\cdot}$ and all parameters $\psi_{\cdot}$. Then , the stationary distribution $\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})$ will satisfy the detail balance*

$$\mathbb{P}(\boldsymbol{N}|\boldsymbol{\eta})q(\boldsymbol{N}, \mathbb{T}_j(\boldsymbol{N})) = \mathbb{P}(\mathbb{T}_j(\boldsymbol{N})|\boldsymbol{\eta})q(\mathbb{T}_j(\boldsymbol{N}), \boldsymbol{N}) \tag{4.13}$$

$$\textit{with} \quad \mathbb{P}(\boldsymbol{n}_t = \boldsymbol{N}|\boldsymbol{\eta}) = c(\boldsymbol{\eta})^{-1}\psi_{n_1}\prod_{j=2}^{J}\psi_{n_j|n_{j-1}}\prod_{j=1}^{J}\left(\frac{\lambda}{\varsigma}\right)^{n_j}$$

where $c(\boldsymbol{\eta})$ represents the normalising constant.

Proof.

(1) Assume that $n_j^{'} = n_j - 1$, which illustrates the detachment of 1 myosin molecule.

We then have

$$q(\boldsymbol{N}, \mathbb{T}_j(\boldsymbol{N})) = \varsigma \times \psi_{n_j'} \psi_{n_{j-1}|n_j'} \psi_{n_{j+1}|n_j'} \quad \text{and} \quad q(\mathbb{T}_j(\boldsymbol{N}), \boldsymbol{N}) = \lambda \times \psi_{n_j} \psi_{n_{j-1}|n_j} \psi_{n_{j+1}|n_j}$$

Substituting the above equations into the detail balance equation, we obtain

$$\frac{\lambda}{\varsigma} c(\boldsymbol{\eta})^{-1} \psi_{n_1} \prod_{j=2}^{J} \psi_{n_j|n_{j-1}} \left(\frac{\lambda}{\varsigma}\right)^{\sum_{j=1}^{J} n_j} \times \varsigma \times \psi_{n_j'} \psi_{n_{j-1}|n_j'} \psi_{n_{j+1}|n_j'} =$$
$$= \frac{\lambda}{\varsigma} c(\boldsymbol{\eta})^{-1} \psi_{n_1} \prod_{j=2}^{J} \psi_{n_j|n_{j-1}} \left(\frac{\lambda}{\varsigma}\right)^{\sum_{j=1}^{J} n_{j-1}} \frac{\psi_{n_{j+1}|n_j'}}{\psi_{n_{j+1}|n_j}} \frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}} \times \lambda \times \psi_{n_j} \psi_{n_{j-1}|n_j} \psi_{n_{j+1}|n_j}$$

Then, the detailed balance equation cancels out to

$$\begin{aligned}
\psi_{n_j'} \psi_{n_{j-1}|n_j'} \psi_{n_{j+1}|n_j'} &= \psi_{n_j} \psi_{n_{j-1}|n_j} \psi_{n_{j+1}|n_j} \frac{\psi_{n_{j+1}|n_j'}}{\psi_{n_{j+1}|n_j}} \frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}} \\
\psi_{n_{j-1}, n_j'} &= \psi_{n_{j-1}, n_j} \frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}} \frac{\psi_{n_{j-1}}}{\psi_{n_{j-1}}} \\
\psi_{n_{j-1}, n_j'} &= \psi_{n_{j-1}, n_j} \frac{\psi_{n_j', n_{j-1}}}{\psi_{n_j, n_{j-1}}} \\
\psi_{n_{j-1}, n_j'} &= \psi_{n_j', n_{j-1}}
\end{aligned}$$

(2) Assume that $n_j' = n_j + 1$, which illustrates the attachment of 1 myosin molecule. We then have

$$q(\boldsymbol{N}, \mathbb{T}_j(\boldsymbol{N})) = \lambda \times \psi_{n_j'} \psi_{n_{j+1}|n_j'} \psi_{n_{j-1}|n_j'} \quad \text{and} \quad q(\mathbb{T}_j(\boldsymbol{N}), \boldsymbol{N}) = \varsigma \times \psi_{n_j} \psi_{n_{j+1}|n_j} \psi_{n_{j-1}|n_j}$$

Substituting the above equations into the detail balance equation, we obtain

$$c(\boldsymbol{\eta})^{-1} \psi_{n_1} \prod_{j=2}^{J} \psi_{n_j|n_{j-1}} \left(\frac{\lambda}{\varsigma}\right)^{\sum_{j=1}^{J} n_j} \times \lambda \times \psi_{n_j'} \psi_{n_{j+1}|n_j'} \psi_{n_{j-1}|n_j'} =$$
$$= c(\boldsymbol{\eta})^{-1} \psi_{n_1} \prod_{j=2}^{J} \psi_{n_j|n_{j-1}} \left(\frac{\lambda}{\varsigma}\right)^{\sum_{j=1}^{J} n_j + 1} \frac{\psi_{n_{j+1}|n_j'}}{\psi_{n_{j+1}|n_j}} \frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}} \times \varsigma \times \psi_{n_j} \psi_{n_{j+1}|n_j} \psi_{n_{j-1}|n_j}$$

And the detailed balance cancels out to

$$\begin{aligned}
\psi_{n_j'}\psi_{n_{j-1}|n_j'}\psi_{n_{j+1}|n_j'} &= \frac{\psi_{n_{j+1}|n_j'}}{\psi_{n_{j+1}|n_j}}\frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}}\psi_{n_j}\psi_{n_{j+1}|n_j}\psi_{n_{j-1}|n_j} \\
\psi_{n_j'}\psi_{n_{j-1}|n_j'} &= \frac{\psi_{n_j'|n_{j-1}}}{\psi_{n_j|n_{j-1}}}\frac{\psi_{n_{j-1}}}{\psi_{n_{j-1}}}\psi_{n_j}\psi_{n_{j-1}|n_j} \\
\psi_{n_{j-1},n_j'} &= \psi_{n_{j-1},n_j}\frac{\psi_{n_j',n_{j-1}}}{\psi_{n_j,n_{j-1}}} \\
\psi_{n_{j-1},n_j'} &= \psi_{n_j',n_{j-1}}
\end{aligned}$$

□

APPENDIX B

EXTRA ANALYSIS FOR 5NM ACTIN KYMOGRAPHS

We present the simulation results obtained from applying the reversible jump MCMC algorithm on two kymographs (datasets) with the same conditions of 5nM actin.

B.1 Dataset 2 with 5nM actin

Table B.1: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
926	1047	1074	1082	1105	1429	50.8

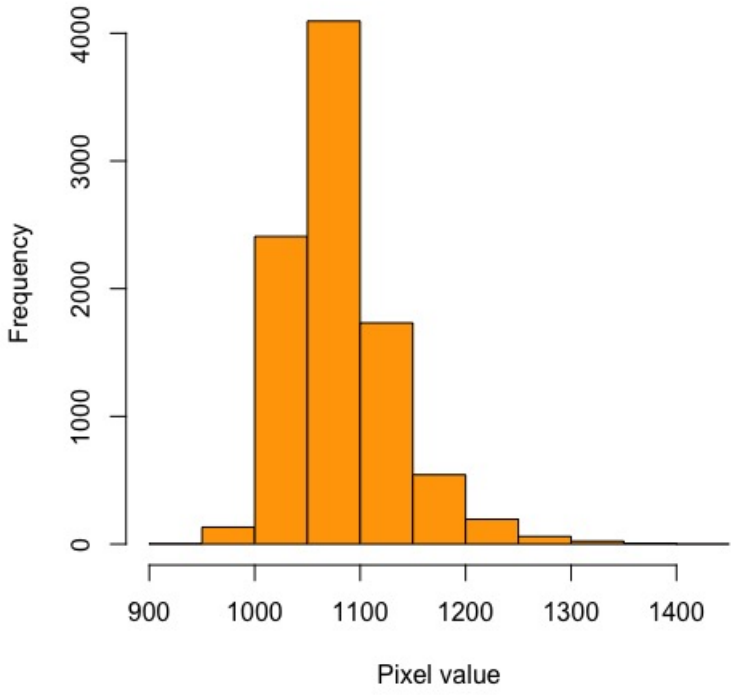


Figure B.1: Histogram of dataset with conditions actin only and myosin II = 5nM

Table B.2: Starting values used for the RJMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.19	0.2	0.15	0.16	0.2	0.1
Mean	1080	970	1000	1080	1130	1190	1250
Variance	50	45	50	40	62	43	52

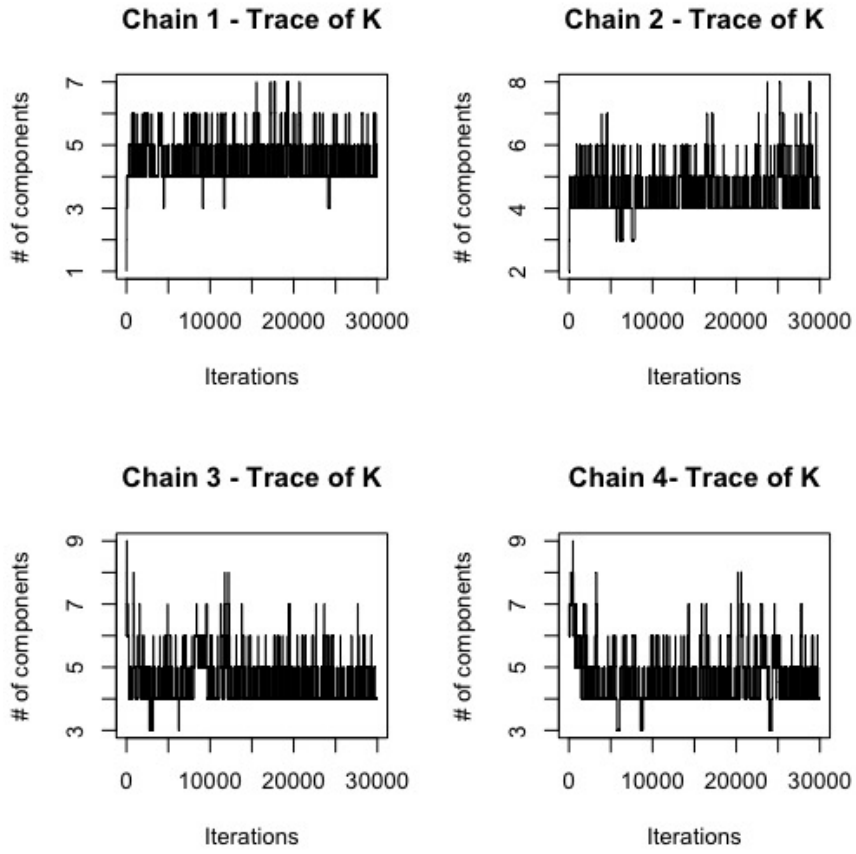


Figure B.2: Traces for mixing over K for a concentration of actin only and myosin II = 5nM, over 30000 sweeps

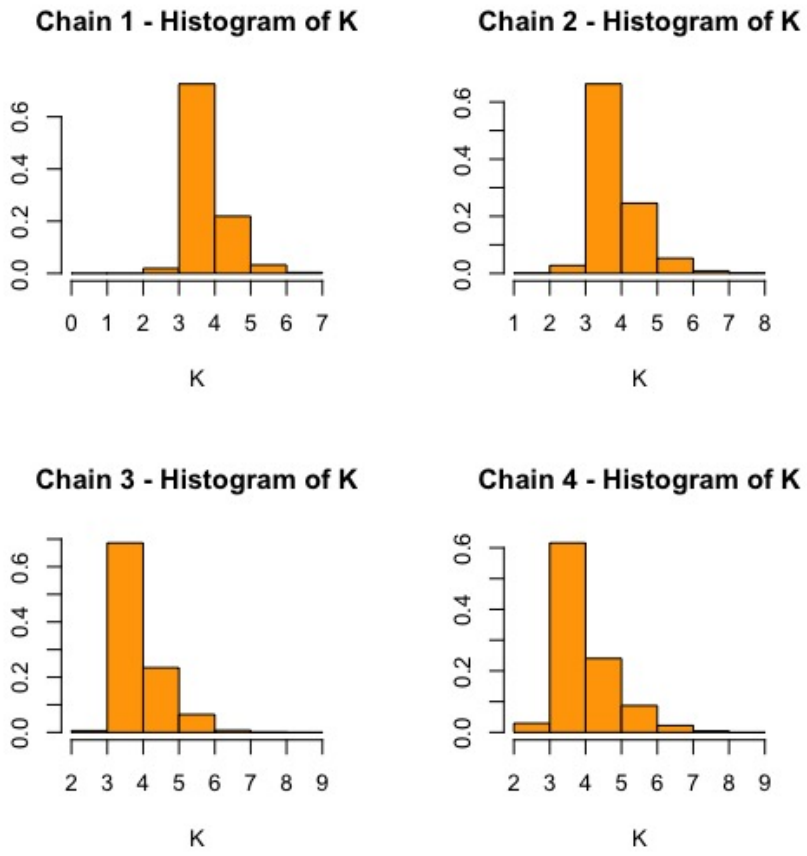


Figure B.3: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

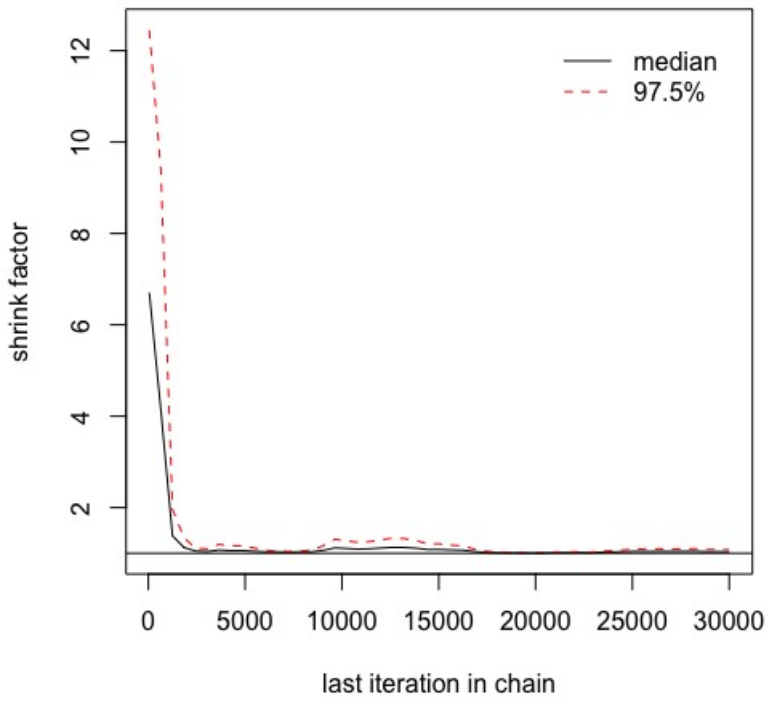


Figure B.4: Gelman plot for all four chains with different starting points

Table B.3: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.03328	1.076271

Table B.4: The spread of data resulting from the RjMCMC algorithm

# of components	**3**	**4**	**5**	**6**	**7**	**8**
Chain 1	378	18753	5018	752	99	0
Chain 2	812	16754	5938	1234	232	30
Chain 3	32	17301	6020	1501	144	2
Chain 4	888	15965	5985	1888	254	20

