An analysis of intrinsically disordered proteins using hidden Markov models and experimental design of stochastic kinetic models

Nina Wilkinson

Thesis submitted for the degree of
Doctor of Philosophy

School of Mathematics & Statistics
Institute for Cell & Molecular Biosciences
Newcastle University
Newcastle upon Tyne
United Kingdom

September 2014

Acknowledgements

To begin, I would like to express my gratitude to my supervisors Richard Boys and Colin Gillespie for their patience, support and encouragement throughout the last three years. In addition I would like to thank my supervisors Doug Gray and Viktor Korolchuk for their biological expertise and Doug Gray's laboratory (University of Ottawa) who completed all laboratory experiments included in this thesis.

I would also like to thank my parents who always have more confidence in me than I have in myself and have provided invaluable support during my education. I would like to thank Kevin Wilson for putting up with me during a stressful period in my PhD and also for proof reading my thesis.

I would like to express how grateful I am to those who have helped me to solve computational issues, in particular Anthony Youd, Michael Beaty, George Stagg and Keith Newman. I am also thankful to Holly Ainsworth for her discussions on Gaussian Processes and her help and advice whenever it was needed. My time as a PhD student would not have been that same without my office mates and the friends I have made and I would like to thank them for making my PhD an enjoyable experience.

Finally I would like to acknowledge the financial support provided by the Biotechnology and Biological Sciences Research Council.

Abstract

An intrinsically disordered protein (IDP) is a protein without a stable secondary or tertiary structure and just over one third of human proteins can be described as IDPs. There has been shown to be a link between neurodegenerative diseases, cancer and protein misfolding, with many of these misfolded proteins being intrinsically disordered. These IDPs may be cytotoxic by interacting and contributing to the aggregation process, which is why cells need to regulate these proteins carefully. Research has shown that hydrophobicity and charge may be important in determining if the amino acid sequence has unstructured areas. We study the sequence structure by first recoding amino acid sequences according to their hydrophobicity and charge and then fitting a hidden Markov model using Markov chain Monte Carlo methods to analyse the sequence structure and use a power posterior analysis to determine the number of distinct transition structures. The results show there to be distinct segment types within the amino acid sequences of the FET proteins which may have biological importance. The location of these segments can be used to guide laboratory work which tests the biological significance of these segment types within cells. One particular segment found in the FET proteins has been linked to oncogenic fusion proteins and experimental analysis has shown a link between this segment and oncogenic activity.

When conducting an experiment, an experimenter needs to determine when and under what conditions they should take measurements. Often the choice of optimal design is made with respect to some statistical criteria. The aim of this work is to determine, for a stochastic kinetic model, the optimal location of the timepoints at which observations are taken. Commonly the statistical criteria involves maximising a utility function over the prior predictive distribution of possible experimental outcomes. Current methodologies for experimental design for models with intractable likelihoods are very computationally expensive as, within the iterative search for the optimal design, the calculation of the utility function requires the determination of the parameter posterior distribution at each iteration. We show how to use delta methods and a Gaussian process as an emulator for the utility to reduce the computational cost and illustrate their application for the simple death process and the Lotka–Volterra model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Using statistical models to guide biological experiments is important as optimising the experimental design and analysis can save money and time. In this thesis we present methodologies to guide two different types of experiment. The first is a sequence analysis using hidden Markov models to identify segments of protein which may have biological relevance. This can then be tested experimentally by biologists. The second aids in the decisions of when to observe a system, maximising the information resulting from an experiment.

This thesis consists of two components; the first investigates intrinsically disordered proteins (IDPs) using hidden Markov models in order to determine biologically relevant segments. The second part considers Bayesian experimental design, in the context of stochastic kinetic models with intractable likelihoods (Wilkinson, 2011).

The aims of this thesis are:

- Part one: Using hidden Markov models analyse IDP sequences to find biologically relevant segments. This can then be tested experimentally in the laboratory.
- Part two: Estimate the optimal design in the context of stochastic kinetic models using a Gaussian process to approximate the utility.

1.1 Overview of thesis

As it is important to all methods developed, we begin with a general introduction to Bayesian inference. In part one we have the advantage of being able to incorporate our prior knowledge about the sequence structure into the analysis and for part two Bayesian techniques are useful when the likelihood is intractable. In both parts of the thesis we use Markov–chain Monte Carlo (MCMC) algorithms to learn about parameters and take advantage of likelihood–free algorithms in Part two.

Part one begins in Chapter 3 by introducing the biological problem, and discussing how intrinsically disordered proteins (IDPs) are linked to diseases including neurodegenerative diseases and cancer. We describe the properties of IDPs, and why we choose the properties hydrophobicity and charge in our modelling. We introduce the proteins of interest and describe why we are interested in analysing the amino acid sequences of these proteins. In addition, we consider the existing tools used to predict the structure of IDPs.

Chapter 4 introduces the statistical techniques that we use to analyse amino acid sequences. We use a hidden Markov model and simplify the sequences using two properties of IDPs, hydrophobicity and charge. MCMC techniques are used to analyse the sequence structure. We discuss how to determine the number of distinct transition structures.

Chapter 5 demonstrates the techniques described in Chapter 4. We analyse the sequences of some intrinsically disordered proteins and describe how the results can be used to guide laboratory experiments. We describe the laboratory experiments that can be used to show the significance of the segment types discovered in the proteins. The results of these experiments and the biological relevance of one of the segments found in the Bayesian analysis are discussed.

We complete Part one of the thesis in Chapter 6 by discussing the advantages and disadvantages of the work and possible future directions.

In Chapter 7 we introduce stochastic kinetic models and discuss simulation algorithms. We focus on two particular models, namely the death model and the Lotka–Volterra (LV) model.

Chapter 8 introduces Bayesian experimental design, its motivation and a solution using utility functions. We illustrate recent work in this area by Drovandi and Pettitt (2013) which uses an ABC method to find optimal designs in models with intractable likelihoods. We illustrate this method for the death model and discuss a drawback of their solution, namely that it can produce inaccurate results due to ignoring variability in the ABC datasets.

Chapter 9 introduces Gaussian processes. The intention here is to find a fast and accurate Gaussian process approximation to the utility function and thereby produce a method which determines optimal designs both quickly and via an algorithm which scales to larger models better than the ABC method. The chapter begins by describing how to fit a Gaussian process to data and introduces some diagnostic tools that can be used to determine whether the Gaussian process is a good fit. The Gaussian process approximation we seek has timepoints and model parameters as inputs and the outputs are a utility. We investigate methods to select training data (used for fitting the Gaussian process) when the inputs are ordered timepoints. We show how these fitted Gaussian processes can be used in Bayesian experimental design and we implement the methods for the death model.

The main aim of part two of the thesis is to show how Gaussian processes can be used

in Bayesian experimental design for models with intractable likelihoods, and we illustrate this by focussing on the Lotka–Volterra model. We discuss techniques to reduce the design space and then fit a Gaussian process to this reduced design space, where the utility is calculated using a linear noise approximation to the Lotka–Volterra model. We then determine the optimal design using this Gaussian process via MCMC methods described in Müller (1999). Finally we discuss our results, provide some conclusions and suggest possible future work in Chapter 11.

Chapter 2

Introduction to Bayesian inference

In both parts of this thesis we are approaching the statistical analysis within a Bayesian framework. In the first part we use Bayesian methods to find segments in intrinsically disordered proteins and in the second part we use Bayesian methods to find optimal designs for stochastic kinetic models, which will involve the need to perform parameter inference in order to estimate a utility function. In this chapter we introduce the concept of Bayesian statistics and introduce methods for parameter inference.

2.1 Bayesian Inference

Suppose we have a model which describes how likely different datasets $\boldsymbol{y}$ are to occur. The model, written $\pi(\boldsymbol{y}|\boldsymbol{\theta})$, is either a probability density function (if $\boldsymbol{y}$ is continuous), a probability function (if $\boldsymbol{y}$ is discrete) or a mixture (if $\boldsymbol{y}$ has both continuous and discrete components). Throughout this thesis we will refer to $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ as a density.

Given observed data $\boldsymbol{y}$, the likelihood function for the parameters $\boldsymbol{\theta}$ is $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ but now this is regarded as a function in $\boldsymbol{\theta}$. Suppose our prior knowledge of $\boldsymbol{\theta}$ is described via a prior distribution, $\pi(\boldsymbol{\theta})$. Then, using Bayes Theorem, we can incorporate both pieces of information to obtain the posterior distribution

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{2.1}$$

As the integral in the denominator is just a normalising constant that is not a function of $\boldsymbol{\theta}$, we can write the posterior distribution as being proportional to the prior times the likelihood, that is

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta}).$$

2.1.1 Prior elicitation

The prior distribution, with density $\pi(\boldsymbol{\theta})$, should encapsulate all information to hand about the parameters $\boldsymbol{\theta}$. Optimally this information would be elicited from an expert with considerable knowledge about the problem/context. In such situations, if there is only a small amount of data available the prior distribution dominates the posterior distribution, with $\pi(\boldsymbol{\theta}|\boldsymbol{y}) \sim \pi(\boldsymbol{\theta})$. However, one common problem with obtaining *substantial prior knowledge* is that the prior distribution is often complex and results in an intractable posterior distribution.

In situations when little information is available about $\boldsymbol{\theta}$, a practical approach is to use a prior distribution that is conjugate to the likelihood in order to make the mathematics simpler and to give the prior a large variance. This means that most of the information about $\boldsymbol{\theta}$ is from the data, and so as a function of $\boldsymbol{\theta}$ the posterior density has a very similar profile to that of the likelihood, $\pi(\boldsymbol{\theta}|\boldsymbol{y}) \sim \pi(\boldsymbol{y}|\boldsymbol{\theta})$.

2.2 Markov chain Monte Carlo (MCMC)

Apart from trivial cases, finding the normalising constant is difficult as we need to evaluate the integral given in the denominator of Equation (2.1) which can have multiple dimensions and not produce a density function available in standard form. MCMC algorithms are standard ways to sample the density of interest, $\pi(\boldsymbol{\theta}|\boldsymbol{y})$, as the algorithm converges to this distribution.

2.2.1 Gibbs sampling

We can use the Gibbs algorithm to sample from a multivariate density of interest by simulating from the full conditional densities of the parameters (Turchin, 1971; Geman and Geman, 1984). We define $\pi(\boldsymbol{\theta})$ to be the density of interest where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$ and the full conditional densities to be $\pi(\theta_i|\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p)$ for $i = 1, \ldots, p$. The algorithm works by alternatively sampling from the conditional density of each parameter given the previous samples of the other parameters; see Algorithm 1. The Gibbs algorithm is useful when the conditional densities are of a standard form and easy to sample from or when it is not possible to sample from the marginal distribution (Gelman et al., 2013). This algorithm produces a homogeneous Markov chain, as when we simulate a parameter value it only depends on the previous parameter values and $\pi(\boldsymbol{\theta})$ is the stationary distribution of the chain.

Algorithm 1 Gibbs Sampling algorithm.

1. Set $j = 1$. Choose values of $\boldsymbol{\theta}^0 = (\theta_1^0, \ldots, \theta_p^0)'$ to initialise the chain.

2. Find a new value of $\boldsymbol{\theta}^j$ from $\boldsymbol{\theta}^{j-1}$ by successively simulating from the conditional densities:

$$\begin{aligned}
&\theta_1^j \sim \pi(\theta_1|\theta_2^{j-1}, \ldots, \theta_d^{j-1}) \\
&\theta_2^j \sim \pi(\theta_2|\theta_1^j, \theta_3^{j-1}, \ldots, \theta_d^{j-1}) \\
&\vdots \\
&\theta_d^j \sim \pi(\theta_d|\theta_1^j, \ldots, \theta_{d-1}^j).
\end{aligned}$$

3. Let $j = j + 1$ go back to step 2.

2.2.2 Metropolis–Hastings algorithm

The Metropolis–Hastings (MH) algorithm, given in Algorithm 2, was first introduced by Metropolis et al. (1953) and then generalised by Hastings (1970). We can use this method to obtain posterior samples from $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. It works by obtaining a stationary distribution equivalent to the distribution of interest. The algorithm starts by initialising the parameters, $\boldsymbol{\theta}$. Then at each iteration we propose a new value of the parameters, $\boldsymbol{\theta}^*$, from the proposal distribution, $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})$. This distribution will be chosen so that it is easy to simulate values from the distribution. The proposal, $\boldsymbol{\theta}^*$ is then compared to the previous $\boldsymbol{\theta}$ or the initial $\boldsymbol{\theta}$ when $i = 1$. This is done by calculating the acceptance probability, $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta})$ which depends on $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})$ and $\pi(\cdot|\boldsymbol{y})$. The proposal is either accepted or rejected according to this probability. This means that the chain either moves to $\boldsymbol{\theta}^*$ or stays at $\boldsymbol{\theta}$. The main advantage of using MCMC is that due to the acceptance probability being a ratio of $\pi(\cdot|\boldsymbol{y})$, the normalising constants cancel, so we are not required to calculate them and only need to know the target distribution up to the constant of proportionality.

The proposal distribution

The proposal distribution, $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})$, proposes the next value of $\boldsymbol{\theta}$ that the Markov chain might take. The simplest proposal distribution to use is a symmetric proposal which cancels in the acceptance probability as $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$. Therefore the acceptance probability simplifies to

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta})}\right\}.$$

Having a symmetric proposal distribution means that a proposal, $\boldsymbol{\theta}^*$, will always be accepted if it moves the chain to an area of higher density. A proposal distribution can

Algorithm 2 Metropolis–Hastings algorithm.

1. Set $j = 1$. Choose values of $\boldsymbol{\theta}^0 = (\theta_1^0, \ldots, \theta_p^0)'$ to initialise the chain.
2. Sample a proposed value of $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta}^{j-1})$ using the proposal distribution q.
3. Calculate the acceptance probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{j-1})$ of the proposed value, $\boldsymbol{\theta}^*$

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)q(\boldsymbol{\theta}^{j-1}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{j-1})\pi(\boldsymbol{y}|\boldsymbol{\theta}^{j-1})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{j-1})}\right\}.$$

4. Let $\boldsymbol{\theta}^j = \boldsymbol{\theta}^*$ with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta})$, otherwise $\boldsymbol{\theta}^j = \boldsymbol{\theta}^{j-1}$.
5. Let $j = j + 1$ go back to step 2.

also be chosen which is independent of the current position of the chain: $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = g(\boldsymbol{\theta}^*)$ for some density, g. This gives the acceptance probability as

$$\begin{aligned}\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)g(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta})g(\boldsymbol{\theta}^*)}\right\} \\ &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)/\pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta})}{g(\boldsymbol{\theta}^*)/g(\boldsymbol{\theta})}\right\}.\end{aligned}$$

Here the acceptance probability is a ratio of $\pi(\cdot)\pi(y|\cdot)$ to $g(\cdot)$. Therefore if $\pi(\cdot)\pi(y|\cdot)$ is similar to $g(\cdot)$ then the acceptance probability will be high. Another choice of proposal distribution for $\boldsymbol{\theta}$ is to use a random walk,

$$\boldsymbol{\theta}^* = \boldsymbol{\theta} + \omega,$$

where ω are independent and identically distributed random variables called innovations. Often a zero mean Gaussian distribution or a symmetric uniform distribution is used for ω. This case of the Metropolis–Hastings algorithm is also called a random walk sampler.

Tuning a random walk proposal

In MCMC the term mixing is used to describe how well a chain moves around the parameter space and as a result of this the length of time until convergence of the chain. Mixing depends on the distribution of ω and therefore the parameters of the distribution of ω. If we choose ω to be a multivariate zero mean Gaussian distribution

$$\omega \sim N_p(0, V),$$

then it is V that controls the mixing of the chain. If V is too big then the distribution will not move much due to large moves being proposed to areas of low density. Thus many of proposed moves are rejected. If V is chosen to be too small then small moves are proposed which are often accepted but the space is not explored very fast or efficiently. Efficient exploring of the space can be done by allowing for correlation in $\boldsymbol{\theta}$, which results in V having non-zero covariance values. If ω has a Gaussian distribution, it has been shown that 0.234 is the optimal acceptance probability (Roberts et al., 1997; Roberts and Rosenthal, 2001). A common technique to tune a random walk is to use

$$V = \frac{2.38^2 \Sigma_{\Pi}}{p},$$

where Σ_{Π} is the variance–covariance matrix of the target distribution Π, and p is the number of parameters, $\boldsymbol{\theta}$. In practice, Σ_{Π} is unknown but it can be approximated using trial runs of the MCMC scheme (Roberts and Rosenthal, 2001; Roberts et al., 1997).

2.2.3 Analysing MCMC output

Once we have output from an MCMC scheme, it is important to check for convergence of the chain. This can be done in an informal manner by plotting trace plots and checking that the Markov chains look like they have converged. The samples that occur before the chain has converged should be removed, this is called the burn–in. Autocorrelation between the samples should also be checked.

There are more formal ways of checking for convergence. Gelman and Rubin (1992) check that chains initialised in different places converge to the same distribution. Heidelberger and Welch (1983); Geweke (1991) and Raftery and Lewis (1992) suggest a technique to decide on the burn–in length, and the appropriate amount of thinning that is needed dependent on how accurate we choose the posterior samples to be.

2.2.4 Likelihood free MCMC

If the likelihood function $\pi(\boldsymbol{y}|\boldsymbol{\theta})$, is analytically intractable or time consuming computationally to evaluate, then we can use likelihood free methods. The basic likelihood free algorithm starts by simulating a proposed parameter $\boldsymbol{\theta}^*$ then simulating a dataset from the model, $\boldsymbol{x} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta}^*)$. Next the simulated data $\boldsymbol{x}$ is compared to the observed data $\boldsymbol{y}$. If $\boldsymbol{x}$ is similar to $\boldsymbol{y}$, then the proposed parameter $\boldsymbol{\theta}^*$ is accepted as a sample from the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. However, if $\boldsymbol{x}$ is not similar to $\boldsymbol{y}$ then it is rejected as it is not likely that the value of $\boldsymbol{\theta}$ would have created the observed dataset for that model (Beaumont et al., 2002; Sisson and Fan, 2010). The algorithm is given in Algorithm 3. Therefore, in likelihood free algorithms, evaluation of the likelihood is approximated by comparing

Algorithm 3 Likelihood–free rejection sampling algorithm (Tavare et al., 1997).

1. Sample $\boldsymbol{\theta}^* \sim \pi(\boldsymbol{\theta})$ where $\pi(\boldsymbol{\theta})$ is the prior for $\boldsymbol{\theta}$.
2. Simulate data $x \sim \pi(x|\boldsymbol{\theta}^*)$ from the model.
3. Accept $\boldsymbol{\theta}^*$ if $\boldsymbol{x} \simeq \boldsymbol{y}$.

a simulated dataset to the observed data. It works by augmenting the target posterior distribution

$$\pi(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

where $\boldsymbol{x}$ is the simulated data. Note that if $\boldsymbol{x} = \boldsymbol{y}$, then $\pi_{LF}(\boldsymbol{\theta}, \boldsymbol{y}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ so we sample from the target posterior exactly. We choose $\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ so that when $\boldsymbol{x}$ and $\boldsymbol{y}$ are very close it takes larger values. The aim is to evaluate the marginal posterior by integrating out the simulated data $\boldsymbol{x}$ (Sisson and Fan, 2010).

$$\pi_{LF}(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta}) \int_{\boldsymbol{x}} \pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}.$$

The marginal posterior, $\pi_{LF}(\boldsymbol{\theta}|\boldsymbol{y})$, estimates the actual posterior, $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. There are methods that target the marginal posterior, however, in general the MCMC scheme will target $\pi_{LF}(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$. $\pi_{LF}(\boldsymbol{\theta}|\boldsymbol{y})$ is estimated by removing the simulated datasets, $\boldsymbol{x}$, from the results of the MCMC scheme. In order to decide how similar the simulated data $\boldsymbol{x}$ is to the observed data $\boldsymbol{y}$, we need to choose a function to measure this, $\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$. This can be a function of the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, for example

$$\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\epsilon} K\left(\frac{|\boldsymbol{x} - \boldsymbol{y}|}{\epsilon}\right)$$

where K is a kernel density and ϵ is a scale parameter. Another choice could compare $\boldsymbol{x}$ and $\boldsymbol{y}$ using summary statistics, $T(\cdot)$, for example

$$\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\epsilon} K\left(\frac{|T(\boldsymbol{x}) - T(\boldsymbol{y})|}{\epsilon}\right).$$

If the summary statistics chosen are sufficient for $\boldsymbol{\theta}$, then it is identical to comparing the actual datasets, and we do not introduce another approximation. Both of these functions have high values when $\boldsymbol{x}$ is similar to $\boldsymbol{y}$ and low values when they are dissimilar. A common choice for $\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ is the uniform kernel density (Marjoram et al., 2003; Tavare et al., 1997),

$$\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) \propto= \begin{cases} 1 & \text{if } \rho(T(\boldsymbol{x}), T(\boldsymbol{y})) \leq \epsilon; \\ 0 & \text{otherwise,} \end{cases}$$

Algorithm 4 Likelihood–Free Metropolis–Hastings algorithm.

1. Set $j = 1$. Choose values of $\boldsymbol{\theta}_0$ and $\boldsymbol{x}_0$ to initialise the chain and choose ϵ. Let $i = 0$.
2. Sample $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_{j-1})$ using the proposal distribution q.
3. Simulate $\boldsymbol{x}^* \sim \pi(\boldsymbol{x}|\boldsymbol{\theta}^*)$ using the model.
4. Calculate the acceptance probability $\alpha(\boldsymbol{\theta}^*, \boldsymbol{x}^*|\boldsymbol{\theta}_{j-1}, \boldsymbol{x}_{j-1})$ of the proposed $\boldsymbol{\theta}^*$ and $\boldsymbol{x}^*$
$$\alpha(\boldsymbol{\theta}^*, \boldsymbol{x}^*|\boldsymbol{\theta}_{j-1}, \boldsymbol{x}_{j-1}) = \min\left\{1, \frac{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}^*)\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}_{j-1}|\boldsymbol{\theta}^*)}{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}_{j-1}, \boldsymbol{\theta}_{j-1})\pi(\boldsymbol{\theta}_{j-1})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_{j-1})}\right\}.$$
5. Let $\boldsymbol{\theta}^j = \boldsymbol{\theta}^*$ and $\boldsymbol{x}^j = \boldsymbol{x}^*$ with probability $\alpha(\boldsymbol{\theta}^*, \boldsymbol{x}^*|\boldsymbol{\theta}, \boldsymbol{x})$, otherwise $\boldsymbol{\theta}^j = \boldsymbol{\theta}^{j-1}$ and $\boldsymbol{x}^j = \boldsymbol{x}^{j-1}$.
6. Let $j = j + 1$ and go back to step 2.

where ρ is a distance measure between $T(\boldsymbol{x})$ and $T(\boldsymbol{y})$. It is possible to use a Metropolis–Hastings algorithm which targets the augmented likelihood free posterior, $\pi_{LF}(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$. The acceptance ratio in this algorithm is $\alpha = \min(1, A)$, where

$$\begin{aligned} A &= \frac{\pi_{LF}(\boldsymbol{\theta}^*, \boldsymbol{x}^*|\boldsymbol{y})q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)\pi(\boldsymbol{x}|\boldsymbol{\theta})}{\pi_{LF}(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)} \\ &= \frac{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}^*)\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)\pi(\boldsymbol{x}|\boldsymbol{\theta})}{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)} \\ &= \frac{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}^*)\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi_\epsilon(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\pi(\boldsymbol{\theta})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}. \end{aligned}$$

We can see that evaluation of the likelihood is avoided. The Metropolis–Hastings algorithm which targets $\pi_{LF}(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$ is given in Algorithm 4. A number of improvements have been proposed to improve the accuracy and efficiency of the basic likelihood free method. See Marin et al. (2012) for an overview.

Part I

An Analysis of intrinsically disordered proteins using hidden Markov models

Chapter 3

Introduction

This chapter introduces part one of the thesis, giving important background information which will be used throughout this part. It first defines intrinsically disordered proteins (IDPs), describes their properties and discusses why they are important as well as their links to many diseases. We consider computational tools for predicting protein disorder, introduce the sets of IDPs that we will analyse and discuss the use of hidden Markov models (HMMs) to analyse sequences.

3.1 Background

For many years it was believed that proteins require a set structure in order to complete their function. This is known as the protein structure–function paradigm. However, more recently it became apparent that there are proteins that do not fit this idea as they have a flexible structure. These proteins became known as 'intrinsically disordered proteins' (IDPs). An IDP is a protein which is biologically active and does not have a stable secondary or tertiary structure (Breydo and Uversky, 2011). Recently, researchers have shown an increased interest in IDPs due to their important biological functions, making research in this area very active (Dosztányi et al., 2010; Uversky, 2011; Uversky et al., 2014; Cumberworth et al., 2013; Fuxreiter et al., 2014).

Just over one third of all human proteins contain long unstructured sections (arbitrarily defined as greater than 30 amino acid residues in length) which allow IDPs to have flexible functions and makes them ideal for signalling and regulation (Dunker and Kriwacki, 2011). It has been found that IDPs have more hydrophobic amino acids when compared to structured proteins so it is suggested that the levels of hydrophobic and hydrophilic amino acids could predict whether a protein is fully folded, completely unfolded or has regions which are unfolded (Dunker and Kriwacki, 2011).

Several algorithms have been created to predict disorder. For example Prilusky et al.

Group 1 (the 'FET' proteins)	Group 2
TATA–binding protein–associated factor 2N (TAF15)	Murine double minute protein (MDM2)
RNA–binding protein FUS (FUS)	CREB–binding protein (CBP)
RNA–binding protein EWS (EWS)	tumour protein 53 (p53)

Table 3.1: Proteins of interest: Group 1 are almost entirely unstructured and group 2 are partially unstructured.

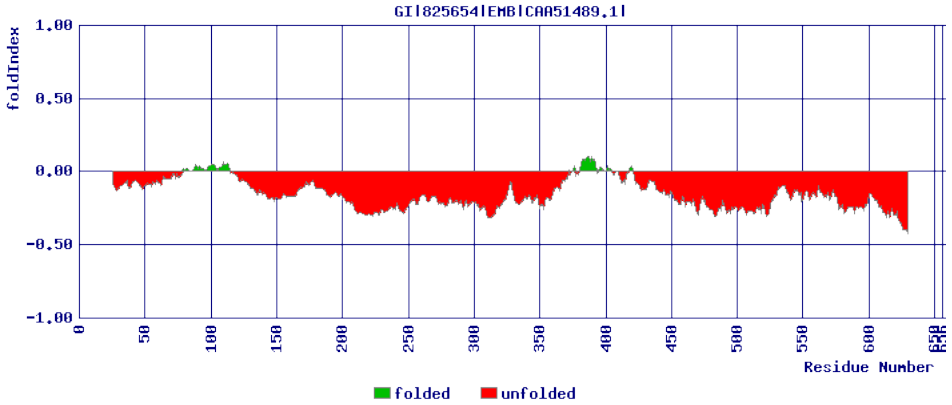
(2005) discuss an Internet tool called FoldIndex in which an amino acid sequence for a protein can be inserted and the tool predicts if the protein is intrinsically unfolded using an algorithm based on the hydrophobicity and net charge. This algorithm is useful to predict which sections of a protein are disordered but can not identify similar structures between proteins or within disordered regions. We hope to use a statistical analysis of the amino acid sequences of IDPs to find segments of the sequence that may have biological reference in order to guide laboratory experiments.

3.1.1 Description of the data

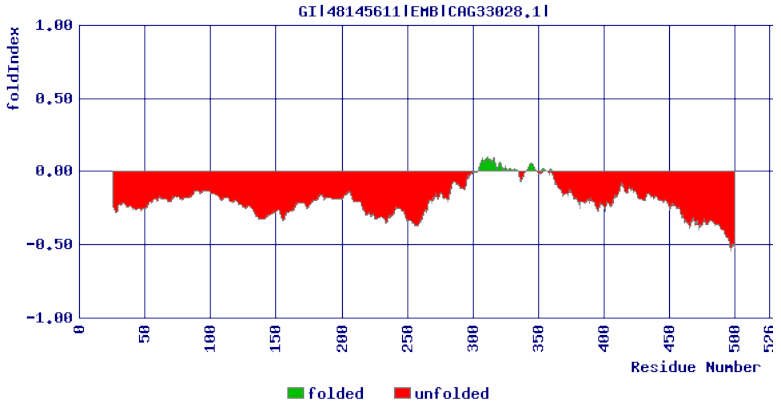
We are particularly interested in two groups of proteins as shown in Table 3.1. We are interested in the first group of proteins (the 'FET' proteins) as they are almost entirely unstructured according to available algorithms. An example algorithm is FoldIndex which is an indicator algorithm that is based on a linear function acting as a boundary between ordered and disordered proteins (Prilusky et al., 2005). This tool produces a graph in which green areas correspond to ordered regions of an amino acid sequence and red areas correspond to disordered regions as shown in Figure 3.1. These proteins are known to form aggregates in neurodegenerative diseases and EWS has been linked to bone and soft tissue cancer (Arvand, 2001). There may be hidden properties of the group one sequences that are common to the FET proteins, which this project may reveal.

The second group are partially unstructured as shown in the FoldIndex graphs in Figure 3.2, but less inclined to aggregate. However, CBP has been found in the inclusion bodies of neurodegenerative diseases and p53 has also been linked to several cancers (Uversky et al., 2014; McCampbell, 2000). These proteins physically and functionally interact through their unstructured regions. We would like to find out if there is something that their unstructured regions have in common, and whether this is different than the FET proteins. We will use Hidden Markov models to investigate this.

(a)



(b)



(c)

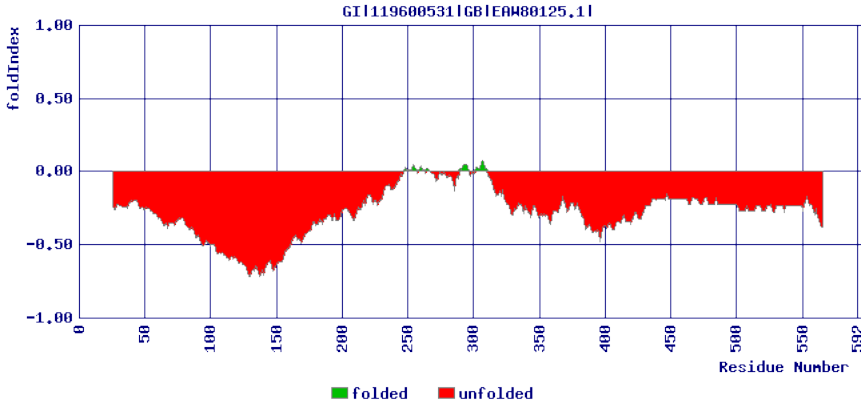
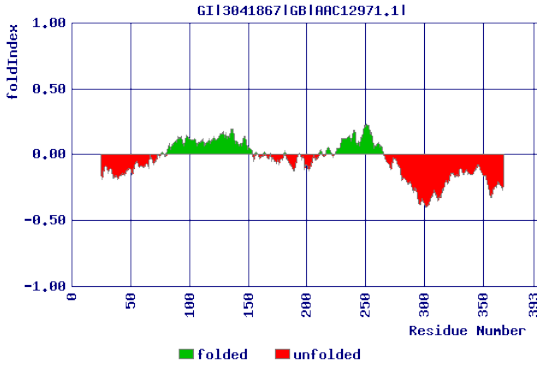
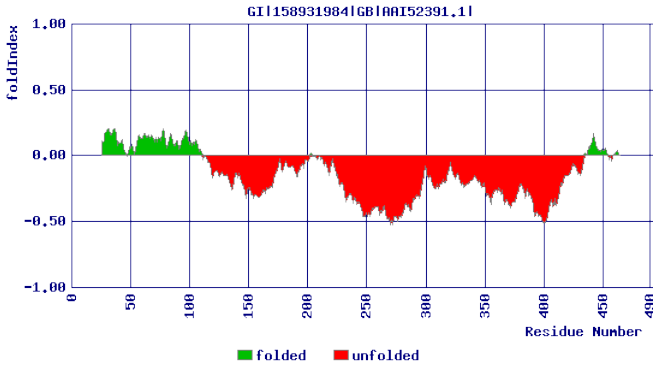


Figure 3.1: Structured and unstructured regions of the group 1 proteins according to the FoldIndex algorithm (Prilusky et al., 2005) for (*a*) EWS, (*b*) FUS and (*c*) TAF15. Residue number is the amino acid number in the protein amino acid sequence. The foldIndex is a number between -1 and 1. Positive values correspond to regions that are predicted to be folded (green) and negative values correspond to regions that are predicted to be unfolded (red).

(a)



(b)



(c)

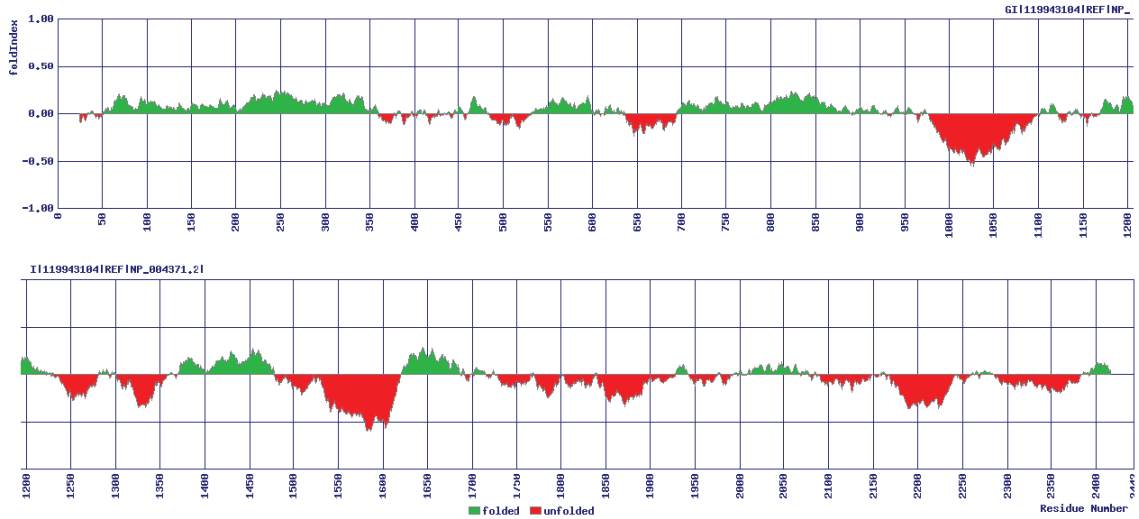


Figure 3.2: Structured and unstructured regions of the group 2 proteins according to the FoldIndex algorithm (Prilusky et al., 2005) for (*a*) p53, (*b*) MDM2 and (*c*) CBP. Residue number is the amino acid number in the protein amino acid sequence. The foldIndex is a number between -1 and 1. Positive values correspond to regions that are predicted to be folded (green) and negative values correspond to regions that are predicted to be unfolded (red).

3.1.2 Aims and objectives

The key aim of this work is to see whether Hidden Markov Models (HMMs) can be used to reveal particular regions of IDPs that are of biological importance or common to these proteins in order to address biological questions about their properties. This analysis will be used to guide future experiments. The main aim of pre–existing algorithms is to predict regions of order and disorder using the amino acid sequences. However, we would like techniques which allow comparison of groups of proteins for similar or different structures within these regions of disorder so that we can guide lab experiments to look at particular segments and their biological importance.

3.2 Literature review of intrinsically disordered proteins

3.2.1 Introduction

Analysis of the amino acid sequences of IDPs is an area which is being actively researched. It is the amino acid sequence which is responsible for the disorder as it leads to particular properties which do not allow ordered proteins to form (Turoverov et al., 2010).

3.2.2 Why find disordered segments?

Disordered proteins are very common in eukaryotes. It is estimated that 30–50% of eukaryotic proteins have a disordered section (Dosztányi et al., 2010). These proteins are involved in important biological functions, for example regulation and cell signalling and have links to diseases including cancer and neurodegenerative diseases. The disorder allows one protein to bind to multiple partners with high specificity (Joerger and Fersht, 2008).

The importance of these proteins means that in order to discover how they work we need to know about their structure and where the disordered segments are located (Dosztányi et al., 2010). There are different types of disorder such as random coil, induced structure after interaction and molten globules. The type of disorder could also be important in how a protein functions (Dosztányi et al., 2010).

3.2.3 IDPs and disease

The cell cycle is maintained by proteins and therefore if a protein is not functioning correctly, this can cause many serious diseases (Uversky et al., 2009). It could be that the protein is not folded correctly, has lost its function, gained a toxic function or formed a protein aggregate. Diseases caused by a protein not taking its correct functional form are called *protein–misfolding diseases*. These diseases can be restricted to one organ or spread through many tissues (Uversky et al., 2009, 2014). The proteins related to protein–misfolding diseases tend to be part of important cellular processes, for example

cell signalling and regulation, and a great number of these proteins are IDPs. Examples of protein–misfolding diseases linked to IDPs include neurodegenerative diseases, cancer, prion diseases, cardiovascular diseases and Type II diabetes (Uversky et al., 2008, 2009, 2014).

IDPs and cancer

The IDP, p53, has strong links to cancer. p53 is important in cancer prevention due to it's role in apoptosis (programmed cell death). Therefore if p53 does not function correctly, this can be one of the main factors leading to cancer developing (Uversky et al., 2014).

p53 is an important protein within cells as it is involved in many processes; for example apoptosis, repairing DNA and in stress response (Uversky et al., 2014). It has been shown that p53 either induces or inhibits 150 genes (Uversky et al., 2009). If p53 does not function, the cell can become cancerous.

Many types of cancer, including breast, liver, colon, lung and brain cancer, have been linked to p53 mutations (Uversky et al., 2008, 2009). It has been shown that 79% of cancer–associated proteins have predicted disordered regions of 30 or more amino acids in length (Uversky et al., 2008). Examples of cancer associated IDPs found experimentally include p53 in several cancers, BRACA1 in breast cancer, AFP in foetal cancer, EWS in bone and soft tissue cancer and HPV proteins in cervical cancer (Uversky et al., 2014).

IDPs and Neurodegenerative disease

Neurodegenerative diseases are associated with cell death in particular areas of the brain which can cause numerous symptoms including movement problems and dementia. Several neurodegenerative diseases have been linked to protein misfolding and the development of protein aggregates and most of the proteins that do not fold correctly in neurodegenerative diseases have been shown to be intrinsically disordered (Breydo and Uversky, 2011).

It has been suggested that overproduction of IDPs can be toxic to cells as particular IDPs could interact and form inclusion bodies (IBs) which indicates that IDPs may play a role in neurodegenerative diseases including Huntington's Disease (HD), Parkinson's Disease (PD) and Alzheimer's Disease (AD) (Dunker and Kriwacki, 2011). As a result cells must regulate these proteins very carefully in comparison to folded proteins.

IDPs and aggregation

Aggregation is not restricted to the proteins associated with neurodegenerative diseases. It has been shown that proteins not linked to neurodegeneration have shown aggregation properties under the correct conditions (Turoverov et al., 2010). Aggregation has been shown to be related to segments of high hydrophobicity, good β–sheet propensity and a

low net charge (Linding et al., 2004). These areas are protected in folded proteins and aggregation happens during times when the protein is not fully folded. Aggregation links IDPs to neurodegenerative diseases. However, experiments have shown that in their native state IDPs are not prone to aggregation. Under some conditions aggregation of IDPs may be encouraged; for example at high temperatures. An analysis comparing IDPs to structured proteins found more segments which are susceptible to aggregation are present in structured proteins. This is thought to be because lack of structure and aggregation resistance require the same properties (Linding et al., 2004).

Linding et al. (2004) found that segments of the amino acid sequence with high hydrophobicity and a low net charge are more likely to aggregate. We would expect that these segments are hidden within a protein to prevent aggregation, which has been shown to be the case when tested experimentally (Linding et al., 2004). Experiments also revealed that aggregation tends to occur during folding (Linding et al., 2004).

3.2.4 Properties of disordered proteins

Amino acid sequence

The properties of a protein originate from the amino acids. A list of amino acids and the corresponding `FASTA` IDs are given in Table 3.2. Previous studies have reported that disordered regions tend to lack W, F, I, Y, V and L and are enriched with G, S and P which suggests that the first set of amino acids are 'order promoting' and the second set are 'disorder promoting' amino acids (Dunker et al., 2001; Ferron et al., 2006). H and T are thought to be unbiased in regards to disorder. Although there is evidence to suggest amino acid frequencies can be used to predict disordered segments (Weathers et al., 2004), there are cases which suggest that using the frequencies is not sufficient without additional information. For example, RNA cap 2'–O–methyltransferase domain of dengue virus polymerase is structured but has high levels of disorder promoting amino acids and a lack of order promoting amino acids (Ferron et al., 2006).

Disordered regions have been associated with low hydrophobicity. That is, whether a protein can easily interact in an aqueous environment. Disordered regions are associated with a high net charge which means they are repulsive (Uversky, 2011). Charge is important in disordered proteins for the extended structure to occur. A previous study has reported that nucleoporins have large disordered domains and if they have a low net charge they have a more structured arrangement than if they have a high net charge (Uversky, 2011; Yamada et al., 2010). Both of these factors are thought to be important for a protein to lack order (Uversky, 2011). This has been shown by an indicator algorithm FoldIndex, which is based on a linear function acting as a boundary between ordered and disordered proteins. Sample output of FoldIndex is shown in Figures 3.1 and 3.2 (Prilusky et al.,

	FASTA ID	Amino Acid	Charge	Hydrophobicity
1	A	Alanine	Neutral	Hydrophobic
2	C	Cysteine	Neutral	Hydrophobic
3	D	Aspartic acid	Negative	Hydrophilic
4	E	Glutamic acid	Negative	Hydrophilic
5	F	Phenylalanine	Neutral	Hydrophobic
6	G	Glycine	Neutral	Hydrophobic
7	H	Histidine	10% Positive, 90% Neutral	Hydrophilic
8	I	Isoleucine	Neutral	Hydrophobic
9	K	Lysine	Positive	Hydrophilic
10	L	Leucine	Neutral	Hydrophobic
11	M	Methionine	Neutral	Hydrophobic
12	N	Asparagine	Neutral	Hydrophilic
13	P	Proline	Neutral	Hydrophobic
14	Q	Glutamine	Neutral	Hydrophilic
15	R	Arginine	Positive	Hydrophilic
16	S	Serine	Neutral	Hydrophilic
17	T	Threonine	Neutral	Hydrophilic
18	V	Valine	Neutral	Hydrophobic
19	W	Tryptophan	Neutral	Hydrophobic
20	Y	Tyrosine	Neutral	Hydrophobic

Table 3.2: List of amino acids.

2005).

Other factors

Other factors used to discriminate between ordered and disordered regions include hydropathy, flexibility, coordination number, β-sheet propensity, volume and bulkiness (Uversky, 2011). Low complexity is associated with disorder. However, this complexity alone is not sufficient, since certain ordered proteins share this property. For example, fibrous proteins (Dosztányi et al., 2010). Evolution speed can also be used to differentiate between disordered and ordered proteins as disordered proteins evolve at a faster rate since it is less important for the sequence to remain the same to conserve function. However, exceptions to this rule can be found as there are some IDPs which are conserved. This is particularly true of those which form complexes (Dosztányi et al., 2010). It has been suggested that posttranslational modification sites and proteolytic attack sites occur frequently in disordered segments (Uversky, 2009).

3.2.5 Predicting protein disorder

Numerous algorithms exist to predict protein disorder. We have already discussed FoldIndex which is based on charge and hydrophobicity. Other algorithms are described in Table 3.3. The predictions from algorithms of protein disorder are used to guide many laboratory experiments and these experiments reveal other segments of disorder which influences the disorder predictors. This creates a cycle of improvement of knowledge (Turoverov et al., 2010).

Protein prediction algorithms depend on various physiochemical properties. Specific predictors are better at predicting certain types of disorder (Ferron et al., 2006). For example, the PONDR algorithm labels disordered regions as those that contain random coils, partially unstructured regions and molten gobules, whereas using charge and hydrophobicity detects fully disordered regions which consist of random coils (Ferron et al., 2006).

Some algorithms use machine learning techniques in which a list of disordered proteins or regions are used to train the algorithms. The databases on disordered proteins are small. For example, Disprot contains only 694 proteins (`http://www.disprot.org/`). There are problems with these protein selection methods as the datasets are not always consistent and are biased due to the difficulty in crystallising proteins with long disordered segments (Ferron et al., 2006). Therefore, algorithms that exist fall into two categories; those which use propensities of the amino acids to predict disorder such as FoldIndex and IUPred and machine learning techniques based on neural networks such as PONDR and DisPro (Ferron et al., 2006). The first category does not have the problem of bias so they can make better predictions for proteins different to the training proteins used in the machine learning techniques (Ferron et al., 2006).

3.2.6 Structure of the Group 1 and 2 proteins

Group 1 proteins: EWS, TAF15 and FUS

FUS, EWS and TAF15 are members of the FET protein family and have similar structures. The amino acid terminus of the FET proteins is intrinsically disordered and can act as a trans–activating domain (TAD) when bound to a DNA binding domain. It contains many Q, G, S and Y amino acids but EWS has many P and T and many repeats of S–Y–G–Q–Q–S (Tan and Manley, 2009). The C–terminal domain consists of a conserved RNA recognition motif with disordered RGG domains either side which contribute to the RNA binding of the FET proteins (Kovar, 2011).

Group 2 proteins: p53, MDM2 and CBP

The protein p53 has an intrinsically disordered N–Terminal region and this has a transactivation domain (TAD) which can bind many partners including CBP and MDM2 (Joerger

Predictor	Information used	Technique
PONDR	Amino acid sequence,various propensities.	
FoldIndex	Hydrophobicity and charge.	Moving average.
GlobPlot	Russell/Linding scale (propensities of an amino acid to be part of a random coil structure or regular secondary structure).	Running sum of propensities for residues.
DISpro	Secondary structure, relevant solvent accessibility.	Machine learning process with 1D–recursive neural networks.
IUPred	Inter–amino acid interactions (negative free energy).	
NORSp	Transmembrane helices and coiled–coil.	Defines disorder as sections with less than 12% in a helix or coil.
DISOPRED/2	High resolution X–ray crystal structures.	Support vector machine learning algorithm.
RONN	Sequence alignment.	Bio–basis function neural network trained on a set of disordered proteins.
SEG	Trigger complexity and extension complexity.	
Disembl	Areas lacking secondary structure, mobile loops, regions devoid of electron density when crystallised.	Neural networks (trained with X–ray structure data).

Table 3.3: List of disorder predictors (Liu and Rost, 2003; Ward et al., 2004; Dosztányi et al., 2010; Prilusky et al., 2005; Ferron et al., 2006).

and Fersht, 2008). Short segments of about 20 amino acids change from disordered to ordered when they form a complex. For example amino acids 15–29 form an α–helix when they interact with the N–terminal domain of MDM2 (Joerger and Fersht, 2008). The MDM2 binding section also forms a helix. This section of MDM2 covers part of the binding area for CBP which is required for transcription to be activated (Joerger and Fersht, 2008). When CBP binds to p53 it is believed to relax the chromatin form of p53. CBP and MDM2 are in competition to bind to the same area of p53 (Joerger and Fersht, 2008). If CBP binds, then MDM2 can not also bind and so p53 is not degraded by the proteasome (Joerger and Fersht, 2008). Conversely, if MDM2 binds, CBP can not also

bind and activation of transcription can not occur. It is thought that posttranslational alterations, for example phosphorylation, affects the binding of MDM2 or CBP (Joerger and Fersht, 2008).

The extreme C terminus is also disordered, however small areas may change from disordered to ordered when interactions occur (Joerger and Fersht, 2008). For example, CBP takes a β–turn form when lysene 382 is acetylated and it is bound to the bromodomain of CBP. The PGGS motif (amino acids 359–362) is extended and it binds to peptides derived from MDM2 (Joerger and Fersht, 2008).

3.3 Statistical review

A considerable amount of literature has been published using hidden Markov models (HMMs) to analyse sequences. For example, HMMs have been used to detect homogeneous segments in DNA sequences (Boys et al., 2000; Boys and Henderson, 2002, 2004). Boys et al. (2000) show that using Markov chain Monte Carlo (MCMC) techniques for an HMM may be used to segment intron 7 of the chimpanzee α–fetoprotein gene. These techniques are particularly useful to combine prior knowledge about particular quantities, such as segment length with the information provided by the data. The authors found 3 segments which approximately agreed with other analyses of the sequence (Churchill, 1989). The analysis did not identify the Xba target site, but this was incorporated in a segment with similar structure (Churchill, 1989). This analysis also produced segment type and changepoint probability plots and found these ideal to identify homogeneous segments. The methodology assumes that the number of segments is known. However, techniques are available to find the most appropriate number of segments which involve finding the posterior probability function for the number of segments (Boys and Henderson, 2001a). Boys and Henderson (2001a) use reversible jump MCMC to find the posterior probability function for the number of segments. These techniques have been successfully applied to the genome of the bacteriophage *lambda* (Boys and Henderson, 2001a). Further work on the bacteriophage *lambda* genome used MCMC methods to determine the order of dependence of the Markov chain which produced results which agreed with other analyses (Boys and Henderson, 2004, 2002, 2001a). A description of how to apply these methods is outlined in Chapter 5.

Other approaches to analyse DNA sequences include multiple changepoint methods where the changepoints determine the edges of the segments. Examples include Braun et al. (2000) who use quasi–likelihood techniques and Liu and Lawrence (1999) who use Bayesian methods. For a review on methods for DNA segmentation see Braun and Müller (1998).

HMMs have been used on protein sequences for sequence alignment, motif detection and

classification. These methods involve using training datasets to estimate the parameters for the HMM (Baldi et al., 1994; Schmidler et al., 2000).

3.4 Summary

In this chapter we have discussed what IDPs are and why they are biologically important. We have stated the aims of this part of the thesis which is to find biologically important segments in IDPs in order to guide laboratory experiments. We have described the properties of IDPs and their amino acid sequences as we will use this information to simplify the amino acid sequence of the IDPs. Finally, we described examples in the literature that have used HMMs in sequence analysis.

Chapter 4

Bayesian analysis using hidden Markov models

This chapter introduces hidden Markov models and describes how they can be used for modelling the amino acid sequence of IDPs. We introduce Bayesian techniques to obtain the number of segment types within a sequence and to find where these segment types are. An important step in the analysis is evaluating the marginal likelihood which is defined in Section 4.2.7. We suggest using the power posterior method to do this (Friel and Pettitt, 2008).

4.1 Introduction

Extensive work has been carried out on DNA base sequences to look for a hidden structure using HMMs (Boys et al., 2000; Boys and Henderson, 2002, 2004). We use this methodology to investigate the structure of amino acid sequences of IDPs.

4.1.1 First order Markov chain model

Suppose the observed sequence is Y_1, Y_2, ..., Y_n. This might be a sequence of amino acids in which case Y_t is the amino acid present at position t. In general we assume that Y_t takes values $1, 2, \ldots, f$, so that, in the amino acid case we have $f = 20$, as there are 20 possible amino acids. The reason we describe this for the general case f is that we relabel the amino acids according to their properties (Table 3.2).

The observed sequence is modelled using a first order Markov chain model. That is, the probability of, for example, an amino acid being present at a certain position is only dependent on the previous amino acid. Hence

$$Pr(Y_t = j|Y_{t-1} = i, Y_{t-2}, Y_{t-3}, \ldots, Y_1) = Pr(Y_t = j|Y_{t-1} = i) = p_{ij} \tag{4.1}$$

for $i, j = 1, 2, \ldots, f$. We can collate these transition probabilities in a transition matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1f} \\ p_{21} & p_{22} & \cdots & p_{2f} \\ \vdots & \vdots & \ddots & \vdots \\ p_{f1} & p_{f2} & \cdots & p_{ff} \end{pmatrix}.$$

The rows represent the probabilities of having each of the f amino acids at position t, given that there is a particular amino acid at position $t-1$. As the elements in a row add up to one, P is called a stochastic transition matrix. We assume that the transition matrix does not change along the sequence, that is, P does not depend on t.

4.1.2 Extension to the HMM

In a hidden Markov model (HMM), the observed sequence develops at a particular position according to one of r different transition matrices. Of course, we also need to determine a sensible value for r which can be based on the marginal posterior probability function for r. This is described in Section 4.2.7. We denote $\mathcal{P} = (P^{(1)}, P^{(2)}, \ldots, P^{(r)})$ where $P^{(k)} = (p_{ij}^{(k)})$ for $k = 1, \ldots, r$. These transitions structures are unknown and must be inferred. We define the sequence $\mathbf{S} = (S_1, S_2, \ldots, S_n)$ to identify which transition structure is used at each point in the sequence, where n is the length of the sequence: here, at position t, S_t is an integer from 1 to r identifying the transition structure used when moving from position t to position $t+1$. For example, given a sequence of ten amino acids and $r = 2$, one possible transition structure is $\mathbf{S} = (1, 1, 1, 1, 2, 2, 2, 1, 1, 1)$.

In reality, we only observe the amino acids at each position and do not observe the sequence $\mathbf{S}$. This is why the model is called a hidden Markov model (HMM). The different transition structures $P_1, \ldots, P_r$ are also unknown. We call the sequence $\mathbf{S}$ our segmentation process and will assume it follows a first order Markov chain with transition matrix

$$\Lambda = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1r} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{r1} & \lambda_{r2} & \cdots & \lambda_{rr} \end{pmatrix}.$$

This transition matrix Λ describes the probability of changing between transition structures along the sequence. Therefore a probabilistic description of the HMM is given by the observed system equation

$$Pr(Y_t = j | Y_1, Y_2, \ldots, Y_{t-1} = i, S_1, S_2, \ldots, S_t = k) = Pr(Y_t = j | S_t = k, Y_{t-1} = i) = p_{ij}^{(k)}$$

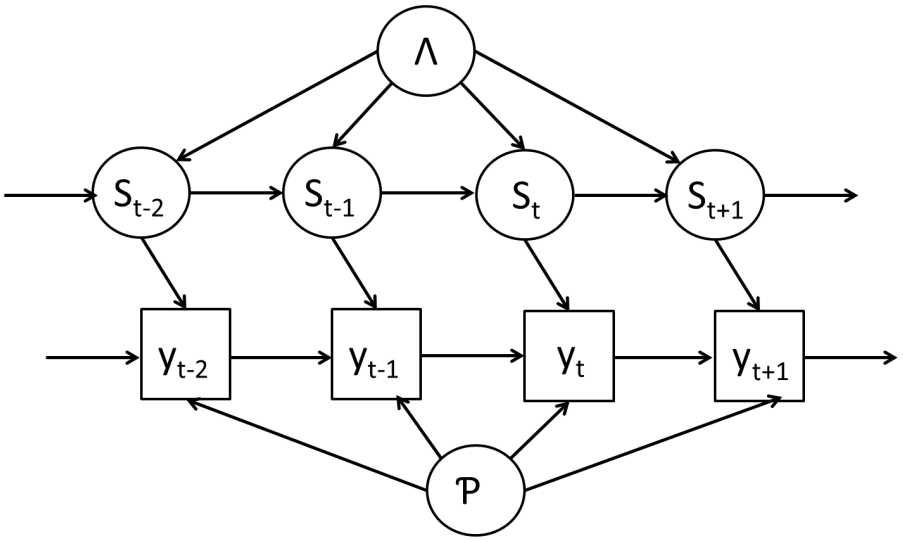


Figure 4.1: DAG to represent the HMM.

for $i, j = 1, 2, \ldots, f$ and $k = 1, 2, \ldots, r$ and the unobserved system equation

$$Pr(S_t = k|S_1, S_2, \ldots, S_{t-1} = j) = Pr(S_t = k|S_{t-1} = j) = \lambda_{jk}$$

for $j, k = 1, 2, \ldots, r$. Another way to represent the model is to use a directed acyclic graph (DAG). The DAG for this model is given in Figure 4.1. In a DAG, due to the local Markov property, node N is conditionally independent of nodes which are not descendents of N given the parents of N. In terms of our model, this means that Y_{t+1} and S_{t+1} are conditionally independent given Y_t and S_t. See Spiegelhalter (1998) for more details on DAGs.

4.2 Bayesian analysis

The aim is to determine the posterior distribution of the parameters in the model, the segmentation and the number of segment types. Following the Bayesian paradigm allows us to combine prior knowledge about the proteins with our sequence data. For example, if the amino acid at a given position results from an update from the transition structure of a particular segment type then it is very likely that the next amino acid will also be an update from this same transition structure.

Using Bayes Theorem, we construct the posterior distribution for the unknown transition structures and the unknown segmentation. However the posterior distribution is complex and only known up to a constant. Therefore we will resort to computationally intensive Bayesian methods to simulate realisations from this posterior distribution.

In this model, if the segmentation was known then the likelihood function is a product

of multinomials as it is a product of the conditional probabilities of the observed and unobserved process as shown in Section 4.2.2. Therefore we can take independent Dirichlet prior distributions for each row of our transition matrices $P_1, \ldots, P_r$ and Λ as these are conjugate and can be used for a range of prior beliefs due to their flexibility (Boys et al., 2000; Boys and Henderson, 2004). Thus we incorporate our prior knowledge into the analysis by an appropriate choice of the parameters of these Dirichlet distributions.

Let $\mathbf{p}_i^{(k)}$ be the i^{th} row of the k^{th} transition matrix, with $i = 1, 2, \ldots, f$ and $k = 1, 2, \ldots, r$ so $\mathbf{p}_i^{(k)}$ is a vector with f elements $(p_{i,j}^{(k)})$, $j = 1, 2, \ldots, f$. Let $\boldsymbol{\theta} = (\mathcal{P}, \Lambda)$ where $\mathcal{P} = \left\{P^{(1)}, P^{(2)} \ldots, P^{(r)}\right\}$. Take $\pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{s}, r)$ to be the likelihood of the parameters $\boldsymbol{\theta}$ given the observed data $\boldsymbol{y} = (y_1, \ldots, y_t)$ and $\pi(\boldsymbol{\theta}|r)$ to be the prior for the parameters. We assume *a priori* independence

$$\pi(\boldsymbol{\theta}|r) = \pi(\mathcal{P}|r)\pi(\Lambda|r) \tag{4.2}$$

and take independent Dirichlet distributions for each row of the transition matrices in $\mathcal{P}$ and Λ, with

$$\mathbf{p}_i^{(k)} \sim \mathcal{D}(\mathbf{a}_i^{(k)}), \qquad \boldsymbol{\lambda}_k \sim \mathcal{D}(\mathbf{b}_k), \tag{4.3}$$

where $i = 1, \ldots, f$, and $k = 1, \ldots, r$. Therefore the prior density is

$$\begin{aligned}\pi(\boldsymbol{\theta}|r) &= \prod_{k=1}^{r}\left\{\pi(\boldsymbol{\lambda}_k)\prod_{i=1}^{f}\pi(\boldsymbol{p}_i^{(k)})\right\} \\ &= \prod_{k=1}^{r}\left\{\left\{\frac{\Gamma(\sum_{\ell=1}^{r} b_{k\ell})}{\prod_{\ell=1}^{r}\Gamma(b_{k\ell})}\prod_{\ell=1}^{r}\lambda_{k\ell}^{b_{k\ell}-1}\right\}\prod_{i=1}^{f}\left\{\frac{\Gamma(\sum_{j=1}^{f} a_{ij}^{(k)})}{\prod_{j=1}^{f}\Gamma(a_{ij}^{(k)})}\prod_{j=1}^{f}(p_{ij}^{(k)})^{a_{ij}^{(k)}-1}\right\}\right\}.\end{aligned}$$

It is possible to use other flexible prior distributions, for example the logistic normal distribution. However, this distribution is not conjugate to the multinomial distribution (Boys et al., 2000). A mixture of Dirichlet distributions is another option, but this would require many extra parameters to be specified (Boys et al., 2000).

4.2.1 Specification of prior parameters

At this stage we have very little idea about the values taken by the transition structures $\mathcal{P}$ and this is expressed through the choice of the prior parameters, $\boldsymbol{a}_i^{(k)}$. This can be achieved by choosing parameters which give each transition probability the same mean. As each row must sum to one, the transition probability means become $1/f$. We also want to express our lack of knowledge about these probabilities by giving them a reasonably large standard deviation. We can achieve this by taking $\boldsymbol{a}_i^{(k)} = (1, 1, \ldots, 1)' \ \forall \ i, k$ where $\boldsymbol{a}_i$ is a vector of length f. As the marginal distribution of the Dirichlet distribution is the Beta distribution then $p_{i,j}^{k} \sim Beta(1, f-1)$, giving us a mean of $1/f$ and standard deviation

of $\sqrt{(f-1)/\{f^2(f+1)\}}$. For each row, $\boldsymbol{a}_i^{(k)}$, the prior represents a theoretical sample of length $f+1$. As we have fr rows this is equivalent to observing a sequence of length $(f+1)fr$. Due to the increasing computational complexity of Bayesian techniques for large values of r, it is unlikely that we will use a value for r greater than 10 and the maximum value for f we consider is 4. This means that the prior is at most equivalent to observing a sequence of length 200. This represents weak prior knowledge as we will use the techniques on sequences over 1500 amino acids in length.

We have stronger prior knowledge for the hidden state transition matrix Λ. This is because we would expect segments to be quite long so if a particular residue is in one segment type we expect it to be highly likely that the next residue is in the same segment type. Therefore, we assume that the diagonal elements λ_{kk} have a mean value of approximately but less than one and the off–diagonal elements λ_{kj} have a very small mean value which is very close to zero. Our beliefs about these off–diagonal elements are exchangeable and so we take $\boldsymbol{b}_k$ to be of the form $\boldsymbol{b}_k = (d, \ldots, d, c, d, \ldots, d)$ where the c is in the k^{th} position in the vector of length r. Therefore, to give λ_{kk} prior mean m and standard deviation s we take

$$d = \frac{c(1-m)}{(r-1)m} \qquad \text{and} \qquad c = \frac{m^2(1-m)}{s^2} - m.$$

These equations can be derived since the marginal distributions of the Dirichlet distribution are Beta and so $\lambda_{kk} \sim Beta(c, (r-1)d)$. Note that if we also take the prior parameters $\boldsymbol{a}_i^{(k)}$ to be the same, with

$$\mathbf{a}_i^{(k)} = a\mathbf{1}, \qquad \text{and} \qquad \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_r \end{pmatrix} = cI_{r\times r} + d(\mathbf{1}\mathbf{1}^T - I_{r\times r}), \tag{4.4}$$

then $\sum_{\ell=1}^{r} b_{k\ell} = c + (r-1)d$, and $\sum_{j=1}^{f} a_{ij}^{(k)} = fa$. Therefore the prior for $\pi(\boldsymbol{\theta}|r)$ is

$$\begin{aligned}\pi(\boldsymbol{\theta}|r) &= \prod_{k=1}^{r}\left\{\frac{\Gamma(c+(r-1)d)}{\Gamma(c)\Gamma(d)^{r-1}}\lambda_{kk}^{c-1}\prod_{\ell=1,\ell\neq k}^{r}\lambda_{k\ell}^{d-1}\right\}\times\prod_{k=1}^{r}\prod_{i=1}^{f}\left\{\frac{\Gamma(fa)}{\Gamma(a)^f}\prod_{j=1}^{f}(p_{ij}^{(k)})^{a-1}\right\}\\ &= \frac{\Gamma(c+(r-1)d)^r\Gamma(fa)^{fr}}{\Gamma(c)^r\Gamma(d)^{r(r-1)}\Gamma(a)^{f^2r}}\left\{\prod_{k=1}^{r}\lambda_{kk}\right\}^{c-1}\left\{\prod_{k=1;\ell=1,\ell\neq k}^{r}\lambda_{k\ell}\right\}^{d-1}\left\{\prod_{k=1}^{r}\prod_{i,j=1}^{f}p_{ij}^{(k)}\right\}^{a-1}.\end{aligned}$$

This prior is equivalent to $c + (r-1)d$ transitions per row of Λ. Therefore the prior for $\boldsymbol{\theta}$ is the equivalent to a sequence of length $r\{c + (r-1)d\} + 1$.

4.2.2 Likelihood

The complete data likelihood, $\pi(\boldsymbol{y}, \boldsymbol{s}|\boldsymbol{\theta}, r)$ is the joint probability function of the observed data, $\boldsymbol{y}$, and the segmentation process, $\boldsymbol{s}$. In the following it will be useful to drop dependence on r for simplicity. The complete data likelihood can be written as

$$\begin{aligned}
\pi(\boldsymbol{y}, \boldsymbol{s}|\boldsymbol{\theta}) &= \pi(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{\theta})\pi(\boldsymbol{s}|\boldsymbol{\theta}) \\
&= \pi(\boldsymbol{y}|\boldsymbol{s}, \mathcal{P})\pi(\boldsymbol{s}|\Lambda) \\
&= Pr(Y_1 = y_1|S_1 = s_1)Pr(S_1 = s_1) \\
&\quad \times \prod_{t=2}^{n} Pr(Y_t = y_t|S_t = s_t, Y_{t-1} = y_{t-1})Pr(S_t = s_t|S_{t-1} = s_{t-1}) \\
&= Pr(Y_1 = y_1|S_1 = s_1)Pr(S_1 = s_1) \prod_{t=2}^{n} p_{y_{t-1},y_t}^{s_t} \lambda_{s_{t-1},s_t}.
\end{aligned}$$

We will assume that the first amino acid, Y_1 and segmentation S_1 are independent and take

$$Pr(S_1 = k) = \frac{1}{r} \quad \text{and} \quad Pr(Y_1 = i) = \frac{1}{f} \quad \text{for} \quad k = 1, \ldots, r \quad \text{and} \quad i = 1, \ldots, f.$$

This means that each amino acid has equal chance of occurring as the first amino acid, Y_1, and each segment type is equally likely in the first position of the segmentation, S_1. Thus the complete data likelihood is

$$\pi(\boldsymbol{y}, \boldsymbol{s}|\boldsymbol{\theta}) = \frac{1}{fr} \prod_{t=2}^{n} p_{y_{t-1},y_t}^{s_t} \lambda_{s_{t-1},s_t}.$$

Further

$$\prod_{t=2}^{n} p_{y_{t-1},y_t}^{s_t} = \prod_{k=1}^{r} \prod_{i=1}^{f} \prod_{j=1}^{f} \left(p_{ij}^{(k)}\right)^{n_{ij}^{(k)}} \quad \text{and} \quad \prod_{t=2}^{n} \lambda_{s_{t-1},s_t} = \prod_{k=1}^{r} \prod_{l=1}^{r} \left(\lambda_{kl}\right)^{m_{kl}},$$

where $n_{ij}^{(k)}$ is the number of transitions from state i to state j in segment type k and m_{kl} is the number of transitions from segment type k to segment type l, and so

$$\pi(\boldsymbol{y}, \boldsymbol{s}|\boldsymbol{\theta}) = \frac{1}{fr} \left\{ \prod_{k=1}^{r} \prod_{i=1}^{f} \prod_{j=1}^{f} \left(p_{ij}^{(k)}\right)^{n_{ij}^{(k)}} \right\} \left\{ \prod_{k=1}^{r} \prod_{l=1}^{r} \left(\lambda_{kl}\right)^{m_{kl}} \right\}.$$

The observed data likelihood, $\pi(\boldsymbol{y}|\boldsymbol{\theta})$, can be written as the sum of the complete data likelihood over all possible segmentations, that is

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{\boldsymbol{s}} \pi(\boldsymbol{y}, \boldsymbol{s}|\boldsymbol{\theta}).$$

This can also be written as

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}) = Pr(Y_1 = y_1) \prod_{t=2}^{n} Pr(Y_t = y_t|\boldsymbol{y}_{t-1}), \tag{4.5}$$

where $\boldsymbol{y}_t = (y_1, \ldots, y_n)$. In Section 4.2.4 we describe an algorithm in which one step calculates the observed data likelihood. Essentially, Equation (4.5) can easily be calculated as it is a product of the normalising constants from the forward filter in the forward–backward algorithm (Baum and Eagon, 1966; Baum and Petrie, 1966; Baum et al., 1970).

4.2.3 The posterior distribution

Using Bayes theorem, we can determine the posterior density as

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

where $\pi(\boldsymbol{\theta})$ is the prior density and $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ is the observed data likelihood. When this density cannot be found analytically we can obtain realisations from the posterior distribution via Markov chain Monte Carlo methods. In this hidden Markov model, the most efficient way to determine the posterior distribution is to use data augmentation, that is, include the segmentation process $\boldsymbol{s}$ but treat it as missing data: we sample the hidden states as part of the MCMC scheme and thereby obtain their posterior distribution. The joint posterior density for the model parameters and segmentation process is

$$\pi(\boldsymbol{s}, \boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{\theta})\pi(\boldsymbol{s}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}).$$

Note that $\pi(\boldsymbol{s}, \boldsymbol{\theta}|\boldsymbol{y})$ is actually not strictly a density as, although $\boldsymbol{\theta}$ is continuous, the hidden states $\boldsymbol{s}$ are discrete. We can sample from this distribution easily via MCMC by alternating between sampling the segmentation process $\boldsymbol{s}$ from $\pi(\boldsymbol{s}|\boldsymbol{\theta}, \boldsymbol{y})$, and the parameters $\boldsymbol{\theta}$ from $\pi(\boldsymbol{\theta}|\boldsymbol{s}, \boldsymbol{y})$. The MCMC algorithm is straightforward as we use a conjugate prior for $\boldsymbol{\theta}$ and we can simulate realisations of $\boldsymbol{s}$ using a forward-backward algorithm (Baum and Eagon, 1966; Baum and Petrie, 1966; Baum et al., 1970). We can obtain a posterior sample for $\boldsymbol{\theta}$ by averaging over the uncertainty of the segmentation $\boldsymbol{s}$. These techniques have been used by Albert and Chib (1993) and Boys and Henderson (2004). Due to the assumption of

prior independence for the parameters $\boldsymbol{\theta}$ we have that

$$\pi(\boldsymbol{\theta}|\boldsymbol{s},\boldsymbol{y}) = \pi(\mathcal{P}|\boldsymbol{s},\boldsymbol{y})\pi(\Lambda|\boldsymbol{s})$$

where

$$\begin{aligned}\pi(\mathcal{P}|\boldsymbol{s},\boldsymbol{y}) &\propto \pi(\mathcal{P})\pi(\boldsymbol{y}|\boldsymbol{s},\mathcal{P}) \\ &\propto \prod_{k=1}^{r}\prod_{i=1}^{f}\prod_{i=1}^{f}(p_{ij}^{(k)})^{a_{ij}^{(k)}-1} \times \prod_{k=1}^{r}\prod_{i=1}^{f}\prod_{i=1}^{f}(p_{ij}^{(k)})^{n_{ij}^{(k)}} \\ &\propto \prod_{k=1}^{r}\prod_{i=1}^{f}\prod_{i=1}^{f}(p_{ij}^{(k)})^{a_{ij}^{(k)}+n_{ij}^{(k)}-1}\end{aligned}$$

and

$$\begin{aligned}\pi(\Lambda|\boldsymbol{s}) &\propto \pi(\Lambda)\pi(\boldsymbol{s}|\Lambda) \\ &\propto \prod_{k=1}^{r}\prod_{l=1}^{r}(\lambda_{kl})^{b_{kl}} \times \prod_{k=1}^{r}\prod_{l=1}^{r}(\lambda_{kl})^{m_{kl}} \\ &\propto \prod_{k=1}^{r}\prod_{l=1}^{r}(\lambda_{kl})^{b_{kl}+m_{kl}}\end{aligned}$$

therefore we recognise that the (full) conditional posterior distribution for $(\mathcal{P},\Lambda)$ has independent components

$$\boldsymbol{p}_i^{(k)}|\boldsymbol{s},r,\boldsymbol{y} \sim \mathcal{D}(\boldsymbol{a}_i^{(k)} + \boldsymbol{n}_i^{(k)}) \tag{4.6}$$

and

$$\boldsymbol{\lambda}_k|\boldsymbol{s},r \sim \mathcal{D}(\boldsymbol{b}_k + \boldsymbol{m}_k) \tag{4.7}$$

where $i = 1,2,\ldots,f$, $k = 1,2,\ldots,r$, $\boldsymbol{n}_i^{(k)} = (n_{ij}^{(k)})$, $\boldsymbol{m}_k = (m_{kl})$, $n_{ij}^{(k)}$ is the number of transitions from state i to state j in segment type k and m_{kj} is the number of transitions from segment type k to segment type l.

We will describe the Gibbs sampling algorithm that can be used to obtain samples from the conditional posterior distributions for $\boldsymbol{\theta}$ and $\boldsymbol{s}$ in the next section.

4.2.4 Gibbs sampling

The Gibbs sampler is an MCMC technique used when the joint distribution is unknown or difficult to sample from but the conditional distributions of the parameters are known and easy to sample from. This technique is useful in our case as we can not estimate $\mathcal{P}$ and Λ, since we do not know $\boldsymbol{s}$, but we do know their full conditional distributions. We use the Gibbs sampler to obtain realisations of the model parameters $\mathcal{P}$ and Λ and of the

Algorithm 5 Gibbs sampling.

1. Simulate $\boldsymbol{\theta}^{(0)}$ from the prior distribution. Set counter $j = 1$.
2. Simulate the segmentation $\boldsymbol{s}^{(j)}$ from $\pi(\boldsymbol{s}|\boldsymbol{\theta} = \boldsymbol{\theta}^{(j-1)}, r, \boldsymbol{y})$ using the forward–backward algorithm (Algorithm 6).
3. Simulate the transition parameters $\boldsymbol{\theta}^{(j)}$ from $\pi(\boldsymbol{\theta}|\boldsymbol{s} = \boldsymbol{s}^{(j)}, r, \boldsymbol{y})$ using Equations (4.6) and (4.7).
4. If $j = N$ stop, otherwise set $j = j + 1$ and return to step 2.

segmentation $\boldsymbol{s}$ given a particular value for r. Let $\boldsymbol{\theta} = (\mathcal{P}, \Lambda)$ and N be the number of iterations of the Gibbs sampler. The Gibbs sampler is shown in Algorithm 5.

We check for convergence and remove the 'burn–in' period from our realisations of $\boldsymbol{s}$ and $\boldsymbol{\theta}$. As realisations from consecutive iterations can be correlated we have the option to reduce this autocorrelation by thinning the realisations, that is, taking only every M iterates for some integer M.

Simulation of the transition parameters

In step 3 of the Gibbs sampler given in Algorithm 5, we simulate realisations of the transition structures. We have shown in Section 4.2.3 that the (full) conditional posterior distribution for $(\mathcal{P}, \Lambda)$ has independent components

$$\boldsymbol{p}_i^{(k)}|\boldsymbol{s}, r, \boldsymbol{y} \sim \mathcal{D}(\boldsymbol{a}_i^{(k)} + \boldsymbol{n}_i^{(k)})$$

and

$$\boldsymbol{\lambda}_k|\boldsymbol{s}, r \sim \mathcal{D}(\boldsymbol{b}_k + \boldsymbol{m}_k)$$

where $i = 1, 2, \ldots, f$, $k = 1, 2, \ldots, r$, $\boldsymbol{n}_i^{(k)} = (n_{ij}^{(k)})$, $\boldsymbol{m}_k = (m_{kl})$, $n_{ij}^{(k)}$ is the number of transitions from state i to state j in segment type k and m_{kj} is the number of transitions from segment type k to segment type l (Boys et al., 2000).

The forward–backward algorithm

During Gibbs sampling we need to simulate a hidden sequence $\boldsymbol{s}$ from $\pi(\boldsymbol{s}|\boldsymbol{\theta}, r, \boldsymbol{y})$. We can do so using the forward–backward algorithm (Baum and Eagon, 1966; Baum and Petrie, 1966; Baum et al., 1970; Boys and Henderson, 2002). This algorithm works be simulating the whole hidden sequence $\boldsymbol{s}$ in a single component block from $\pi(\boldsymbol{s}|\boldsymbol{\theta}, \boldsymbol{y}, r)$. As $\boldsymbol{s}$ follows a

Algorithm 6 The forward–backward algorithm.