Table B.5: Summary results including the mean, variance and weight of having a total of either 3 or 4 bound myosins

# of	Total of 3 binders (4 components)			Total of 4 binders (5 components)		
binders	Mean = μ_3	Variance= σ_3	Weight=w_3	Mean = μ_4	Variance= σ_4	Weight=w_4
0	1048	26	0.36	1044	26	0.3
1	1082	31	0.41	1074	30	0.33
2	1129	48	0.18	1103	38	0.22
3	1200	72	0.04	1144	53	0.12
4				1203	69	0.04

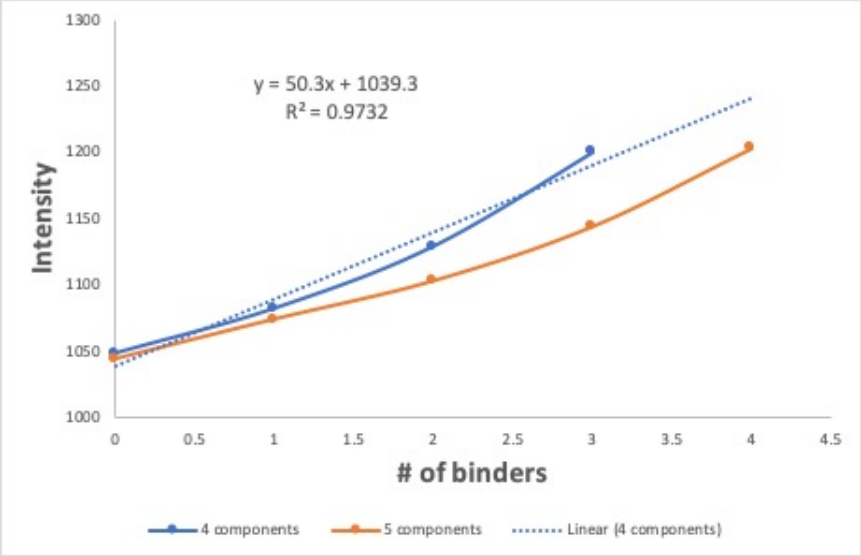


Figure B.5: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 4 or 5 total components for linear plots of the mean intensity vs. the number of binders.

B.2 Dataset 3 with 5nM actin

Table B.6: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
901	1047	1074	1081	1105	2453	64

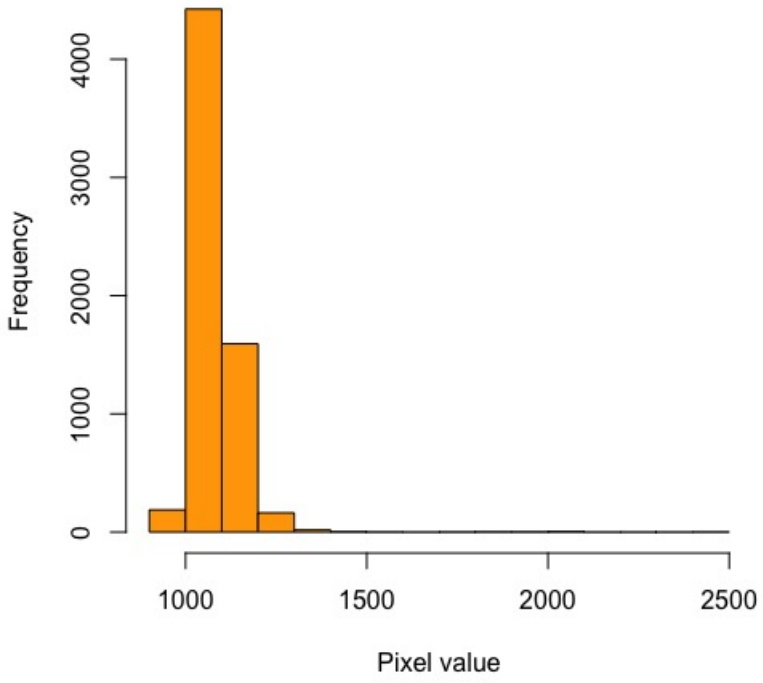


Figure B.6: Histogram of dataset with conditions actin only and myosin II = 5nM

Table B.7: Starting values used for the RJMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.19	0.2	0.15	0.16	0.2	0.1
Mean	1080	970	1000	1080	1130	1190	1250
Variance	50	45	50	40	62	43	52

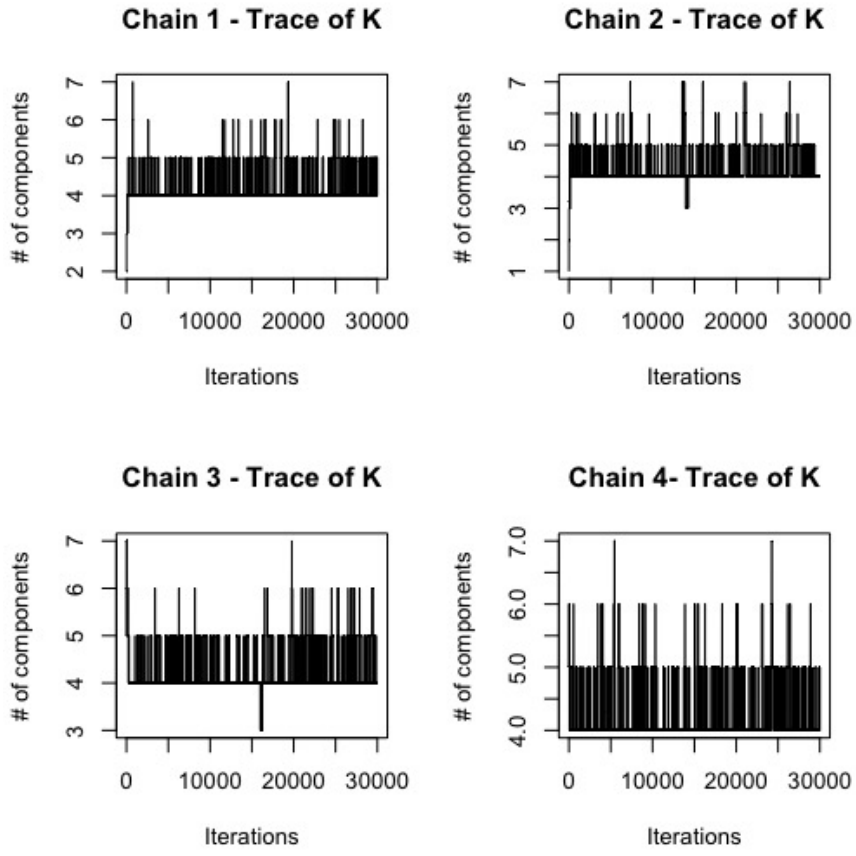


Figure B.7: Traces for mixing over K for a concentration of actin only and myosin II = 5nM, over 30000 sweeps

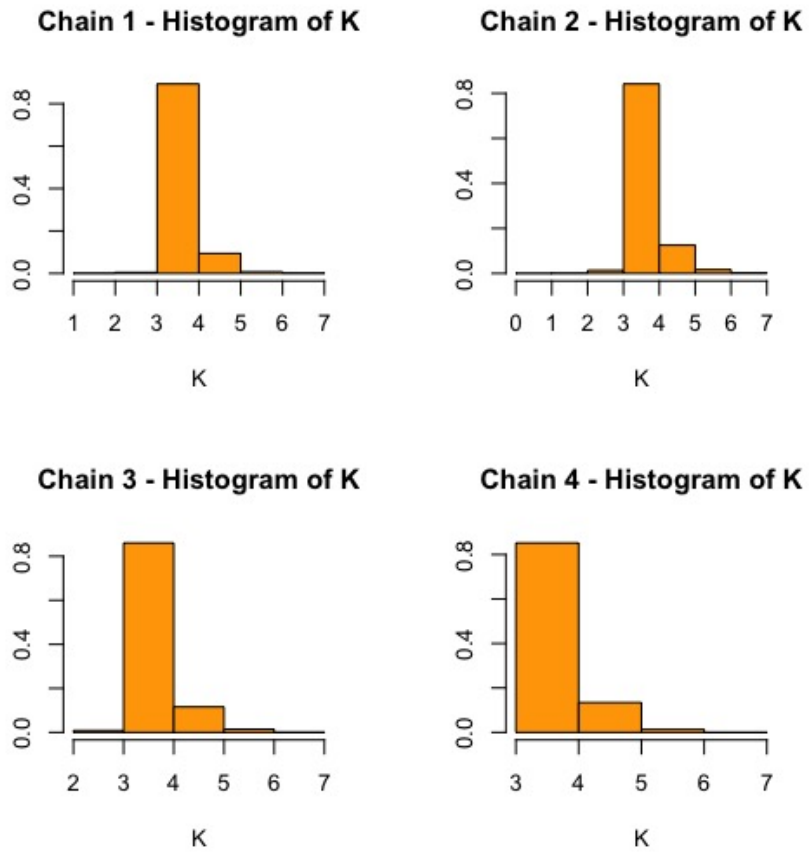


Figure B.8: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

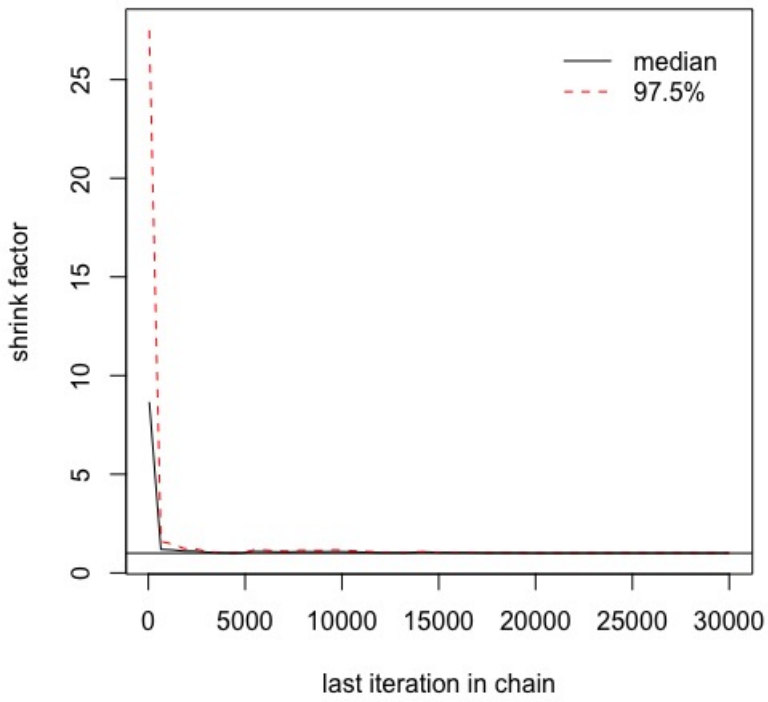


Figure B.9: Gelman plot for all four chains with different starting points

Table B.8: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.005483	1.009718

Table B.9: The spread of data resulting from the RJMCMC algorithm

# of components	**3**	**4**	**5**	**6**	**7**
Chain 1	0	22315	2494	165	26
Chain 2	319	20945	3204	471	61
Chain 3	256	21336	3098	308	2
Chain 4	0	21190	3425	373	12

Table B.10: Summary results including the mean, variance and weight of having a total of either 3 or 4 bound myosins

# of	Total of 3 binders (4 components)			Total of 4 binders (5 components)		
binders	Mean $=\mu_3$	Variance$=\sigma_3$	Weight$=w_3$	Mean $=\mu_4$	Variance$=\sigma_4$	Weight$=w_4$
0	1055	264	0.5	1041	197	0.41
1	1094	39	0.38	1083	43	0.36
2	1176	67	0.12	1136	53	0.18
3	1700	296	0.01	1254	86	0.04
4				1788	282	0.01

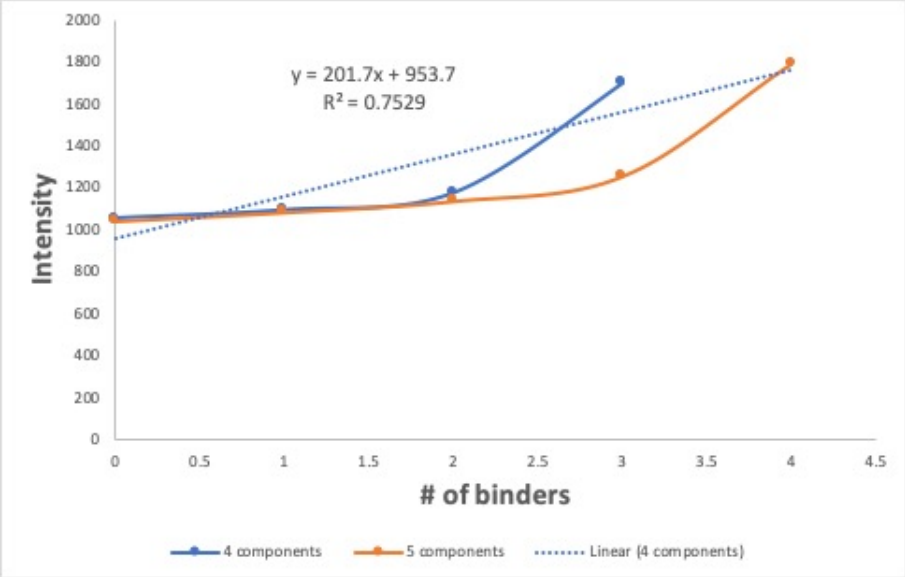


Figure B.10: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 4 or 5 total components for linear plots of the mean intensity vs. the number of binders.