1. Calculate the forward probabilities, $f_k(t)$, using the forward filter given in Algorithm 7, for $k = 1, \ldots, r$ and $t = 1, \ldots, n$.
2. Calculate the backward probabilities, $b_j(k, t)$, using the backward filter given in Algorithm 8, for $j = 1, \ldots, r$, $k = 1, \ldots, r$ and $t = 1, \ldots, n$.
3. Sample from $1, \ldots, r$ with probabilities $f_k(n)$ for $k = 1, \ldots, r$ to obtain s_n.
4. For $i = n-1, \ldots, 1$ and $j = 1, \ldots, r$ sample from $1, \ldots, r$ with probabilities $b_j(k, i-1)$ to obtain $s_{n-1}, \ldots, s_1$.

first order Markov chain we have

$$\pi(\boldsymbol{s}|\boldsymbol{\theta}, \boldsymbol{y}) = \pi(\boldsymbol{s}_n|\boldsymbol{y}, \boldsymbol{\theta}) \prod_{t=1}^{n-1} \pi(\boldsymbol{s}_t|\boldsymbol{s}_{t+1}, \boldsymbol{y}^t, \boldsymbol{\theta}),$$

where $\boldsymbol{y}^t = (y_1, y_2, \ldots, y_t)$. This algorithm works by first determining *filtered* probabilities through a forward sweep through the sequence ($t = 1, \ldots, n$), sampling s_n from its marginal posterior distribution, then sampling the remainder of the s_t through a backward sweep ($t = n-1, \ldots, 1$) in which s_t is sampled from it's conditional distribution given s_{t+1}. Dropping the explicit dependence on r and $\boldsymbol{\theta}$ for notational simplicity, the forward part of the algorithm calculates the filtered probabilities $f_k(t) = Pr(S_t = k|\boldsymbol{y}^t)$, $t = 1, \ldots, n$. The backward part of the algorithm calculates the conditional probabilities $b_j(k, t) = Pr(S_t = j|S_{t+1} = k, \boldsymbol{y}^t)$ using the filtered probabilities. A more detailed description of the forward–backward algorithm is given in Algorithm 6.

An alternative way to sample from the conditional distribution $\pi(\boldsymbol{s}|\boldsymbol{\theta}, \boldsymbol{y})$ is to use Gibbs sampling with n univariate component blocks, one for each s_t. In each block we sample from $\pi(s_t|\boldsymbol{s}_{-t}, \boldsymbol{\theta}, \boldsymbol{y})$ for $t = 1, 2, \ldots, n$ where $\boldsymbol{s}_{-t}$ is the sequence $\boldsymbol{s}$ with s_t omitted. This univariate conditional distribution can be shown to be

$$\begin{aligned}\pi(s_t|\boldsymbol{s}_{-t}, \boldsymbol{\theta}, \boldsymbol{y}) &= \pi(s_t|s_{t-1}, s_{t+1}, y_t, y_{t-1}, \mathcal{P}, \Lambda) \\ &= \frac{\lambda_{s_{t-1},s_t} \lambda_{s_t,s_{t+1}} P^{(s_t)}_{y_{t-1},y_t}}{\sum_{k=1}^{r} \lambda_{s_{t-1},k} \lambda_{k,s_{t+1}} P^{(k)}_{y_{t-1},y_t}}.\end{aligned}$$

Unfortunately this one-at-a-time algorithm usually suffers from high dependence between the large number of component blocks, and so does not converge to the posterior distribution as efficiently as when using the single block method. Therefore we will use the single block method, that is, use the forward–backward algorithm (Boys et al., 2000; Germain, 2010).

Algorithm 7 The forward filter.

1. Initialise the forward probabilities

$$f_k(1) = Pr(S_1 = k|y_1) = \frac{Pr(Y_1 = y_1|S_1 = k)Pr(S_1 = k)}{\sum_{l=1}^{r} Pr(Y_1 = y_1|S_1 = l)Pr(S_1 = l)} = \frac{\pi_{y_1}^{(k)}\pi_k}{\sum_{l=1}^{r} \pi_{y_1}^{(l)}\pi_l},$$

   where $Pr(S_1 = k) = \pi_k$ and $Pr(Y_1 = y_1|S_t = k) = \pi_{y_1}^{(k)}$.

2. Calculate the forward probabilities using forward recursions

$$\begin{aligned} f_k(t) = Pr(S_t = k|\boldsymbol{y}^t) &= \frac{Pr(Y_t = y_t, S_t = k|\boldsymbol{y}^{t-1})}{Pr(Y_t = y_t|\boldsymbol{y}^{t-1})} = \frac{Pr(Y_t = y_t, S_t = k|\boldsymbol{y}^{t-1})}{\sum_{k'=1}^{r} Pr(Y_t = y_t, S_t = k'|\boldsymbol{y}^{t-1})} \\ &= \frac{Pr(Y_t = y_t|S_t = k, \boldsymbol{y}^{t-1})Pr(S_t = k|\boldsymbol{y}^{t-1})}{\sum_{k'=1}^{r} Pr(Y_t = y_t|S_t = k', \boldsymbol{y}^{t-1})Pr(S_t = k'|\boldsymbol{y}^{t-1})} \\ &= \frac{p_{y_{t-1},y_t}^{(k)} \sum_{l=1}^{r} \lambda_{lk} f_l(t-1)}{\sum_{k'=1}^{r} \left\{ p_{y_{t-1},y_t}^{(k')} \sum_{l=1}^{r} \lambda_{lk'} f_l(t-1) \right\}}, \end{aligned}$$

   where

$$\begin{aligned} Pr(S_{t+1} = k|\boldsymbol{y}^t) &= \sum_{l=1}^{r} Pr(S_t = l, S_{t+1} = k|\boldsymbol{y}^t) \\ &= \sum_{l=1}^{r} Pr(S_{t+1} = k|S_t = l)Pr(S_t = l|\boldsymbol{y}^t) \\ &= \sum_{l=1}^{r} \lambda_{lk} f_l(t). \end{aligned}$$

4.2.5 Label switching

Label switching occurs when the likelihood of a model's parameters are symmetric and this is true in our case as the likelihood is invariant under permutations of r (Stephens, 2000; Giles, 2001). For example, if $r = 2$ all of the subscripts equal to 1 could switch with the subscripts 2. This means in general there are $r!$ combinations. If we look at trace plots of the parameters of the model and can see jumps after convergence then this is an indication that the labels have switched. We can implement an online process to help prevent label switching (Stephens, 2000; Giles, 2001) which attempts to avoid the problem by identifying how the labels have switched and changing them back (relabelling). This is done as shown

Algorithm 8 The backward filter.

1. Calculate the backward probabilities using backward recursions

$$\begin{aligned} b_j(k,t) &= Pr(S_t = j|S_{t+1} = k, \boldsymbol{y}) = Pr(S_t = j|S_{t+1} = k, \boldsymbol{y}^t) \\ &= \frac{Pr(S_{t+1} = k|S_t = j)Pr(S_t = j|\boldsymbol{y}^t)}{Pr(S_{t+1} = k|\boldsymbol{y}^t)} \\ &= \frac{\lambda_{jk} f_j(t)}{\sum_{l=1}^{r} \lambda_{lk} f_l(t)}, \end{aligned}$$

   where

$$\begin{aligned} Pr(S_{t+1} = k|\boldsymbol{y}^t) &= \sum_{l=1}^{r} Pr(S_t = l, S_{t+1} = k|\boldsymbol{y}^t) \\ &= \sum_{l=1}^{r} Pr(S_{t+1} = k|S_t = l)Pr(S_t = l|\boldsymbol{y}^t) \\ &= \sum_{l=1}^{r} \lambda_{lk} f_l(t). \end{aligned}$$

Algorithm 9 Label switching algorithm.

At iteration i

1. Simulate the new segmentation $\boldsymbol{s}^{(i)}$.
2. Calculate all $r!$ permutations of $\boldsymbol{s}^{(i)}$ and the number of matches between each permutation of $\boldsymbol{s}^{(i)}$ and the previous segmentation $\boldsymbol{s}^{(i-1)}$.
3. Apply the permutation which provides the maximum number of matches to $\boldsymbol{s}^{(i-1)}$ to $\boldsymbol{s}^{(i)}$. Also apply this permutation to $\mathcal{P}^{(i)}$ and $\Lambda^{(i)}$.

in Algorithm 9. Richardson and Green (1997) and Robert et al. (2000) use an alternative approach where they choose an order (ascending order) so that the parameters can be identified. However, Stephens (2000) and Celeux et al. (2000) have shown that this may cause problems with inference.

4.2.6 Parameter reduction

As there are $f = 20$ amino acids, we have 20×20 matrices for $\mathcal{P}$ which will be difficult to estimate if the sequences we are analysing are not long enough to provide sufficient information. The large matrices will also slow down the algorithm. To help reduce this problem we consider three different recodings of the amino acids – these are based on

$f = 2$	$f = 3$	$f = 4$
Hydrophobic = 1	Neutral = 1	Neutral, Hydrophobic = 1
Hydrophilic = 2	Positive = 2	Neutral, Hydrophilic = 2
	Negative = 3	Positive, Hydrophilic =3
		Negative, Hydrophilic = 4

Table 4.1: Properties used to convert amino acid sequences.

the properties of hydrophobicity and charge as these have been shown to be important predictors of disorder (Prilusky et al., 2005).

1. hydrophobic or hydrophilic ($f = 2$) giving 2×2 matrices,
2. positive, negative or neutral ($f = 3$) giving 3×3 matrices,
3. hydrophobic and neutral, or hydrophilic with either a positive, neutral or negative charge ($f = 4$) giving 4×4 matrices.

These recodings are shown in Table 4.1.

4.2.7 Calculating the posterior probability function for r

In order to infer an appropriate number of segment types (r) we need to be able to determine the posterior distribution for r, through its probability function $\pi(r|\boldsymbol{y})$. Using Bayes theorem we have that

$$\pi(r|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|r)\pi(r).$$

To make use of this formulation we need the prior for r, $\pi(r)$, and the marginal likelihood $\pi(\boldsymbol{y}|r)$. Using Bayes Theorem, the posterior density for the parameters $\boldsymbol{\theta}$ given r is

$$\pi(\boldsymbol{\theta}|r, \boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta}|r)\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)}{\pi(\boldsymbol{y}|r)}, \tag{4.8}$$

where $\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)$ is the observed data likelihood and can be determined from the forward filter (of the forward–backward scheme) as described in Section 4.2.2. Integrating both sides with respect to $\boldsymbol{\theta}$ and rearranging gives

$$\pi(\boldsymbol{y}|r) = \int \pi(\boldsymbol{\theta}|r)\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)\, d\boldsymbol{\theta} = E_{\boldsymbol{\theta}|r}\{\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)\}. \tag{4.9}$$

This expectation can sometimes be well approximated using realisations $\{\boldsymbol{\theta}^{(i)}; i = 1, \ldots, N\}$ from the prior distribution, as

$$\pi(\boldsymbol{y}|r) \simeq \frac{1}{N}\sum_{i=1}^{N} \pi(\boldsymbol{y}|\boldsymbol{\theta}^{(i)}, r). \tag{4.10}$$

Unfortunately calculating the marginal likelihood directly by averaging the likelihood over the prior distribution usually gives numerical problems due to the (very) small values taken by the likelihood. However, this can be overcome by using the log-sum-exp trick in which rescaling constants are used to make the calculation more numerically stable. Lets define a rescaling constant

$$k_r = \max_{i=1,\ldots,N} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}^{(i)}, r)$$

so that

$$\frac{e^{k_r}}{N} \leq \pi(\boldsymbol{y}|r) \leq e^{k_r}.$$

Let

$$\pi^*(\boldsymbol{y}|r) \equiv e^{-k_r}\pi(\boldsymbol{y}|r) = \frac{1}{N}\sum_{i=1}^{N} \exp[-k_r + \log \pi(\boldsymbol{y}|\boldsymbol{\theta}^{(i)}, r)\}]. \tag{4.11}$$

The largest value taken by the summand is one and so

$$\frac{1}{N} \leq \pi^*(\boldsymbol{y}|r) \leq 1.$$

Noting that $\pi(r|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|r)\pi(r)$, a numerically stable calculation of the posterior probability function for r is found using the log-sum-exp trick. We define

$$k^* = \max_r \log \pi(\boldsymbol{y}|r) = \max_r \left\{k_r + \log \pi^*(\boldsymbol{y}|r)\right\},$$

as

$$\pi(r|\boldsymbol{y}) = ke^{k_r - k^*}\pi^*(\boldsymbol{y}|r)\pi(r),$$

where

$$k^{-1} = \sum_r e^{k_r - k^*}\pi^*(\boldsymbol{y}|r)\pi(r).$$

There are several alternative methods for determining this marginal likelihood; for example, using the power posterior method (Friel and Pettitt, 2008) or Chib's method (Chib, 1995). We review these methods in the next two sections.

4.2.8 Calculating the marginal likelihood exactly

We can write the marginal likelihood as

$$\pi(\boldsymbol{y}|r) = \sum_{\boldsymbol{s}} \pi(\boldsymbol{y}|\boldsymbol{s}, r)\pi(\boldsymbol{s}|r), \tag{4.12}$$

where the sum is over all r^n possible segmentations for a sequence of length n. It is possible to evaluate Equation (4.12) exactly as in our HMM, we have conditional independence

of $\mathcal{P}$ and Λ and the use of conjugate priors. However this calculation can be very time consuming as the number of possible segmentations is very large for most sequences. For example if $n = 100$ and $r = 2$ there are $2^{100} \simeq 1.27 \times 10^{30}$ combinations, and this makes direct evaluation unfeasible in reality. However, for short sequences we can easily compute the marginal likelihood; for example when $n = 10$ and $r = 2$ there are only $2^{10} = 1024$ possible segmentations. The marginal likelihood is calculated using $\pi(\boldsymbol{s}|r)$ and $\pi(\boldsymbol{y}|\boldsymbol{s}, r)$. These are determined as follows:

$$\begin{aligned}
\pi(\boldsymbol{s}|r) &= \int \pi(\boldsymbol{s}|\Lambda, r)\pi(\Lambda|r)d\Lambda \\
&= \int Pr(S_1 = s_1|r) \prod_{t=2}^{n} Pr(S_t = s_t|S_{t-1} = s_{t-1}, \Lambda, r)\pi(\Lambda|r)d\Lambda \\
&= \int \frac{1}{r} \left\{ \prod_{k=1}^{r} \prod_{l=1}^{r} (\lambda_{kl})^{m_{kl}} \right\} \left\{ \prod_{k=1}^{r} \left\{ \frac{\Gamma(\sum_{l=1}^{r} b_{kl})}{\prod_{l=1}^{r} \Gamma(b_{kl})} \prod_{l=1}^{r} \lambda_{kl}^{b_{kl}} \right\} \right\} d\Lambda \\
&= \frac{1}{r} \prod_{k=1}^{r} \left\{ \frac{\Gamma(\sum_{l=1}^{r} b_{kl})}{\prod_{l=1}^{r} \Gamma(b_{kl})} \int \prod_{l=1}^{r} \lambda_{kl}^{m_{kl}+b_{kl}-1} d\boldsymbol{\lambda}_k \right\} \\
&= \frac{1}{r} \prod_{k=1}^{r} \frac{\Gamma(\sum_{l=1}^{r} b_{kl}) \prod_{l=1}^{r} \Gamma(m_{kl} + b_{kl})}{\prod_{l=1}^{r} \Gamma(b_{kl}) \Gamma\left\{\sum_{l=1}^{r} (m_{kl} + b_{kl})\right\}}
\end{aligned}$$

and

$$\begin{aligned}
\pi(\boldsymbol{y}|\boldsymbol{s}, r) &= \int \pi(\boldsymbol{y}|\boldsymbol{s}, \mathcal{P}, r)\pi(\mathcal{P}|r)d\mathcal{P} \\
&= \int Pr(Y_1 = y_1|S_1 = s_1, \mathcal{P}) \prod_{t=2}^{n} Pr(Y_t = y_t|S_t = s_t, Y_{t-1} = y_{t-1}, \mathcal{P})\pi(\mathcal{P}|r)d\mathcal{P} \\
&= \int \frac{1}{f} \prod_{k=1}^{r} \prod_{i=1}^{f} \prod_{j=1}^{f} (p_{ij}^{k})^{n_{ij}^{(k)}} \prod_{k=1}^{r} \prod_{i=1}^{f} \left\{ \frac{\Gamma(\sum_{j=1}^{f} a_{ij}^{(k)})}{\prod_{j=1}^{f} \Gamma(a_{ij}^{(k)})} \prod_{j=1}^{f} (p_{ij}^{(k)})^{a_{ij}^{(k)}-1} \right\} dp_{ij}^{(k)} \\
&= \frac{1}{f} \prod_{k=1}^{r} \prod_{i=1}^{f} \left\{ \frac{\Gamma(\sum_{j-1}^{f} a_{ij}^{(k)})}{\prod_{j=1}^{f} \Gamma(a_{ij}^{(k)})} \int \prod_{j=1}^{f} (p_{ij}^{(k)})^{n_{ij}^{(k)}+a_{ij}^{(k)}-1} dp_{ij}^{(k)} \right\} \\
&= \frac{1}{f} \prod_{k=1}^{r} \prod_{i=1}^{f} \left\{ \frac{\Gamma(\sum_{j=1}^{f} a_{ij}^{(k)}) \prod_{j=1}^{f} \Gamma(n_{ij}^{(k)} + a_{ij}^{(k)})}{\prod_{j=1}^{f} \Gamma(a_{ij}^{k}) \Gamma(\sum_{j=1}^{f} (n_{ij}^{(k)} + a_{ij}^{(k)}))} \right\}.
\end{aligned}$$

4.2.9 Power posterior method and application to HMMs

The power posterior method is another way of determining the marginal likelihood (Friel and Pettitt, 2008). This method works by sampling from

$$\pi_T(\boldsymbol{\theta}|\boldsymbol{y}, r) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)^T \pi(\boldsymbol{\theta}|r) \tag{4.13}$$

Algorithm 10 The power posterior method.

1. Initialise $\boldsymbol{\theta}_0^{(0)}$. Typically we set $\boldsymbol{\theta}_0^{(0)}$ to be the prior mean, so convergence occurs immediately to the power posterior with $T = 0$.
2. Sample $\boldsymbol{\theta}_i^{(j)}$ for $j = K+1, \ldots R$ using MCMC sampling from $\pi_{T_i}(\boldsymbol{\theta}|\boldsymbol{y}, r)$.
3. Approximate the expectation in expression (4.14) by using Monte Carlo integration
$$E_{\boldsymbol{\theta}|\boldsymbol{y},T}[\log \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)] \approx \frac{1}{R-K} \sum_{j=K+1}^{R} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}_i^{(j)}, r).$$
4. While $i<n$ start the next chain with
$$\boldsymbol{\theta}_{i+1}^{(0)} = \frac{1}{R-K} \sum_{j=K+1}^{R} \boldsymbol{\theta}_i^{(j)}.$$
5. Calculate the $\log \pi(\boldsymbol{y}|r)$ approximately using Equation 4.15.

where $T \in [0, 1]$ is known as the temperature parameter. Expression (4.13) is known as the power posterior. This method uses ideas from path sampling (Gelman and Meng, 1998) to show that the log marginal likelihood can be written in terms of an integral with respect to T, namely

$$\log \pi(\boldsymbol{y}|r) = \int_0^1 E_{\boldsymbol{\theta}|\boldsymbol{y},T}[\log \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)]dT. \tag{4.14}$$

Here the integrand is the expectation of the half mean deviance is taken with respect to the power posterior at temperature T. The integral can be estimated by discretising T over its interval and using the trapezoidal rule. Thus if we take $0 = T_0 < T_1 < \ldots < T_{n-1} < T_n = 1$, then the log marginal likelihood can be estimated by using

$$\log \hat{\pi}(\boldsymbol{y}|r) = \sum_{i=0}^{n-1} (T_{i+1} - T_i) \frac{E_{\boldsymbol{\theta}|\boldsymbol{y},r,T_{i+1}}[\log \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)] + E_{\boldsymbol{\theta}|\boldsymbol{y},r,T_i}[\log \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)]}{2}. \tag{4.15}$$

In turn these expectations can be estimated by using the output $\{\boldsymbol{\theta}_i^{(j)},\ j = 1, \ldots, R\}$ of an MCMC sampler targeting the power posterior for temperature T_i. The power posterior method is described in detail in Algorithm 10.

To calculate the Monte Carlo standard error of the log marginal likelihood we follow the techniques described by Friel and Pettitt (2008). Let $X_i = \hat{E}_{\boldsymbol{\theta}|\boldsymbol{y},T_i}[\log \pi(\boldsymbol{y}|\boldsymbol{\theta}, r)]$ denote this expectation estimated using the MCMC output. Then we can rewrite equation (4.15)

as

$$\begin{aligned}\log \hat{\pi}(\boldsymbol{y}|r) &= \sum_{i=0}^{n-1} \left(\frac{T_{i+1} - T_i}{2} \right) (X_{i+1} + X_i) \\ &= \frac{1}{2} \sum_{i=0}^{n-1} (T_{i+1}X_{i+1} - T_iX_i + T_{i+1}X_i - T_iX_{i+1}) \\ &= \frac{1}{2} \left\{ \sum_{i=0}^{n} T_iX_i - \sum_{i=0}^{n-1} T_iX_i + \sum_{i=0}^{n-1} T_{i+1}X_i - \sum_{i=1}^{n} T_{i-1}X_i \right\} \\ &= \frac{1}{2} \left\{ T_nX_n - T_0X_0 + \sum_{i=1}^{n-1} (T_{i+1} - T_{i-1})X_i \right\}.\end{aligned}$$

Therefore

$$\begin{aligned}Var\{\log \hat{\pi}(\boldsymbol{y}|r)\} = \frac{(T_1 - T_0)^2}{4} \mathrm{Var}(X_0) + \frac{(T_n - T_{n-1})^2}{4} \mathrm{Var}(X_n) \\ + \sum_{i=1}^{n-1} \frac{(T_{i+1} - T_{i-1})^2}{4} \mathrm{Var}(X_i). \qquad (4.16)\end{aligned}$$

Thus we can estimate $Var\{\log \hat{\pi}(\boldsymbol{y}|r)\}$ by using estimates of the $Var(X_i) = MCSE_i^2$, where the $MCSE_i$ are simply the individual Monte Carlo standard errors (Roberts, 1996). We will use the batch means method to estimate these individual Monte Carlo standard errors, as this is a straightforward method to implement. Essentially, the batch means method estimates the standard error by looking at the variability of a collection of mean values, where these means are obtained after splitting the data into a collection of batches. Suppose we have $N = ab$ iterations of a Markov chain $\{Z_i\}$, where b is the batch size and a is the number of batches. We can estimate $\mu = E[g(Z)]$ using $\hat{\mu} = \sum_{i=1}^{N} g(Z_i)/N$ (Flegal, 2008). Consider the a batch means

$$Y_k = \frac{1}{b} \sum_{i=(k-1)b+1}^{kb} g(Z_i), \qquad k = 1, \ldots, a.$$

If the batch size b is chosen so that the Y_k are (almost) independent then the Y_k can be thought of as a random sample with variance $Var\{g(Z)\}/b$ and so we can estimate $Var\{g(Z)\}$ using

$$\widehat{Var}\{g(Z)\} = \frac{b}{a-1} \sum_{k=1}^{a} (Y_k - \hat{\mu})^2.$$

We use $b = \sqrt{N}$ as suggested in Alexopoulos et al. (1997). Thus we can estimate the Monte

Carlo error of $\hat{\mu}$ using

$$\widehat{MCSE}(\hat{\mu})^2 = \frac{\widehat{Var}\{g(Z)\}}{N}.$$

Using this method we can determine estimates for the $Var(X_i) = MCSE_i^2$ and thereby estimate the standard error of the estimate of log marginal likelihood as

$$\sqrt{\left\{ \frac{(T_1 - T_0)^2}{4} \widehat{MCSE}_0^2 + \sum_{i=1}^{n-1} \frac{(T_{i+1} - T_{i-1})^2}{4} \widehat{MCSE}_i^2 + \frac{(T_n - T_{n-1})^2}{4} \widehat{MCSE}_n^2 \right\}}.$$

Application to our hidden Markov model

In order to use to power posterior method on HMMs we can use data augmentation or we can marginalise over the hidden segmentation, $\boldsymbol{s}$. If we use data augmentation, we have the advantage that $\pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{s}, r)$ follows an exponential family as it is a multinomial distribution with a fixed sequence length. This means that when we take this distribution to the power of t, it is still a member of the same exponential family. Therefore sampling from the power posterior $\pi_T(\boldsymbol{\theta}, \boldsymbol{s}|\boldsymbol{y}, r)$ is simple when a conjugate prior is used. In order to simulate realisations from the power posterior $\pi_T(\boldsymbol{\theta}, \boldsymbol{s}|\boldsymbol{y}, r)$ for the HMM at temperature T we construct an MCMC scheme with two blocks: $\boldsymbol{\theta}$ and $\boldsymbol{s}$. Now

$$\begin{aligned}
\pi_T(\boldsymbol{\theta}, \boldsymbol{s}|\boldsymbol{y}, r) &\propto \pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{s}, r)^T \pi(\boldsymbol{\theta}, \boldsymbol{s}|r) \\
&= \pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{s}, r)^T \pi(\boldsymbol{s}|\Lambda, r) \pi(\Lambda|r) \pi(\mathcal{P}|r) \\
&= \left\{ \prod_{k=1}^{r} \prod_{i=1}^{f} \prod_{j=1}^{f} (p_{ij}^{(k)})^{n_{ij}^{(k)} T} (p_{ij}^{(k)})^{a_{ij}^{(k)} - 1} \right\} \left\{ \prod_{k=1}^{r} \prod_{l=1}^{r} (\lambda_{kl})^{m_{kl}} (\lambda_{kl})^{b_{kl}} \right\}.
\end{aligned}$$

Therefore the full conditional power posterior distributions for $\boldsymbol{\theta}$ are

$$p_i^{(k)}|\boldsymbol{s}, r, \boldsymbol{y}, T \sim \mathcal{D}(\boldsymbol{a}_i^{(k)} + T\boldsymbol{n}_i^{(k)}) \quad \text{and} \quad \lambda_k|\boldsymbol{s}, r, \boldsymbol{y} \sim \mathcal{D}(\boldsymbol{b}_k + \boldsymbol{m}_k),$$

and so iterates from the $\boldsymbol{\theta}$ block are straightforward to obtain. Also iterates from the $\boldsymbol{s}$ block can be obtained by using an adapted version of the forward–backward algorithm; see Algorithm 11.

4.2.10 Chib's method and application to HMMs

A further method for estimating the marginal likelihood using parameter samples from the posterior distribution can be found in Chib (1995). This approach is based on a rearrangement of Bayes theorem:

$$\pi(\boldsymbol{y}|r) = \frac{\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)\pi(\boldsymbol{\theta}|r)}{\pi(\boldsymbol{\theta}|\boldsymbol{y}, r)}. \tag{4.17}$$

Algorithm 11 The adapted forward–backward algorithm.

1. Calculate the forward probabilities, $f_k(t)$, using the adapted forward filter given in Algorithm 12, for $k = 1, \ldots, r$ and $t = 1, \ldots, n$.
2. Calculate the backward probabilities, $b_j(k, t)$, using the backward filter given in Algorithm 8, for $j = 1, \ldots, r$, $k = 1, \ldots, r$ and $t = 1, \ldots, n$.
3. Sample from $1, \ldots, r$ with probabilities $f_k(n)$ for $k = 1, \ldots, r$ to obtain S_n.
4. For $i = n-1, \ldots, 1$ and $j = 1, \ldots, r$ sample from $1, \ldots, r$ with probabilities $b_j(k, i-1)$ to obtain $S_{n-1}, \ldots, S_1$.

Here the numerator is the likelihood multiplied by the prior and the denominator is the posterior density for $\boldsymbol{\theta}$. A key point to note is that this equality holds for all values of $\boldsymbol{\theta}$. Also the observed data likelihood $\pi(\boldsymbol{y}|\boldsymbol{\theta}, r)$ can be obtained from the forward filter (Algorithm 7). Further the marginal posterior density $\pi(\boldsymbol{\theta}|\boldsymbol{y}, r) = E_{\boldsymbol{s}|\boldsymbol{y},r}[\pi(\boldsymbol{\theta}|\boldsymbol{s}, \boldsymbol{y}, r)]$ can be approximated by using the output of the standard HMM MCMC scheme (Algorithm 5) as, marginally, this gives realisations from $\boldsymbol{s}|\boldsymbol{y}, r$, that is, use the approximation

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}, r) \simeq \frac{1}{M} \sum_{p=1}^{M} \pi(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{s}^{(p)}, r). \tag{4.18}$$

We note that Chib recommends that a value $\boldsymbol{\theta}^*$ is used in Equation (4.17) which is a point with high posterior density as this produces a more accurate value of the marginal likelihood. The rationale here is that for an MCMC run of a given length, we can expect the posterior density to be more accurate at high density points as there are more samples at high density points in comparison to low density points in the tails of the distribution. The algorithm is given in Algorithm 13. In our case we use

$$\begin{aligned}\hat{\pi}(\boldsymbol{\theta}|\boldsymbol{y}, r) &= \frac{1}{M} \sum_{p=1}^{M} \pi(\mathcal{P}, \Lambda|\boldsymbol{y}, \boldsymbol{s}^{(p)}, r) \\ &= \frac{1}{M} \sum_{p=1}^{M} \left\{ \prod_{k=1}^{r} \prod_{i=1}^{f} \mathcal{D}(\boldsymbol{p}_i^{(k)}|\boldsymbol{a}_i^{(k)} + \boldsymbol{n}_i^{(k)}) \mathcal{D}(\boldsymbol{\lambda}_k|\boldsymbol{b}_k + \boldsymbol{m}_k) \right\},\end{aligned}$$

where $\boldsymbol{n}_i^{(k)}$ and $\boldsymbol{m}_k$ depend on $\boldsymbol{s}^{(p)}$. Thus, using Equation (4.17), the estimate of log marginal likelihood is

$$\log \hat{\pi}(\boldsymbol{y}|r) = \log \left\{ \frac{\pi(\boldsymbol{y}|\boldsymbol{\theta}^*, r)\pi(\boldsymbol{\theta}^*|r)}{\hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r)} \right\} = \log \left\{ \pi(\boldsymbol{y}|\boldsymbol{\theta}^*, r)\pi(\boldsymbol{\theta}^*|r) \right\} - \log \hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r).$$

Algorithm 12 The adapted forward filter.

1. Initialise the forward probabilities

$$f_k(1) = Pr(S_1 = k|y_1) = \frac{Pr(Y_1 = y_1|S_1 = k)^T Pr(S_1 = k)}{\sum_{l=1}^{r} Pr(Y_1 = y_1|S_1 = l)^T Pr(S_1 = l)} = \frac{\pi_{y_1}^{(k)T} \pi_k}{\sum_{l=1}^{r} \pi_{y_1}^{(l)T} \pi_l},$$

   where $Pr(S_1 = k) = \pi_k$ and $Pr(Y_1 = y_1|S_t = k) = \pi_{y_1}^{(k)}$.

2. Calculate the forward probabilities using forward recursions

$$f_k(t) = \frac{Pr(Y_t = y_t|S_t = k, \boldsymbol{y}^{t-1})^T Pr(S_t = k|\boldsymbol{y}^{t-1})}{\sum_{k'=1}^{r} Pr(Y_t = y_t|S_t = k', \boldsymbol{y}^{t-1})^T Pr(S_t = k'|\boldsymbol{y}^{t-1})} = \frac{\left\{p_{y_{t-1},y_t}^{(k)}\right\}^T \sum_{l=1}^{r} \lambda_{lk} f_l(t-1)}{\sum_{k'=1}^{r} \left[\left\{p_{y_{t-1},y_t}^{(k')}\right\}^T \sum_{l=1}^{r} \lambda_{lk'} f_l(t-1)\right]},$$

   where

$$Pr(S_{t+1} = k|\boldsymbol{y}^t) = \sum_{l=1}^{r} Pr(S_t = l, S_{t+1} = k|\boldsymbol{y}^t) = \sum_{l=1}^{r} Pr(S_{t+1} = k|S_t = l) Pr(S_t = l|\boldsymbol{y}^t) = \sum_{l=1}^{r} \lambda_{lk} f_l(t).$$

The Monte Carlo variability of this estimate is

$$Var\{\log \hat{\pi}(\boldsymbol{y}|r)\} = Var\{\log \hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r)\} = \frac{Var\{\hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r)\}}{\hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r)^2}$$

using a delta method approximation (Chib, 1995). Let $h^{(p)} = \pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \boldsymbol{s}^{(p)}, r)$ so that our estimate is $\hat{h} = \sum_{p=1}^{M} h^{(p)}/M$. We can determine an estimate of the Monte Carlo variability

Algorithm 13 Chib's method.

1. Run Gibbs sampling (Algorithm 5) to obtain samples from $\pi(\boldsymbol{\theta}, \boldsymbol{s}|r, \boldsymbol{y})$.
2. Choose $\boldsymbol{\theta}^*$ to be a point with high density.
3. Approximate $\pi(\boldsymbol{y}|\boldsymbol{\theta}^*, r)$ from the forward filter (Algorithm 7).
4. Evaluate $\log \pi(\boldsymbol{y}|r) = \log \pi(\boldsymbol{y}|\boldsymbol{\theta}^*, r) - \log \pi(\boldsymbol{\theta}^*|r) + \log \pi(\boldsymbol{\theta}^*|\boldsymbol{y}, r)$.

of the estimate of log marginal likelihood using

$$\widehat{Var}\{\hat{\pi}(\boldsymbol{\theta}^*|\boldsymbol{y}, r)\} = \frac{1}{M}\left(\hat{\gamma}(0) + 2\sum_{p=1}^{2\delta+1}\hat{\gamma}(p)\right) \tag{4.19}$$

where

$$\hat{\gamma}(k) = \frac{1}{M}\sum_{p=1}^{M-k}(h^{(p)} - \hat{h})(h^{(p+k)} - \hat{h})$$

is the estimated autocovariance at lag k and δ is the smallest positive integer satisfying $\hat{\gamma}(2\delta) + \hat{\gamma}(2\delta + 1) > 0$. This latter condition provides an effective way of truncating the sum of the lagged autocovariances in Equation (4.19) as, beyond lag $2\delta + 1$, the autocovariances are negligible.

4.2.11 Choosing a method to determine r

Chib's method is fairly straightforward to implement as it uses existing (and simple) algorithms for inference conditional on r. That said, the power posterior method has been shown to perform more accurately for this type of HMM than Chib's method (Germain, 2010). However the power posterior method has slightly longer computation times though it also is simple to implement.

Yet another strategy to obtaining the (marginal) posterior distribution for r is to use a reversible jump MCMC algorithm in which r is treated as an unknown parameter of the model; see Boys and Henderson (2001b). However, these authors have also shown that such algorithms mix very poorly and are very computationally inefficient.

In the next chapter we compare the performance of the three approaches described above using simulated data.

Chapter 5

Application of methodology

In this chapter we will apply the methods described in Chapter 4 to sequence data. We begin by showing that the methods are accurate on a simulated data example before moving on to apply the techniques to the two groups of proteins identified in Chapter 3. All computer codes used in this chapter are available in an R package we have developed at `https://github.com/nina88/HMMs`.

5.1 Gibbs sampling using simulated data

We now examine the performance of the Gibbs sampling algorithm (Algorithm 5) by examining the results when using simulated data. We also use the label switching algorithm (Algorithm 9) so that the thinned MCMC output (after convergence) can be used to (i) estimate the transition matrices for $\mathcal{P}$ and Λ; (ii) plot the probability of being in each segment type for each amino acid and (iii) plot the probability of changing segment types. Space restrictions prevent this thesis from listing the results for all simulated datasets; here we only include example output which is typical of the full set of results.

We begin by simulating a sequence with a $f = 4$ letter alphabet (four states) and of length $5k$. There are $r = 3$ different types of segment and the transition matrices are

$$P^{(1)} = \begin{pmatrix} 0.35 & 0.15 & 0.25 & 0.25 \\ 0.35 & 0.20 & 0.05 & 0.40 \\ 0.35 & 0.20 & 0.20 & 0.25 \\ 0.25 & 0.15 & 0.20 & 0.40 \end{pmatrix}, \qquad P^{(2)} = \begin{pmatrix} 0.25 & 0.30 & 0.15 & 0.30 \\ 0.05 & 0.45 & 0.05 & 0.45 \\ 0.15 & 0.25 & 0.25 & 0.35 \\ 0.05 & 0.40 & 0.05 & 0.50 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.30 & 0.10 & 0.55 & 0.05 \\ 0.30 & 0.25 & 0.15 & 0.30 \\ 0.35 & 0.20 & 0.35 & 0.10 \\ 0.10 & 0.10 & 0.75 & 0.05 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.9990 & 0.0005 & 0.0005 \\ 0.0010 & 0.9980 & 0.0010 \\ 0.0050 & 0.0050 & 0.9900 \end{pmatrix}.$$

The analysis for the 'typical' sequence (after correcting for label switching) gives the posterior means

$$P^{(1)} = \begin{pmatrix} 0.35 & 0.14 & 0.25 & 0.26 \\ 0.35 & 0.21 & 0.05 & 0.40 \\ 0.34 & 0.23 & 0.19 & 0.24 \\ 0.23 & 0.17 & 0.21 & 0.39 \end{pmatrix}, \qquad P^{(2)} = \begin{pmatrix} 0.21 & 0.34 & 0.15 & 0.30 \\ 0.06 & 0.39 & 0.08 & 0.48 \\ 0.11 & 0.29 & 0.18 & 0.42 \\ 0.05 & 0.40 & 0.05 & 0.49 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.28 & 0.09 & 0.57 & 0.06 \\ 0.26 & 0.25 & 0.15 & 0.34 \\ 0.42 & 0.19 & 0.27 & 0.11 \\ 0.12 & 0.04 & 0.81 & 0.03 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.9995 & 0.0004 & 0.0001 \\ 0.0018 & 0.9964 & 0.0018 \\ 0.0044 & 0.0016 & 0.9940 \end{pmatrix}$$

and (element-wise) posterior standard deviations

$$P^{(1)} = \begin{pmatrix} 0.0136 & 0.0098 & 0.0126 & 0.0125 \\ 0.0181 & 0.0156 & 0.0080 & 0.0188 \\ 0.0173 & 0.0152 & 0.0142 & 0.0154 \\ 0.0117 & 0.0101 & 0.0113 & 0.0135 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.0574 & 0.0646 & 0.0481 & 0.0624 \\ 0.0138 & 0.278 & 0.0151 & 0.0286 \\ 0.0402 & 0.0565 & 0.0474 & 0.0626 \\ 0.0115 & 0.0257 & 0.0115 & 0.0268 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.0481 & 0.0312 & 0.0522 & 0.0263 \\ 0.0697 & 0.0667 & 0.0538 & 0.0744 \\ 0.0466 & 0.0385 & 0.0433 & 0.0305 \\ 0.0574 & 0.0351 & 0.0701 & 0.0310 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.0004 & 0.0003 & 0.0002 \\ 0.0015 & 0.0021 & 0.0015 \\ 0.0036 & 0.0022 & 0.0041 \end{pmatrix}.$$

These summary statistics show that the algorithm provides good and accurate estimates for the transition matrices.

Figure 5.1 shows the probability of being in a particular segment type, the probability of changing segment type and the actual segmentation. Figure 5.1(a) shows the probability of being in segment type one. The areas where the probability is almost one corresponds to the actual segmentation given in Figure 5.1(e). This is also the case for segment type two in Figure 5.1(b) and segment type three in Figure 5.1(c). The change point plot given in Figure 5.1(d) corresponds to the positions that the actual segmentation changes between segments in Figure 5.1(e). Therefore the MCMC scheme provides a accurate indication of where the segment types and the changepoints are when compared to the actual segmentation.

We now look at the effect of increasing the sequence length to $10k$. The analysis of a

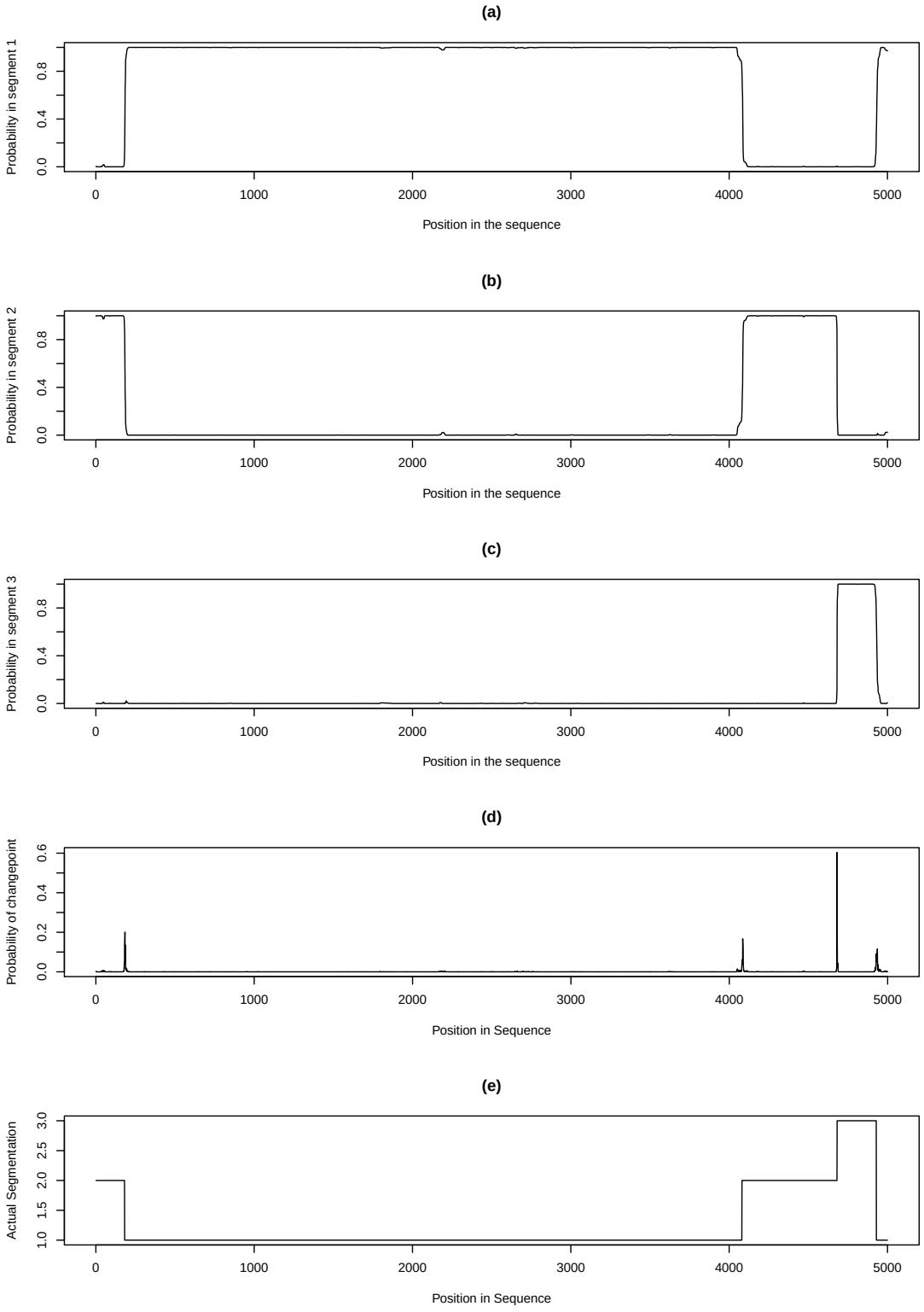


Figure 5.1: (a)–(c) are the probability plots of being in each segment type, (d) is the probability of changing segment type and (e) is the actual segmentation using simulated data of length $5k$.

'typical' sequence gives posterior means

$$P^{(1)} = \begin{pmatrix} 0.33 & 0.17 & 0.26 & 0.25 \\ 0.35 & 0.21 & 0.05 & 0.39 \\ 0.34 & 0.22 & 0.19 & 0.25 \\ 0.25 & 0.16 & 0.20 & 0.39 \end{pmatrix}, \qquad P^{(2)} = \begin{pmatrix} 0.28 & 0.31 & 0.12 & 0.29 \\ 0.05 & 0.45 & 0.05 & 0.44 \\ 0.14 & 0.27 & 0.25 & 0.34 \\ 0.05 & 0.40 & 0.06 & 0.50 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.33 & 0.12 & 0.50 & 0.06 \\ 0.25 & 0.27 & 0.16 & 0.32 \\ 0.38 & 0.18 & 0.33 & 0.11 \\ 0.11 & 0.21 & 0.62 & 0.06 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.9989 & 0.0006 & 0.0006 \\ 0.0016 & 0.9973 & 0.0011 \\ 0.0024 & 0.0042 & 0.9934 \end{pmatrix}$$

and (element-wise) posterior standard deviations

$$P^{(1)} = \begin{pmatrix} 0.0105 & 0.0083 & 0.0101 & 0.0100 \\ 0.0144 & 0.0120 & 0.0065 & 0.0145 \\ 0.0141 & 0.0118 & 0.0116 & 0.0125 \\ 0.0098 & 0.0084 & 0.0087 & 0.0112 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.0331 & 0.0340 & 0.0243 & 0.0327 \\ 0.0068 & 0.0154 & 0.0067 & 0.0152 \\ 0.0248 & 0.0321 & 0.0314 & 0.0343 \\ 00064 & 0.0142 & 0.0066 & 0.0145 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.0286 & 0.0202 & 0.0315 & 0.0143 \\ 0.0339 & 0.0354 & 0.0283 & 0.0358 \\ 0.0258 & 0.0209 & 0.0250 & 0.0158 \\ 0.0295 & 0.0399 & 0.0472 & 0.0236 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.0004 & 0.0003 & 0.0003 \\ 0.0009 & 0.0011 & 0.0007 \\ 0.0019 & 0.0022 & 0.0027 \end{pmatrix}.$$

The posterior means again show that the algorithm provides a good estimate for the transition matrices. Also, as expected, the longer sequence provides a smaller posterior standard deviation due to it providing more information and thereby reducing uncertainty.

In Figure 5.2 we see the probability plots of being in each segment type, the probability of changing segment type and the actual segmentation. Again the method provides a good indication of where the segment types and the changepoints are when compared to the actual segmentation. We note that there is one small section of segment type 3 which is not picked up at position 3425–3437.

5.2 Comparing methods to calculate the marginal likelihood for r

We investigate the performance of averaging the observed data likelihood over the prior, the power posterior method and Chib's method in order to estimate the marginal likelihood. In order to determine which method to use we compare each approximation of the marginal likelihood to the exact marginal likelihood. As the exact marginal likelihood given subsection 4.2.8 is a sum over all possible segmentation sequences, we use a short

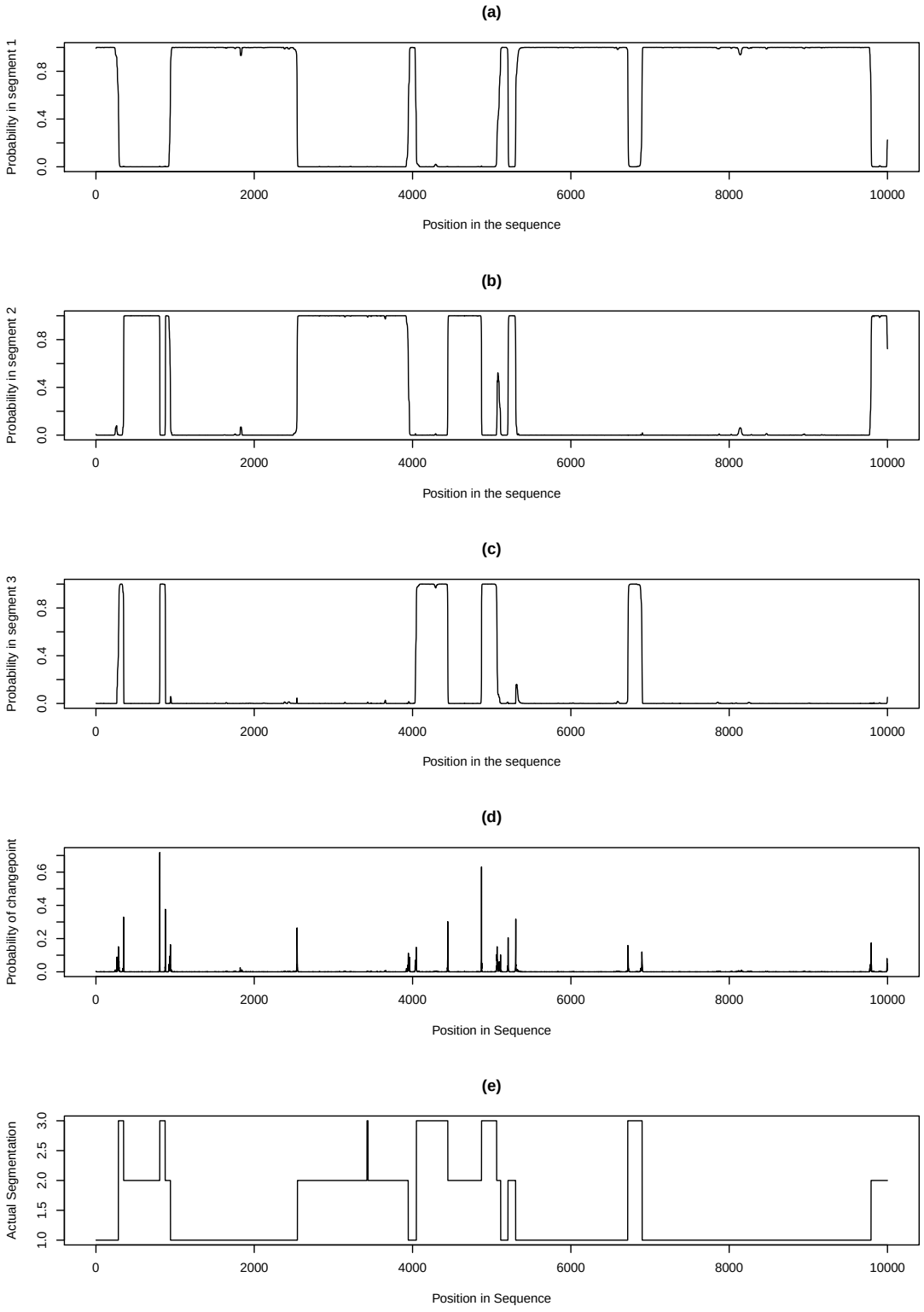


Figure 5.2: (a)–(c) are the probability plots of being in each segment type, (d) is the probability of changing segment type and (e) is the actual segmentation using simulated data of length $10k$.

simulated sequence of length 10 for this simulation study. We fix the number of segment types to be $r = 2$ and so there are $2^{10} = 1024$ possible segmentations. The transition matrices we use for this comparison are

$$P^{(1)} = \begin{pmatrix} 0.35 & 0.65 \\ 0.30 & 0.70 \end{pmatrix}, \qquad P^{(2)} = \begin{pmatrix} 0.05 & 0.95 \\ 0.70 & 0.30 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix}.$$

In the simulation study, we simulate 100 different sequences of length 10 for the observed data $\boldsymbol{y}$ with a $f = 2$ letter alphabet (two states). The marginal likelihood is then calculated exactly by averaging the observed data likelihood over the prior, via Chib's method and using the power posterior method. Note that when averaging the observed data likelihood over the prior we used $10k$ realisations from the prior.

For Chib's algorithm we run the Gibbs sampling algorithm twice, once to choose a high density point and another to estimate the posterior ordinate at the high density point. To estimate a high density point, we run Gibbs sampling without correcting for label switching and use the posterior means of $\mathcal{P}$ and Λ to be the high density points. We then run Gibbs sampling again, correcting for label switching and use the segmentation samples along with the observed data $\boldsymbol{y}$ to evaluate the posterior ordinate at the high density point. For both runs of Gibbs sampling we use $6k$ samples, after removing a burn–in of $2k$ and thinning by 20.

In the power posterior method, we use $10k$ iterations for each value of T after removing a burn in of $4k$ iterations. We also use $T_i = \left(\frac{i}{40}\right)^4$, $i = 1, \ldots, 40$, recommended by Friel and Pettitt (2008), which gives $T_0 = 0$ and $T_{40} = 1$ and also uses more values of T near zero to improve the estimate of the marginal likelihood for r.

We can investigate the accuracy of these estimates of log–marginal likelihood by calculating the variability of the estimation error for the 100 different simulated datasets, this variability being measured as the deviation from the correct value $\log \hat{\pi}(\boldsymbol{y}|r = 2) - \log \pi(\boldsymbol{y}|r = 2)$. Figure 5.3 shows box–and–whisker plots of the empirical distribution of each estimator off the log–marginal likelihood. It shows that all three methods perform very well as the estimation errors (on the log scale) are close to zero. The square root deviations are 0.0116 for the power posterior method, 0.1356 for Chib's method and 0.2742 when averaging the observed data likelihood over the prior. It is clear that the power posterior algorithm has a much smaller error and so, from this limited simulation study, appears to be the most accurate of the three methods. We note that the power posterior method has been shown to perform well for HMMs of a similar type (Germain, 2010).

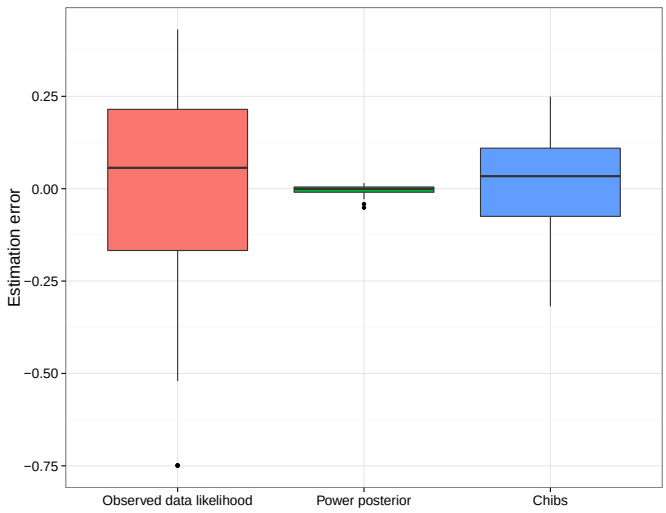


Figure 5.3: Boxplots showing the distribution of the estimation error, $\log \hat{\pi}(\boldsymbol{y}|r=2) - \log \pi(\boldsymbol{y}|r=2)$ for three methods to approximate the marginal likelihood which are averaging the observed data likelihood over the prior, the power posterior method and Chib's method.

5.2.1 The power posterior method

We now investigate the performance of the power posterior method to determine the correct value for r using a simulated dataset. Essentially this method works by obtaining the marginal likelihood for r and then calculating the posterior probability function using Bayes theorem. Full details of the power posterior method are given in Section 4.2.9. Here we illustrate the method by using a simulated dataset of length 10 for the observed data $\boldsymbol{y}$. We use $r = 2$ segment types, $f = 2$ possible states for each element of the sequence, and the same transition matrices for $\mathcal{P}$ and Λ as used in Section 5.2. We assume a prior distribution for r which is based on a truncated version of a Poisson distribution with mean a, where the distribution is truncated above to have largest value u. We will adopt the notation $\text{Pois}(a, u)$ for this distribution. Our chosen prior is $r \sim 1 + \text{Pois}(1, 4)$, with sample space $\{1, 2, 3, 4, 5\}$, as our analysis is an attempt to find a segmentation with relatively few transition structures and is one chosen in agreement with Prof. Doug Gray.

In this section we present two independent analyses of the simulated dataset using the power posterior method. The independent analyses use different seeds for the stochastic elements and so give an indication of the accuracy and stability of the method. In the power posterior method, we use $10k$ iterations for each value of T after removing a burn in of $4k$ iterations. We also use $T_i = \left(\frac{i}{40}\right)^4$, $i = 1, \ldots, 40$, recommended by Friel and Pettitt (2008), which gives $T_0 = 0$ and $T_{40} = 1$ and also uses more values of T near zero to improve the estimate of the marginal likelihood for r. Table 5.2 shows the results for the simulated dataset. Each repeat (independent run) gives a similar value for $\log \pi(\boldsymbol{y}|r)$ and hence for

r	$\log \pi(\boldsymbol{y}\|r)$	$\pi(r\|\boldsymbol{y})$
1	-5.703782	0.3763
2	-5.733802	0.3652
3	-5.734157	0.1825
4	-5.734276	0.0608
5	-5.734336	0.0152

Table 5.1: Exact calculations of the log–marginal likelihood for r and the posterior probability function for r.

	Repeat 1			Repeat 2		
r	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$
1	-5.704	-	0.3749	-5.704	-	0.3749
2	-5.734	0.005	0.3637	-5.731	0.005	0.3646
3	-5.719	0.005	0.1847	-5.722	0.005	0.1840
4	-5.723	0.005	0.0613	-5.724	0.005	0.0612
5	-5.717	0.005	0.0154	-5.725	0.005	0.0153

Table 5.2: This table shows two repeats of the log marginal likelihood estimated using the power posterior approach, the standard error of this approximation and the posterior probability for r for $r = 2, \ldots, 5$. Note that the exact value is used for $r = 1$.

$\pi(r|\boldsymbol{y})$. Also the posterior probability for $r = 1$ is the largest (at over 0.37), although this is not $r = 2$ which is the value we used to simulate the data we can see that this matches the exact calculation as shown in Table 5.1 which also gives the highest posterior probability to $r = 1$. The reason that even the exact calculation picks $r = 1$ is due to the lack of information provided by such a short sequence. The standard error of the estimate of $\log \pi(\boldsymbol{y}|r)$ is close to 0.005 for both runs. Note that we have used the exact value for $r = 1$ as this is easily calculated for any length sequence so we will use the exact value throughout this chapter.

5.2.2 Chib's method

We also implement Chib's method to calculate the marginal likelihood and therefore the posterior probability function for r. We use the same simulated sequences as used in Section 5.2.1. We use $M = 6k$ in Equation (4.17) in order to approximate the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{s}, r)$. The results are given in Table 5.3. We can see that this method gives $r = 2$ the highest posterior probability in each repeat which does not match the $r = 1$ value given by the exact method in Table 5.1 and the standard error of $\log \pi(\boldsymbol{y}|r)$ is larger than the power posterior method at values over 0.1.

	Repeat 1			Repeat 2		
r	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$
1	-5.704	-	0.3643	-5.704	-	0.3617
2	-5.657	0.148	0.3816	-5.654	0.164	0.3803
3	-5.730	0.142	0.1775	-5.704	0.132	0.1809
4	-5.693	0.145	0.0614	-5.679	0.138	0.0618
5	-5.704	0.127	0.0152	-5.698	0.133	0.0152

Table 5.3: This table shows two repeats of the log marginal likelihood estimated using Chib's method, the standard error of this approximation and the posterior probability for r for $r = 2, \ldots, 5$. Note that the exact value is used for $r = 1$.

	Repeat 1			Repeat 2		
r	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$	$\log \pi(\boldsymbol{y}\|r)$	$SE\{\log \pi(\boldsymbol{y}\|r)\}$	$\pi(r\|\boldsymbol{y})$
1	-5.704	-	0.3540	-5.704	-	0.3621
2	-5.634	1.699	0.3795	-5.675	1.707	0.3725
3	-5.648	1.568	0.1872	-5.679	1.573	0.1856
4	-5.635	1.480	0.0632	-5.651	1.516	0.0636
5	-5.818	1.431	0.0161	-5.635	1.423	0.0162

Table 5.4: This table shows two repeats of the log marginal likelihood estimated using the forward filter approach, the standard error of this approximation and the posterior probability for r for $r = 2, \ldots, 5$. Note that the exact value is used for $r = 1$.

5.2.3 Using the forward filter

This method estimates the posterior probability function for r using the observed data likelihood which can be determined from the forward filter (of the forward-backward scheme) as described in Section 4.2.7. Table 5.4 gives the results for the same dataset to those used for the power posterior method in Section 5.2.1. We can see that using this method the posterior probability for $r = 2$ is the highest in both repeats which does not match the exact calculation and the standard error is over 280 times larger in comparison to the standard error of the power posterior method. Therefore we will use the power posterior method to find the marginal likelihood for r for the real data as it is the most accurate of the three methods we have investigated.

5.3 Applying the power posterior method to the group 1 and group 2 proteins

The simulation study indicates that the Gibbs sampling techniques work to segment sequences and the power posterior method is accurate at approximating the marginal

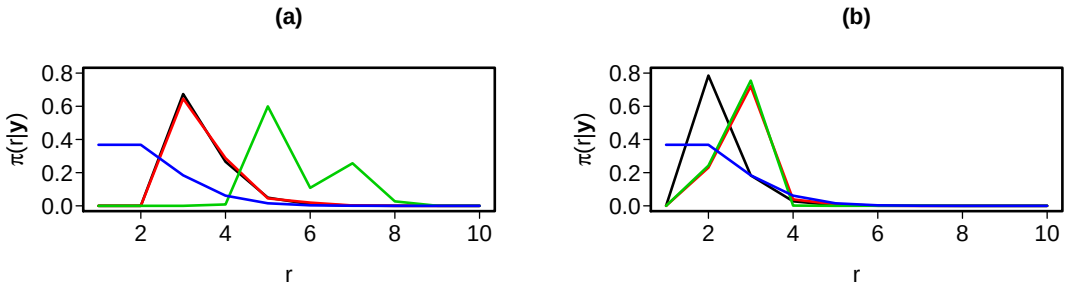


Figure 5.4: Plot of the posterior probability functions $\pi(r|\boldsymbol{y})$ for group 1 and group 2. Plot (a) is a plot of the posterior probability functions for the group 1 proteins for $f = 2$ (black), $f = 3$ (red) and $f = 4$ (green) respectively. Plot (b) is a plot of the posterior probability functions for the group 2 proteins with $f = 2$ (black), $f = 3$ (red) and $f = 4$ (green) respectively. The prior distribution is given in blue.

likelihood and hence choosing a value of r. We now apply the power posterior method and the Gibbs sampling techniques to the two groups of proteins of interest. We are interested to find whether there are common patterns within each group of proteins and so we analyse separately a concatenated sequence of the three proteins in each group. Note that we must also make a slight change in the transition counts to remove any counts of transitions over protein boundaries. These analyses will study patterns in relation to the hydrophobicity and charge of the amino acids. Thus we will have either $f = 2$ when we use hydrophobicity, $f = 3$ when we use charge or $f = 4$ when we use a combination of the two. The appropriate codings are given in Table 4.1. Throughout this section we assume the prior $r \sim 1 + \text{Pois}(1, 9)$, with sample space $\{1, 2, \ldots, 10\}$. We begin by applying the power posterior method to find the posterior probability distribution of r for the different codings ($f = 2, 3, 4$) for the group 1 and group 2 proteins.

5.3.1 Group 1: TAF15, FUS and EWS

In Figure 5.4(a) we can see that for coding using hydrophobicity ($f = 2$) the power posterior method gives the highest probability to $r = 3$, when coding via charge ($f = 3$), the highest posterior probability is when $r = 3$ and coding via both charge and hydrophobicity ($f = 4$) finds that $r = 5$ is most plausible *a posteriori*. We will examine these three cases more closely by determining the transition matrices and segmentation for $r = 3$ when $f = 2$ and $f = 3$ and $r = 5$ for $f = 4$.

5.3.2 Group 2: p53, MDM2 and CBP

In Figure 5.4 (b), we can see that for coding using hydrophobicity ($f = 2$) the power posterior method gives the highest probability to $r = 2$, when coding via charge ($f = 3$), $r = 3$ has the highest posterior probability and coding via both charge and hydrophobicity ($f = 4$) finds that $r = 3$ has the highest posterior probability. We therefore investigate the transition structures by using the Gibbs algorithm with values of $r = 2$ when $f = 2$, $r = 3$ when $f = 3$ and $r = 3$ when $f = 4$.

5.4 Inference for the transition structures in the group 1 and group 2 proteins

5.4.1 Group 1

The power posterior method identified $r = 3$ as the model with the highest posterior probability when $f = 2$ and $f = 3$ and $r = 5$ when $f = 4$. We use the Gibbs sampling algorithm to estimate the segmentations as shown in Figures 5.5, 5.6 and 5.7 for $100k$ iterations. Even though convergence occurs almost immediately we use a burn in of $10k$ and we thin by 20. In Figure 5.5(a) we see that the proteins all tend to be in segment type one for roughly the first half of the protein with a probability of almost one, Figure 5.5(b) and Figure 5.5(c) show that with probability of almost one all three proteins are a combination of segment type 2 and 3 in the second half. In Figure 5.6(a) we see that the proteins all tend to be in segment type one at the beginning of the proteins with a probability of almost one, Figure 5.6(b) shows that the majority of the rest of the proteins are in segment type two and Figure 5.6(c) shows that with probability of almost one TAF15 is in segment type 3 at the end of the protein. Figure 5.7(a) we see that the proteins all tend to be in segment type one at the beginning of the proteins with a probability of almost one, Figure 5.7(b) and Figure 5.7(d) show that the middle of the proteins mostly consist of segment type two with a small section of segment type 4 in TAF15 and FUS and Figure 5.7(c) shows that with probability of almost one the proteins FUS and EWS are in segment type 3 at the end of the protein, where as TAF15 is in segment type 5 as shown in Figure 5.7(e).

This segmentation is interesting as each protein has a very similar segmentation towards the start of the proteins. As $f = 4$ essentially encodes the information for $f = 2$ and $f = 3$, we will concentrate on $f = 4$ for the Group 1 proteins. The posterior mean transition

matrices for $\mathcal{P}$ and Λ when $f = 4$ and $r = 5$ are

$$P^{(1)} = \begin{pmatrix} 0.5035 & 0.4497 & 0.0080 & 0.0387 \\ 0.4015 & 0.5434 & 0.0134 & 0.0418 \\ 0.3804 & 0.2754 & 0.1335 & 0.2106 \\ 0.1912 & 0.6865 & 0.0531 & 0.0693 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.4377 & 0.2449 & 0.1805 & 0.1369 \\ 0.6504 & 0.1375 & 0.1342 & 0.0779 \\ 0.6032 & 0.1515 & 0.1288 & 0.1165 \\ 0.5892 & 0.1657 & 0.0952 & 0.1499 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.6964 & 0.0398 & 0.1713 & 0.0926 \\ 0.3609 & 0.1263 & 0.3350 & 0.1777 \\ 0.8311 & 0.0587 & 0.0698 & 0.0404 \\ 0.1137 & 0.1607 & 0.6606 & 0.0650 \end{pmatrix}, \quad P^{(4)} = \begin{pmatrix} 0.4190 & 0.2165 & 0.1848 & 0.1797 \\ 0.2368 & 0.2037 & 0.1946 & 0.3649 \\ 0.2256 & 0.4725 & 0.1224 & 0.1796 \\ 0.2362 & 0.5686 & 0.1203 & 0.0749 \end{pmatrix},$$

$$P^{(5)} = \begin{pmatrix} 0.7042 & 0.0675 & 0.0334 & 0.1948 \\ 0.5685 & 0.1756 & 0.1585 & 0.0974 \\ 0.5524 & 0.3075 & 0.0657 & 0.0744 \\ 0.0641 & 0.1117 & 0.7656 & 0.0585 \end{pmatrix}, \quad \Lambda = \begin{pmatrix} 0.993 & 0.003 & 0.002 & 0.002 & 0.001 \\ 0.001 & 0.984 & 0.012 & 0.002 & 0.001 \\ 0.002 & 0.007 & 0.980 & 0.008 & 0.003 \\ 0.015 & 0.011 & 0.003 & 0.970 & 0.002 \\ 0.002 & 0.002 & 0.003 & 0.003 & 0.990 \end{pmatrix},$$

and the (element–wise) posterior standard deviations are

$$P^{(1)} = \begin{pmatrix} 0.0315 & 0.0315 & 0.0058 & 0.0155 \\ 0.0284 & 0.0286 & 0.0072 & 0.0124 \\ 0.1764 & 0.1582 & 0.1123 & 0.1359 \\ 0.0853 & 0.1033 & 0.0526 & 0.0502 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.0577 & 0.0386 & 0.0311 & 0.0277 \\ 0.0586 & 0.0406 & 0.0412 & 0.0325 \\ 0.0846 & 0.0620 & 0.0440 & 0.0413 \\ 0.0837 & 0.0650 & 0.0497 & 0.0491 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.0390 & 0.0230 & 0.0383 & 0.0270 \\ 0.1605 & 0.0853 & 0.1351 & 0.1199 \\ 0.0654 & 0.0476 & 0.0337 & 0.0285 \\ 0.0905 & 0.0881 & 0.1319 & 0.0530 \end{pmatrix}, \quad P^{(4)} = \begin{pmatrix} 0.1739 & 0.1145 & 0.1213 & 0.1142 \\ 0.1590 & 0.1229 & 0.1201 & 0.1584 \\ 0.1495 & 0.1611 & 0.0851 & 0.1103 \\ 0.1099 & 0.1324 & 0.1006 & 0.0622 \end{pmatrix},$$

$$P^{(5)} = \begin{pmatrix} 0.1923 & 0.1144 & 0.0730 & 0.0738 \\ 0.1922 & 0.1470 & 0.1126 & 0.1087 \\ 0.1574 & 0.1129 & 0.0857 & 0.1051 \\ 0.0841 & 0.1809 & 0.2611 & 0.0696 \end{pmatrix}, \quad \Lambda = \begin{pmatrix} 0.004 & 0.004 & 0.002 & 0.003 & 0.001 \\ 0.002 & 0.007 & 0.007 & 0.004 & 0.001 \\ 0.003 & 0.006 & 0.009 & 0.007 & 0.004 \\ 0.015 & 0.011 & 0.005 & 0.015 & 0.004 \\ 0.005 & 0.004 & 0.004 & 0.005 & 0.009 \end{pmatrix}.$$

The standard deviations for the transition matrices for segment types 1–3 are generally lower than the transition matrices for segment types 4 and 5 which indicates more uncertainty around segment types 4 and 5.

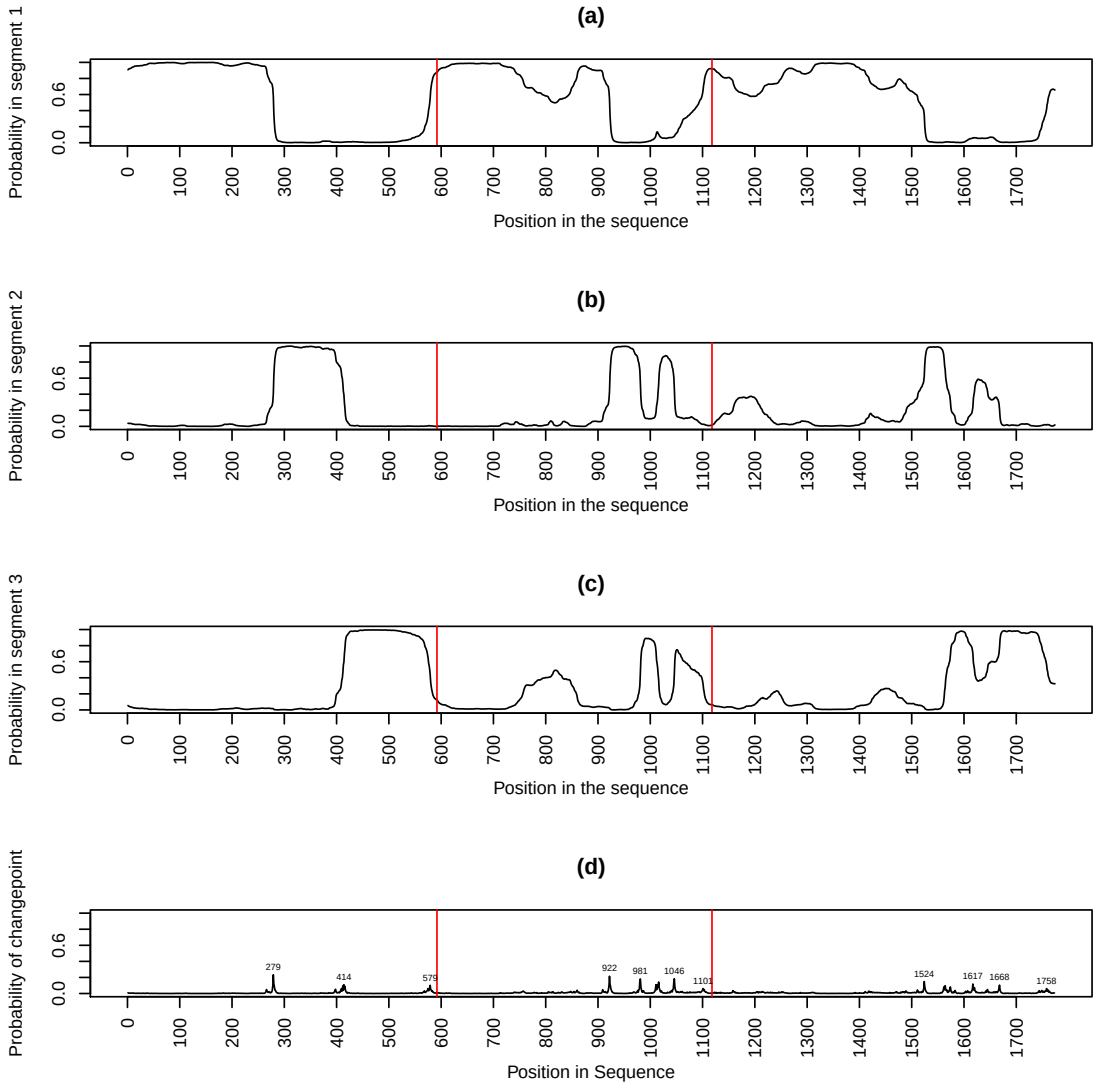


Figure 5.5: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group one proteins. The proteins are joined together with TAF15 first, FUS second and EWS last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

5.4.2 TDP–43

TDP–43 has similar functions and pathologies to FUS but the sequence is very different. It would be interesting to see if these structures are in TDP–43. We have used the forward–backward algorithm with the transition matrices ($f = 4$ case) estimated for the group 1 proteins to estimate a segmentation for TDP–43 to see if these transition structures exist in TDP–43. The plot of the segmentation is given in Figure 5.8 showing that segment type one is present at the end of TDP–43 and segment type 2 at the start until roughly

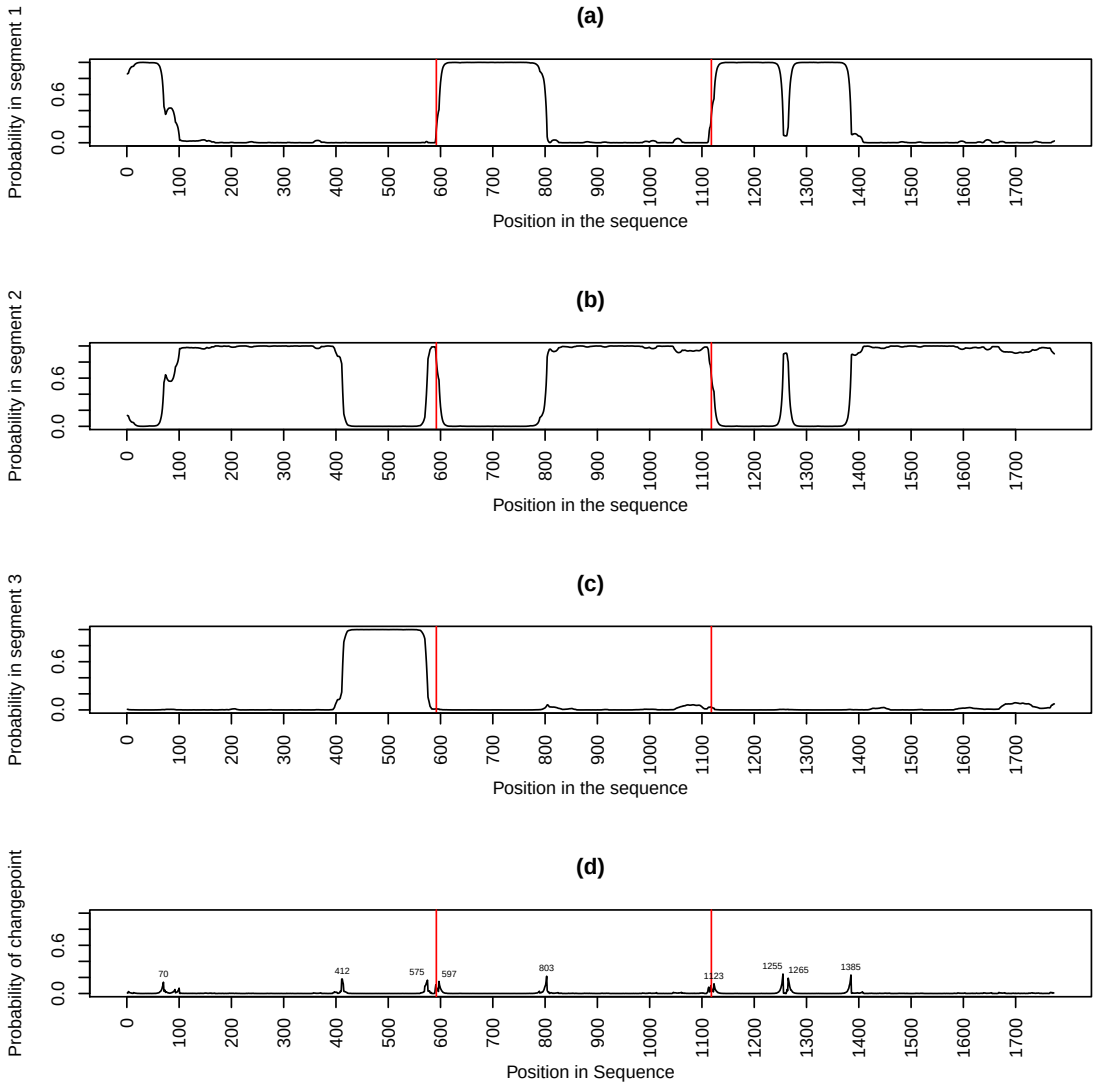


Figure 5.6: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group one proteins. The proteins are joined together with TAF15 first, FUS second and EWS last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

two thirds of the way along the length of the protein. There is a small section of segment type four around amino acids $101 - 119$. Segment types 3 and 5 are not present in the protein. We use the same methods on the concatenated proteins as shown in Figure 5.9 and a similar pattern for the segmentation of TDP–43 occurs.