APPENDIX C

EXTRA ANALYSIS FOR 10 NM MYOSIN AT PCA6

We present the simulation results obtained from applying the reversible jump MCMC algorithm on two kymographs (datasets) with the same conditions of 10 nM myosin at pCa6.

C.1 Dataset 2 with 10 nM myosin at pCa6

Table C.1: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
2745	3064	3162	3179	3274	3900	158

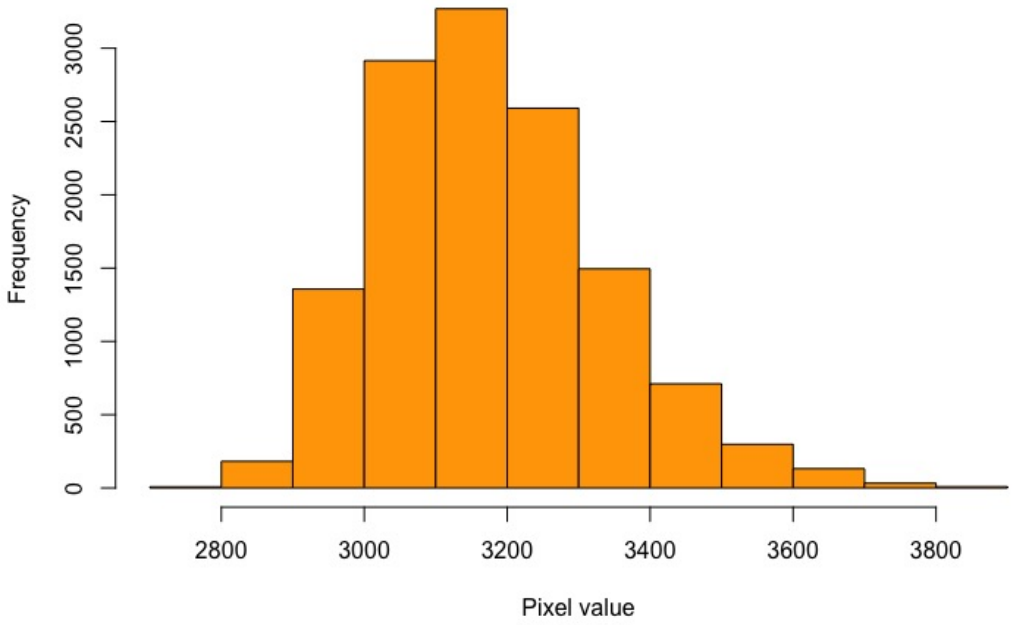


Figure C.1: Histogram of dataset with conditions actin only and myosin II = 5nM

Table C.2: Starting values used for the RJMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.19	0.2	0.15	0.16	0.2	0.1
Mean	3179	2900	3000	3160	3300	3450	3600
Variance	158	105	120	160	133	151	210

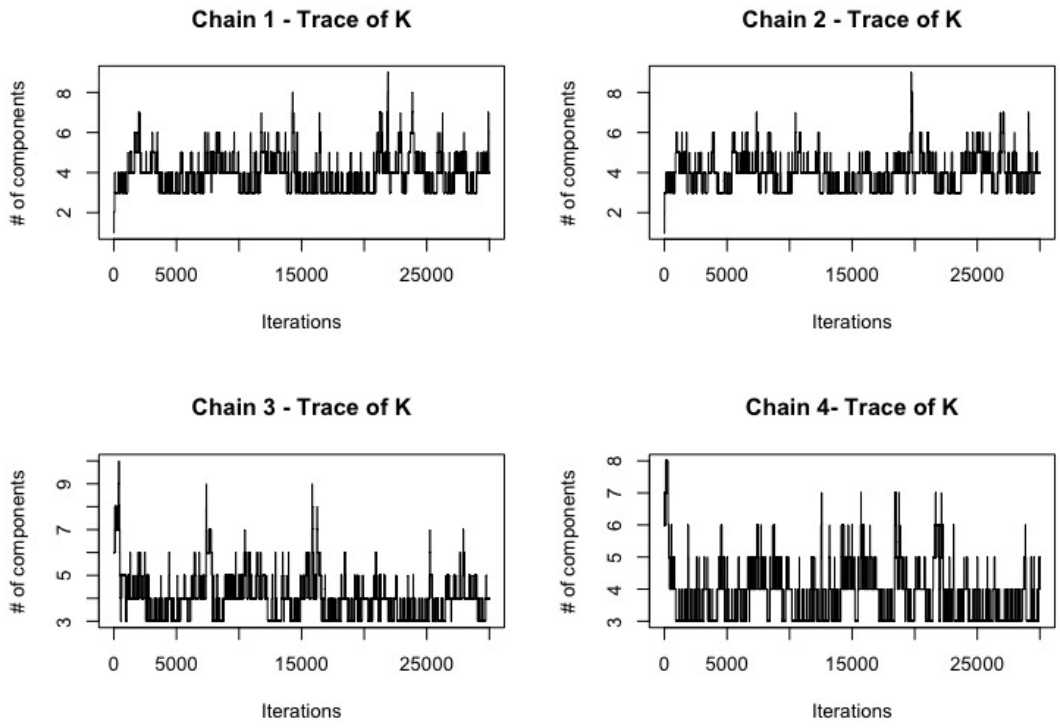


Figure C.2: Traces for mixing over K for thin filaments with a concentration of pCa6 and S1=10nM, over 30000 sweeps

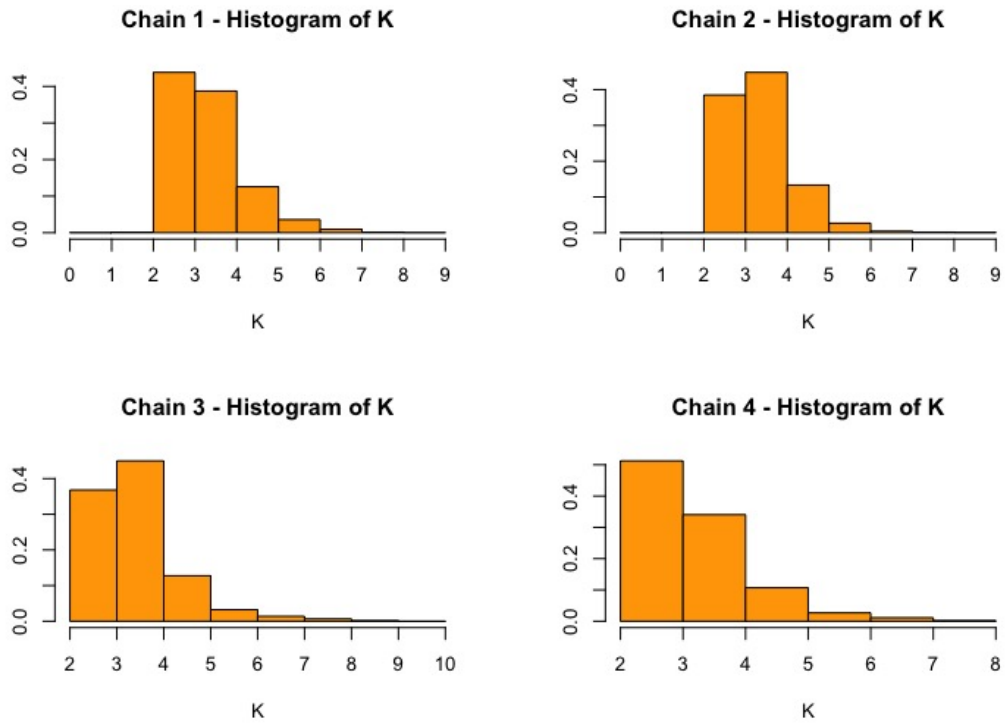


Figure C.3: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

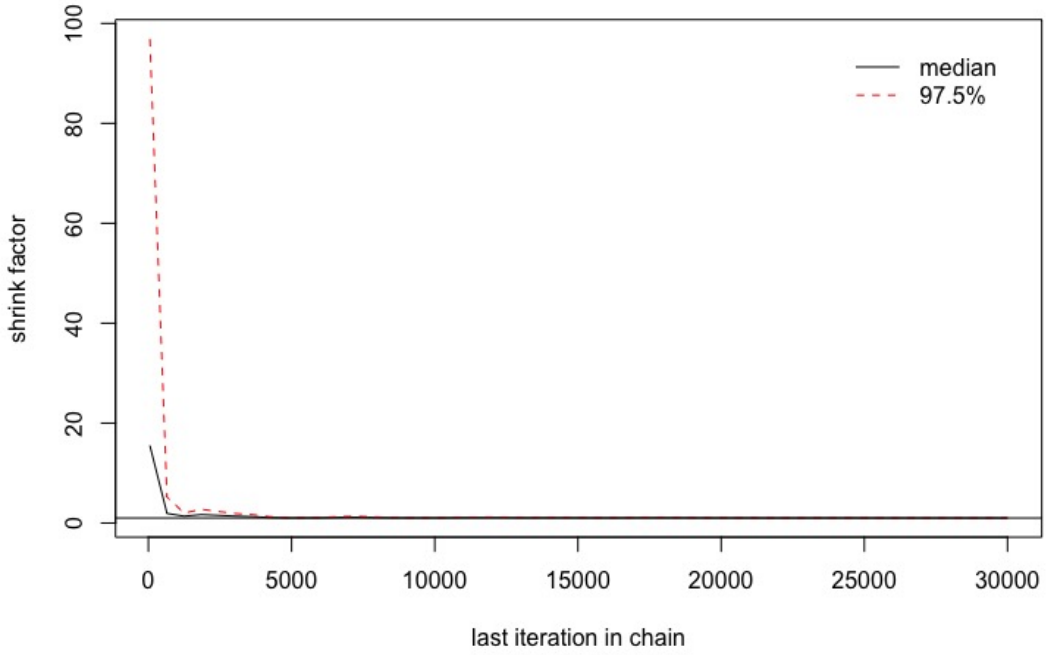


Figure C.4: Gelman plot for all four chains with different starting points

Table C.3: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.007235	1.018919

Table C.4: The spread of data resulting from the RJMCMC algorithm

# of components	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Chain 1	11070	9651	3135	833	260	46	5
Chain 2	9501	11244	3361	699	131	48	16
Chain 3	9276	11607	3033	825	202	54	3
Chain 4	12756	8772	2659	662	151	0	0

Table C.5: Summary results including the mean, variance and weight of having a total of either 2 or 3 bound myosins

# of	**Total of 2 binders (3 components)**			**Total of 3 binders (4 components)**		
binders	Mean = μ_2	Variance= σ_2	Weight=w_2	Mean =μ_3	Variance=σ_3	Weight=w_3
0	3039	80	0.27	3029	76	0.23
1	3189	116	0.56	3149	99	0.33
2	3369	161	0.18	3254	119	0.31
3				3409	156	0.13

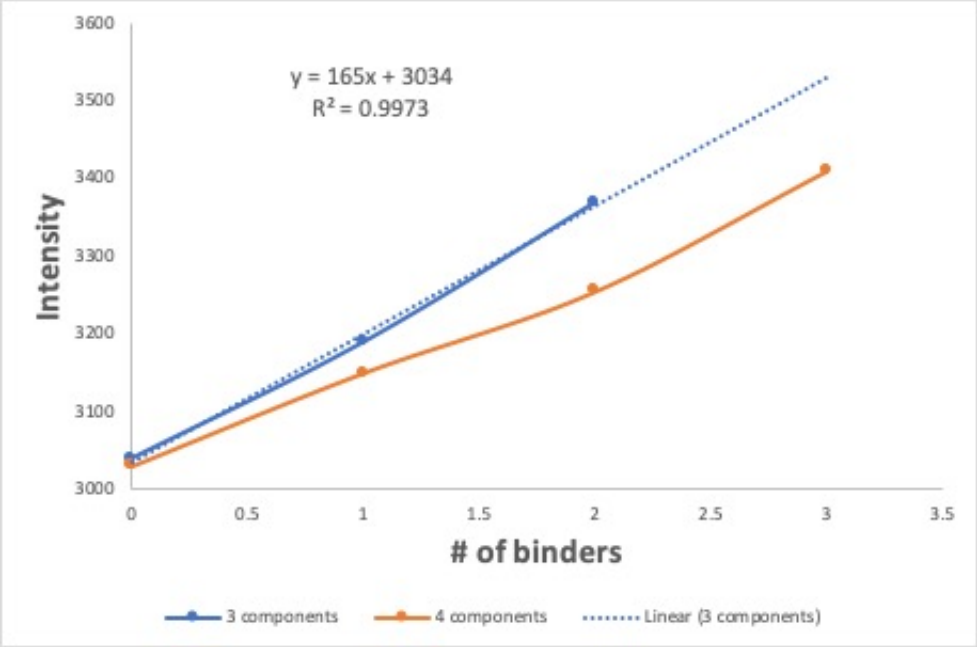


Figure C.5: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 3 or 4 total components for linear plots of the mean intensity vs. the number of binders.