If we run the power posterior analysis with a concatenation of the group one proteins with TDP–43 for $f = 2, 3$ and 4, then $r = 3, 4$ and 6 are chosen to be the best models respectively. This is shown in Figure 5.10 as $r = 3, 4$ and 6 have the highest posterior probabilities for r.

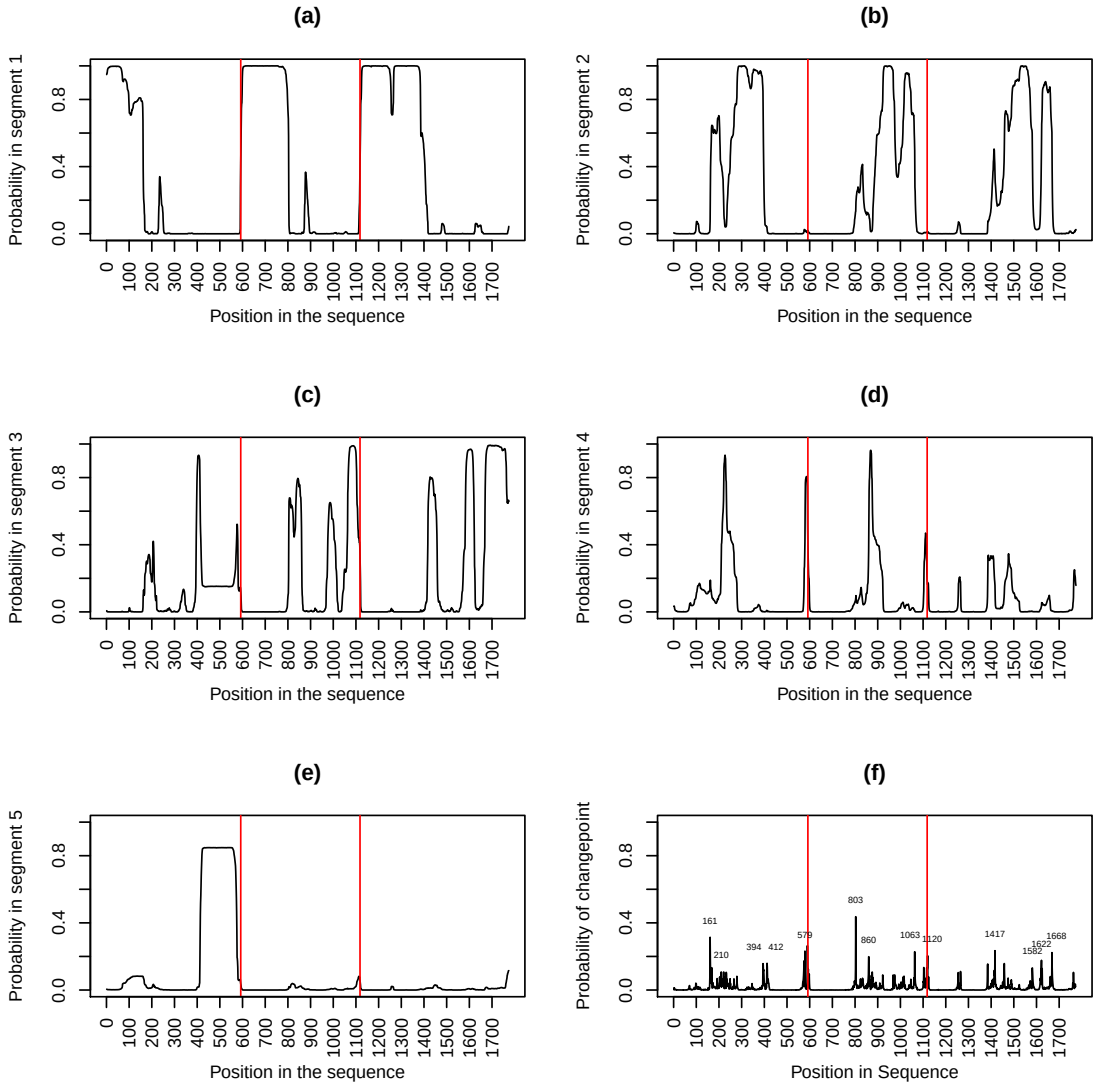


Figure 5.7: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group one proteins. The proteins are joined together with TAF15 first, FUS second and EWS last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

If we run the Gibbs sampling algorithm with the 4 proteins concatenated with $f = 2$ and $r = 3$ and predict transition structures we obtain the segmentation shown in Figure 5.11. We see that there is a difference in the segmentation for TDP–43 as this only consists of segment type 1. For $f = 3$ and $f = 4$ in Figures 5.12 and 5.13 respectively have a majority of TDP–43 in one segment type (labelled segment type two) and one section towards the end of the protein in a different segment type (labelled segment type one). From this analysis it does not look like the structure of TDP–43 is similar to the group one proteins.

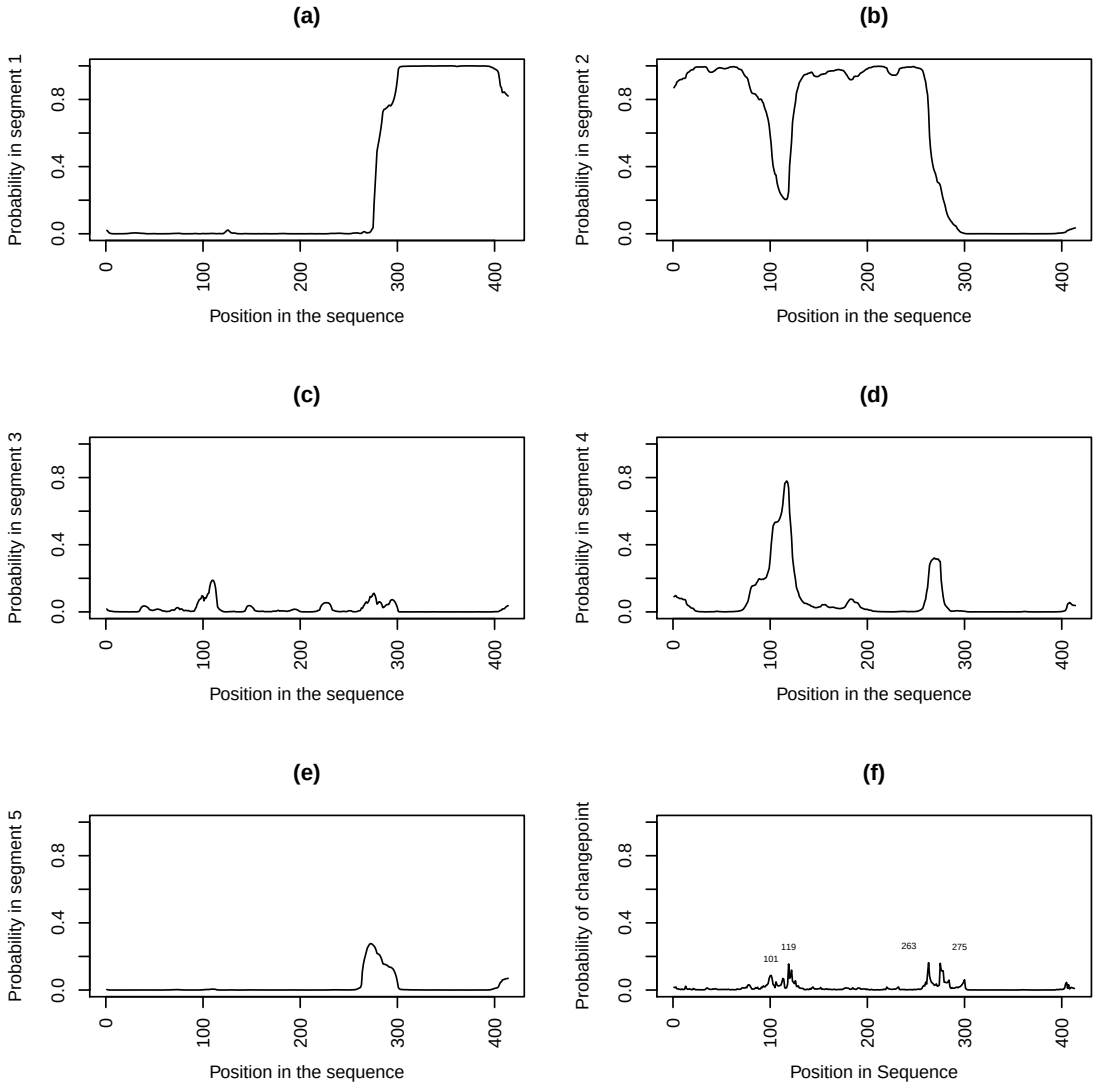


Figure 5.8: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for TDP–43. On the changepoint plot the position of the most probable changepoints are labelled.

5.4.3 Are the group 1 segment types in the group 2 proteins?

We simulate segmentations using the forward–backward algorithm for the group 2 proteins using the posterior mean transition matrices found for the group 1 proteins when $f = 4$. We want to see if a similar segmentation pattern occurs and which sections of the group 2 proteins are similar to the group 1 proteins. Figure 5.14 shows that all of p53 is in segment type 2, MDM2 is a mixture of segment types 2 and 4 and in CBP segment types 1 and 2 are favoured with small sections of 4. Segment type 5 is not present. The segmentation of the group two proteins does not look similar to the group 1 segmentation given in Figure 5.7.

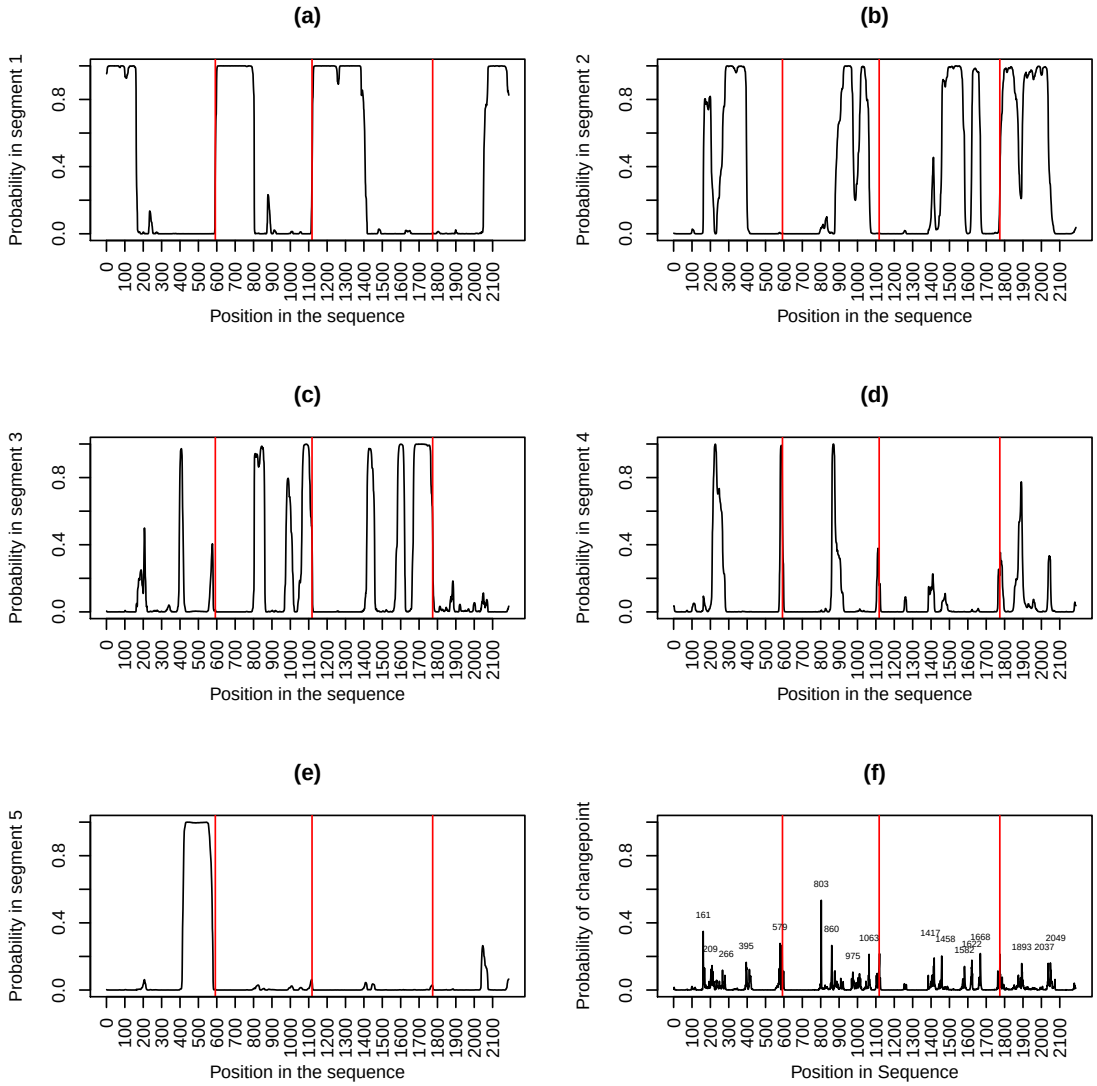


Figure 5.9: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group 1 proteins and TDP–43. The proteins are joined together with TAF15 first, FUS second, EWS third and TDP–43 last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

5.4.4 Are these structures found in the homologues of FUS?

Protein homologues are proteins that are derived from a common ancestor. They may be in different species. We are going to look at two FUS homologues which are Cabeza from *Drosophila* (fruit fly) and FUST–1 from the nematode worm. We first look for the group 1 mean transition structures in Cabeza and FUST–1 using the forward–backward algorithm. In Figures 5.15 and 5.16 we can see that segment type 1 appears at the beginning and a

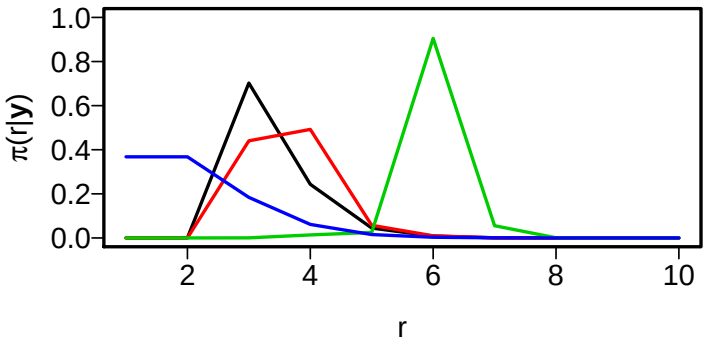


Figure 5.10: Plot of the posterior probability functions for r, $\pi(r|\boldsymbol{y})$ for the group 1 proteins and TDP–43 for $f = 2$ (black), $f = 3$ (red) and $f = 4$ (green) respectively. The prior distribution is given in blue.

mixture of segment types 2,3 and 4 in the second half which is a similar pattern to the group 1 proteins. As we are forcing Cabeza and FUST–1 to choose between the 5 segment types we wanted to check that no other structures exist. Therefore, suppose we have $r = 7$ segment types, 5 similar to those in group 1 and 2 other possible structures. For the structures similar to those in group 1 we take the independent priors, $\boldsymbol{p}_i^{(k)} \sim \mathcal{D}(c_k \boldsymbol{p}_i^{(k)*})$ which gives

$$E(p_{ij}^{(k)}) = p_{ij}^{(k)*} \quad \text{and} \quad Var(p_{ij}^{(k)}) = \frac{1}{c_k + 1} p_{i,j}^{(k)*} \left\{1 - p_{ij}^{(k)*}\right\}.$$

where $\boldsymbol{p}_i^{(k)*}$ is the mean posterior transition matrices for the group 1 proteins. Here we take $c_k = 50$. For the other structures, we use our previous weak independent priors $\boldsymbol{p}_i^{(k)} \sim \mathcal{D}(1, 1, 1, 1)$. Figure 5.17 shows that segments similar to those in group 1 are favoured.

We can concatenate all 5 proteins (group 1 proteins with Cabeza and FUST–1) and perform a power posterior analysis. The results are given in Figure 5.18 which are that the highest posterior probabilities are for $r = 3$ when $f = 2$, $r = 3$ when $f = 4$ and $r = 7$ when $f = 4$ therefore we run the Gibbs sampling algorithm for these values. The results for $f = 2$ and $f = 3$ are shown in Figure 5.19 and 5.20 respectively. We can see that in both cases Cabeza and FUST–1 favour other segment types to the group 1 proteins at the start of the proteins. Figure 5.21 shows the results for $f = 4$ and $r = 7$. These graphs show that Cabeza and FUST–1 follow a similar segmentation structure to the group one proteins. We have run the Gibbs sampling algorithm with $f = 4$ and $r = 5$ and the results are in

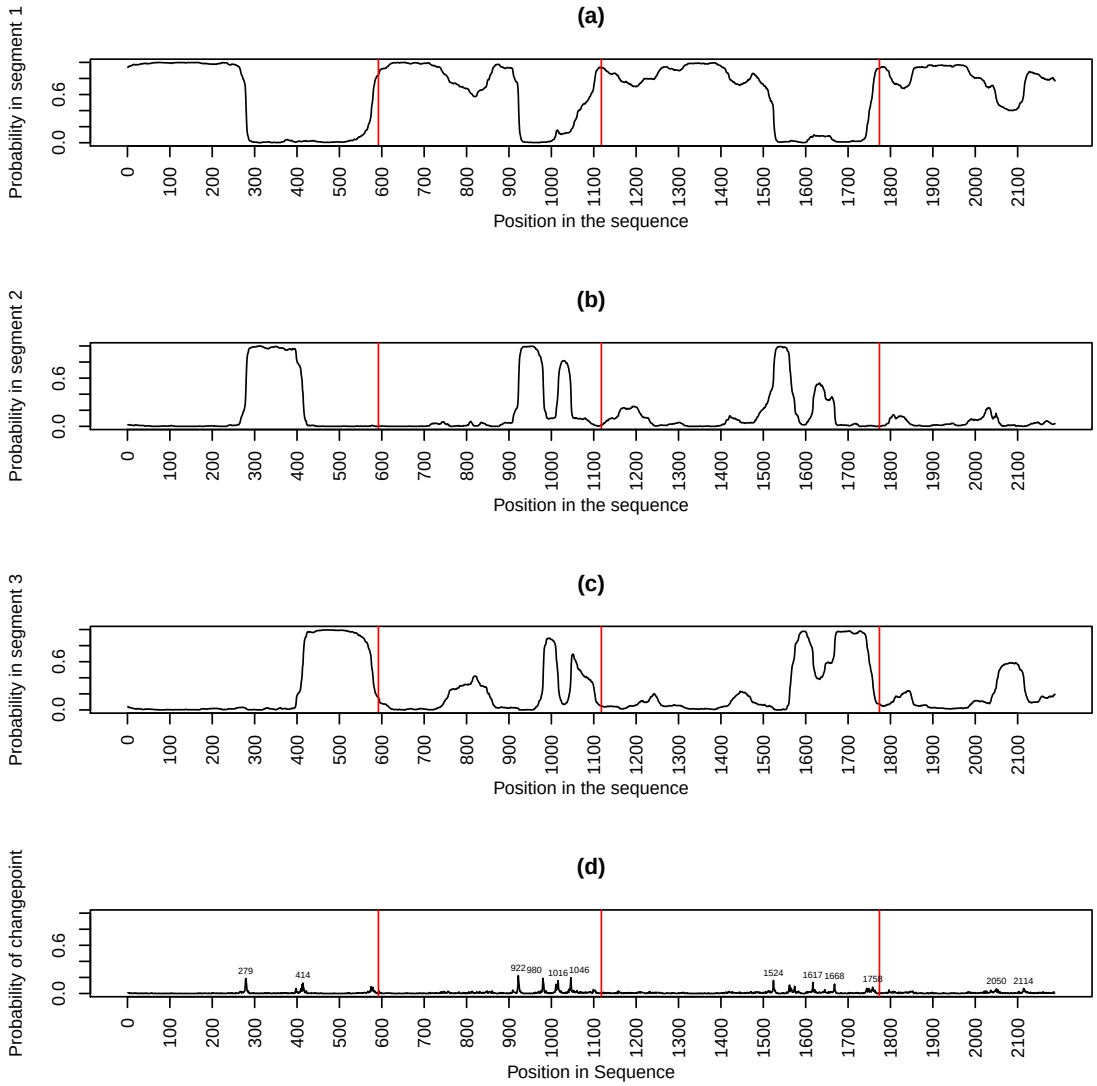


Figure 5.11: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group 1 proteins and TDP–43. The proteins are joined together with TAF15 first, FUS second, EWS third and TDP–43 last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

Figure 5.22. This figure shows that Cabeza and FUST–1 follow a similar segmentation pattern to the group 1 proteins.

5.4.5 How could this information be used to guide experiments?

The hypothesis to come from the segmentation of the group one proteins is that the segments have biological importance. There is little information about the natural function

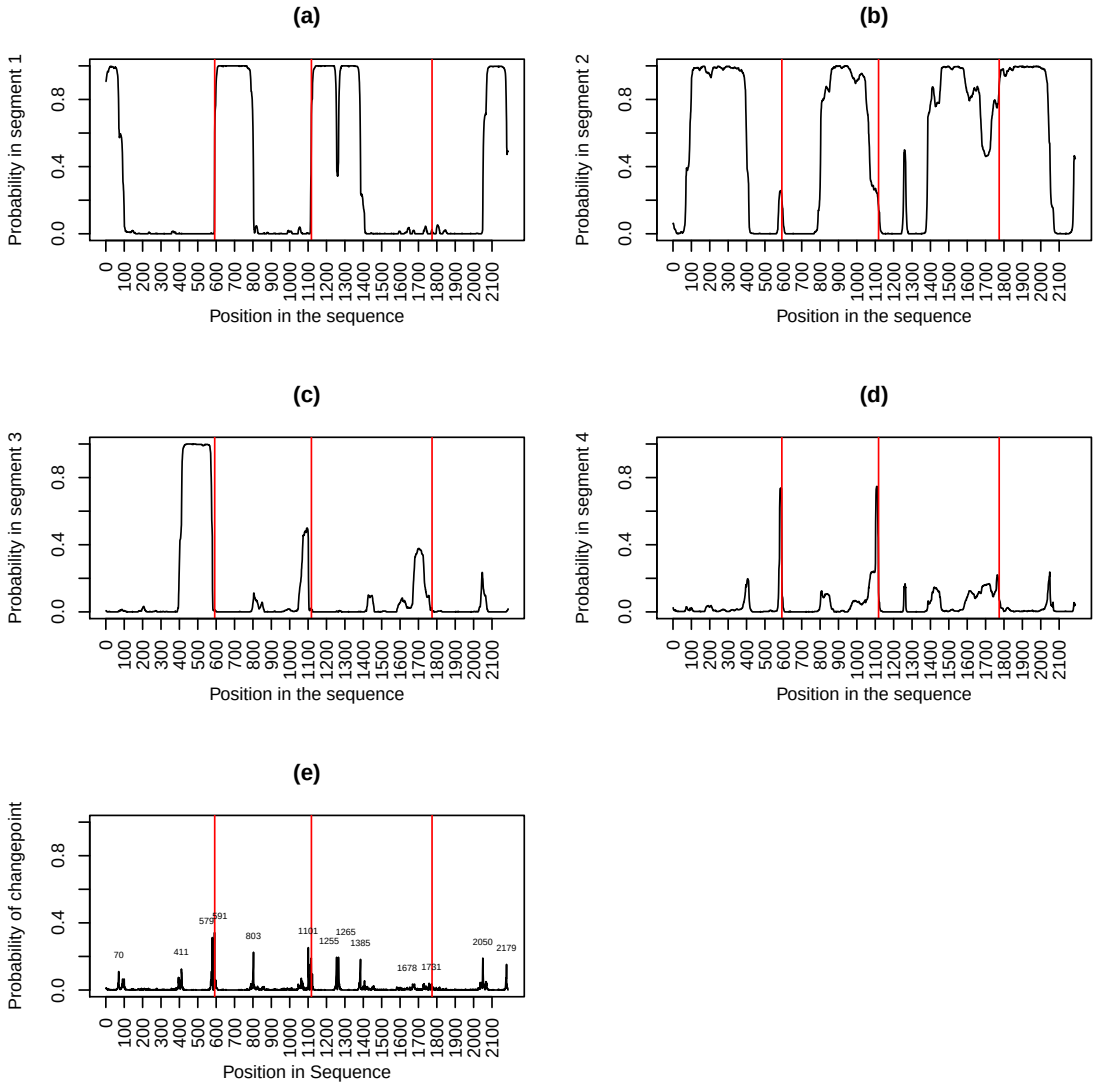


Figure 5.12: Plots of (a)–(d): the probability of being in each segment and (e) the probability of changing segments for the group 1 proteins and TDP–43. The proteins are joined together with TAF15 first, FUS second, EWS third and TDP–43 last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

of the FET proteins making this difficult to test experimentally. Therefore, we test the hypothesis on the abnormal role of the FET proteins following choromosomal translocations as oncogenic fusions.

A fusion protein is a protein that is made when two or more genes from different proteins join together. If this fusion gene is translated, then polypeptides are made which can have properties from all of the original proteins. Cancer cells often contain fusion proteins and these fusion proteins may function as oncoproteins. An oncoprotein is a

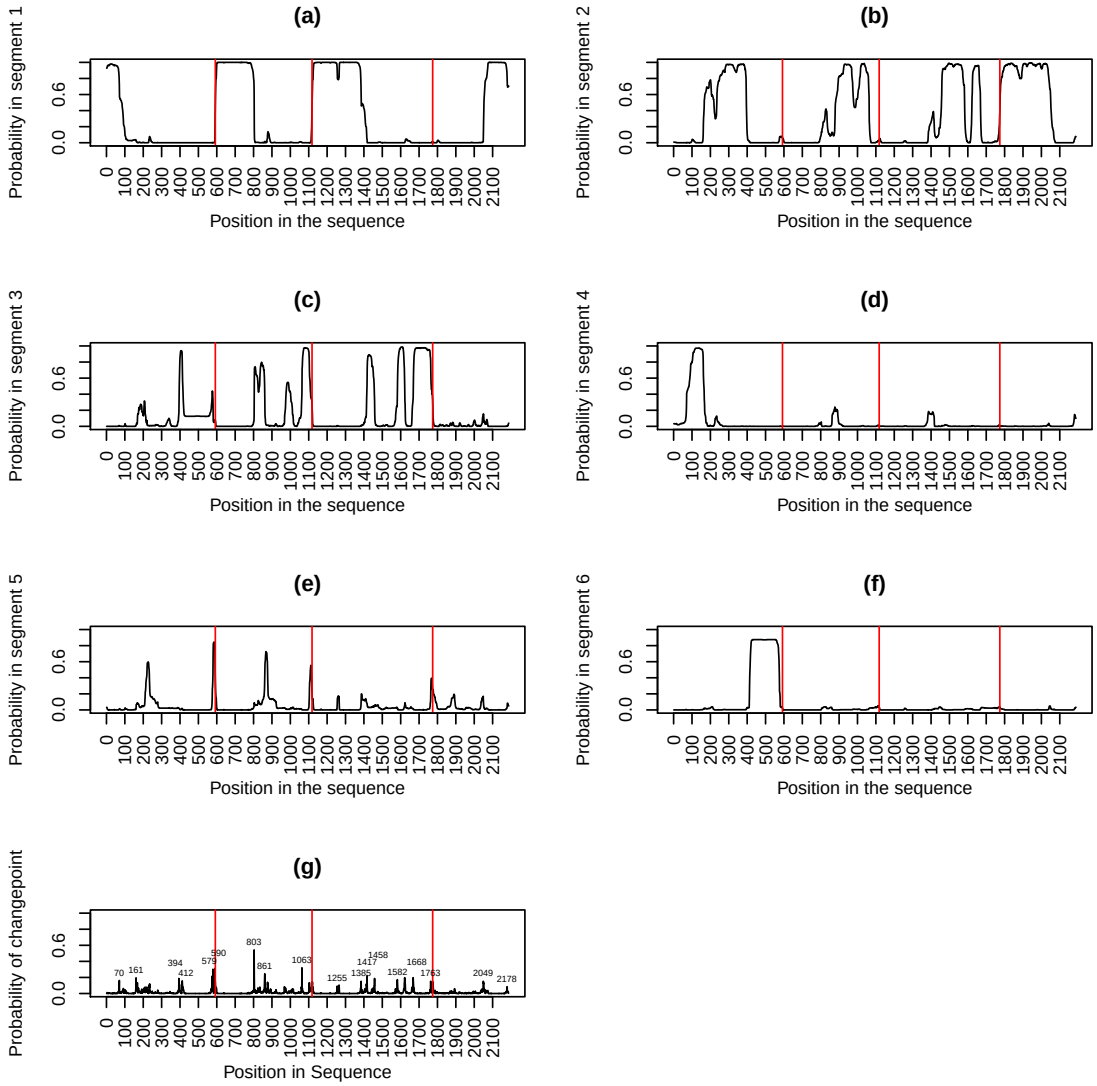


Figure 5.13: Plots of (a)–(f): the probability of being in each segment and (g) the probability of changing segments for the group 1 proteins and TDP–43. The proteins are joined together with TAF15 first, FUS second, EWS third and TDP–43 last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

protein that can cause a cell to transform into a tumour cell. The fusion gene that encodes the oncoprotein is therefore called an oncogene.

The main abnormality in FET genes are fusions to transcription factor genes (Kovar, 2011). In this fusion the RNA–binding domain of FET is replaced with the transcription factor DNA binding domain (Kovar, 2011). FET oncoproteins have been shown to transform cells in culture. An example that has been extensively researched is EWS–FLI1 in Ewing's sarcoma family tumours (Kovar, 2011).

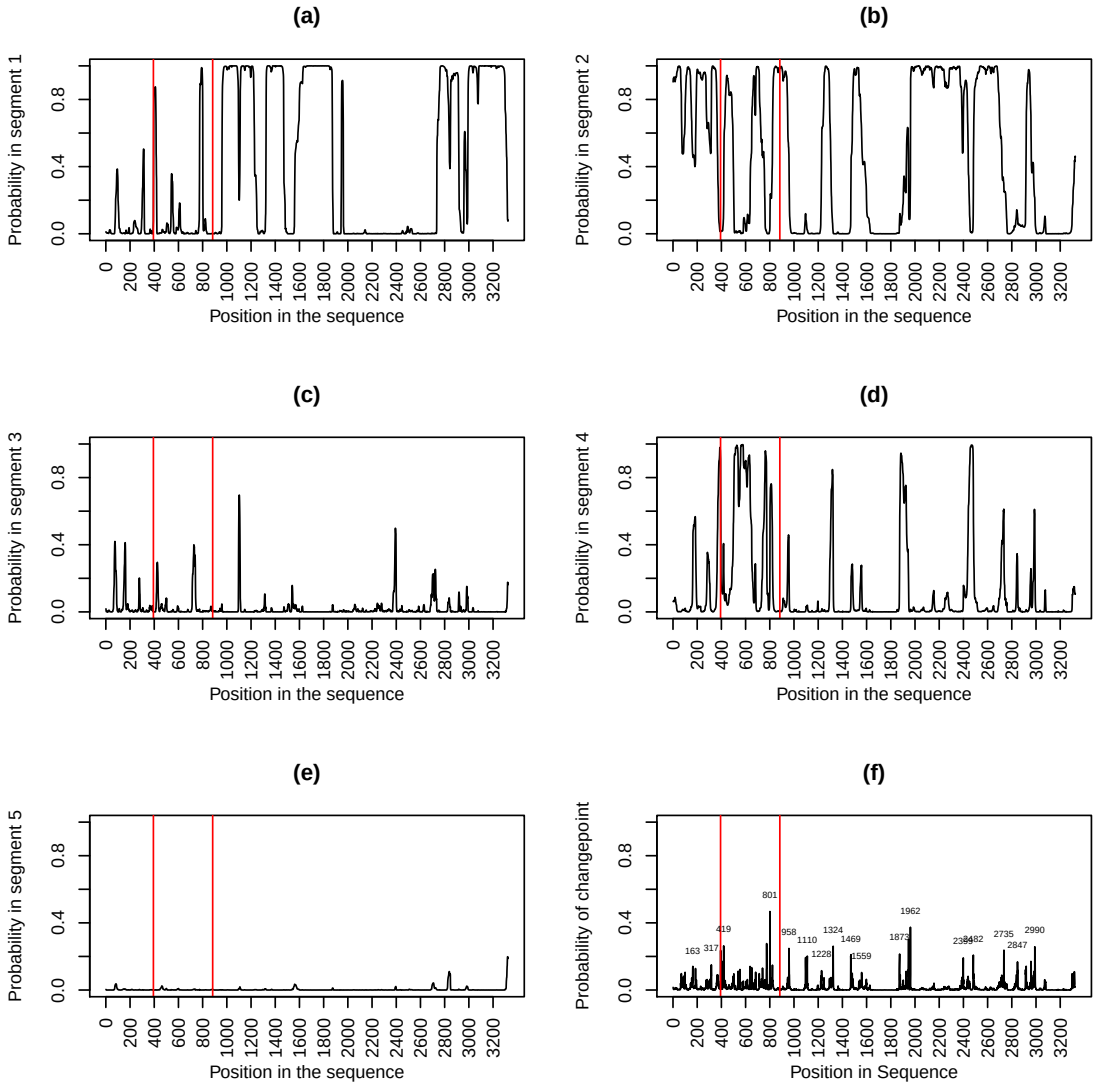


Figure 5.14: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group 2 proteins. The proteins are joined together with p53 first, MDM2 second and CBP last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

Ewing's family tumours are a type of cancer found in bone or nearby soft tissues. This type of cancer is usually found in young adults (Arvand, 2001). It has been found that 85% of these tumours have a chromosome translocation that is detectable. There have been several different transcription factors found fused with EWS which are linked to Ewing's family of tumours. In three of these fusions the location of the break in EWS is very close to the position of the end of segment 1 as shown in Table 5.5.

Another fusion involving TAF15 has been linked to acute myelogenous leukemia which is a cancer of the bone and blood marrow. The point of this fusion is very close to the end

Paper	Protein	With	Location	Segment 1 ends
Delattre et al. (1992)	EWS	FLI1	265	266
Sorensen et al. (1994)	EWS	ERG	264	266
Urano et al. (1996)	EWS	EIA–F	264	266
Martini et al. (2002)	TAF15	TAF15	162	161

Table 5.5: Comparison of where segment type 1 ends and where the FET proteins are cut when oncogenic fusion proteins are made.

of the segment 1 in TAF15 as shown in Table 5.5.

Therefore, regarding oncogenic fusion proteins, see Table 5.5, it appears that segment type 1 is linked, within about 1 or 2 amino acids, to the section of the FET proteins that is involved in an oncogenic fusion.

We use this information to guide experiments which have fusions with and without segment one and compare the oncogenic activity. The prediction is that the strongest oncogenic activity will occur when segment one of a FET protein is used in a fusion.