C.2 Dataset 3 with 10 nM myosin at pCa6

Table C.6: Summary statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
2792	3096	3190	3208	3304	4038	154

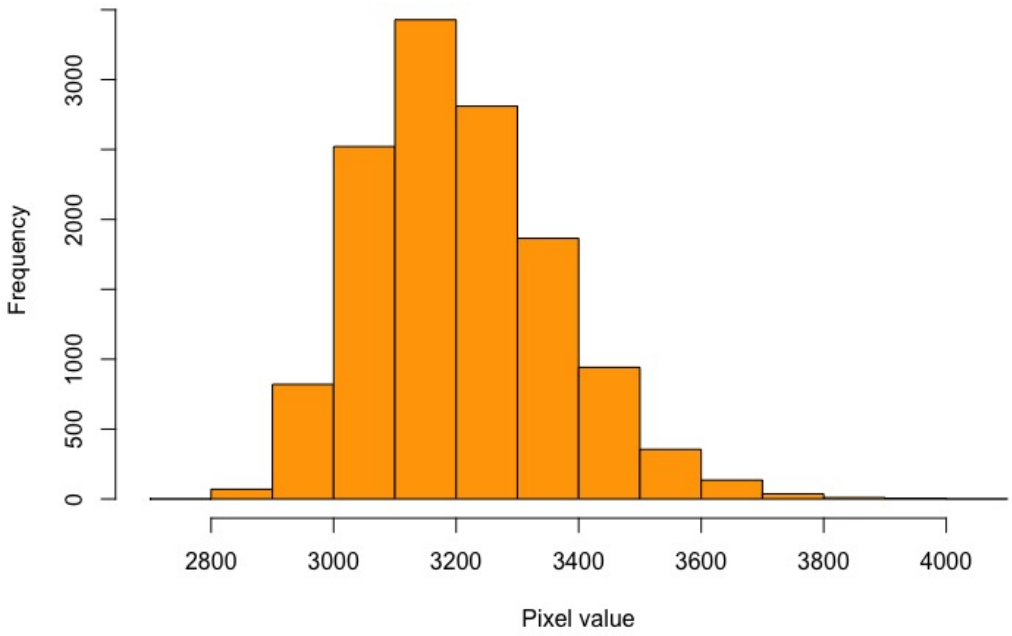


Figure C.6: Histogram of dataset for thin filament with conditions pCa6 and S1=10nM

Table C.7: Starting values used for the RJMCMC algorithm

	Chain 1 and 2	**Chain 3 and 4**					
Weight	1	0.19	0.2	0.15	0.16	0.2	0.1
Mean	3208	2900	3000	3160	3300	3450	3600
Variance	154	105	120	160	133	151	210

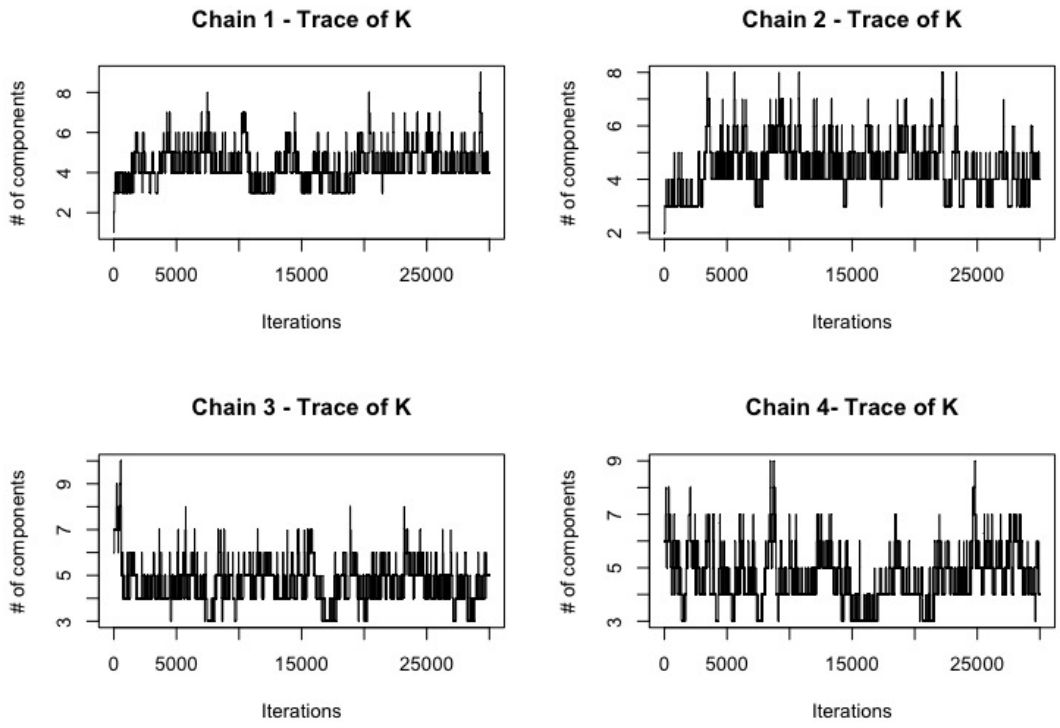


Figure C.7: Traces for mixing over K for thin filaments with a concentration of pCa6 and S1=10nM, over 30000 sweeps

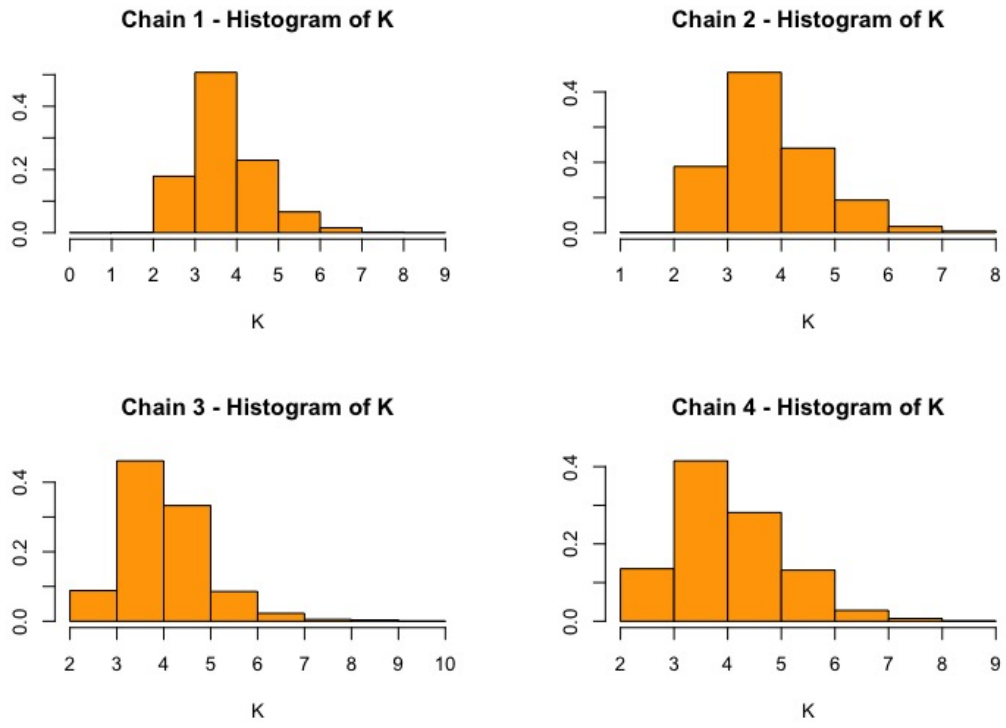


Figure C.8: Posterior distribution of 30 000 Ks produced by the reversible jump MCMC algorithm

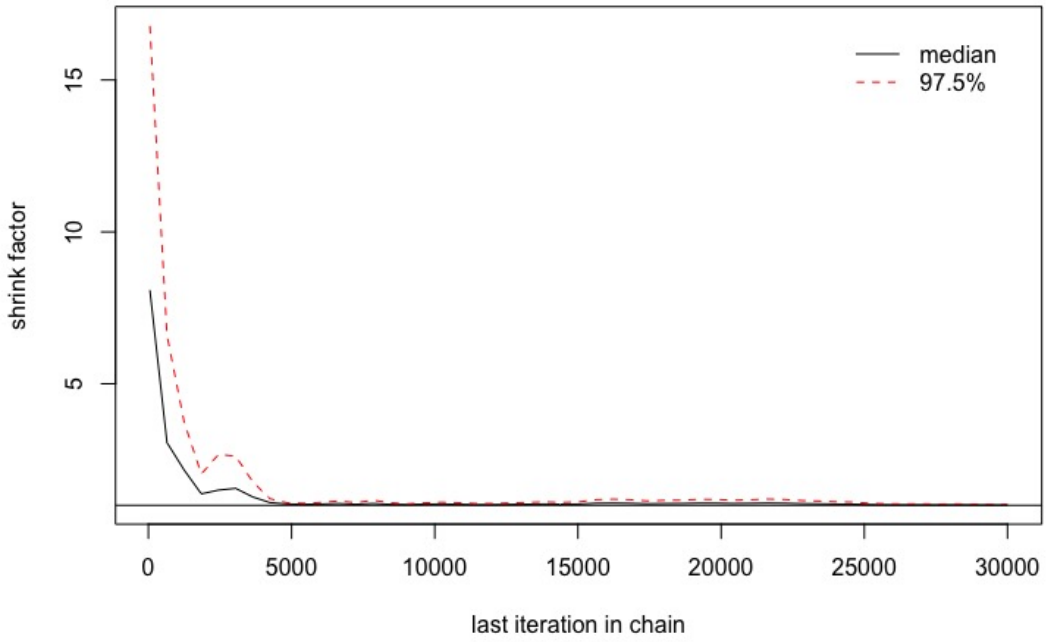


Figure C.9: Gelman plot for all four chains with different starting points

Table C.8: Potential scale reduction factor for all chains

Point est.	Upper C.I.
1.010237	1.02688

Table C.9: The spread of data resulting from the RJMCMC algorithm

# of components	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Chain 1	4015	12911	5780	1789	443	57	5
Chain 2	3147	12110	6631	2571	433	108	0
Chain 3	2563	10866	8868	2310	372	21	0
Chain 4	3589	10688	7121	2883	556	127	36

Table C.10: Summary results including the mean, variance and weight of having a total of either 3 or 4 bound myosins

# of	Total of 3 binders (4 components)			Total of 4 binders (5 components)		
binders	Mean $=\mu_3$	Variance$=\sigma_3$	Weight$=w_3$	Mean $=\mu_4$	Variance$=\sigma_4$	Weight$=w_4$
0	3069	81	0.28	3050	76	0.2
1	3193	99	0.37	3155	91	0.31
2	3328	135	0.28	3262	110	0.28
3	3483	159	0.07	3382	133	0.16
4				3539	152	0.04

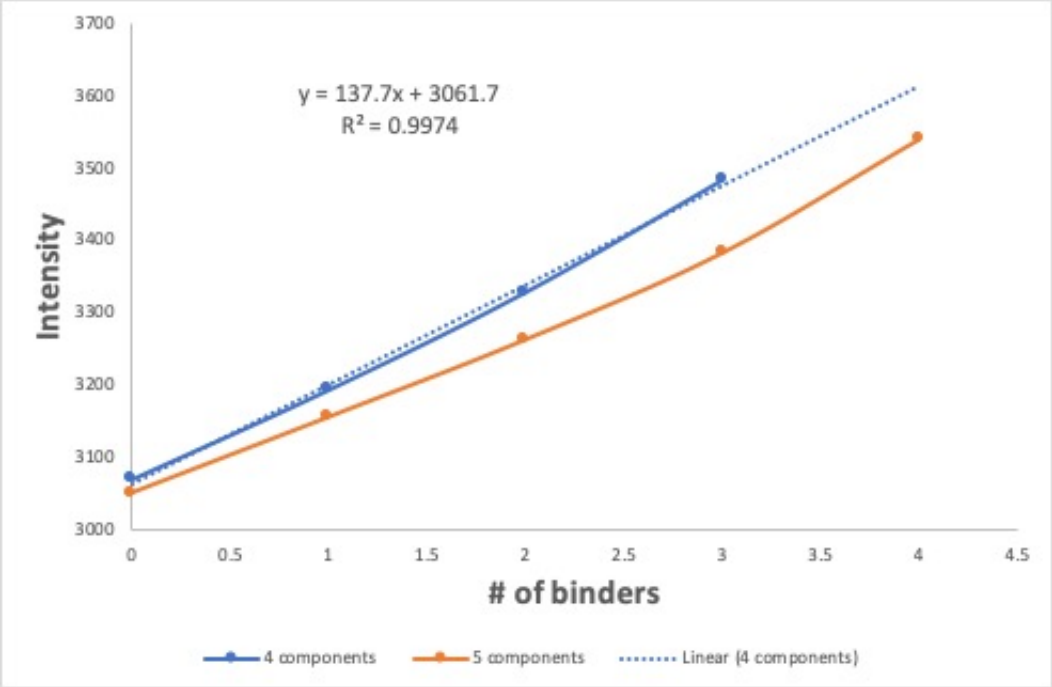
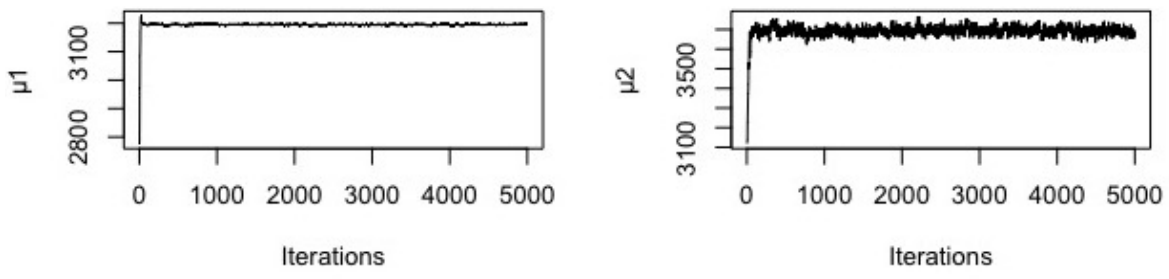


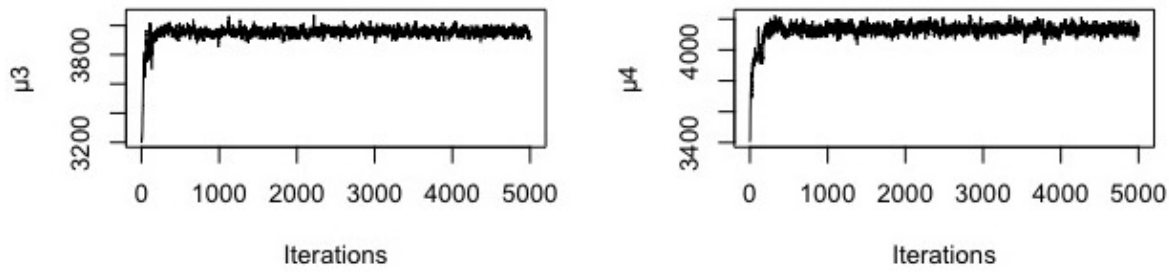
Figure C.10: Comparing the number of components used in the RJMCMC analysis. Using the RJMCMC analysis with either 4 or 5 total components for linear plots of the mean intensity vs. the number of binders.