5.4.6 Group 2

The power posterior method gave $r = 2, 3$ and 3 the highest posterior probabilities when $f = 2, 3$ and 4 respectively. Therefore we chose to investigate the structure of these proteins for these choices of r and f and see whether these analyses provided any insights. Figure 5.23 shows the results when $f = 2$. From this plot we can see that CBP and p53 almost entirely consist of segment type one (Figure 5.23(a)) apart from a small section of segment type 2 in CBP (Figure 5.23(b)). MDM2 is a mixture of both segment types. Figures 5.24(a)–(c) show the results for $f = 3$. The protein p53 is a mixture of all three segment types, MDM2 consists only of segment type two and three and CBP consists of segment type one and two. In the $f = 4$ case (Figures 5.25(a)–(c)) p53 contains only segment type two and three, MDM2 contains one and two only and CBP is a mixture of segment types two and three with a very small section of segment type one. From these plots it is often the case that more than 50% of p53 and/or MDM2 are within one segment type with only small sections in other segment types. There also seems to be no similarity between the segmentations between the group 2 proteins. It is likely that this is due to CBP dominating the analysis due to its size (length 2442) in comparison to p53 and MDM2 (lengths 393 and 490 respectively). Unfortunately our biological expert (Prof. Doug Gray) could not ascribe any biologically interesting features to the different locations of the segment types within the segmentations.

5.5 Experimental methods and results

The experiments resulting from the analysis in this chapter have been completed by Prof. Doug Gray's Laboratory (University of Ottawa) and have used fusions of the FET protein, FUS and the transcription factor, CHOP. They have also used a fusion with Cabeza to see if segment one of Cabeza fused to CHOP will produce the same transforming activity in mouse cells.

The first experiment involves transfecting NIH3T3 cells (introducing DNA into cells) to give cells expressing each of the CHOP fusions shown in Figure 5.26(a), (b), (c) and (e). Figure 5.26(a) represents a construct with the entire sequence of FUS fused to CHOP, Figure 5.26(b) has segment one of FUS fused to CHOP, Figure 5.26(c) have segment one of Cabeza fused to CHOP and Figure 5.26(e) is the control which consists only of CHOP. Six fields from each of three dishes of cells were captured for each construct to obtain microscope images similar to those shown in Figure 5.27. The dark spots in these images are the colonies of cells. These thresholded images were quantified using ImagePro software as shown in Figure 5.28. This software detects the dark spots, counts them and calculates the area of the spots. The results of this analysis are shown in Figure 5.29(a)–(c). We compared each construct to the control (CHOP) using a two–sided two sample t–test.

The first segment identified in FUS by the statistical analysis shows enhanced transforming activity when fused to the CHOP transcription factor (SEGCHOP in Figure 5.26) relative to CHOP alone as shown in Figure 5.29. It is statistically significant in terms of total area, mean area and number of colonies which means the SEGCHOP fusion generates more colonies and there are significantly more cells in a colony. As expected the full length FUSCHOP fusion has transforming activity. The surprise is that segment 1 from the Drosophila Cabeza protein (labelled CABCHOP) shows the strongest transforming activity in mammalian cells as it is statistically significant in terms of total area and number of colonies. This probably relates to expression levels. From a western analysis the CABCHOP protein had the highest level of expression.

Another set of experiments compare the FUSCHOP fusion, segment one of FUS fused to CHOP and a randomised segment one fused to CHOP. These three constructs are shown in Figure 5.26 (a), (b) and (d). The randomised segment one is generated by randomly reordering the amino acids present in segment one of FUS. The resulting fusion is generated commercially. The constructs are transfected into NIH3T3 cells and 15 images of random fields for each fusion is captured. Colony counts and size are determined using ImagePro software.

The results are shown in Figure 5.29 (d)–(f). We compared the constructs to the control (CHOP) using a two–sided two sample t–test. We find that FUSCHOP generate slightly more colonies than SEGCHOP and RCHOP but the colonies are roughly the

same size. This indicates that most of the transforming activity in FUS is in the first segment, but the actual sequence is not important. It is the frequency of amino acids that is important. This matches the computational analysis as when a segmentation is found for the sequence for FUS with a randomised segment one, this is still classified as segment type one as shown in Figure 5.30(a) as the first part of each protein is in segment type one with probability of almost one. The transition structures found are very similar to those found when the segment is not randomised. This could be because the segment still has the same proportion of amino acid types as segment one has roughly equal proportions of ones and twos. This represent amino acids that are neutral and hydrophobic or neutral and hydrophilic respectively. The posterior mean transition matrices for $r = 5$ are

$$P^{(1)} = \begin{pmatrix} 0.5042 & 0.4647 & 0.0047 & 0.0264 \\ 0.3961 & 0.5565 & 0.0099 & 0.0374 \\ 0.4206 & 0.2025 & 0.1539 & 0.2230 \\ 0.2892 & 0.5780 & 0.0735 & 0.0593 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.3642 & 0.2758 & 0.2047 & 0.1552 \\ 0.6933 & 0.1026 & 0.1418 & 0.0623 \\ 0.6437 & 0.1160 & 0.1253 & 0.1150 \\ 0.6592 & 0.1357 & 0.0600 & 0.1450 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.7078 & 0.0505 & 0.1438 & 0.0978 \\ 0.4416 & 0.1597 & 0.2578 & 0.1408 \\ 0.8283 & 0.0833 & 0.0584 & 0.0300 \\ 0.1274 & 0.0971 & 0.7001 & 0.0754 \end{pmatrix}, \quad P^{(4)} = \begin{pmatrix} 0.4674 & 0.2375 & 0.1251 & 0.1699 \\ 0.3834 & 0.2385 & 0.1239 & 0.2542 \\ 0.1351 & 0.5081 & 0.1428 & 0.2140 \\ 0.1936 & 0.6438 & 0.1018 & 0.0608 \end{pmatrix},$$

$$P^{(5)} = \begin{pmatrix} 0.5710 & 0.1052 & 0.1120 & 0.2117 \\ 0.3837 & 0.1687 & 0.2644 & 0.1833 \\ 0.5681 & 0.2417 & 0.0855 & 0.1047 \\ 0.1573 & 0.2245 & 0.5391 & 0.0791 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.994 & 0.001 & 0.002 & 0.002 & 0.001 \\ 0.001 & 0.978 & 0.015 & 0.002 & 0.004 \\ 0.002 & 0.005 & 0.982 & 0.007 & 0.004 \\ 0.007 & 0.016 & 0.002 & 0.973 & 0.002 \\ 0.007 & 0.003 & 0.003 & 0.005 & 0.981 \end{pmatrix},$$

and the (element–wise) posterior standard deviations are

$$P^{(1)} = \begin{pmatrix} 0.0295 & 0.0280 & 0.0040 & 0.0122 \\ 0.0257 & 0.0255 & 0.0057 & 0.0103 \\ 0.1742 & 0.1393 & 0.1201 & 0.1411 \\ 0.0982 & 0.1159 & 0.0720 & 0.0440 \end{pmatrix}, \quad P^{(2)} = \begin{pmatrix} 0.0625 & 0.0395 & 0.0415 & 0.0312 \\ 0.0629 & 0.0489 & 0.0471 & 0.0327 \\ 0.0853 & 0.0641 & 0.0430 & 0.0428 \\ 0.0916 & 0.0667 & 0.0423 & 0.0515 \end{pmatrix},$$

$$P^{(3)} = \begin{pmatrix} 0.0329 & 0.0285 & 0.0328 & 0.0379 \\ 0.1070 & 0.0853 & 0.0949 & 0.0814 \\ 0.0789 & 0.0599 & 0.0288 & 0.0222 \\ 0.1058 & 0.0671 & 0.1431 & 0.0603 \end{pmatrix}, \quad P^{(4)} = \begin{pmatrix} 0.1322 & 0.0641 & 0.0708 & 0.0853 \\ 0.1381 & 0.1116 & 0.0872 & 0.1186 \\ 0.1050 & 0.1255 & 0.0757 & 0.0936 \\ 0.0751 & 0.0965 & 0.0750 & 0.0473 \end{pmatrix},$$

$$P^{(5)} = \begin{pmatrix} 0.2524 & 0.1185 & 0.1083 & 0.1508 \\ 0.2294 & 0.1473 & 0.1626 & 0.1499 \\ 0.2157 & 0.1457 & 0.0818 & 0.1188 \\ 0.1627 & 0.2241 & 0.2882 & 0.0761 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} 0.003 & 0.002 & 0.002 & 0.002 & 0.002 \\ 0.001 & 0.009 & 0.009 & 0.004 & 0.008 \\ 0.003 & 0.006 & 0.009 & 0.007 & 0.004 \\ 0.013 & 0.010 & 0.004 & 0.013 & 0.004 \\ 0.011 & 0.005 & 0.005 & 0.007 & 0.015 \end{pmatrix},$$

These matrices are very similar to the posterior transition structures found for the FET proteins without segment one randomised for segments types one to three which shows that randomising the sequence of segment type one does not change the structure of segment type one. This agrees with the experimental results. The standard deviations for the transition matrices for segment types 1–3 are generally lower than the transition matrices for segment types 4 and 5 which indicates more uncertainty around segment types 4 and 5.

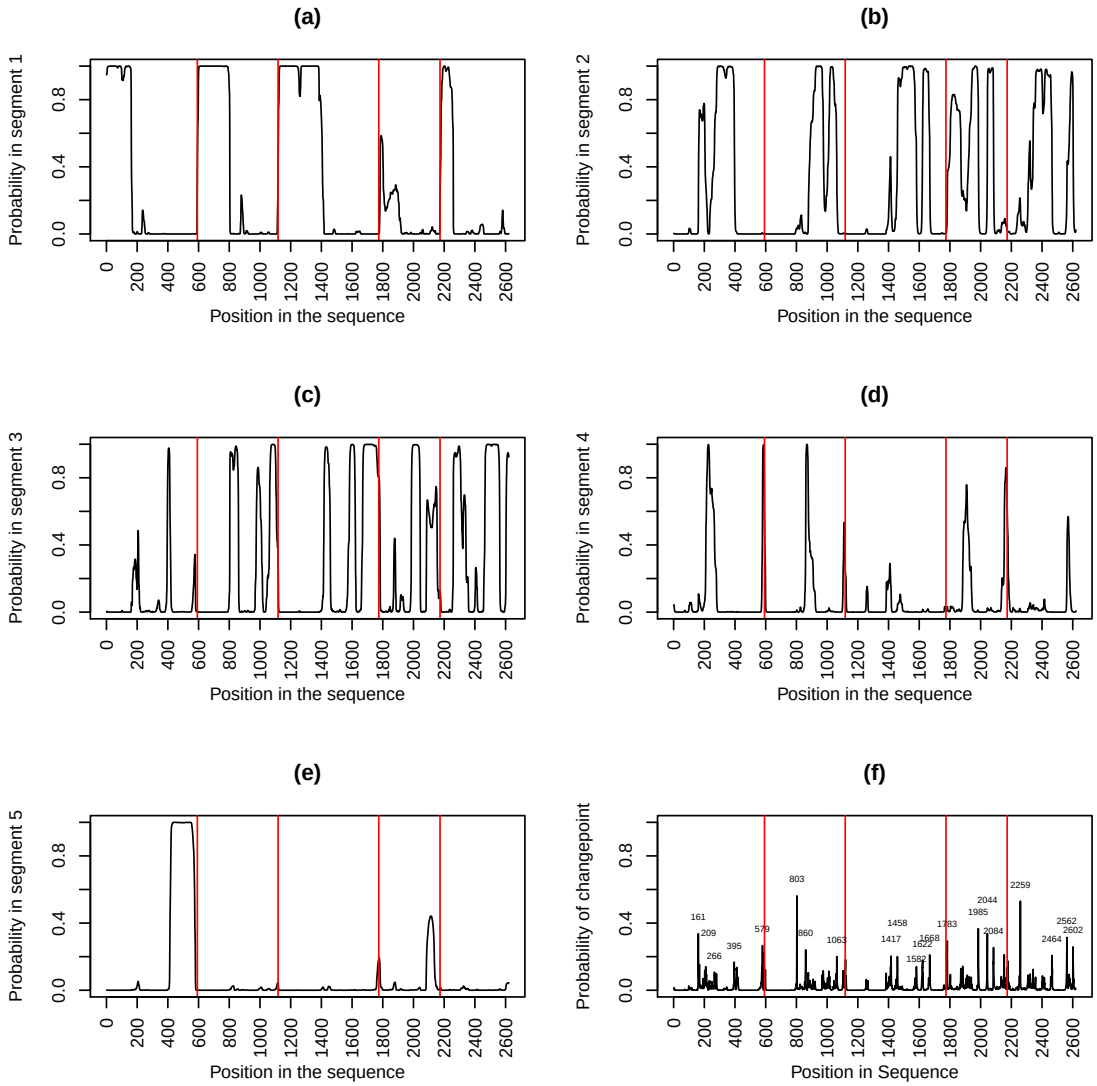


Figure 5.15: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group 1 proteins, Cabeza and FUST–1. The proteins are joined together with TAF15 first, FUS second, EWS third, Cabeza forth and FUST–1 fifth. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

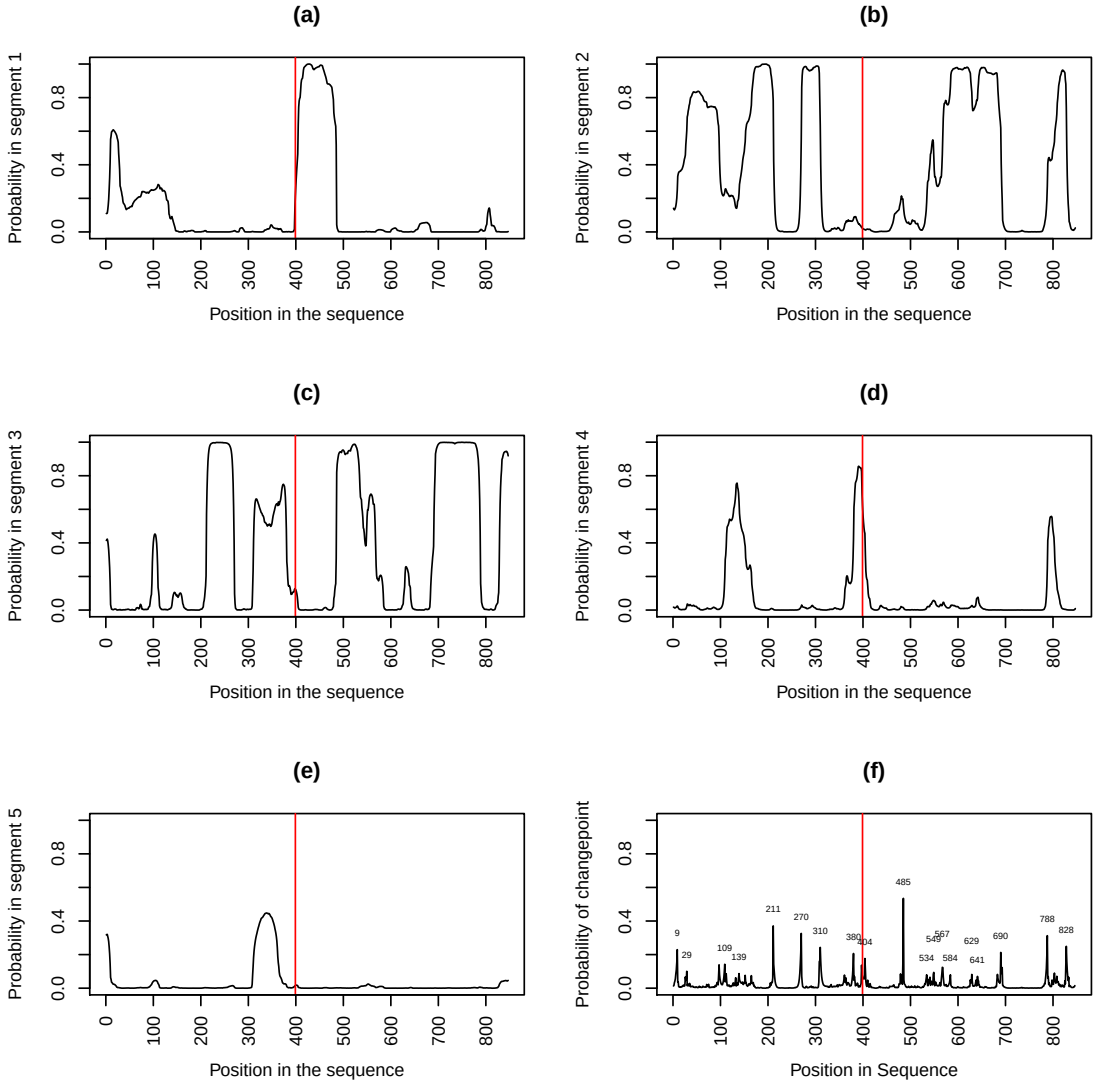


Figure 5.16: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for Cabeza and FUST–1. The proteins are joined together with Cabeza first and FUST–1 second. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

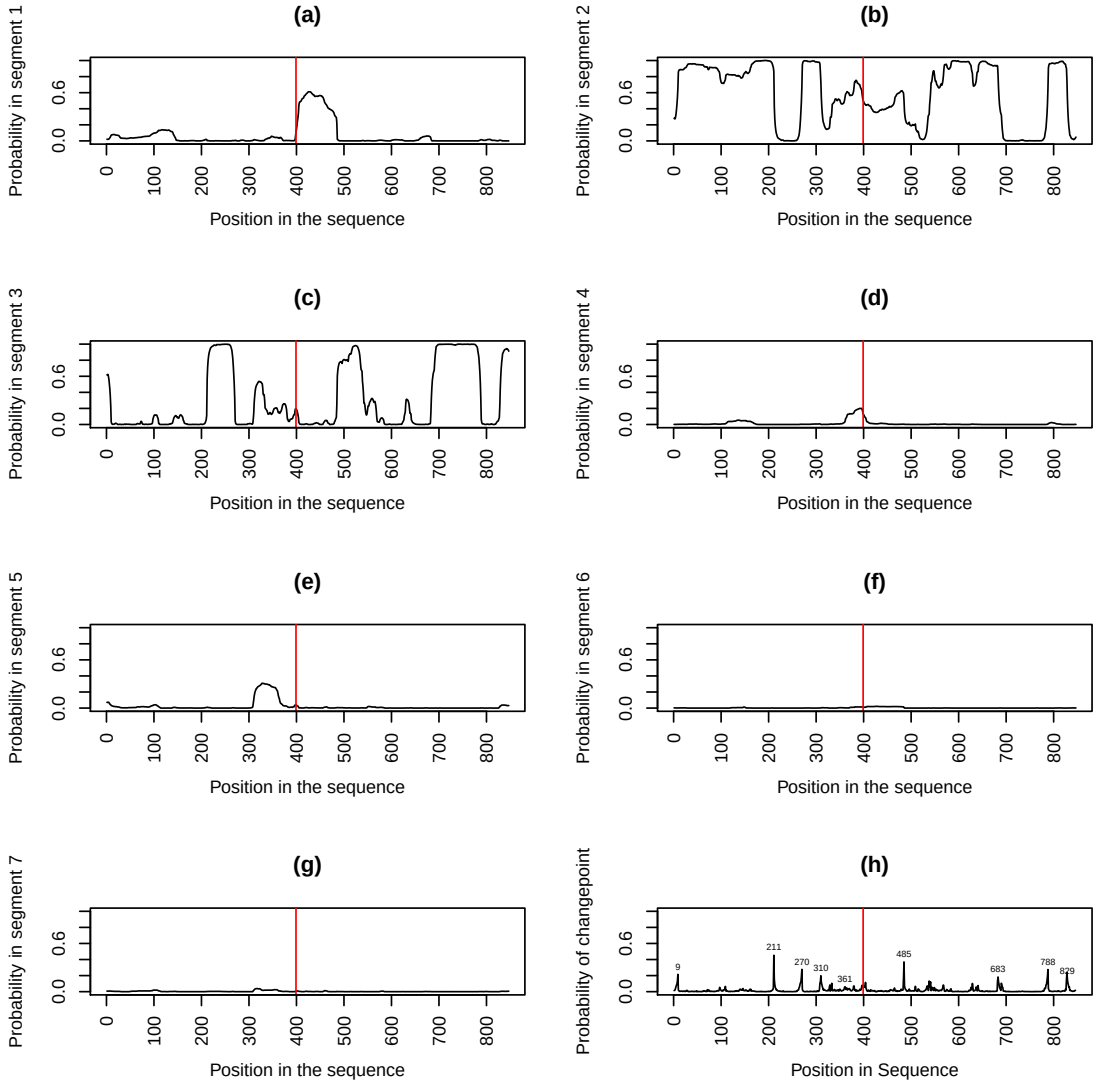


Figure 5.17: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for Cabeza and FUST–1. The proteins are joined together with Cabeza first and FUST–1 second. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

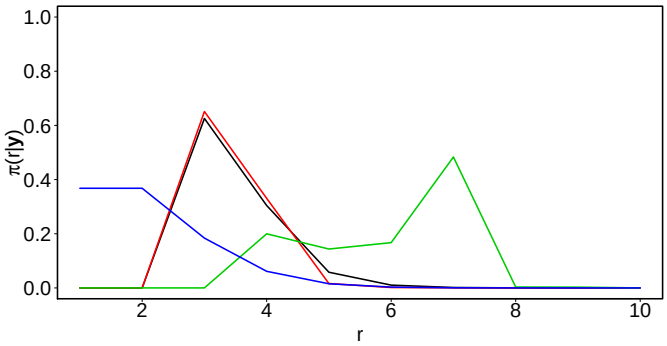


Figure 5.18: Plot of the posterior probability functions for r, $\pi(r|\boldsymbol{y})$ for the group 1 proteins, Cabeza and FUST–1 for $f = 2$ (black), $f = 3$ (red) and $f = 4$ (green) respectively. The prior distribution is given in blue.

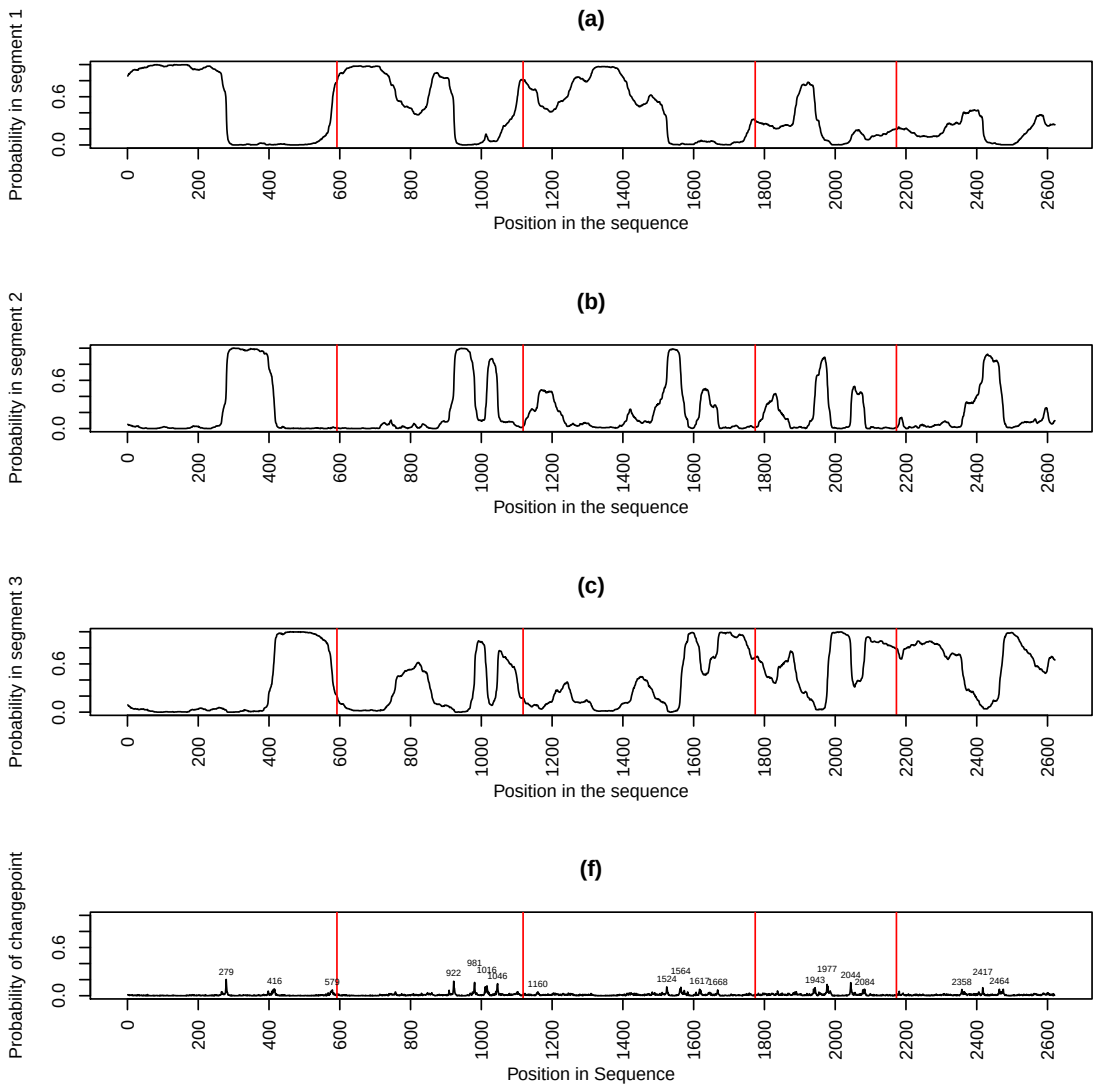


Figure 5.19: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group 1 proteins, Cabeza and FUST–1. The proteins are joined together with TAF15 first, FUS second, EWS third, Cabeza forth and FUST–1 fifth. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

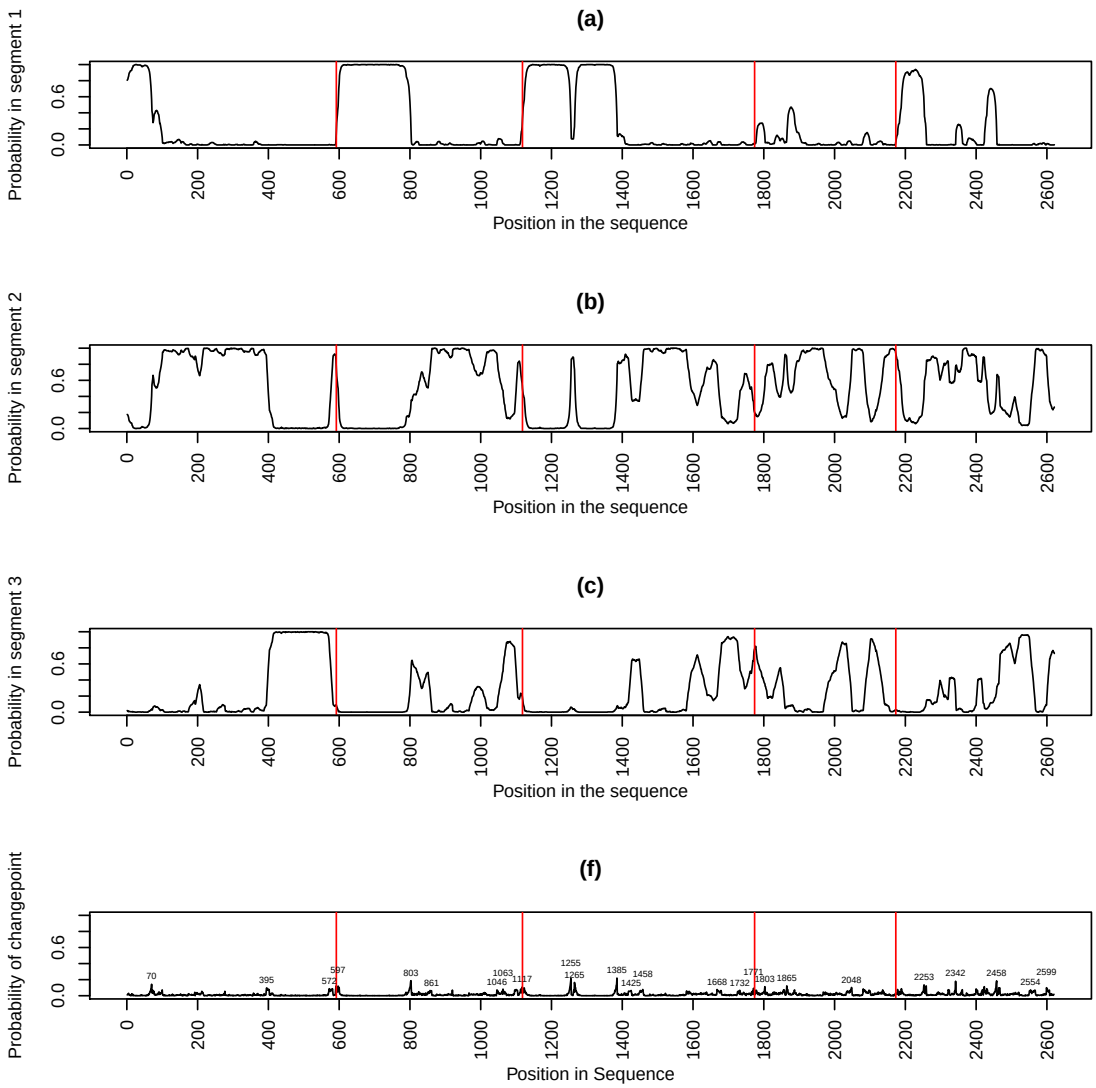


Figure 5.20: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group 1 proteins, Cabeza and FUST–1. The proteins are joined together with TAF15 first, FUS second, EWS third, Cabeza forth and FUST–1 fifth. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

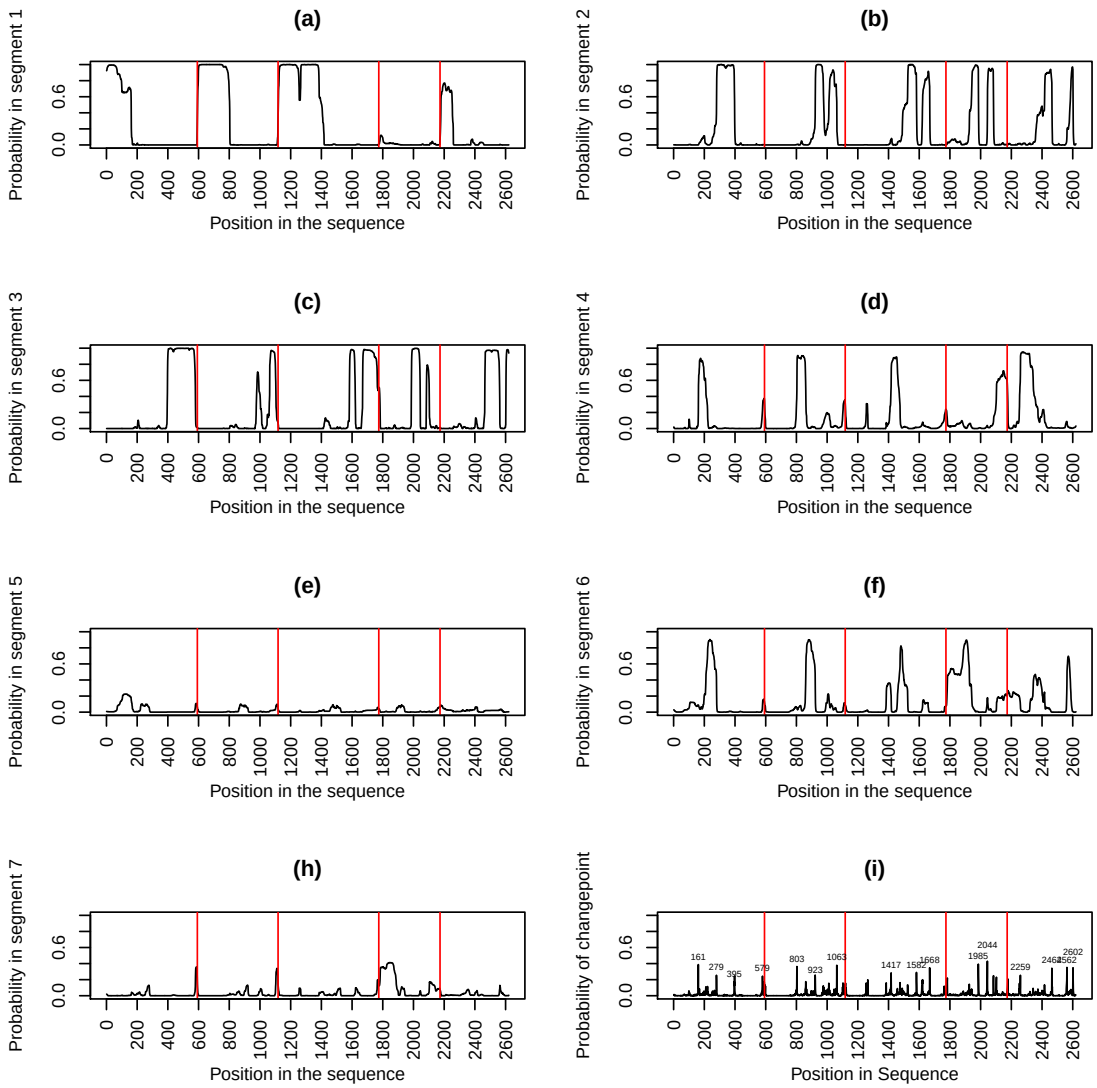


Figure 5.21: Plots of (a)–(h): the probability of being in each segment and (i) the probability of changing segments for the group 1 proteins, Cabeza and FUST–1. The proteins are joined together with TAF15 first, FUS second, EWS third, Cabeza forth and FUST–1 fifth. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

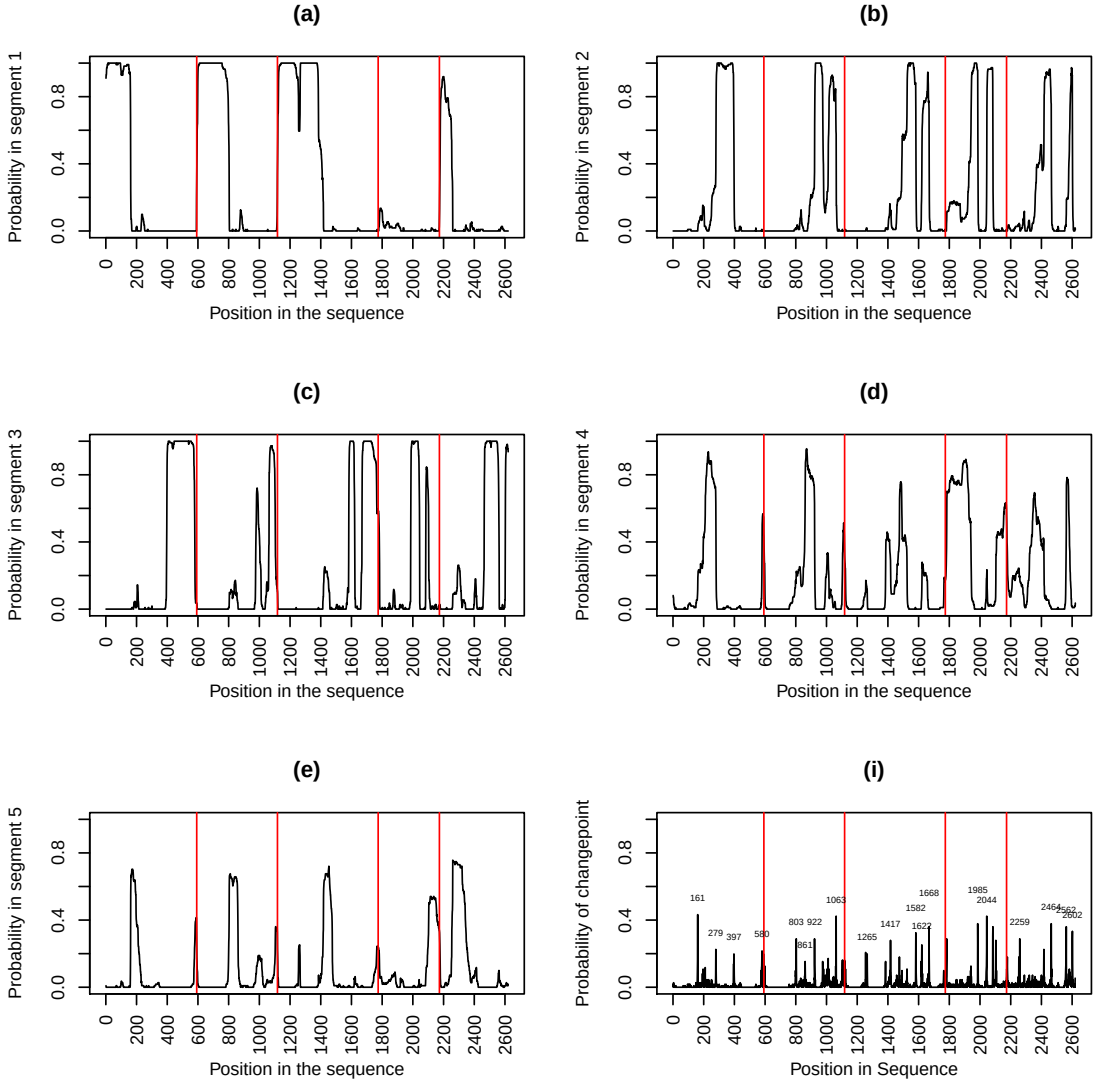


Figure 5.22: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group 1 proteins, Cabeza and FUST–1. The proteins are joined together with TAF15 first, FUS second, EWS third, Cabeza forth and FUST–1 fifth. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

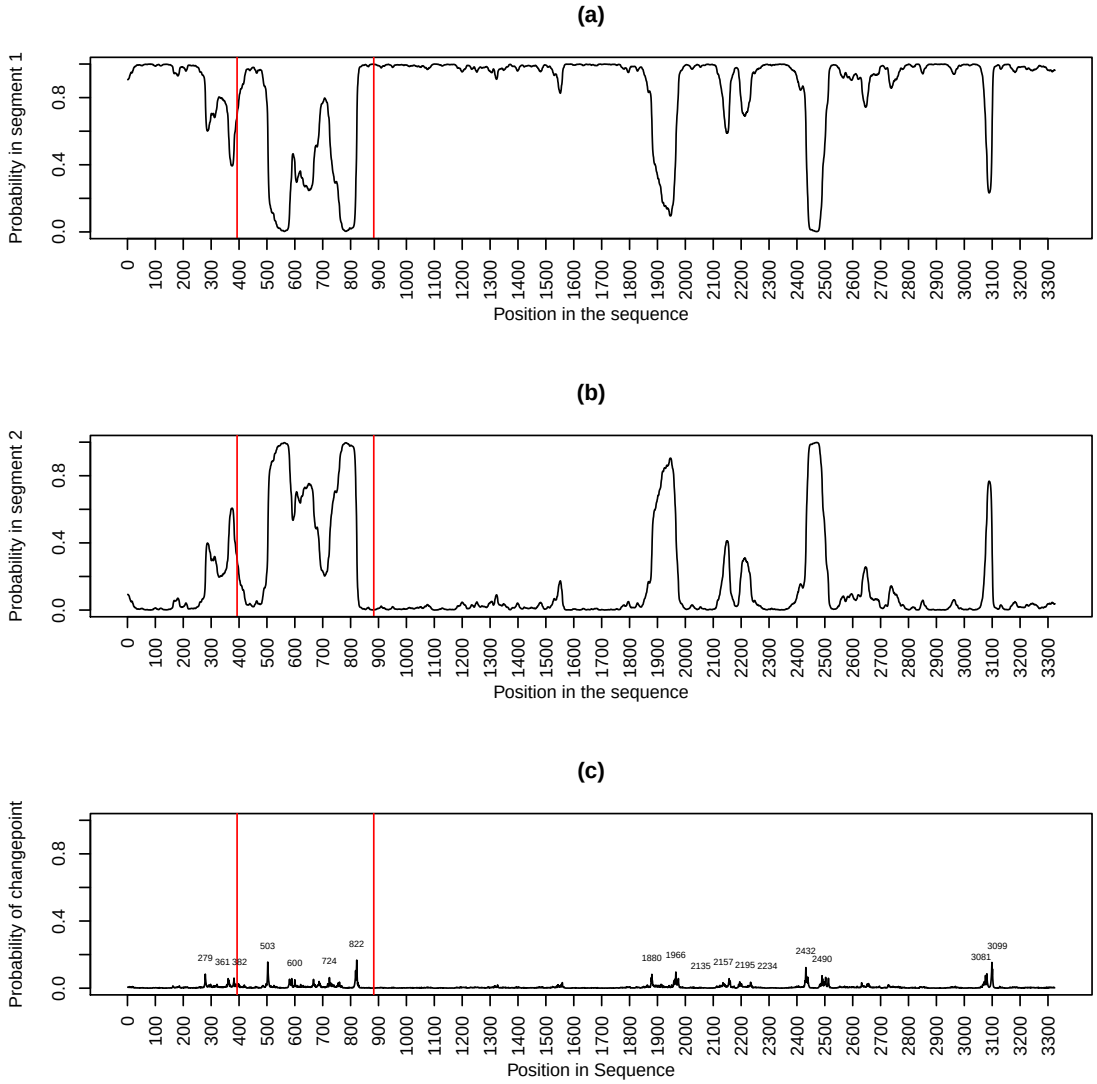


Figure 5.23: Plots of (a)–(b): the probability of being in each segment and (c) the probability of changing segments for the group 2 proteins with $f = 2$. The proteins are joined together with p53 first, MDM2 second and CBP third. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

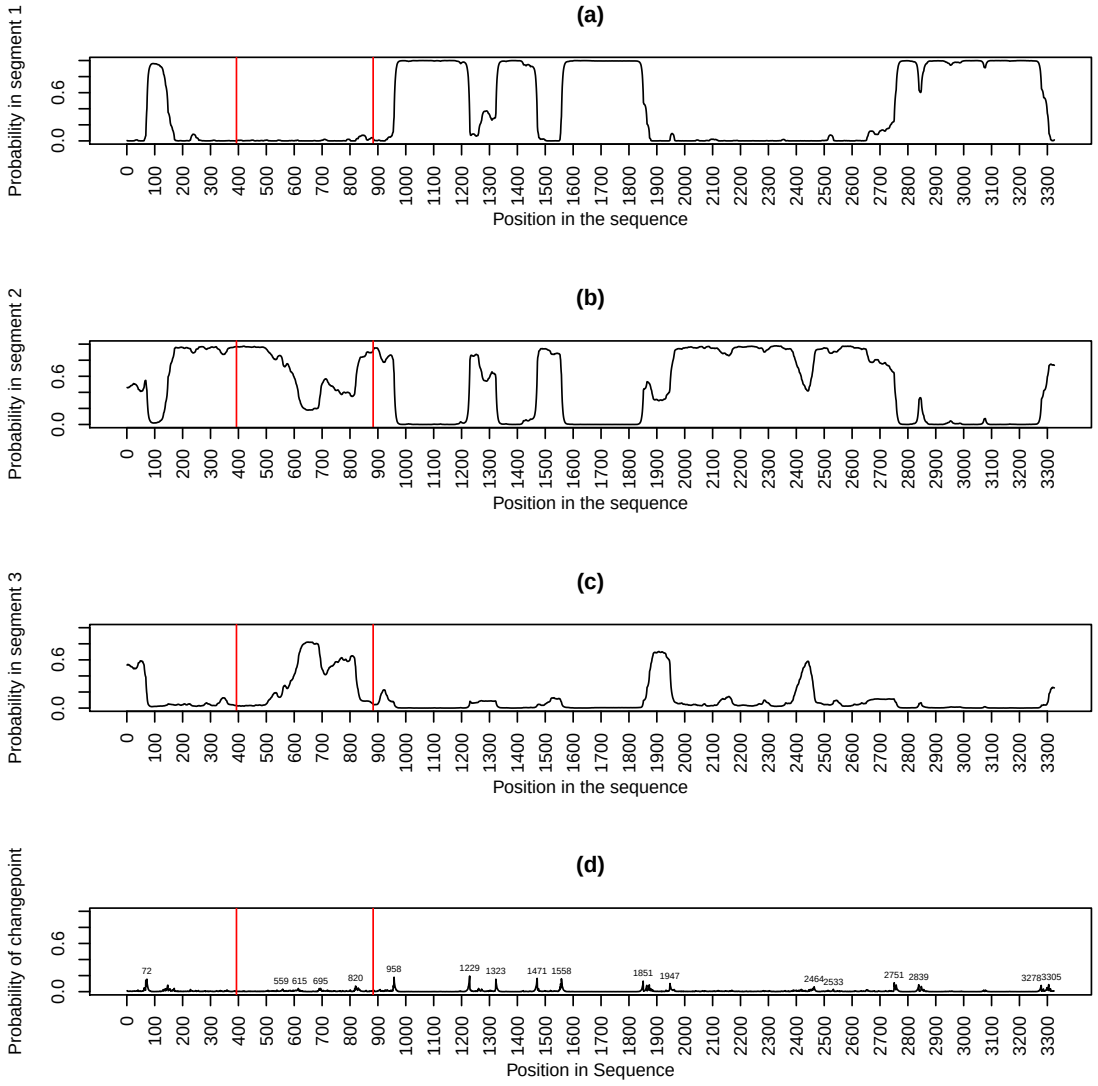


Figure 5.24: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group 2 proteins with $f = 3$. The proteins are joined together with p53 first, MDM2 second and CBP third. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

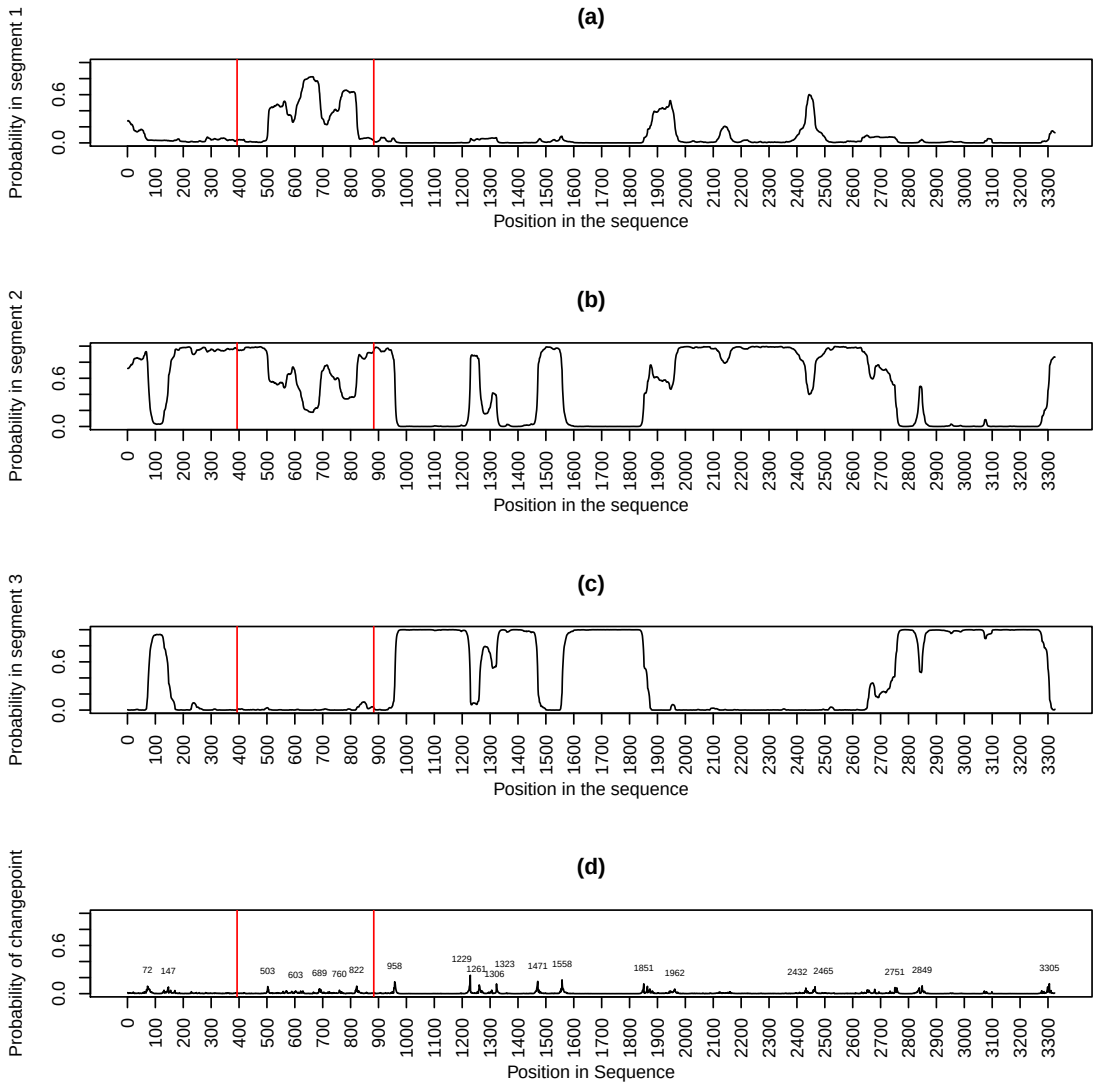


Figure 5.25: Plots of (a)–(c): the probability of being in each segment and (d) the probability of changing segments for the group 2 proteins with $f = 4$. The proteins are joined together with p53 first, MDM2 second and CBP third. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

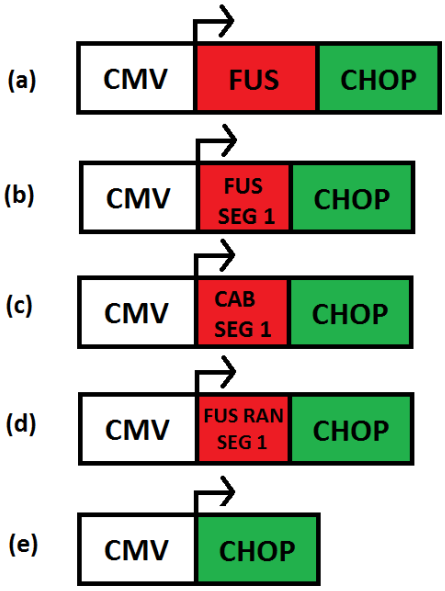


Figure 5.26: Constructs used in the biological experiments. Construct (a) is FUSCHOP which is the whole protein FUS fused to CHOP, (b) is SEGCHOP which is segment 1 of FUS fused to chop, (c) is CABCHOP which is segment 1 of Cabeza fused to CHOP, (d) is RCHOP which is randomised segment one of FUS fused to CHOP and (e) is CHOP alone which is the control. CMV stands for cytomegalovirus, a DNA virus which is the source of the promoter element used in the plasmid which is then transfected into cells.

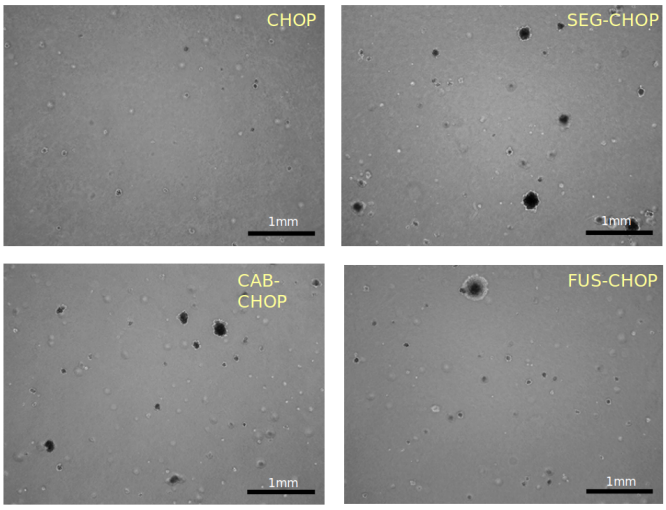


Figure 5.27: Microscope images of colonies of cells (black spots).

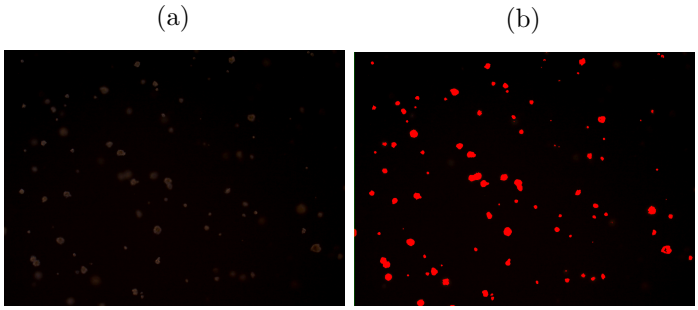


Figure 5.28: Images to show ImagePro software in use. Image (a) is the original microscope image and Image (b) shows how the software has identified the colonies.

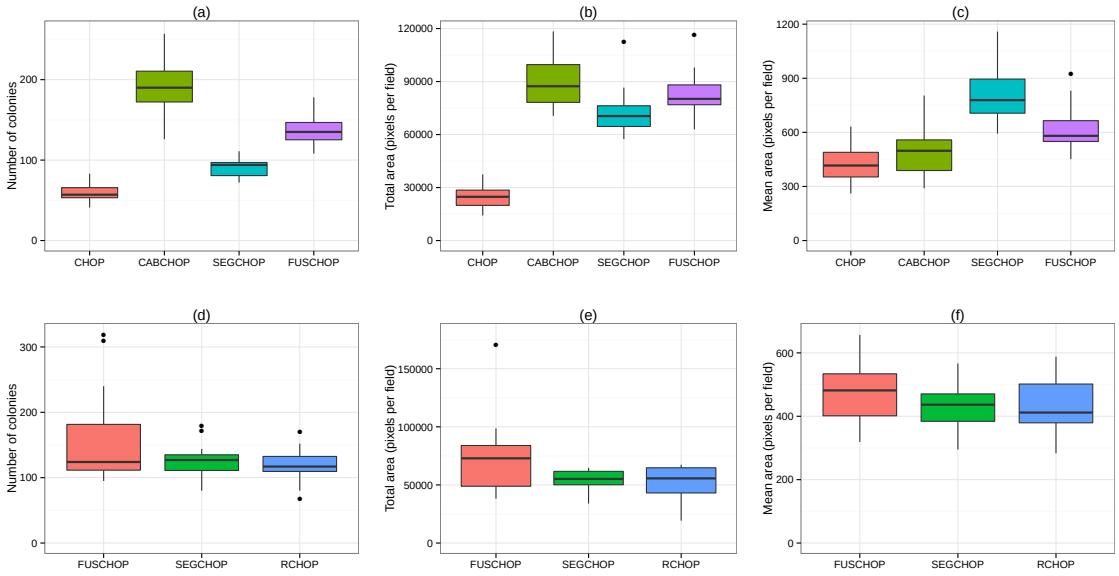


Figure 5.29: Plots to show the results of Prof. Doug Gray's experiments. We have boxplots of the number of colonies, total area and mean area for each of the constructs for the first set of experiments in plots (a), (b) and (c) and the second set of experiments in plots (d), (e) and (f).

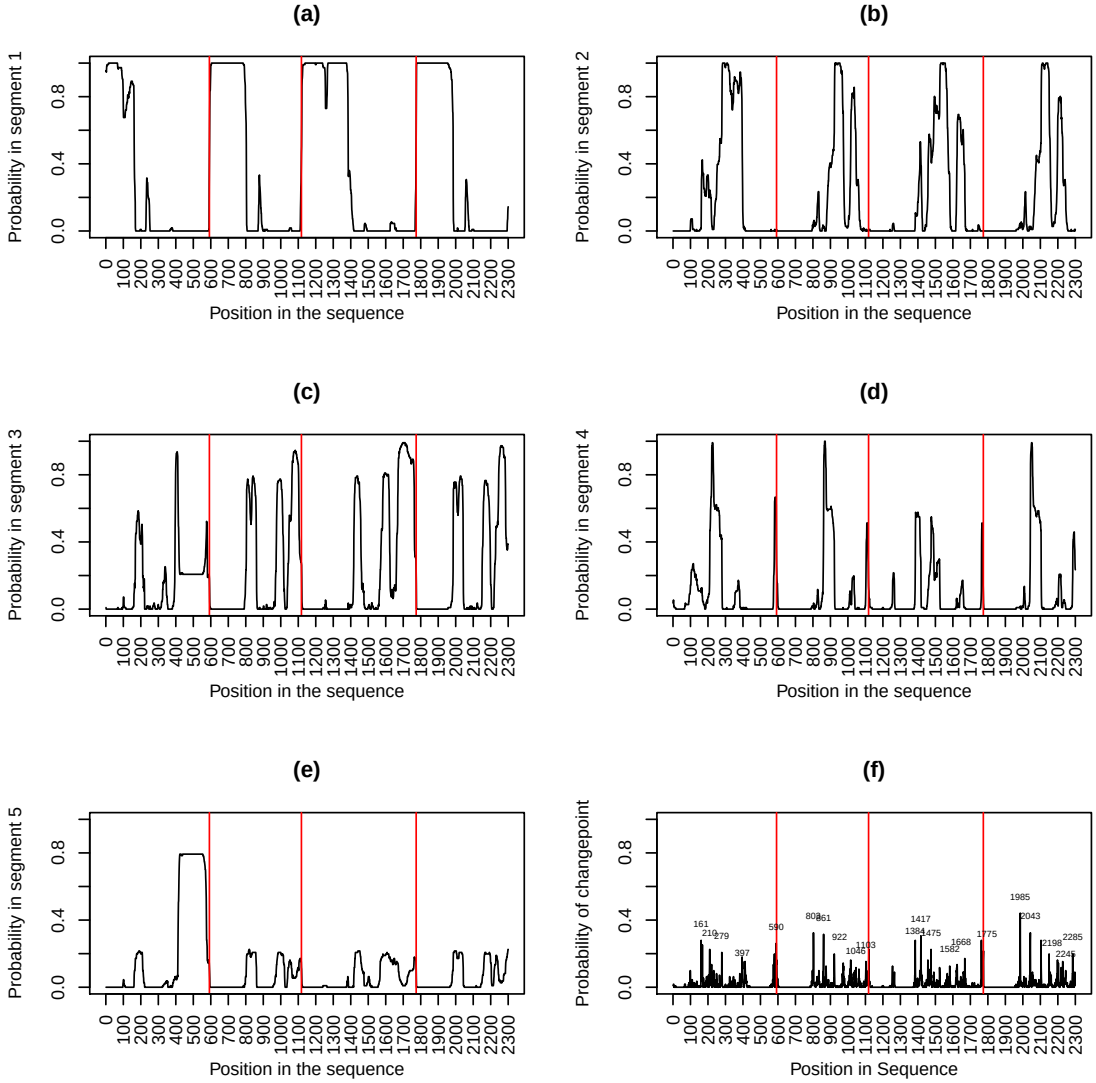


Figure 5.30: Plots of (a)–(e): the probability of being in each segment and (f) the probability of changing segments for the group 1 proteins and FUS with a randomised segment one. The proteins are joined together with TAF15 first, FUS second, EWS third and FUS with a randomised segment one last. The separation between the proteins is shown by the vertical red lines. On the changepoint plot the position of the most probable changepoints are labelled.

Chapter 6

Discussion and conclusion

6.1 Statistical conclusion

One of the strengths of the statistical techniques we have used are that it is straightforward to include prior knowledge about important quantities, for example the length of segment types. Also the MCMC method used is fairly simple to implement as it is just Gibbs sampling and, as we can simulate from the conditional distribution of the entire segmentation using a forward-backward algorithm, it has good convergence properties. The analysis allows us to detect the (rough) locations of different segment types and allows for uncertainty in their structure and location. The MCMC sampler converges very quickly, often within 100 iterations, and autocorrelation is often very low. In addition, determining an appropriate number of segment types via the power posterior analysis is accurate and fairly easy to implement. The joint posterior density for the model parameters and the segmentation process is relatively insensitive to small changes in the prior distributions for the transition structures. Similarly, the marginal posterior distribution for the number of segment types is also relatively insensitive to small changes in the Poisson prior distribution.

A potential weakness of this method of analysis is that of label switching and if we do not correct for it we would obtain similar estimates for the transition structure of each segment type. However we can correct for label switching using an 'on–line' algorithm as described in Section 4.2.5. Unfortunately this does slow down the code considerably, especially for larger values of r. For example, when $f = 4$ the Gibbs sampling algorithm takes 6.9 seconds when $r = 3$ and 1588.1 seconds when $r = 6$ to complete 100 iterations. Another problem concerns the value for r calculated using the marginal likelihood (from the power posterior method). Often the posterior mode is the maximum value of r allowed by the prior distribution. However the algorithm takes much longer to run for large values of r, and therefore to allow large values for r in the analysis is not feasible. Another weakness of this type of analysis is that, although the method gives a segmentation of a protein sequence

with corresponding transition structures for the segment types, there is the possibility that these segment types do not have an obvious biologically relevance. That said, it might lead experimentalists to undertake additional experiments to investigate possible biological reasons behind the segmentation. Additionally there is also a possible problem that the segmentation we get depends on how we categorise the protein sequences, for example, using hydrophobicity ($f = 2$), charge ($f = 3$) or both ($f = 4$). Thus these HMM analyses are fundamentally empirical investigations and provide suggestions of possible biological structure for further scientific analysis and hypothesis driven investigations. We have shown that these HMMs can give insight into the location of biologically significant areas but further experimental work is required to confirm or disprove this.

6.2 Biological conclusion

In this part of the thesis we have shown that HMMs can be used to reveal particular regions of IDPs that have biological relevance. We found that in the FET proteins the first segment can be linked to oncogenic fusion proteins which we have backed up using experimental data from Prof. Doug Gray's laboratory at the University of Ottawa. In these experiments we found that when segment one from FUS was fused with the CHOP transcription factor it had similar transforming properties to the whole of FUS fused to CHOP. In addition, it was found that segment one of cabeza has the strongest transforming activity. In a separate experiment, it was found that if segment one in FUS was randomised, the resulting construct had almost the same level of transforming ability, which was backed up when analysed computationally. This indicates that the order of amino acids is not as important as the frequency of amino acids.

Overall, we have shown that our HMM analysis reveals a pattern in the sequence of amino acids in the FET proteins that has biological meaning. Also these segments can be revealed by using only the specific physical properties of the amino acid sequence without the need to consider linear sequence motifs (small sections of protein that bring about protein–protein interaction) or three dimensional structures (Gray, 2014). We have demonstrated that using statistical techniques can help guide biological experiments, and with the biologists' knowledge and expertise, further insight into sequence structure can be found. We have shown this in particular with the structure and function of FUS.

6.3 Future work

There are many ways in which this work could be extended. For example, a natural extension to our model would be to generalise the dependence between the amino acids within the HMM. Currently we assume this has first order dependence, that is the distribution of the

amino acid at location t depends only on that at the previous location (and the segment type). This assumption could be generalised to allow this distribution to depend on the amino acids at the previous q locations, that is, we allow the observation model to be a qth order Markov chain. Within this extended model we could use the methods in Boys and Henderson (2004) to facilitate inference on both the number of segments (r) and the order of dependence (q) in the analysed sequence.

The current HMM assumes conditional independence of the amino acids in different segments. In general this is not appropriate for protein sequences as the different segments would interact in 3D space (Schmidler et al., 2000). Therefore it might be interesting to extend the model to allow interactions between segments via the use of higher orders in the HMM.

The data we have studied come from experiments that look into the oncogenic properties of FUS. However it might be interesting to look at links with neurodegenerative diseases. FUS has been linked to normal and abnormal function in neurodegenerative diseases; for example, Amyotrophic lateral sclerosis (ALS), where it is likely that the gene for FUS contains a mutation (Da Cruz and Cleveland, 2011). Here the C terminus contains most of these mutations; however some have been found in segment one of FUS (Da Cruz and Cleveland, 2011). Some mutations delete one or more of the residues glycine or serine or both. In segment one of FUS there are many of these residues so it is difficult to understand why this should make a difference. Experiments have shown that if we randomise the sequence of segment one then there is still transforming activity and so it would be interesting to see if the same is true for neurodegeneration. All neuronal functions are linked to the metabolism of RNA, that is, RNA splicing, RNA transport and RNA stability. Therefore we could investigate the effect of introducing the constructs given in Figure 5.26 into neurons using lentiviral vectors and analysing the RNA using sequencing methods (RNA–SEQ) (Pareek et al., 2011). It could be that transforming activity acts similarly or differently to RNA regulatory functions.

This far we have looked at experimental validation on segment one of the FET proteins. We could look at the other segments but we believe that these segments are less interesting as the RNA binding domain of FUS in segment two will lose function if it is altered and the RGG repeats in segment three of FUS have already been well studied due to their abundance in RNA binding proteins (Da Cruz and Cleveland, 2011).

Finally, we may gain more insight into protein structure by considering different properties of the amino acids in the analysis. This would result in a new recoding of the amino acid sequence and might lead to the discovery of structures with a more plausible biological relevance.

Part II

Experimental design of stochastic kinetic models

Chapter 7

Stochastic kinetic models

In this chapter we introduce stochastic kinetic models using chemical reaction notation. The aim of the second part of this thesis is to construct optimal experimental designs for stochastic kinetic models and the key problem here is that all but the most simple models suffer from having intractable likelihoods. The solution we provide to this problem is based on that of Müller (1999), who describes an algorithm where forward simulation can be used to avoid likelihood evaluations in the Metropolis–Hastings ratio.

We begin by examining exact methods of simulation including the direct method and then consider approximate methods of simulation, such as the τ–leap algorithm, the Chemical Langevin equation (CLE) and the linear noise approximation (LNA). We introduce two stochastic kinetic models in this chapter, namely the (pure) death model and the Lotka–Volterra (LV) model, and show example realisations from both models.

7.1 Introduction to stochastic kinetic models

Biological systems are often represented using a stochastic kinetic model (SKM) as these models include random variability that would occur in nature. It is important to represent such variation because it takes into account the different ways that a process can evolve. For example, a population could become extinct or explode with the same parameter set in a stochastic kinetic model, but not in a deterministic model. Reactions are stochastic and they are driven by Brownian motion. A reaction results in species changing by integer amounts which means that the species numbers do not change smoothly, species numbers jump in discrete values. These models come under the heading of stochastic kinetic models. For an overview of SKMs, see Wilkinson (2011) and Golightly and Gillespie (2013).

7.2 Chemical reaction notation

Consider a system at a cellular level. When molecules collide and a change occurs, this is called a chemical reaction. We are interested in the levels of each molecule type or chemical species in a cell and describe this system by a set of chemical reactions. Denote the number of molecules or species as $\mathcal{Y}_1, \ldots, \mathcal{Y}_u$ and the reactions as $R_1, \ldots, R_v$. We represent v reactions involving u species in chemical reaction notation as

$$\begin{array}{lllll} R_1: & p_{1,1}\mathcal{Y}_1 + p_{1,2}\mathcal{Y}_2 + \ldots + p_{1,u}\mathcal{Y}_u \to q_{1,1}\mathcal{Y}_1 + q_{1,2}\mathcal{Y}_2 + \ldots + q_{1,u}\mathcal{Y}_u \\ R_2: & p_{2,1}\mathcal{Y}_1 + p_{2,2}\mathcal{Y}_2 + \ldots + p_{2,u}\mathcal{Y}_u \to q_{2,1}\mathcal{Y}_1 + q_{2,2}\mathcal{Y}_2 + \ldots + q_{2,u}\mathcal{Y}_u \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_v: & p_{v,1}\mathcal{Y}_1 + p_{v,2}\mathcal{Y}_2 + \ldots + p_{v,u}\mathcal{Y}_u \to q_{v,1}\mathcal{Y}_1 + q_{v,2}\mathcal{Y}_2 + \ldots + q_{v,u}\mathcal{Y}_u \end{array} \tag{7.1}$$

where $P = (p_{i,j})$ is a matrix with dimension $v \times u$ which contains the coefficients of the reacting species and $Q = (q_{i,j})$ is of the same dimension as P and it contains the coefficients of the products (Wilkinson, 2011). The stoichiometry matrix is $S = (Q - P)'$ which represents the net change in molecules or species each time a reaction occurs. Let $Y_{t,j}$ be the number of molecules of species $\mathcal{Y}_j$ at time t then we can represent the numbers of molecules for all species at time t as $Y_t = (Y_{1,t}, \ldots, Y_{u,t})'$.

7.2.1 Markov jump process

An example of a reaction in chemical reaction notation is

$$\mathcal{Y}_1 + \mathcal{Y}_2 \to \mathcal{Y}_3. \tag{7.2}$$

In this case, a reaction occurs due to a collision between one molecule of $\mathcal{Y}_1$ and one molecule of $\mathcal{Y}_2$ to give a molecule of $\mathcal{Y}_3$. Define $y = (y_1, \ldots, y_u)'$ to be the current state of the system. This means in this example that y_1 and y_2 reduce by one and y_3 increases by one. This reaction occurs randomly due to Brownian motion. If we think about molecules in a container that has a fixed volume, Gillespie (1992) showed that for a small interval Δt, the reaction rate is constant. If we consider again reaction (7.2), the probability of $\mathcal{Y}_1$ reacting with $\mathcal{Y}_2$ in a small time interval of length Δt is $\theta \Delta t$, where θ is is the constant rate at which the reaction occurs. If the number of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are y_1 and y_2 respectively then these molecules can pair up in $y_1 y_2$ ways. Hence the probability of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ reacting and producing $\mathcal{Y}_3$ in a small time interval of length Δt is $\theta y_1 y_2 \Delta t$. Note that this probability only depends on the current state of the system. This system is an example of a Markov jump process as it has this Markov property and it is a continuous time, discrete valued

Order	Reaction	Hazard	Description
0	$\emptyset \to \mathcal{Y}_1$	θ_1	Influx
1	$\mathcal{Y}_1 \to \emptyset$	$\theta_2 y_1$	Degradation
2	$\mathcal{Y}_1 + \mathcal{Y}_2 \to \mathcal{Y}_3$	$\theta_3 y_1 y_2$	Catalysation
2	$2\mathcal{Y}_1 \to \mathcal{Y}_2$	$\theta_4 y_1(y_1 - 1)/2$	Dimerisation
3	$3\mathcal{Y}_1 \to \mathcal{Y}_3$	$\theta_5 y_1(y_1 - 1)(y_1 - 2)/6$	Trimerisation

Table 7.1: Examples of possible reactions and hazards.

process. The overall hazard for this reaction occurring is defined as

$$h(y, \theta) = \theta y_1 y_2.$$

An example of a more complex system is given by the reactions and their hazards in Table 7.1. Such systems are typically governed by the law of mass action kinetics which states that the rate of a chemical reaction is proportional to the product of the concentration of reactants (Waage and Gulberg, 1986). We can consider the general system of reactions given in Equations (7.1) and write the overall hazard for each reaction as

$$h_i(y, \theta_i) = \theta_i \prod_{j=1}^{u} \binom{y_j}{p_{i,j}}, \quad \text{for } i = 1, \ldots, v.$$

Here the hazards are a product of the binomial coefficients multiplied by the rate constant due to the fact that the hazard depends on the number of ways that the reactants can collide.

7.2.2 Chemical master equation

Define $p_y(t)$ to be the probability of $y = (y_1, y_2, \ldots, y_u)'$ molecules of each species at a particular time, t, representing the system state at time, t. Another description is that $p_y(t)$ is the transition kernel of the Markov jump process. If we consider Δt to represent a small time interval, by considering all possible ways in which state, y, can be obtained and adding up the probabilities of these different possibilities, we obtain

$$p_y(t + \Delta t) = \sum_{i=1}^{v} h_i(y - S^i, \theta_i) p_{y - S^i}(t) \Delta t + \left\{ 1 - \sum_{i=1}^{v} h_i(y, \theta_i) \Delta t \right\} p_y(t), \tag{7.3}$$

where S^i is a vector of the ith column of S. The first term in Equation (7.3) represents the probability of the system reaching state y via a reaction R_i, while the second term represents the probability of no reactions occurring in the time interval. We can form the chemical master equation (CME) by rearranging Equation (7.3) and letting $\Delta t \to 0$, to

give

$$\frac{dp_y(t)}{dt} = \sum_{i=1}^{v} \left\{ h_i(y - S^i, \theta_i) p_{y-S^i}(t) - h_i(y, \theta_i) p_y(t) \right\}. \tag{7.4}$$

7.2.3 Direct method

The direct method (Gillespie, 1977) is widely used in stochastic modelling to obtain exact realisations from an SKM. Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_v)'$ where $\boldsymbol{\theta}$ is a vector of the reaction rates for each reaction. The algorithm works by simulating the time of the next reaction. This time follows an exponential distribution with rate parameter $h_0(Y_t, \boldsymbol{\theta})$, where $h_0(Y_t, \boldsymbol{\theta})$ is the sum of the hazards of each individual reaction

$$h_0(Y_t, \boldsymbol{\theta}) = \sum_{i=1}^{v} h_i(Y_t, \theta_i).$$

We can calculate the probability of each reaction taking place at this next reaction time as

$$\frac{h_i(Y_t, \theta_i)}{h_0(Y_t, \boldsymbol{\theta})} \quad i = 1, \ldots, v,$$

and then use these probabilities to sample which reaction has occurred. The direct method is described in Algorithm 14. Although this method is an exact solution to the CME, it can be computationally very time consuming to simulate from models, especially as they become more complex, for example, as the number of reactions or the number of species increases. Increasing the reaction rates θ_i can also increase the time it takes to obtain a realisation in a fixed time interval as this decreases the time until the next reaction and so increases the total number of reactions in the interval.

7.3 Example systems

7.3.1 The death model

This process has a single reaction

$$\mathcal{Y} \xrightarrow{\theta} \emptyset \tag{7.5}$$

and the population is monotonically decreasing. As this model has only one species, we write the species as $\mathcal{Y}$ instead of $\mathcal{Y}_1$. The hazard function corresponding to reaction (7.5) is

$$h(y, \theta) = \theta y. \tag{7.6}$$

It is often illuminating to compare the solution of a stochastic kinetic model with that of its deterministic equivalent. For example, the solution to the deterministic system may not be too far from the mean of the stochastic system. Such points raise questions that are

Algorithm 14 Direct method (Gillespie, 1977)

1. Initialise $t = 0$, rate constants $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_v)'$ and the molecule numbers $y = (y_1, \ldots, y_u)$ at time t.
2. For every possible reaction calculate the hazard $h_i(y, \theta_i)$ for $i = 1, \ldots, v$.
3. Calculate the overall hazard rate $h_0(y, \boldsymbol{\theta}) = \sum_{i=1}^{v} h_i(y, \theta_i)$.
4. Calculate the time of the next reaction by simulating $t^{\diamond} \sim \exp(h_0(y, \boldsymbol{\theta}))$ and set $t = t + t^{\diamond}$.
5. Determine which reaction has occurred by simulating from a discrete distribution with probabilities
$$\frac{h_i(y, \theta_i)}{h_0(y, \boldsymbol{\theta})}, \quad i = 1, \ldots, v,$$
and update y accordingly.
6. Record t and y. If $t < t_{max}$ go to step 2.

interesting in their own right but not considered further here. Let $Y(t)$ be the number of individuals of species $\mathcal{Y}$ at time t. Then in a small time interval of length Δt, we have

$$Y(t + \Delta t) = Y(t) - \theta Y(t)\Delta t.$$

Rearranging and letting $\Delta t \to 0$, yields

$$\frac{dY(t)}{dt} = -\theta Y(t).$$

This ordinary differential equation (ODE) can be solved to obtain

$$Y(t) = y_0 \exp(-\theta t)$$

where $Y(0) = y_0$.

The stochastic version of this model assumes that in a small time step Δt, the probability of a death occurring in the population of species $\mathcal{Y}$ is

$$P(Y_{t+\Delta t} = y - 1 | Y_t = y) = \theta y \Delta t + o(\Delta t), \qquad y = y_0, y_0 - 1, \ldots, 0.$$

where $Y_0 = y_0$. Suppose that $p_y(t)$ is the probability that there are y individuals of species $\mathcal{Y}$ at time t. Then in time interval $(t, t + \Delta t)$ we have

$$p_y(t + \Delta t) = p_{y+1}(t)\theta(y + 1)\Delta t + p_y(t)(1 - \theta y \Delta t). \tag{7.7}$$

Rearranging and letting $\Delta t \to 0$, we obtain the forward Kolmogorov equation (also known as the CME)

$$\frac{dp_y(t)}{dt} = \theta(y+1)p_{y+1}(t) - \theta y p_y(t).$$

This system can be solved analytically, to obtain the probability of y individuals at time t as

$$p_y(t) = \binom{y_0}{y} \exp^{-\theta y t}(1 - \exp^{-\theta t})^{y_0 - y},$$

where $y = y_0, y_0 - 1, \ldots, 0$ and $Y_0 = y_0$. We can see that the solution of the stochastic model is binomially distributed and therefore the mean of this stochastic process is $m(t) = y_0 \exp^{-\theta t}$ which is equivalent to the solution of the deterministic process.