C.3 Hidden Markov models simulation

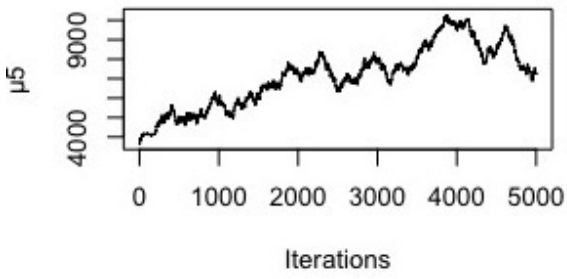
Figures C.11, C.12 and C.13 show the trace plots for Dataset 1 analysed using the proposed variation of the hidden Markov model.



(a)

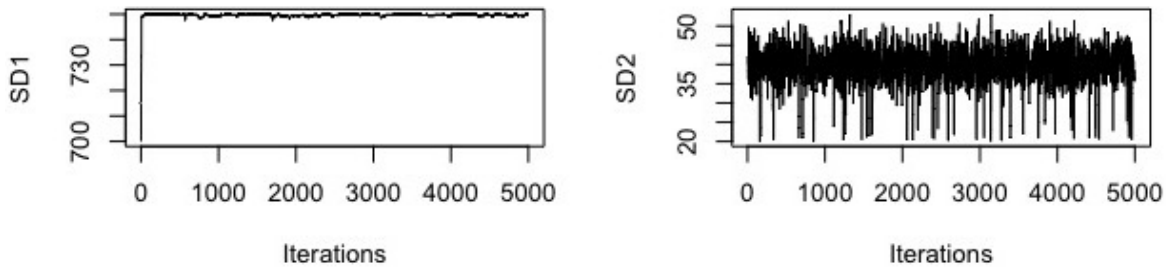


(b)

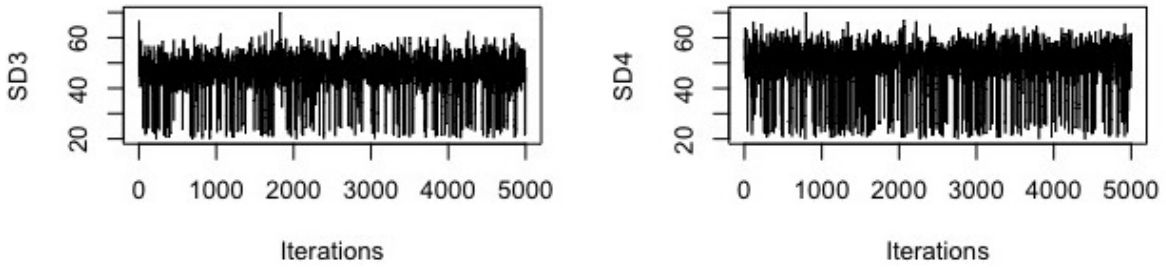


(c)

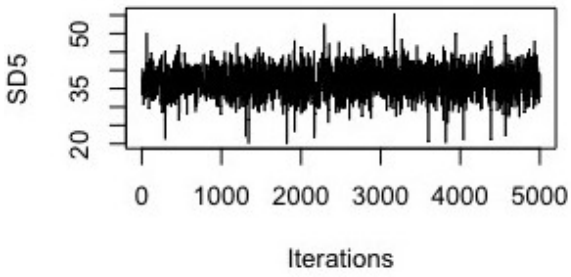
Figure C.11: MCMC trace plots of $\mu_1, \mu_2, \mu_3, \mu_4$ and μ_5



(a)

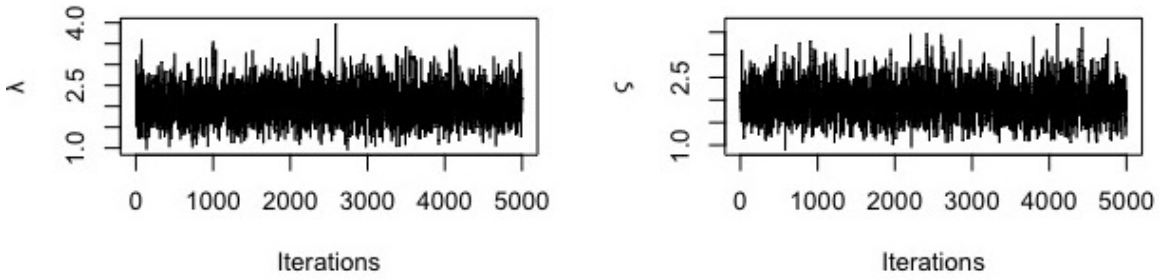


(b)

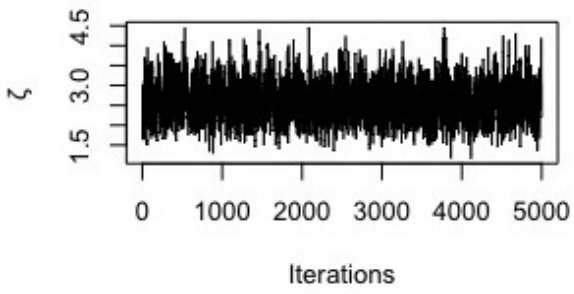


(c)

Figure C.12: MCMC trace plots of $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ and σ_5



(a)



(b)

Figure C.13: MCMC trace plots of λ, ς and ζ

APPENDIX D

COMPUTATIONAL CODE FOR STATISTICAL SIMULATIONS USING R SOFTWARE

D.1 RJMCMC simulations

```
##############################
#### STANDARDISATION ######
##############################
set.seed(2345)
library(standardize)
library("ggplot2")

actin_1_4_v=as.vector(as.matrix(actin_1_4))
actin_5_4_v=as.vector(as.matrix(actin_5_4))
actin_5_6_v=as.vector(as.matrix(actin_5_6))
```

```
movie_33_v=as.vector(as.matrix(movie_33))
movie_34_v=as.vector(as.matrix(movie_34))
movie_37_v=as.vector(as.matrix(movie_37))

summary(actin_1_4_v)
hist(actin_1_4_v,col="orange",main="", xlab="Pixel value")
hist(actin_5_4_v, col="green", add = TRUE)
summary(actin_5_4_v)
hist(actin_5_4_v,col="orange",main="", xlab="Pixel value")
summary(actin_5_6_v)
hist(actin_5_6_v,col="orange",main="", xlab="Pixel value")

summary(movie_33_v)
hist(movie_33_v,col="orange",main="", xlab="Pixel value")
summary(movie_34_v)
hist(movie_34_v,col="orange",main="", xlab="Pixel value")
summary(movie_37_v)
hist(movie_37_v,col="orange",main="", xlab="Pixel value")

################################################
######### OVERLAPPING HISTOGRAMS ###############
################################################
hist(actin_1_4_v, col="grey",ylim=c(0,5000), xlim=c(0,6000),main="
    Overlapping the actin datasets", xlab="Mean light intensity")
hist(actin_5_4_v, col="blue", add=T)
hist(actin_5_6_v, col="red", add=T)

legend(2500, 5000, legend=c("actin dataset 1", "actin dataset 2","actin
    dataset 3"),
       lwd=5,col=c("grey","red", "blue"), lty=1:1, cex=0.8,
       text.font=2)

hist(movie_33_v, col="grey",ylim=c(0,5000), xlim=c(2000,5000),main="
    Overlapping the calcium datasets", xlab="Mean light intensity")
hist(movie_34_v,  col="blue", add=T)
```

```
hist(movie_37_v,   col="red", add=T)
legend(3000, 5000, legend=c("pCa 6 dataset 1", "pCa 6 dataset 2","pCa 6 dataset 3"),
       lwd=5,col=c("grey","red", "blue"), lty=1:1, cex=0.8,
       text.font=2)

################################################
##### PREPARE EACH DATASET IN THIS STEP ######
################################################
library(miscF)
library(coda)
scaled=movie_37_v
y=scaled

#######CHAINS 1 and 2################
w_one <- c(1)
mu_one <- c(3208)
sigma2_one <- c(154)

#constructs and executes a function call from a function and a list of arguments to be passed to it
Z <- do.call(cbind, lapply(1, function(i)w_one[i]*dnorm(y, mu_one[i], sqrt(sigma2_one[i]))))
#returns a vector or array or list of values obtained by applying a function to margins of an array or matrix
Z <- apply(Z, 1, function(x) which(x==max(x))[1])
# simulation of RJMCMC algorithm for Chain 1
chain1<-uvnm.rjmcmc(y, nsweep=30000, kmax=11, k=1, w_one, mu_one, sigma2_one, Z,delta=1,
                    xi=NULL, kappa=NULL, alpha=2, beta=NULL, g=0.2, h=NULL)
# simulation of RJMCMC algorithm for Chain 2
chain2<-uvnm.rjmcmc(y, nsweep=30000, kmax=11, k=1, w_one, mu_one, sigma2_one, Z,delta=1,
```

```
                    xi=NULL, kappa=NULL, alpha=2,beta=NULL, g=0.2, h=NULL
    )

########## CHAINS 3 and 4 ###############
w <-c(0.19,0.20, 0.15, 0.16,0.20,0.10)
mu <-c(2900,3000,3160,3300,3450,3600)
sigma2 <-c(105, 120,160,133,151,210)
Z <-do.call(cbind, lapply(1:6, function(i)w[i]*dnorm(y, mu[i], sqrt(
    sigma2[i]))))
Z <- apply(Z, 1, function(x) which(x==max(x))[1])
chain3 <-uvnm.rjmcmc(y, nsweep=30000, kmax=10, k=6, w, mu, sigma2, Z,
    delta=1, xi=NULL, kappa=NULL, alpha=2,beta=NULL, g=0.2, h=NULL)
chain4 <-uvnm.rjmcmc(y, nsweep=30000, kmax=10, k=6, w, mu, sigma2, Z,
    delta=1, xi=NULL, kappa=NULL, alpha=2,beta=NULL, g=0.2, h=NULL)

k_chain1=mcmc(as.mcmc(chain1$k.save))
k_chain2=mcmc(as.mcmc(chain2$k.save))
k_chain3=mcmc(as.mcmc(chain3$k.save))
k_chain4=mcmc(as.mcmc(chain4$k.save))
combinedchains = mcmc.list(k_chain1,k_chain2,k_chain3,k_chain4)

########PLOTS########
par(mfrow=c(2,2))
#outputs density plots of each chain
densplot(k_chain1, show.obs = TRUE, col="orange",xlab = "K", type="l",
    main="Chain 1 - Histogram of K")
densplot(k_chain2, show.obs = TRUE,col="orange",xlab = "K", type="l",main
    ="Chain 2 - Histogram of K")
densplot(k_chain3, show.obs = TRUE,col="orange",xlab = "K", type="l",
    main="Chain 3 - Histogram of K")
densplot(k_chain4, show.obs = TRUE,col="orange", xlab = "K", type="l",
    main="Chain 4 - Histogram of K")

#outputs trace plots of each chain
```

```
traceplot(k_chain1,type="l",smooth = FALSE,ylab="# of components",main="
    Chain 1 - Trace of K")
traceplot(k_chain2,type="l",smooth = FALSE,ylab="# of components",main="
    Chain 2 - Trace of K")
traceplot(k_chain3,type="l",smooth = FALSE,ylab="# of components",main="
    Chain 3 - Trace of K")
traceplot(k_chain4,type="l",smooth = FALSE,ylab="# of components",main="
    Chain 4- Trace of K")

#######DIAGNOSTIC OF CHAINS###########
gelman_diagnostic=gelman.diag(combinedchains)
gelman.plot(combinedchains)
psrf=gelman_diagnostic$psrf

########AUTOCORRELATIN OF COMBINED CHAINS######## #testing for
    autocorrelation and plotting autocorrelation
autocorr(combinedchains,lags=c(1,5,10,100,500), relative =TRUE)
autocorr.plot(combinedchains,lag.max=100,ask=dev.interactive())

burnin=5000

######## SPREAD OF DATA #########
table(chain1$k.save[-(1:burnin)])
table(chain2$k.save[-(1:burnin)])
table(chain3$k.save[-(1:burnin)])
table(chain4$k.save[-(1:burnin)])

#######WEIGHT FOR 4 COMPONENTS######
w_chain1=mcmc(chain1$w.save[-(1:burnin)])
w_chain2=mcmc(chain2$w.save[-(1:burnin)])
w_chain3=mcmc(chain3$w.save[-(1:burnin)])
w_chain4=mcmc(chain4$w.save[-(1:burnin)])

weight_4_1=array(w_chain1[which(k_chain1[-(1:burnin)]==4)])
weight_4_2=array(w_chain2[which(k_chain2[-(1:burnin)]==4)])
```

```
weight_4_3=array(w_chain3[which(k_chain3[-(1:burnin)]==4)])
weight_4_4=array(w_chain4[which(k_chain4[-(1:burnin)]==4)])

weight<-matrix(unlist(weight_4_1,weight_4_2,weight_4_3), ncol = 4, byrow
    = TRUE)
weight_4_4=matrix(unlist(weight_4_4),ncol=4,byrow=TRUE)
weight=rbind(weight,weight_4_4)
mean_w<-apply(weight,2,mean)

######## WEIGHT FOR 5 COMPONENTS #########
weight_5_1=array(w_chain1[which(k_chain1[-(1:burnin)]==5)])
weight_5_2=array(w_chain2[which(k_chain2[-(1:burnin)]==5)])
weight_5_3=array(w_chain3[which(k_chain3[-(1:burnin)]==5)])
weight_5_4=array(w_chain4[which(k_chain4[-(1:burnin)]==5)])

weight<-matrix(unlist(weight_5_1,weight_5_2,weight_5_3), ncol = 5, byrow
    = TRUE)
weight_5_4=matrix(unlist(weight_5_4),ncol=5,byrow=TRUE)
weight=rbind(weight,weight_5_4)
mean_w<-apply(weight,2,mean)

####### MEAN FOR 4 COMPONENTS #########
musave1<-mcmc(chain1$mu.save[-(1:burnin)])
musave2<-mcmc(chain2$mu.save[-(1:burnin)])
musave3<-mcmc(chain3$mu.save[-(1:burnin)])
musave4<-mcmc(chain4$mu.save[-(1:burnin)])

musavek1 = musave1[which(k_chain1[-(1:burnin)]==4)]
musavek2 = musave2[which(k_chain2[-(1:burnin)]==4)]
musavek3 = musave3[which(k_chain3[-(1:burnin)]==4)]
musavek4 = musave4[which(k_chain4[-(1:burnin)]==4)]
mu<-matrix(unlist(musavek1,musavek2,musavek3), ncol = 4, byrow = TRUE)
musavek4=matrix(unlist(musavek4),ncol=4,byrow=TRUE)
mu=rbind(mu,musavek4)
mean<-apply(mu,2,mean)
```

```
159
    ######## SD for 4 components ##########
161 sigmasave1<-chain1$sigma2.save[-(1:burnin)]
    sigmasave2<-chain2$sigma2.save[-(1:burnin)]
163 sigmasave3<-chain3$sigma2.save[-(1:burnin)]
    sigmasave4<-chain4$sigma2.save[-(1:burnin)]
165
    sigma1 = sigmasave1[which(k_chain1[-(1:burnin)]==4)]
167 sigma2 = sigmasave2[which(k_chain2[-(1:burnin)]==4)]
    sigma3 = sigmasave3[which(k_chain3[-(1:burnin)]==4)]
169 sigma4=sigmasave4[which(k_chain4[-(1:burnin)]==4)]

171 SD<-matrix(unlist(sigma1,sigma2,sigma3), ncol = 4, byrow = TRUE)
    SD2<-matrix(unlist(sigma4),ncol=4,byrow=TRUE)
173 sigma=rbind(SD,SD2)
    mean<-sqrt(apply(sigma,2,mean))
175
    ########### MEAN FOR 5 COMPONENTS ################
177 musavek1 = musave1[which(k_chain1[-(1:burnin)]==5)]
    musavek2 = musave2[which(k_chain2[-(1:burnin)]==5)]
179 musavek3 = musave3[which(k_chain3[-(1:burnin)]==5)]
    musavek4 = musave4[which(k_chain4[-(1:burnin)]==5)]
181 mu<-matrix(unlist(musavek1,musavek2,musavek3), ncol = 5, byrow = TRUE)
    musavek4=matrix(unlist(musavek4),ncol=5,byrow=TRUE)
183 mu=rbind(mu,musavek4)
    mean<-apply(mu,2,mean)
185
    ######## SD for 5 components ##########
187 sigma1 = sigmasave1[which(k_chain1[-(1:burnin)]==5)]
    sigma2 = sigmasave2[which(k_chain2[-(1:burnin)]==5)]
189 sigma3 = sigmasave3[which(k_chain3[-(1:burnin)]==5)]
    sigma4=sigmasave4[which(k_chain4[-(1:burnin)]==5)]
191 SD<-matrix(unlist(sigma1,sigma2,sigma3), ncol = 5, byrow = TRUE)
    SD2<-matrix(unlist(sigma4),ncol=5,byrow=TRUE)
193 sigma=rbind(SD,SD2)
```

```
mean<-sqrt(apply(sigma,2,mean))
```

D.2 Hidden Markov MCMC

```
####################
### IMPORT DATA ###
####################
set.seed(1234)
library(truncnorm)
library('MASS')

movie_33=read.delim("/Users/mdmiha/Documents/PhD Biostats/Chapter 4 -
    including analysis and data/New approach using raw datasets/Compare
    untreated-rollball50/Thin filamentes pCa6 S1 10 nM/Reslice of
    2016-04-06_33 untreated.txt",header=FALSE)
binders=movie_33
data=as.matrix(binders)#define the dataset as a matrixo
data=data[-c(1:50),-c(51:200)]

p <- ncol(data) # number of columns for data
nn <- nrow(data) # number of rows for data
S=p
J=nn

#data transformed using RJMCMC and is being used to simulate from it
load("/Users/mdmiha/Documents/PhD Biostats/Chapter 4 - including analysis
     and data/Ch4-transformed datasets/transformed-dataset-calcium-movie33
    .Rdata")
scaled=z
scaled_rjmcmc=as.matrix(scaled)
scaled_rjmcmc=scaled[-c(1:50),-c(51:200)] #has to match the dimension of
    the poisson process
MAXNO_COMPONENTS=5 # up to 11 binders plus when there is no binding

########################
### STARTING VALUES ###
```

```
28 ##########################
   #these starting values are based on results from previous run of the
       algorithm
30 mu_start=c(2700,3155,3220,3417,3609)
   Var_start=c(705,41,45,49,42)
32
   # we do a random generation for psi because we do not simulate it in this
        thesis
34 #psi is a rate
   psi=matrix(1,MAXNO_COMPONENTS,MAXNO_COMPONENTS)
36 psi=rpois(matrix(2,MAXNO_COMPONENTS,MAXNO_COMPONENTS),2)/20
   psi<-matrix(psi,byrow=MAXNO_COMPONENTS, ncol=MAXNO_COMPONENTS)+0.01 #
       adding 0.01 because otherwise we get Inf in                    "
       poisson process"
38 psi_start=psi

40 lambda=2 #choose values for rate of attaching; this is the mean so it
       should be very small considering that we have mostly 0s in the dataset
   lambda_start=2
42 varsigma=2 #choose values for rate of dettaching; in our study, the
       dettaching rate is similar to the attaching rate
   varsigma_start=2
44 zeta=2.5
   zeta_start=2.5
46 delta=0.1 #because it represents the time interval given by one pixel

48 #standard deviation should be very small
   sdupdate_mu=c(32,29,20,28,34)
50 sdupdate_var=c(15,9,9,9,9)
   sdupdate_lambda=0.6
52 sdupdate_varsigma=0.5
   sdupdate_zeta=0.6
54
   parameter=list(mu_start,Var_start,lambda_start,varsigma_start,zeta_start)
56 startvalue=c(mu_start,Var_start,lambda_start,varsigma_start,zeta_start)
```

```
58 ##############################################
   ### DEFINING INITIAL MATRIX FOR LATENT ###
60 ##############################################
   N=scaled_rjmcmc+1 # we increase the labelling by 1 because we cannot have
       0s in this matrix;
62
   ################################################
64 ### CONSTRUCTING THE PRIOR DISTRIBUTIONS ###
   ################################################
66 K=MAXNO_COMPONENTS
   alpha=c(100,120,120,130,100) #this is the shape in gamma distribution
68 beta=c(3,3,2.5,2.5,2.7) #this is represented by scale in gamma
       distribution

70 prior=function(parameter,K,alpha,beta){
     Mean=parameter[[1]]
72   sigma=sqrt(parameter[[2]]) # this is the standard deviation that I
       believe fits the dataset based on results from rjmcmc
     muprior=rep(0,K)
74   varprior=rep(0,K)
     for(k in 1:K){
76     mu=parameter[[1]]
       muprior[k]=dnorm(mu[k],Mean[k],sigma[k],log=TRUE)
78   }
     for (k in 1:K){
80     Var=parameter[[2]]
       varprior[k]=dgamma(Var[k],alpha[k],beta[k],log=TRUE)
82   }
     lamb=parameter[[3]]
84   lambda_prior=dgamma(lamb,5.5,2.3,log = TRUE)
     varsig=parameter[[4]]
86   varsigma_prior=dgamma(varsig,5.7,2.5,log = TRUE)
     zet=parameter[[5]]
88   zeta_prior=dgamma(zet,6,2,log = TRUE)
```

```
  #return(list(muprior,varprior,lambda_prior,varsigma_prior,zeta_prior))
  return(sum(muprior+varprior+lambda_prior+varsigma_prior+zeta_prior)) #
    sum instead of multiplication because we have taken logs of the priors
     above
}

######################
### DECAY FACTOR ###
######################
decay_fun = function(j,k,zeta){
  if(j-k==0){
    return(1)
  }
  else if (abs(j-k)>3){
    return(0)
  }
  else{
    return( exp(-zeta*abs(j-k)) )
  }
}
decay=matrix(0,nn,nn)
for(j in 1:nn){
  for(k in 1:nn){
    decay[j,k]=decay_fun(j,k,zeta)
  }
}

########################
### LATENT PROCESS ###
########################
birth_temp=matrix(0,J,S)
death_temp=matrix(0,J,S)
nothing_temp=matrix(0,J,S)
latent_fun=function(J,S,N){
  for(s in 2:S){
```

```
    for(j in 1:J){
      if(N[j,s]-N[j,s-1]>0){
        birth_temp[j,s]=N[j,s]-N[j,s-1]
      }
      else if (N[j,s]-N[j,s-1]<0){
        death_temp[j,s]=-N[j,s]+N[j,s-1]
      }
      else {
        nothing_temp[j,s]=0
      }
    }
  }
  y_list=list(birth_temp, death_temp, nothing_temp)
}
latent<-latent_fun(J,S,N) #recording the function "latent_fun", which is
    also recorded in the function below

####################
### LIKELIHOOD ###
####################
likelihood=function(K, parameter, data, N, decay){
  mu=parameter[[1]]
  var=parameter[[2]]
  Mean=matrix(0,J,S)
  Var=matrix(0,J,S)
  likeli=matrix(0,nn,p)
  for(j in 1:J){
    for(s in 2:S){
      for(k in (1:J)[-j]){
        if(N[k,s]>1){
          Mean[j,s] = Mean[j,s] + (mu[N[k,s]]-mu[1])*decay[j,k]
          Var[j,s]= Var[j,s] + var[N[k,s]]*(decay[j,k]^2)
        }
      }
    }
```

```
156   }
   X_Mean=Mean+mu[1] # mean intensity value plus the noise (the noise is
     also equivalent to having no binding)
158   Variance=Var+var[1] # variance plus noise
   likeli = dnorm(data,mean=X_Mean ,sd=sqrt(Variance),log=TRUE)
160   sumlikeli=sum(likeli) #sum instead of multiplication because we have
     taken logs of the likelihood
   return(sumlikeli)
162 }

164 ######################################################
### BIRTH - DEATH MODEL => POISSON PROCESS ###
166 ######################################################
poisson_process=function(S,J,N,parameter,psi_start,lambda,varsigma,delta
    =0.1){
168   latent = latent_fun(J,S,N)
   birth_temp = latent[[1]]
170   death_temp = latent[[2]]
   nothing_temp = latent[[3]]
172   psi=psi_start
   birth_rate=matrix(0.2,J,S)
174   death_rate=matrix(0.3,J,S)
   pois_process=matrix(0,J,S)
176   temp_one=matrix(0,J,1)
  # we take logs of the whole process because otherwise the product will
     be equal to zero when we do the final product of the matrix
178   for(s in 2:S){
     for(j in 2:J){
180       if(j<J){
         #cat(j,",",s,",",N[j+1,s-1],",",N[j,s],"\n")
182         birth_rate[j,s]=log(delta*lambda*psi[N[j,s]]*psi[N[j-1,s-1],N[j,s
    ]]*psi[N[j+1,s-1],N[j,s]])
         death_rate[j,s]=log(delta*varsigma*psi[N[j,s]]*psi[N[j-1,s-1],N[j
    ,s]]*psi[N[j+1,s-1],N[j,s]])
184       }
```

```
      else if(j==J){
        birth_rate[j,s]=log(delta*lambda*psi[N[j,s]]*psi[N[j-1,s-1],N[j,s]])
        death_rate[j,s]=log(delta*varsigma*psi[N[j,s]]*psi[N[j-1,s-1],N[j,s]])
      }
    }
  }
  for(s in 2:S){
    for(j in 1:J){
      pois_process[j,s]=(birth_rate[j,s]*(birth_temp[j,s]))+(death_rate[j,s]*(death_temp[j,s]))-birth_rate[j,s]-death_rate[j,s]
    }
  }
  for(j in 2:J){
    temp_one[j,]=log(psi[N[j,1],N[j-1,1]])
  }
  pois_process[,1]=temp_one
  sum_pois_process=sum(pois_process)
  #w=list(sum_pois_process,pois_process) #returning a list because we use the sum of the poisson process across time for the posterior
  # we use the matrix of the poisson process (all positions across time) when simulating the latent, N
  #return(pois_process)
  return(sum_pois_process)
}

#################################
### PROPOSAL DISTRIBUTIONS ###
#################################
proposal_random_mu<-function(meanparameter,sdupdate_mu){
  promu=rtruncnorm(1,0,2*meanparameter,mean=meanparameter,sd=sdupdate_mu)
  return(promu)
}

```

```
proposal_random_Var<-function(meanparameter,sdupdate_scale){
  m=rnorm(1,mean=meanparameter,sd=sdupdate_var)
  return(m)
}

proposal_random_N<-function(scaled_rjmcmc,j,t){
  #generate random numbers from the poisson distribution with a dimension
     equal to the latent variable
  #we update each latent parameter at each time point in turn because
    otherwise there would be too many parameters to be updated at once and
     the acceptance probability too low
  # this loop works only if I increase the raw data by 0.01
  # otherwise, the raw data matrix contains too many 0s and the loop gets
     stuck as the random generation cannot get past these 0s
  repeat{
    Nrand=rpois(1,scaled_rjmcmc[j,t]+0.01)
    if (Nrand<=4 & Nrand>0) break
  }
  #cat(j,",",t,Nrand[j,t],"\n")
  return(Nrand)
}

proposal_random_lambda<-function(meanparameter,sdupdate_lambda){
  l=rtruncnorm(1,0,2*meanparameter,mean=meanparameter,sd=sdupdate_lambda)
    #because we have only 1 binding rate
  return(l)
}

proposal_random_mu<-function(meanparameter,sdupdate_mu){
  promu=rtruncnorm(1,0,2*meanparameter,mean=meanparameter,sd=sdupdate_mu)
  return(promu)
}

proposal_random_varsigma<-function(meanparameter,sdupdate_varsigma){
```

```
244   v=rtruncnorm(1,0,2*meanparameter,mean=meanparameter,sd=sdupdate_
        varsigma) #because we have only 1 detaching rate
      return(v)
246 }

248 proposal_random_zeta<-function(meanparameter,sdupdate_zeta){
      z=rtruncnorm(1,0,2*meanparameter,mean=meanparameter,sd=sdupdate_zeta)
250   return(z)
    }
252

    proposal_dens_mu<-function(previousparam,current_proposal,sdupdate_mu){
254   muden = dtruncnorm(current_proposal,0,2*previousparam,mean=
        previousparam,sd=sdupdate_mu)
      return(sum(log(muden)))
256 }

258 proposal_dens_Var<-function(previousparam, current_proposal,sdupdate_var)
        {
      Varden = dtruncnorm(current_proposal,0,2*previousparam,mean=
        previousparam,sd=sdupdate_var)
260   return(sum(log(Varden)))
    }
262

    proposal_dens_N<-function(mat,scaled_rjmcmc,j,t){
264   #We chose a proposal density for 'N=latent' to be a poisson density
        because it counts the number of binders, with lambda=previous step
      #and we generate this distribution based on the poisson process we have
         previously calculated
266   #how does lambda change? based on the previous iteration
      #we do not include the Nden matrix into the chain because we do not
        care about N's path
268   #for N we only keep the previous step matrix and the proposal matrix,
        since we do not save this variable in the Markov chain
      scaled_rjmcmc=scaled_rjmcmc+1
270   #if(scaled_rjmcmc[j,t]>0){
```

```
    Nden=dpois(mat[j,t],scaled_rjmcmc[j,t])
# }
 #else if(scaled_rjmcmc[j,t]==0){
  # Nden=dpois(mat[j,t],scaled_rjmcmc[j,t])+1
 #}
 #else {NULL}
 return(sum(log(Nden))) #this has to be kept as it is the proposal value
    of latent N
}

proposal_dens_lambda<-function(tempvar,previousparam,sdupdate_lambda){
 lambdaden=dtruncnorm(tempvar,0,2*previousparam,mean=previousparam,sd=
   sdupdate_lambda)
 return(log(lambdaden))
}

proposal_dens_varsigma<-function(tempvar,previousparam,sdupdate_varsigma)
   {
 varsigmaden=dtruncnorm(tempvar,0,2*previousparam,mean=previousparam,sd=
   sdupdate_varsigma)
 return(log(varsigmaden))
}

proposal_dens_zeta<-function(tempvar,previousparam,sdupdate_zeta){
 zetaden=dtruncnorm(tempvar,0,2*previousparam,mean=previousparam,sd=
   sdupdate_zeta)
 return(log(zetaden))
}

#################################
### POSTERIOR DISTRIBUTION ###
#################################
posterior <- function(S,J,parameter,psi_start,K,data, N,decay){
 return (likelihood(K,parameter,data,N,decay) + prior(parameter,K,alpha,
   beta)+poisson_process(S,J,N,parameter,psi_start,lambda,varsigma,delta
```

```
    =0.1))
}

##############################
### METROPOLIS HASTINGS ###
##############################
iterations=7000
metropolis_mcmc<-function(startvalue,parameter,iterations,psi_start,k,
    data){
  temp_N=scaled_rjmcmc+1 #because we use the same lambda for both the
    random generation of N as well as for the proposal density
  latent_temp=list()
  param=runif(MAXNO_COMPONENTS*MAXNO_COMPONENTS,0,MAXNO_COMPONENTS)
  param=matrix(param,byrow=MAXNO_COMPONENTS,ncol=MAXNO_COMPONENTS)
  chain=array(dim=c(iterations+1,MAXNO_COMPONENTS*2+3)) #array with nrow=
    iterations+1 and ncol=equal to the number of parameters
  chain[1,]=startvalue #defines the starting values of each column of the
     chain
  previousparameter = parameter
  current_parameter = parameter
  for(i in 2:iterations){
    ########## updating mu ###########
    previousmu=previousparameter[[1]]
    current_mu=previousmu
    for(n in 1:MAXNO_COMPONENTS){
      current_mu[n] = proposal_random_mu(previousmu[n],sdupdate_mu[n])
      current_parameter[[1]][n] = current_mu[n] #updating the nth value
    in 1st item of the list 'parameter'
      temp1=proposal_dens_mu(current_parameter[[1]][n],previousparameter
    [[1]][n],sdupdate_mu[n])
      temp2=proposal_dens_mu(previousparameter[[1]][n],current_parameter
    [[1]][n],sdupdate_mu[n])
      probability=posterior(S,J,current_parameter,psi_start,K,data,temp_N
    ,decay)-posterior(S,J,previousparameter,psi_start,K,data,temp_N,decay)
    +temp2-temp1
```

```
    if (log(runif(1)) < probability){

    }else{
      current_parameter[[1]][n]= previousparameter[[1]][n]
    }
    previousparameter = current_parameter
  }
  chain[i,1:MAXNO_COMPONENTS] = sort(current_parameter[[1]])
  cat(i, ",", "mu", chain[i,1:MAXNO_COMPONENTS],"\n")

  ############ updating sigma ################
  previousvar=previousparameter[[2]]
  current_sigma=previousvar
  for (m in 1:MAXNO_COMPONENTS){
    current_sigma[m] = proposal_random_Var(previousvar[m],sdupdate_var[
m])
    current_parameter[[2]][m] = current_sigma[m]
    temp1 = proposal_dens_Var(current_parameter[[2]][m],
previousparameter[[2]][m],sdupdate_var[m])
    temp2 = proposal_dens_Var(previousparameter[[2]][m],current_
parameter[[2]][m],sdupdate_var[m])
    probability=posterior(S,J,current_parameter,psi_start,K,data,temp_N
,decay)-posterior(S,J,previousparameter,psi_start,K,data,temp_N,decay)
+temp2-temp1
    if(log(runif(1)) < probability & (current_parameter[[2]][m]>=20) &
(current_parameter[[2]][m]<=750)){
    }
    else{
      current_parameter[[2]][m]= previousparameter[[2]][m]
    }
    previousparameter = current_parameter
  }
  chain[i,(MAXNO_COMPONENTS+1):(MAXNO_COMPONENTS*2)]=current_parameter
[[2]]
```

```
352     cat(i, ",", "sigma", chain[i,(MAXNO_COMPONENTS+1):(MAXNO_COMPONENTS*2)
        ],"\n")

354     ############## updating lambda ###################
        previouslambda=previousparameter[[3]]
356     current_lambda=previouslambda
        current_lambda=proposal_random_lambda(previouslambda,sdupdate_lambda)
358     current_parameter[[3]]=current_lambda
        temp1 = proposal_dens_lambda(current_parameter[[3]],previousparameter
        [[3]],sdupdate_lambda)
360     temp2 = proposal_dens_lambda(previousparameter[[3]],current_parameter
        [[3]],sdupdate_lambda)
        probability=posterior(S,J,current_parameter,psi_start,K,data,temp_N,
        decay)-posterior(S,J,previousparameter,psi_start,K,data,temp_N,decay)+
        temp2-temp1
362     if (log(runif(1)) < probability){
        }
364     else{
          current_parameter[[3]]=previousparameter[[3]]
366     }
        previousparameter=current_parameter
368     chain[i,MAXNO_COMPONENTS*2+1]=current_parameter[[3]]
        cat(i, ",","lambda",chain[i,MAXNO_COMPONENTS*2+1],"\n")
370

        ################ updating varsigma ###################
372     previousvarsigma=previousparameter[[4]]
        current_varsigma=previousvarsigma
374     current_varsigma=proposal_random_varsigma(previousvarsigma,sdupdate_
        varsigma)
        current_parameter[[4]]=current_varsigma
376     temp1 = proposal_dens_varsigma(current_parameter[[4]],
        previousparameter[[4]],sdupdate_varsigma)
        temp2 = proposal_dens_varsigma(previousparameter[[4]],current_
        parameter[[4]],sdupdate_varsigma)
```

```
 probability=posterior(S,J,current_parameter,psi_start,K,data,temp_N,
decay)-posterior(S,J,previousparameter,psi_start,K,data,temp_N,decay)+
temp2-temp1
 if (log(runif(1)) < probability){
 }
 else{
   current_parameter[[4]]=previousparameter[[4]]
 }
 previousparameter=current_parameter
 chain[i,MAXNO_COMPONENTS*2+2]=current_parameter[[4]]
 cat(i, ",","varsigma:", chain[i,MAXNO_COMPONENTS*2+2],"\n")

 ################## updating N ######################
 if (i==2){
   previousN=temp_N #previous step of the chain; have set it equal to
scaled_rjmcmc (lambda)
   current_N=previousN
 }
 previousN=temp_N
 current_N=previousN
 for(t in 1:S){
   for(j in 1:J){
     #created a separate chain just for the latent, so it does not get
 stored in the markov chain
      current_N[j,t]=proposal_random_N(scaled_rjmcmc,j,t)+1 #increased
the labelling by 1 because N cannot start from 0
      temp1=proposal_dens_N(current_N-1,scaled_rjmcmc,j,t)
      temp2=proposal_dens_N(previousN-1,scaled_rjmcmc,j,t)
      probability=posterior(S,J,previousparameter,psi_start,K,data,
current_N,decay)-posterior(S,J,previousparameter,psi_start,K,data,
previousN,decay)+temp2-temp1
      if (log(runif(1))<probability){
        temp_N[j,t]=current_N[j,t]-1
      }
      else{
```

```
406             temp_N[j,t]=previousN[j,t]
            }
408         }
      }
      cat(i, ",","latent:",temp_N,"\n")
      latent_temp[[i]]=temp_N

      ################ updating zeta ####################
      previouszeta=previousparameter[[5]]
      current_zeta=previouszeta
      current_zeta=proposal_random_zeta(previouszeta,sdupdate_zeta)
      current_parameter[[5]]=current_zeta
      temp1 = proposal_dens_zeta(current_parameter[[5]],previousparameter
     [[5]],sdupdate_zeta)
      temp2 = proposal_dens_zeta(previousparameter[[5]],current_parameter
     [[5]],sdupdate_zeta)
      probability=posterior(S,J,current_parameter,psi_start,K,data,temp_N,
     decay)-posterior(S,J,previousparameter,psi_start,K,data,temp_N,decay)+
     temp2-temp1
      if (log(runif(1)) < probability){
      }
      else{
        current_parameter[[5]]=previousparameter[[5]]
      }
      previousparameter=current_parameter
      chain[i,MAXNO_COMPONENTS*2+3]=current_parameter[[5]]
      cat(i, ",","zeta:",chain[i,MAXNO_COMPONENTS*2+3],"\n")
    }
    output=list("the chain"=chain,"latent N"=latent_temp)
    return(output)
}
algorithm_4binders=metropolis_mcmc(startvalue,parameter,iterations,psi_
    start,k,data)
save(algorithm_4binders,file="MCMC_12dec_4binders.Rdata")
```

D.2.1 Numerical analysis of the hidden Markov model

```
##############################
###### LOAD LIBRARIES ######
##############################
library(MHadaptive)
library(pheatmap)
library(RColorBrewer)
library(miscF)
library(coda)

##############################
###### IMPORT RESULTS ######
##############################
# loading the results from the M-H algorithm
load("/Users/mdmiha/Documents/PhD Biostats/Sept20-Writing the R code /
    Ceres_results_Sep_20/Long run/MCMC_6jan_4binders.Rdata")
chain_4binders=algorithm_4binders

combined_chains=chain_4binders[["the chain"]]
latent_N=chain_4binders[[2]]

##############################
#### THINNING THE CHAIN ####
##############################
n=2 #where "n" is the thinning factor
mu_1 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,1][seq(1, length(
    combined_chains[,1]), by = n)]))
mu_2 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,2][seq(1, length(
    combined_chains[,1]), by = n)]))
mu_3 <-mcmc(as.mcmc(chain_4binders[["the chain"]][,3][seq(1, length(
    combined_chains[,1]), by = n)]))
mu_4 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,4][seq(1, length(
    combined_chains[,1]), by = n)]))
```

```
mu_5 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,5][seq(1, length(
    combined_chains[,1]), by = n)]))
var_1 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,6][seq(1, length(
    combined_chains[,1]), by = n)]))
var_2 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,7][seq(1, length(
    combined_chains[,1]), by = n)]))
var_3 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,8][seq(1, length(
    combined_chains[,1]), by = n)]))
var_4 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,9][seq(1, length(
    combined_chains[,1]), by = n)]))
var_5 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,10][seq(1, length(
    combined_chains[,1]), by = n)]))
lambda_1 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,11][seq(1, length
    (combined_chains[,1]), by = n)]))
varsigma_1 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,12][seq(1,
    length(combined_chains[,1]), by = n)]))
zeta_1 <- mcmc(as.mcmc(chain_4binders[["the chain"]][,13][seq(1, length(
    combined_chains[,1]), by = n)]))


########################
##### TRACE PLOTS ######
########################
traceplot(mu_1,type="l",smooth = FALSE,ylab="  1 ",xlab="Iterations")
traceplot(mu_2,type="l",smooth = FALSE,ylab="  2 ",xlab="Iterations")
traceplot(mu_3,type="l",smooth = FALSE,ylab="  3 ",xlab="Iterations")
traceplot(mu_4,type="l",smooth = FALSE,ylab="  4 ",xlab="Iterations")
traceplot(mu_5,type="l",smooth = FALSE,ylab="  5 ",xlab="Iterations")
traceplot(var_1,type="l",smooth = FALSE,ylab="SD1",xlab="Iterations")
traceplot(var_2,type="l",smooth = FALSE,ylab="SD2",xlab="Iterations")
traceplot(var_3,type="l",smooth = FALSE,ylab="SD3",xlab="Iterations")
traceplot(var_4,type="l",smooth = FALSE,ylab="SD4",xlab="Iterations")
traceplot(var_5,type="l",smooth = FALSE,ylab="SD5",xlab="Iterations")
traceplot(lambda_1,type="l",smooth = FALSE,ylab="   ",xlab="Iterations")
traceplot(varsigma_1,type="l",smooth = FALSE,ylab="   ",xlab="Iterations")
traceplot(zeta_1,type="l",smooth = FALSE,ylab="   ",xlab="Iterations")
```

```
54
##############################################
56 #### AUTOCORRELATION AND DENSITY PLOTS######
##############################################
58 par(mfrow=c(2,2))
mu_1_na<-mu_1[!is.na(mu_1)] # make sure to remove all NAs first
60 mu_2_na<-mu_2[!is.na(mu_2)] # make sure to remove all NAs first
mu_3_na<-mu_3[!is.na(mu_3)] # make sure to remove all NAs first
62 mu_4_na<-mu_4[!is.na(mu_4)] # make sure to remove all NAs first
mu_5_na<-mu_5[!is.na(mu_5)] # make sure to remove all NAs first
64
autocorr.plot(mu_1_na, lag.max=500, auto.layout = FALSE,main="  1 ")
66 plot(density(mu_1_na))
autocorr.plot(mu_2_na, lag.max=500, auto.layout = FALSE,main="  2 ")
68 plot(density(mu_2_na))
autocorr.plot(mu_3_na, lag.max=500, auto.layout = FALSE,main="  3 ")
70 plot(density(mu_3_na))
autocorr.plot(mu_4_na, lag.max=500, auto.layout = FALSE,main="  4 ")
72 plot(density(mu_4_na))
autocorr.plot(mu_5_na, lag.max=500, auto.layout = FALSE,main="  5 ")
74 plot(density(mu_5_na))

76 var_1_na<-var_1[!is.na(var_1)] # make sure to remove all NAs first
var_2_na<-var_2[!is.na(var_2)] # make sure to remove all NAs first
78 var_3_na<-var_3[!is.na(var_3)] # make sure to remove all NAs first
var_4_na<-var_4[!is.na(var_4)] # make sure to remove all NAs first
80 var_5_na<-var_5[!is.na(var_5)] # make sure to remove all NAs first

82 autocorr.plot(var_1_na, lag.max=500, auto.layout = FALSE,main="SD1")
plot(density(var_1_na))
84 autocorr.plot(var_2_na, lag.max=500, auto.layout = FALSE,main="SD2")
plot(density(var_2_na))
86 autocorr.plot(var_3_na, lag.max=500, auto.layout = FALSE,main="SD3")
plot(density(var_3_na))
88 autocorr.plot(var_4_na, lag.max=500, auto.layout = FALSE,main="SD4")
```

```
plot(density(var_4_na))
autocorr.plot(var_5_na, lag.max=200, auto.layout = FALSE,main="SD5")
plot(density(var_5_na))

lambda_1_na<-lambda_1[!is.na(lambda_1)] # make sure to remove all NAs
    first
autocorr.plot(lambda_1_na, lag.max=200, auto.layout = FALSE,main="  ")
plot(density(lambda_1_na))
varsigma_1_na<-varsigma_1[!is.na(varsigma_1)] # make sure to remove all
    NAs first
autocorr.plot(varsigma_1_na, lag.max=200, auto.layout = FALSE,main="  ")
plot(density(varsigma_1_na))
zeta_1_na<-zeta_1[!is.na(zeta_1)] # make sure to remove all NAs first
autocorr.plot(zeta_1_na, lag.max=200, auto.layout = FALSE,main="  ")
plot(density(zeta_1_na))

#####################################
######ACCEPTANCE PROBABILITY#######
#####################################
burnin=1000 #first 1000 iterations which we may want to discard
combined_chains=chain_4binders[["the chain"]][-(1:burnin),]
latent_N=chain_4binders[[2]][-c(1:burnin)]

MAXNO_COMPONENTS=5
mm=MAXNO_COMPONENTS*2+3
acceptance_rate=rep(0,MAXNO_COMPONENTS*2+3)
for (a in 1:mm){
  acceptance_rate[a]=1-mean(duplicated(combined_chains[,a]))
}
global_acceptance=mean(acceptance_rate)

#############################################
###### PARAMETER MEANS ####################
#############################################
chain_4binders <- as.data.frame(combined_chains)
```

```
param_means=rep(0,MAXNO_COMPONENTS*2+3)
t=matrix(0,J,S)
t_mean=matrix(0,J,S)
for (b in 1:(MAXNO_COMPONENTS*2+3)) {
  param_means[b]=mean(combined_chains[,b],na.rm=TRUE)
}

################################
###### LATENT OUTPUT #######
################################
# remove the blank elements in our list by running code that subsets the
    list by elements that only have a length greater than zero
latent_N <- latent_N[lapply(latent_N,length)>0]
latent_N=lapply(latent_N, matrix, nrow=1) # convert each element to a
    matrix format

tempor=matrix(0,J,S)
tempor_mean=matrix(0,J,S)
z=length(latent_N)
tt=rep(0,J*S)
for (a in 2:z) {
  tt=as.vector(latent_N[[a]])
  tempor=matrix(tt,J,S)
  tempor_mean=tempor+tempor_mean
}

# Average of the latent variable
latent_avr_4binders=round(tempor_mean/(z-1)-1) #because we added 1 to the
    matrix earlier on in the model
cols = colorRampPalette(c("black", "white"))(30)
pheatmap(latent_avr_4binders, cluster_rows = FALSE, cluster_cols = FALSE,
    color=cols, cellwidth = 10, cellheight = 7,main = 'Heatmap
    representing up to 4 components') # distances 0 to 3 are red, 3 to 9
    black
```

BIBLIOGRAPHY

[1] Aitkin, M. (1980). Mixture applications of the em algorithm in glim. *COMPSTAT 1980*, pages 537–541.

[2] Albert, J. H. and Chib, S. (1993). Bayes inference via gibbs sampling of autoregressive time series subject to markov mean and variance shifts. *Journal of Business & Economic Statistics*, 11(1):1–15.

[3] Andrew Gelman, D. B. R. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472.

[4] Archer, G. and Titterington, D. (2002). Parameter estimation for hidden markov chains. *Journal of Statistical Planning and Inference*, 108(1-2):365–390.

[5] Besag, J. (1989). Digital image processing: Towards bayesian image analysis. *Journal of Applied statistics*, 16(3):395–407.

[6] Bouguila, N. and Elguebaly, T. (2012). A fully bayesian model based on reversible jump mcmc and finite beta mixtures for clustering. *Expert Systems with Applications*, 39(5):5946–5959.

[7] Bulla, J. (2006). Application of hidden markov models and hidden semi-markov models to financial time series.

[8] Burnham, K. P. and Anderson, D. R. (2004). Multimodel inference: understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304.

[9] Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

[10] Chaudhary, S. (2014). Implementation and performance analysis of markov random field. *International Journal of Advancements in Research Technology*, 3:37–41.

[11] Chen, J. (1995). Optimal rate of convergence for finite mixture models. *The Annals of Statistics*, pages 221–233.

[12] Chen, M.-H., Shao, Q.-M., and Ibrahim, J. G. (2012). *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media.

[13] Chen, Z., Barbieri, R., and Brown, E. N. (2010). State space modeling of neural spike train and behavioral data. In *Statistical signal processing for neuroscience and neurotechnology*, pages 175–218. Elsevier.

[14] Christian P. Robert, Tobias Rydn, D. M. T. (2000). Bayesian inference in hidden markov models through the reversible jump markov chain monte carlo method. *Journal of the Royal Statistical Society Series B*, 62(1)(1):57–75.

[15] Congdon, P. (2006). *Bayesian Statistical Modelling*. John Wiley & Sons, 2nd edition.

[16] deLeeuw, J. (1992). Introduction to akaike (1973) information theory and an extension of the maximum likelihood principle. In *Breakthroughs in statistics*, pages 599–609. Springer.

[17] Dellaportas, P. and Papageorgiou, I. (2006). Multivariate mixtures of normals with unknown number of components. *Statistics and Computing*, 16(1):57–68.

[18] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[19] Desai, R., Geeves, M. A., and Kad, N. M. (2015). Using fluorescent myosin to directly visualize cooperative activation of thin filaments. *Journal of Biological Chemistry*, 290(4):1915–1925.

[20] Diebolt, J. and Robert, C. P. (1994). Estimation of finite mixture distributions through bayesian sampling. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(2):363–375.

[21] Eddy, S. R. (1996). Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.

[22] Feng, D. (2018). Miscellanous functions. *R News*.

[23] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620.

[24] Frühwirth-Schnatter, S. (2006). *Finite mixture and Markov switching models*. Springer Science & Business Media.

[25] Gamerman, D. and Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC.

[26] Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

[27] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

[28] Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2008). *Bayesian data analysis*. Chapman & Hall/CRC Boca Raton, FL, USA, 2nd edition.

[29] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

[30] Ghahramani, Z. (2001a). An introduction to hidden markov models and bayesian networks. In *Hidden Markov models: applications in computer vision*, pages 9–41. World Scientific.

[31] Ghahramani, Z. (2001b). An introduction to hidden markov models and bayesian networks. In *Hidden Markov models: applications in computer vision*, pages 9–41. World Scientific.

[32] Green, P. J. (1995a). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

[33] Green, P. J. (1995b). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

[34] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

[35] Hespanhol, L., Vallio, C. S., Costa, L. M., and Saragiotto, B. T. (2019). Understanding and interpreting confidence and credible intervals around effect estimates. *Brazilian journal of physical therapy*, 23(4):290–301.

[36] Huxley, H. and Hanson, J. (1954). Changes in the cross-striations of muscle during contraction and stretch and their structural interpretation. *Nature*, 173:149–152.

[37] Kad, N. M., Kim, S., Warshaw, D. M., VanBuren, P., and Baker, J. E. (2005). Single-myosin crossbridge interactions with actin filaments regulated by troponin-tropomyosin. *Proceedings of the National Academy of Sciences of the United States of America*, 102(47):16990–16995.

[38] Leonard E. Baum, M. K. (1965). Convergence rates in the law of large numbers. *Transactions of the American Mathematical Society*, 120(1):108–123.

[39] Leroux, B. G. and Puterman, M. L. (1992). Maximum-penalized-likelihood estimation for independent and markov-dependent mixture models. *Biometrics*, pages 545–558.

[40] McKillop, D. and Geeves, M. A. (1993). Regulation of the interaction between actin and myosin subfragment 1: evidence for three states of the thin filament. *Biophysical Journal*, 65:693–701.

[41] McLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. Wiley Series, Canada.

[42] McLachlan, G. J. (1982). 9 the classification and mixture maximum likelihood approaches to cluster analysis. *Handbook of statistics*, 2:199–208.

[43] McLachlan, G. J. (1987). On bootstrapping the likelihood ratio test stastistic for the number of components in a normal mixture. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 36(3):318–324.

[44] McLachlan, G. J. and Basford, K. E. (1988). *Mixture models: Inference and applications to clustering*, volume 84. M. Dekker New York.

[45] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

[46] Mijailovich, S. M., Kayser-Herold, O., Li, X., Griffiths, H., and Geeves, M. A. (2012). Cooperative regulation of myosin-s1 binding to actin filaments by a continuous flexible tm–tn chain. *European Biophysics Journal*, 41(12):1015–1032.

[47] Neal, R. (1991). Bayesian mixture modelling by monte carlo simulation. Technical report, Technical Report CRG–TR–91–2, Computer Science, Univ. of Toronto.

[48] Nguyen, X. (2013). Convergence of latent mixing measures in finite and infinite mixture models. *The Annals of Statistics*, 41(1):370–400.

[49] Niederer, S. A., Campbell, K. S., and Campbell, S. G. (2019). A short history of the development of mathematical models of cardiac mechanics. *Journal of molecular and cellular cardiology*, 127:11–19.

[50] OpenStax. Anatomy and physiology. `https://cnx.org/contents/FPtK1zmh@8.25:fEI3C8Ot@10/Preface`. Accessed: 2016-09-03.

[51] Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110.

[52] Pietrzykowski, M. and Sałabun, W. (2014). Applications of hidden markov model: state-of-the-art. *Int. J. Comput. Technol. Appl*, 5:1384–1391.

[53] Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11.

[54] Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252.

[55] Rao, C. R. (1948). The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203.

[56] Richardson, S. and Green, P. J. (1997). On bayesian analysis of mixtures with an unknown number of componenets. *Journal of the Royal Statistical Society. Series B*, 4(59):731–792.

[57] Robert, C. and Casella, G. (2010). *Monte Carlo statistical methods*. Springer Science & Business Media, 2nd edition.

[58] Robert, C. P. and Titterington, D. (1998). Reparameterization strategies for hidden markov models and bayesian approaches to maximum likelihood estimation. *Statistics and Computing*, 8(2):145–158.

[59] Roeder, K. and Wasserman, L. (1997). Practical bayesian density estimation using mixtures of normals. *Journal of the American Statistical Association*, 92(439):894–902.

[60] Ross, S. M., Kelly, J. J., Sullivan, R. J., Perry, W. J., Mercer, D., Davis, R. M., Washburn, T. D., Sager, E. V., Boyce, J. B., and Bristow, V. L. (1996). *Stochastic processes*, volume 2. Wiley New York.

[61] Rydn, T. and Titterington, D. M. (1998). Computational bayesian analysis of hidden markov models. *Journal of Computational and Graphical Statistics*, 7(2):194–211.

[62] Silverthorn, D. U., Ober, W. C., Garrison, C. W., Silverthorn, A. C., and Johnson, B. R. (2010). *Human physiology: an integrated approach*. Pearson/Benjamin Cummings San Francisco, CA, USA:, 5th edition.

[63] Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and Van Der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the royal statistical society: Series b (statistical methodology)*, 64(4):583–639.

[64] Spudich, J. A. (2014). Hypertrophic and dilated cardiomyopathy: four decades of basic research on muscle lead to potential therapeutic approaches to these devastating genetic diseases. *Biophysical journal*, 106(6):1236–1249.

[65] Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of statistics 28*, pages 40–74.

[66] Stephens, M. et al. (2000). Bayesian analysis of mixture models with an unknown number of componentsan alternative to reversible jump methods. *the Annals of Statistics*, 28(1):40–74.

[67] Stracher, A. (2013). *Muscle and nonmuscle motility*, volume 1. Academic Press, Inc., London.

[68] Tobacman, L. C. (1996). Thin filament mediated regulation of cardiac contraction. *Annual Reviews Physiology*, 58:447–481.

[69] Wagenmakers, E.-J. and Farrell, S. (2004). Aic model selection using akaike weights. *Psychonomic bulletin & review*, 11(1):192–196.

[70] Walcott, S. and Sun, S. X. (2009). Hysteresis in cross-bridge models of muscle. *Phys. Chem. Chem. Phys."*, 11:4871–4881.

[71] Weldon, W. (1892). Certain correlated variations in crangon vulgaris. 51:2–21.

[72] Weldon, W. (1893). On certain correlated variations in carcinus moenas. 54:318–329.

[73] Zucchini, W., MacDonald, I. L., and Langrock, R. (2017). *Hidden Markov models for time series: an introduction using R*. Chapman and Hall/CRC.

[1]