7.3.2 The Lotka–Volterra (LV) model

The Lotka–Volterra model (LV) model describes the interaction between prey, $\mathcal{Y}_1$ and predators, $\mathcal{Y}_2$ (Lotka, 1925; Volterra, 1926). It has three reactions and two species. The three reactions in the LV model are

$$\mathcal{Y}_1 \xrightarrow{\theta_1} 2\mathcal{Y}_1, \qquad \mathcal{Y}_1 + \mathcal{Y}_2 \xrightarrow{\theta_2} 2\mathcal{Y}_2 \quad \text{and} \qquad \mathcal{Y}_2 \xrightarrow{\theta_3} \emptyset.$$

The parameters can be interpreted as

- θ_1 is the rate of prey reproduction,
- θ_2 is the rate of prey death and predator reproduction,
- θ_3 is the rate of predator death.

Let $y = (y_1, y_2)$ be the current states of the system at time t. By assuming the laws of mass action, the hazard functions for the reactions are

$$h_1(y, \theta_1) = \theta_1 y_1, \qquad h_2(y, \theta_2) = \theta_2 y_1 y_2 \qquad \text{and} \qquad h_3(y, \theta_3) = \theta_3 y_2.$$

The deterministic ODEs for the LV process are

$$\frac{dY_1(t)}{dt} = \theta_1 y_1 - \theta_2 y_1 y_2 \qquad \text{and} \qquad \frac{dY_2(t)}{dt} = \theta_2 y_1 y_2 - \theta_3 y_2.$$

We can also describe the LV model stochastically. Let $Y_{1,t} = y_1$ and $Y_{2,t} = y_2$ be the current states of the system. The evolution of the system in a small time interval $(t, t + \Delta t)$

can be described as

$$
\begin{aligned}
&P(Y_{1,t+\Delta t} = y_1 + 1, Y_{2,t+\Delta t} = y_2 | Y_{1,t} = y_1, Y_{2,t} = y_2) = \theta_1 y_1 \Delta t + o(\Delta t),\\
&P(Y_{1,t+\Delta t} = y_1 - 1, Y_{2,t+\Delta t} = y_2 + 1 | Y_{1,t} = y_1, Y_{2,t} = y_2) = \theta_2 y_1 y_2 \Delta t + o(\Delta t),\\
&P(Y_{1,t+\Delta t} = y_1, Y_{2,t+\Delta t} = y_2 + 1 | Y_{1,t} = y_1, Y_{2,t} = y_2) = \theta_3 y_2 \Delta t + o(\Delta t).
\end{aligned}
$$

Let $p_{y_1,y_2}(t)$ be the probability that $Y_{1,t} = y_1$ and $Y_{2,t} = y_2$. Then

$$
\begin{aligned}
p_{y_1,y_2}(t + \Delta t) = p_{y_1,y_2}(t)&(1 - \theta_1 y_1 \Delta t - \theta_2 y_1 y_2 \Delta t - \theta_3 y_2 \Delta t)\\
&+ p_{y_1-1,y_2}(t)\theta_1(y_1 - 1)\Delta t + p_{y_1+1,y_2-1}(t)\theta_2(y_1 + 1)(y_2 - 1)\Delta t\\
&+ p_{y_1,y_2+1}(t)\theta_3(y_2 + 1)\Delta t. \qquad (7.8)
\end{aligned}
$$

Rearranging Equation (7.8) and letting $\Delta t \to 0$ we obtain the forward Kolmogorov equation

$$
\begin{aligned}
\frac{dp_{y_1,y_2}(t)}{dt} = &\, p_{y_1-1,y_2}(t)\theta_1(y_1 - 1) + p_{y_1+1,y_2-1}(t)\theta_2(y_1 + 1)(y_2 - 1) + p_{y_1,y_2+1}(t)\theta_3(y_2 + 1)\\
&- p_{y_1,y_2}(t)(\theta_1 y_1 + \theta_2 y_1 y_2 + \theta_3 y_2).
\end{aligned}
$$

7.3.3 Example simulations from these stochastic kinetic models

Figure 7.1 shows example stochastic simulations from the death model and the Lotka–Volterra model. Figure 7.1(a) has five realisations from the stochastic version of death model in red and the deterministic solution in black. This process has been initialised with 50 individuals in the population and we observe that the population decreases until extinction.

Figure 7.1(b) has five realisations from the Lotka–Volterra model. The green lines are the prey and the blue lines are the predators. The black line is the deterministic solution. The graph clearly shows the cyclic nature of the process. The logic behind this is that, as the number of prey increase, the predators shortly follow due to an increase in the amount of food available but as the number of predators increase this causes the prey to begin to decrease. The lack of food then causes more predators to die, decreasing the number of predators and this continues. We note that there are other scenarios in which the specie levels do not cycle. For example, this happens when all the prey die out, in which case the predators also eventually die out. Also, if the predator all die then the prey levels simply continue to grow.

7.4 Other methods of simulation from SKMs

The simple death process is analytically tractable as an expression can be found for the transition probability. However, in more complex models (such as the LV model) this

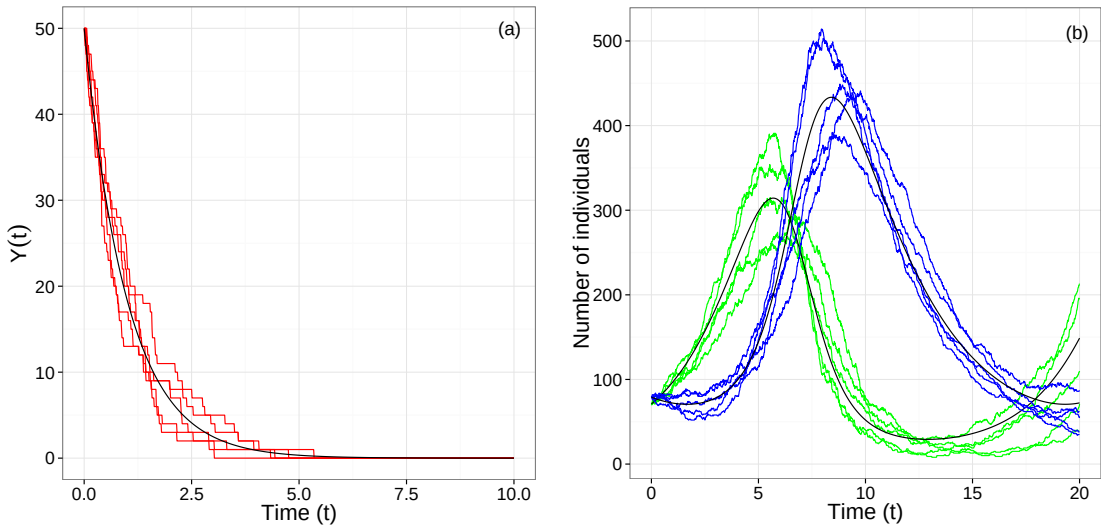


Figure 7.1: Realisations from the two models. Plot (a) is the death model ($\theta = 1$, $Y(0) = 50$). The red lines are stochastic realisations from the models and the black line is the deterministic solution. The LV model ($\theta_1 = 0.5$, $\theta_2 = 0.0025$, $\theta_3 = 0.3$, $Y_1(0) = 71$, $Y_2(0) = 79$) is in plot (b). The green lines are the prey and the blue lines are the predators. The black line is the deterministic solution.

is generally not the case. Fortunately, even when a model has analytically intractable transition probabilities, it can be straightforward to simulate from the model and obtain realisations. This means that the behaviour of the model can be observed. We have already discussed an exact simulation method called the direct method. Simulation methods can be split into two categories: exact and approximate algorithms.

7.4.1 Exact simulation

The advantage of an exact method is that we can simulate exact realisations from the model. However these methods can be very slow to run computationally. In Subsection 7.2.3 we discussed the direct method (Gillespie, 1977). Other popular exact algorithms are the next reaction method and the first reaction method (Gillespie, 1977; Gibson and Bruck, 2000).

The first reaction method is an equivalent version of the direct method but it is computationally more expensive (Gillespie, 1977). The first reaction method works by calculating a time for every reaction to happen if no other reaction had happened first. Then the reaction that is chosen is that which has the smallest next reaction time. This algorithm uses a random number each time it simulates a possible time for a reaction to occur and is slow to update the hazards and find the smallest next reaction time (Gibson and Bruck, 2000).

The next reaction method is an adapted version of the first reaction method which is more computationally efficient. The next reaction method is improved mainly by reusing

the possible times where possible and only recalculating the hazards and times if they are going to change by the use of a dependency graph (Gibson and Bruck, 2000). The next reaction method is also called the Gibson–Bruck algorithm.

7.4.2 Approximate simulation algorithms

As the size of the model increases in terms of the number of reactions and species, exact algorithms become increasingly impractical to use (due to their long computation time) and advantages may be gained by considering approximate simulation methods.

τ–leap algorithm

A commonly used approximate method is the τ–leap algorithm, introduced by Gillespie (2001). This method relies on the assumption that the time interval Δt is small enough to assume that the hazard rate is constant within the interval. It works by simulating the number of each type of reaction by assuming that the number of each type of reaction in each time interval has an independent Poisson distribution. Therefore each reaction, R_i, has a $Po(h_i(y, \theta_i)\Delta t)$ distribution for the number of occurrences of each reaction within each time interval Δt.

Choosing the time step, Δt, is a balance between accuracy and speed, as smaller time steps produce more accurate realisations but slower simulations. The idea is that Δt is chosen so that it is as large as possible without sacrificing accuracy. It is possible to determine if a suitable level of accuracy is met by using a particular timestep and a constraint. An example of a constraint is given by Gillespie (2001) in which the difference is calculated between the system's hazard for the expected new state, y^e, and for the current state, y. This must be less than a value proportional to the total hazard for the current state of the system, that is,

$$|h_i(y^e, \theta_i) - h_i(y, \theta_i)| \leq \epsilon\, h_0(y, \boldsymbol{\theta})$$

where ϵ is a specified value to satisfy $0 < \epsilon < 1$. As $\epsilon \to 0$ the algorithm becomes exact but slower. Other approaches for determining the value of τ to use are given in Gillespie and Petzold (2003) and Cao et al. (2006). A detailed description of the τ–leap algorithm is given in Gillespie (2001). A summary of the τ–leap algorithm and its possible alterations is given in Sandmann (2009).

Chemical Langevin equation (CLE)

We will follow the informal approach of Wilkinson (2011) and Golightly and Gillespie (2013) to construct the Chemical Langevin equation (CLE). To see a formal derivation please see Gillespie (1992, 2000, 2001). As described for the τ–leap algorithm, in a small time

interval $(t, t+\Delta t]$ the hazards of each reaction can be assumed to be constant and reactions occur according to a Poisson process. Let dR_t be the number of reactions that occur in $(t, t+\Delta t]$, then each element of dR_t has a Poisson distribution with $\lambda = h_i(Y_t, \theta_i)$ where $i = 1, \ldots, v$. We can capture the stochastic evolution of the system using the stochastic differential equation (SDE) for dR_t, which is based upon the mean and variance of the Poisson distribution for each element of dR_t with rate $\lambda = h_i(Y_t, \theta_i)$. The SDE is

$$dR_t = h(Y_t, \boldsymbol{\theta})dt + \text{diag}\{\sqrt{h(Y_t, \boldsymbol{\theta})}\}dW_t^*,$$

where Y_t is the current system state at time t and dW_t^* is an increment of Brownian motion. As $dY_t = SdR_t$, where S is the stoichiometry matrix, we obtain

$$dY_t = Sh(Y_t, \boldsymbol{\theta})dt + S\text{diag}\{\sqrt{h(Y_t, \boldsymbol{\theta})}\}dW_t^*,$$

where the Brownian motion W_t^* has dimension v. As $\text{Var}(dY_t) = S\text{diag}\{h(Y_t, \boldsymbol{\theta})\}S'dt$, we obtain the Chemical Langevin equation as

$$dY_t = Sh(Y_t, \boldsymbol{\theta})dt + \sqrt{S\text{diag}\{h(Y_t, \boldsymbol{\theta})\}S'}dW_t, \tag{7.9}$$

where now dW_t is an increment of standard Brownian motion in u dimensions which now matches the dimensions of the state Y_t. This equation provides a continuous approximation to the Markov–Jump process (MJP) in the form of an Itô stochastic differential equation (SDE). The CLE captures the dynamics of the system well, while assuming continuous states, and it returns the stochastic element of an MJP. Here $Sh(Y_t, \theta)$ is the drift term and $S\text{diag}\{h(Y_t, \theta)\}S'$ is the diffusion coefficient.

We can use the CLE to simulate from a SKM by numerically integrating the CLE. This could be done by using an Euler–Maruyama discretisation to approximate the change in the states of the process (Golightly and Wilkinson, 2011). For the simple death model, the CLE is

$$dY_t = -\theta y dt + \sqrt{\theta y}dW_t.$$

We have introduced the CLE in order to be able to introduce the linear noise approximation in the next section, however for a detailed explanation of how to simulate from the CLE please see Golightly and Gillespie (2013).

Linear Noise Approximation

The linear noise approximation (LNA) was first introduced by Kurtz (1970, 1971). It approximates the CLE and has the advantage that it is more numerically and analytically tractable. The LNA is a normal approximation to a system in which the mean and variance

are represented as a set of ODEs. We can use the LNA to simulate close approximate realisations from a SKM (Van Kampen, 2007). It is a popular approximation because it just requires solving a set of ODEs numerically.

A derivation of the LNA and a description of how the LNA is valid for any system that is sufficiently large in terms of the concentration of reacting species is given in Wallace et al. (2012). Ferm et al. (2008) investigate the accuracy of the LNA and Hayot and Jayaprakash (2004) provide examples of the LNA working accurately in simple genetic systems and also a case when the LNA fails due to the probability distribution of proteins being non–Gaussian. Elf and Ehrenberg (2003) use the LNA to produce fast realisations which show the stochasticity of a system at a cellular level, particularly in cases when there are low numbers of reactants and fluctuations can be large. While Komorowski et al. (2009) use a LNA approximation to construct a likelihood function in order to perform parameter inference for gene expression data.

We now derive the LNA approximation of the CLE. Let Ω be the volume of the container in which the reactions happen. Rescaling the hazard function by setting $h(Y_t, \boldsymbol{\theta}) = \Omega f(Y_t/\Omega, \boldsymbol{\theta})$ and replacing the hazard function in the CLE given in Equation (7.9), gives

$$dY_t = \Omega S f\left(\frac{Y_t}{\Omega}, \boldsymbol{\theta}\right) dt + \sqrt{\Omega S \text{diag}\left\{f\left(\frac{Y_t}{\Omega}, \boldsymbol{\theta}\right)\right\} S'} dW_t. \tag{7.10}$$

As the system becomes close to its thermodynamic limit (Ω and Y_t become large) the LNA becomes a more accurate approximation to the CLE (Golightly and Gillespie, 2013). By considering how the system would scale according to the container volume, and thinking about the Central Limit Theorem and Poisson process variation, we can write Y_t as

$$Y_t = \Omega z_t + \sqrt{\Omega} M_t, \tag{7.11}$$

where z_t is the deterministic process and M_t is the residual stochastic process. The idea is that when the average concentration (Y_t/Ω) is constant relative fluctuations will decrease with $1/\sqrt{\Omega}$ (Elf and Ehrenberg, 2003). Therefore, if we replace Y_t in Equation (7.10) with Equation (7.11), the CLE becomes

$$dz_t + \frac{1}{\sqrt{\Omega}} dM_t = S f\left(z_t + \frac{M_t}{\sqrt{\Omega}}, \boldsymbol{\theta}\right) dt + \frac{1}{\sqrt{\Omega}} \sqrt{S \text{diag}\left\{f\left(z_t + \frac{M_t}{\sqrt{\Omega}}, \boldsymbol{\theta}\right)\right\} S'} dW_t.$$

The Taylor expansion of $f\left(z_t + \frac{M_t}{\sqrt{\Omega}}, \boldsymbol{\theta}\right)$ can be used to linearise the rate function and obtain

$$f\left(z_t + \frac{M_t}{\sqrt{\Omega}}, \boldsymbol{\theta}\right) = f(z_t, \boldsymbol{\theta}) + \frac{1}{\sqrt{\Omega}} F_t M_t + O(\Omega^{-1}),$$

where $F_t = \left(\frac{\partial f_i(z_t,\boldsymbol{\theta})}{\partial z_{j,t}}\right)$ for $i = 1, \ldots, v$ and $j = 1, \ldots, u$. Using the Taylor expansion and collecting terms of $O(1)$ we get the deterministic rate equation

$$\frac{dz_t}{dt} = Sf(z_t, \boldsymbol{\theta}) \tag{7.12}$$

which is the large volume asymptotic limit of the CLE. By collecting terms of $O\left(\frac{1}{\sqrt{\Omega}}\right)$ we get the SDE for the residual process

$$dM_t = SF_tM_tdt + \sqrt{S\text{diag}\{f(z_t, \boldsymbol{\theta})\}S'}dW_t. \tag{7.13}$$

Therefore the LNA approximation of the CLE is given by Equations (7.11), (7.12) and (7.13). Initialising the system with $M_{t_1} \sim N(m_{t_0}, V_{t_0})$, where t_0 is the initial timepoint, the solution to Equation (7.13) is

$$(M_t|\boldsymbol{\theta}) \sim N(m_t, V_t),$$

where

$$\frac{dm_t}{dt} = SF_tm_t \qquad \text{and} \qquad \frac{dV_t}{dt} = V_tF'_tS' + S\text{diag}\{h(z_t)\}S' + SF_tV_t.$$

Note that the parameters m_t and V_t depend on z_t and $\boldsymbol{\theta}$ but we have removed this from our notation for simplicity. We now have a set of ODEs to solve to find the solution to Equation (7.13). Using Equation (7.11), it can be shown that

$$Y_t \sim N(\Omega z_t + \sqrt{\Omega}m_t, \Omega V_t).$$

We will assume that $\Omega = 1$ and so obtain

$$Y_t \sim N(z_t + m_t, V_t). \tag{7.14}$$

This solution to the LNA can be used to simulate (approximate) realisations from a SKM by first using numerical integration to find z_t and V_t over a small time interval and using Equation (7.14) to simulate the state of the system.

The LNA can become less accurate over time in a simulation due to the ODE for z_t being numerically integrated over the whole time range as this can lead to a difference between z_t and the true stochastic simulation. Fearnhead et al. (2012) suggest a remedy to this problem by setting z_t to be y_t at each timepoint, in which case we numerically integrate from t to $t + \Delta t$ with $z_t = y_t$ and V_t as a matrix with entries all equal to zero. Note that m_t is zero for all t so we do not need to solve the ODE for m_t. The algorithm for simulation using the LNA is given in Algorithm 15. More information can be found in Fearnhead et al. (2012) and Golightly and Gillespie (2013).

The LNA can also be used for parameter inference. The likelihood for the fully observed

system is approximated as $\pi(y|\boldsymbol{\theta}) = \prod_{i=1}^{n} \hat{\pi}_{LNA}(y_i|y_{i-1}, \boldsymbol{\theta})$ which is a product of normal densities (Fearnhead et al., 2012). MCMC algorithms can be used to approximate the posterior distribution $\pi(\boldsymbol{\theta}|y)$ by using this LNA approximation to the likelihood. This means that the MCMC scheme will target $\pi(\boldsymbol{\theta})\hat{\pi}_{LNA}(y|\boldsymbol{\theta})$. Giagos (2011) shows that the LNA provides a good approximation to $\pi(\boldsymbol{\theta}|y)$ for the LV model.

Algorithm 15 Linear noise approximation (LNA).

1. Set $t = 0$. Initialise values of $\boldsymbol{\theta}$ and initialise y_0 to be the initial values of all of the species. Let $z_0 = y$, $m_0 = y - z_0$ (all elements of m_0 should be zero) and V_0 is a $u \times u$ matrix with all entries being zero.
2. Use numerical integration to solve the set of ODEs for z_t, m_t and V_t over the time interval $(t, t + \Delta t]$.
3. Simulate $y_{t+\Delta t} \sim N(z_{t+\Delta t} + m_{t+\Delta t}, V_{t+\Delta t})$.
4. Set $t = t + \Delta t$, $m_t = y - z_t$ and reset V_t to be a matrix of zeros.
5. If $t < t_{max}$ go to Step 2, otherwise output t and y.

7.4.3 Hybrid simulation techniques

Hybrid simulation techniques combine the use of exact simulation and approximate simulation. If reactions are present in a system where there are a low concentration of reacting species, generally these reactions would happen very slowly. In this case the discrete nature of these reactions become important and should not be ignored and so using the Chemical Langevin equation (CLE) or the linear noise approximation (LNA) can give quite inaccurate realisations. In this situation, hybrid algorithms can be used effectively by splitting reactions into slow and fast reactions and using exact methods for the slow reactions and approximate algorithms for the fast algorithms. This technique takes advantage of the efficiency of the approximate algorithm but also tries to improve the accuracy for reactions where the approximation would not be suitable, and hence providing a middle ground for speed and accuracy. Examples of the use of hybrid methods include Kiehl et al. (2004) and Alfonsi et al. (2005), who numerically solve ODEs for the fast reactions, Higham et al. (2011) and Salis and Kaznessis (2005), who use a CLE approximation, and Puchalka and Kierzek (2004) who use the τ–leap algorithm.

Chapter 8

Introduction to Bayesian Experimental Design

8.1 Introduction

Consider an experimentalist who is thinking about how to design an experiment whose output is essentially a continuous trace of various possible measurements. If we assume that costs prohibit measuring these continuous traces, key questions are at how many time points should measurements be taken and at what times should the measurements be made. We will focus on the determination of such "optimal designs" by choosing the design to optimise a statistical criteria. More generally, a design could comprise of, for example, the sample size, proportions of observations to each treatment, the experimental units, the duration of the experiment, as well as the times to take observations or the species to observe (Chaloner and Verdinelli, 1995). Generally experimentalists design their experiments with reference to information already available from previous and/or similar experiments (Chaloner and Verdinelli, 1995). This leads naturally to framing the problem from a Bayesian perspective. The topic of Bayesian experimental design has had much interest in recent times. Within the literature, there are examples of where the optimal design found using Bayesian techniques has been applied to real experiments (Flournoy, 1993). Also, there are many examples in which previous experimental results have been used to construct a prior distribution for model parameters in order to design a more efficient version of the experiment; see, for example, Clyde et al. (1995a,b); Drovandi and Pettitt (2013).

Choosing an efficient (or optimal) design can save time and money as it reduces the risk of having to undertake repeat experiments. Good designs increase the information content of the experimental data and so it is natural to seek optimal designs which reduce (posterior) parameter uncertainty (Chaloner and Verdinelli, 1995).

The design of experiments from a Bayesian perspective takes into account prior knowledge about unknown quantities or parameters $\boldsymbol{\theta}$, together with the potential cost and benefits of performing the experiment expressed as a utility function. Prior information could be obtained from pilot studies, observational studies or by eliciting expert beliefs (Clyde, 2001). The utility function takes high values for experiments with large benefits and small costs and low values for experiments with small benefits and large costs. It is clear that we should choose the design that maximises the prior expectation of the utility function (Farrow, 2013). The utility function $u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta})$ is a function of the (yet unobserved) experimental data $\boldsymbol{y}$, the parameters $\boldsymbol{\theta}$ in the stochastic model and design chosen $\boldsymbol{d}$. As the utility function depends on the unknown quantities, the design should be chosen by studying the utility function after accounting for the uncertainty of these quantities, that is, chosen to maximise the expectation of $u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta})$ with respect to the joint distribution of $(\boldsymbol{\theta}, \boldsymbol{y})$. We write this expected utility as

$$u(\boldsymbol{d}) = E_{\boldsymbol{\theta},\boldsymbol{y}}[u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta})] = \int_{\boldsymbol{y}} \int_{\boldsymbol{\theta}} u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta}) \pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{y}, \tag{8.1}$$

where $\pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta})$ is the likelihood function of the observations when using design $\boldsymbol{d}$ given the parameters $\boldsymbol{\theta}$ and $\pi(\boldsymbol{\theta})$ is the prior for $\boldsymbol{\theta}$.

8.1.1 The utility function

In general utility functions describe the reward from declaring a value $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$, from declaring a posterior distribution for $\boldsymbol{\theta}$ or, in our case, from declaring an optimal design $\boldsymbol{d}$. In each case, the utility function should reflect the aims of the specific problem.

A popular choice of utility function is the expected gain in Shannon information given by the experiment (Shannon, 1948). Here the optimal design is chosen to be the design that maximises the expected gain in Shannon information. This gain in Shannon information is also the same as the expected Kullbeck–Leibler divergence between the posterior and prior distributions

$$u(\boldsymbol{d}) = \int_{\boldsymbol{y}} \int_{\boldsymbol{\theta}} \log \left(\frac{\pi(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{d})}{\pi(\boldsymbol{\theta})} \right) \pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{d}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{y}.$$

As the prior distribution is independent of the design, this can be simplified to

$$u(\boldsymbol{d}) = \int_{\boldsymbol{y}} \int_{\boldsymbol{\theta}} \log \{\pi(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{d})\} \pi(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{d}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{y},$$

that is, the expected Shannon information of $\pi(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{d})$ (Chaloner and Verdinelli, 1995). This utility is often used when the aim of the experiment is to perform parameter inference on $\boldsymbol{\theta}$, and is the analogue of D–optimality which maximises the determinant of the information matrix over all possible designs.

Another possible choice of utility function is a quadratic loss function, which can be used when the experimental aim is to find a point estimate of the parameters $\hat{\boldsymbol{\theta}}$ (Chaloner and Verdinelli, 1995; Clyde, 2001). This utility is a function linear in $-(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})'A(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})$ and is an analogue of A–optimality which minimises the trace of the inverse of the information matrix over all possible designs (Farrow, 2013). For example, the expected utility could be

$$u(\boldsymbol{d}) = -\int_{\boldsymbol{y}}\int_{\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})'A(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{d})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}d\boldsymbol{y},$$

where A is a symmetric non–negative definite matrix.

8.1.2 Utility functions for models with intractable likelihoods

The choice of optimal experimental design for models with tractable likelihoods is often based on a utility function that is a scalar function of the Fisher information matrix. In Bayesian experimental design, a common utility function uses the expected gain in Shannon information from prior to posterior or considers the concentration of the posterior distribution. In this case, when the posterior distribution can not be determined analytically, evaluations of the likelihood are required in order to sample from the posterior distribution (Drovandi and Pettitt, 2013). In all but the most simple models and designs, the posterior is not analytically tractable due to the discrete nature of the designs within a continuous–time stochastic model.

Pagendam and Ross (2013) base their utility function on the Fisher information matrix and numerically evaluate the matrix by taking the exponential of the matrix of transition rates as described in Podlich et al. (1999). Pagendam and Pollett (2009, 2010) use a Gaussian diffusion approximation for the stochastic kinetic model with large initial values, that is, they approximate the likelihood with a Gaussian distribution. Komorowski et al. (2011) use the linear noise approximation to a Markov process to estimate the likelihood, which they then use to evaluate the information matrix. This approach converts evaluation of the Fisher information matrix to solving a system of ODEs. We note that low molecule numbers reduce the accuracy of this approximation. Finally, moment closure is used in Cook et al. (2008) to approximate the likelihood.

It is possible to have a utility function that does not explicitly depend on $\boldsymbol{y}$ or $\boldsymbol{\theta}$ (Drovandi and Pettitt, 2013). One such example is a utility function based on the posterior distribution for $\boldsymbol{\theta}$. A utility function based on the Fisher information matrix is an example of a utility that does not depend on $\boldsymbol{y}$ as the expectation with respect to $\boldsymbol{y}$ has already been computed (Drovandi and Pettitt, 2013).

Recent work by Drovandi and Pettitt (2013) provides an algorithm based on Markov chain Monte Carlo sampling and approximate Bayesian computation (ABC) which finds optimal Bayesian designs for a Markov process. The method only requires simulation from

the stochastic kinetic model. They use a utility function that is not a function of $\boldsymbol{\theta}$, based on the posterior variance matrix of $\boldsymbol{\theta}$, that is, they use the posterior generalised precision

$$u(\boldsymbol{d}, \boldsymbol{y}) = \frac{1}{\det\{Var(\boldsymbol{\theta}|\boldsymbol{d}, \boldsymbol{y})\}}, \tag{8.2}$$

as their utility function. This function takes higher values when there is less uncertainty about $\boldsymbol{\theta}$. Note that, as the utility function does not depend on $\boldsymbol{\theta}$, we drop the $\boldsymbol{\theta}$ argument in the function's notation. The authors approximate the utility function using ABC techniques. Their aim, like that in this thesis, is to determine the optimal timepoints to observe a stochastic kinetic model (assuming a fixed number of timepoints). Their method is more computationally expensive than typical parameter inference as the parameter posterior distribution needs to be estimated everytime they wish to evaluate the utility function which is required at every iteration of their MCMC scheme.

8.2 The Drovandi and Pettit approach

The optimal design $\boldsymbol{d}^*$ is the design which maximises the expected utility $u(\boldsymbol{d})$ in Equation (8.1). However, $u(\boldsymbol{d})$ is typically analytically intractable. Although it is possible to evaluate $u(\boldsymbol{d})$ using Monte Carlo integration, this technique becomes impractical as the design dimension increases. Müller (1999) suggests an alternative approach to evaluating the expected utility by sampling from the joint density

$$h(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta}) \propto u(\boldsymbol{d}, \boldsymbol{y})\pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta})\pi(\boldsymbol{\theta}) \tag{8.3}$$

using MCMC (see Algorithm 16). This method is very efficient in comparison to Monte Carlo sampling which is the commonly used alternative (Müller, 1999). This method works because this distribution has a marginal distribution (over $\boldsymbol{\theta}$ and $\boldsymbol{y}$) which is proportional to $u(\boldsymbol{d})$. The optimal design is then determined by locating the mode of this marginal distribution using the posterior samples of $\boldsymbol{d}$.

Drovandi and Pettitt (2013) use the MCMC scheme suggested by Müller (1999) which is given in Algorithm 16. This algorithm is useful in experimental design for models with intractable likelihoods as the likelihood terms in the Metropolis–Hastings ratio cancel as a result of simulating from the model as a proposal for the data.

Often the marginal distribution for $\boldsymbol{d}$ is fairly flat and so it can be advantageous to amplify the signal for the location of the mode by modifying the design to have J replicates of the experiment (Müller, 1999). In this case, the target for the MCMC algorithm is the density

$$h(\boldsymbol{d}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_J) \propto \prod_{j=1}^{J} u(\boldsymbol{d}, \boldsymbol{y}_j)\pi(\boldsymbol{y}_j|\boldsymbol{d}, \boldsymbol{\theta}_j)\pi(\boldsymbol{\theta}_j). \tag{8.4}$$

Algorithm 16 MCMC algorithm for experimental design by Müller (1999).

1. Initialise $\boldsymbol{d}^0$, sample $\boldsymbol{\theta}^0 \sim \pi(\boldsymbol{\theta})$ and simulate $\boldsymbol{y}^0 \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}^0, \boldsymbol{d}^0)$.
2. Calculate $u^0 = u(\boldsymbol{d}^0, \boldsymbol{y}^0)$.
3. Set $i = 1$. While $i \leq n$
   (a) Propose $\boldsymbol{d}^c \sim q(\boldsymbol{d}|\boldsymbol{d}^{i-1})$, $\boldsymbol{\theta}^c \sim \pi(\boldsymbol{\theta}), \boldsymbol{y}^c \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}^c, \boldsymbol{d}^c)$.
   (b) Calculate $u^c = u(\boldsymbol{d}^c, \boldsymbol{y}^c)$.
   (c) Accept proposed states with probability
   $$\min\left(1, \frac{u^c q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)}{u^{i-1} q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})}\right).$$
   (d) Set $i = i + 1$ and return to step 3(a).

This distribution has a marginal distribution over $\boldsymbol{\theta}_j$ and $\boldsymbol{y}_j$ for $j = 1, \ldots, J$ which is proportional to $u(\boldsymbol{d})^J$ (Drovandi and Pettitt, 2013; Müller, 1999). Therefore the optimal design is the mode of this marginal distribution. This modification requires an alteration to the MCMC algorithm as, for each design $\boldsymbol{d}$, we need to simulate J independent samples of $\boldsymbol{\theta}$ from the prior and then J independent samples of $\boldsymbol{y}$ given these values of $\boldsymbol{\theta}$. The utility is then calculated for each pair $(\boldsymbol{\theta}_j, \boldsymbol{y}_j)$ and then their product is taken. This is described in Algorithm 17. A larger value of J indicates that they have a tighter distribution around the mode. However, larger values of J will increase the time taken to perform each iteration of the MCMC scheme linearly (Drovandi and Pettitt, 2013).

This work uses the generalised precision as the utility function given in Equation 8.2. Although Drovandi and Pettitt (2013) could have made other choices such as the Kullback–Leibler divergence between the prior distribution and the posterior distribution (Kullback and Leibler, 1951), this type of utility function requires that the likelihood be evaluated: a problem for the types of model we consider in this thesis. That said, Liepe et al. (2013) have suggested estimating this quantity using the difference between histograms for the prior and posterior distribution computed using the samples from each distribution. They estimate their utility function using ABC methods as the ABC posterior distribution does not require the likelihood to be evaluated. If we measure the distance between the simulated "true" data $\boldsymbol{y}$ and an ABC dataset $\boldsymbol{x}$ using a metric $\rho(\boldsymbol{y}, \boldsymbol{x})$ and require this distance to be within ϵ then the ABC posterior is

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}, \epsilon) = \int_{\boldsymbol{x}} \pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})1(\rho(\boldsymbol{y}, \boldsymbol{x}) \leq \epsilon)d\boldsymbol{x}, \tag{8.5}$$

Algorithm 17 MCMC algorithm for experimental design by Müller (1999) amplifying the mode.

1. Initialise $\boldsymbol{d}^0$, sample $\boldsymbol{\theta}_j^0 \sim \pi(\boldsymbol{\theta})$ and simulate $\boldsymbol{y}_j^0 \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}^0, \boldsymbol{d}^0)$ for $j = 1, \ldots, J$.
2. Calculate $u(\boldsymbol{d}^0, \boldsymbol{y}_j^0)$ for $j = 1, \ldots, J$.
3. Calculate $u_0 = \prod_{j=1}^{J} u(\boldsymbol{d}^0, \boldsymbol{y}_j^0)$.
4. Set $i = 1$. While $i \leq n$
   (a) Propose $\boldsymbol{d}^c \sim q(\boldsymbol{d}|\boldsymbol{d}^{i-1})$, $\boldsymbol{\theta}_j^c \sim \pi(\boldsymbol{\theta}), \boldsymbol{y}_j^c \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}_j^c, \boldsymbol{d}^c)$ for $j = 1, \ldots, J$.
   (b) Calculate $u(\boldsymbol{d}^c, \boldsymbol{y}_j^c)$ for $j = 1, \ldots, J$.
   (c) Calculate $u^c = \prod_{j=1}^{J} u(\boldsymbol{d}^c, \boldsymbol{y}_j^c)$.
   (d) Accept proposed states with probability
   $$\min\left(1, \frac{u^c q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)}{u^{i-1} q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})}\right).$$
   (e) Set $i = i + 1$ and return to step 4(a).

where $1(A)$ is a binary function which is one if A is true. Drovandi and Pettitt (2013) use the metric

$$\rho^k \equiv \rho(\boldsymbol{y}, \boldsymbol{x}_k) = \sum_{i=1}^{D} \frac{|y_i - x_{i,k}|}{\text{std}(x_{i,\cdot})},$$

where D is the number of design points and $\text{std}(\cdot)$ is the sample standard deviation. The utility function is calculated by replacing the true posterior with the ABC posterior, and the ABC posterior is determined for each simulated dataset $\boldsymbol{y}$ at every iteration in Algorithm 16. The ABC rejection algorithm is given in Algorithm 18.

The ABC algorithm requires a tuning parameter α, where $\epsilon = \rho^{\alpha N_{\text{ABC}}}$ and N_{ABC} is the number of ABC simulated datasets. This parameter controls the proportion of particles that are used to approximate the ABC posterior. Choosing α is a balance between posterior accuracy and Monte Carlo error. A low value of α provides a sample that is closer to the true posterior distribution. However, for a particular N_{ABC}, a low α means that fewer samples are kept and this increases the Monte Carlo error of the utility function. We note that substantial computational savings can be achieved by performing steps 1 and 2 of Algorithm 18 and storing the output *before* running the full analysis.

The designs $\boldsymbol{d}$ considered consist only of the timepoints at which measurements should be taken. Pragmatically, their method discretises the time axis and so they need only simulate data at each of the (relatively small number of) design points when constructing the ABC datasets $\boldsymbol{x}_k$. This also allows calculation of the sample standard deviations in

Algorithm 18 ABC rejection algorithm.

1. Generate $\boldsymbol{\theta}^k \sim p(\boldsymbol{\theta})$ for $k = 1, \ldots, N_{\text{ABC}}$.
2. Simulate $\boldsymbol{x}^k \sim p(\boldsymbol{y}|\boldsymbol{\theta}^k, \boldsymbol{d})$ for $k = 1, \ldots, N_{\text{ABC}}$.
3. Calculate discrepancies ρ^k for $k = 1, \ldots, N_{\text{ABC}}$, making particles $\{\boldsymbol{\theta}^k, \rho^k\}_{k=1}^{N_{\text{ABC}}}$.
4. Sort the particles according to the discrepancy ρ.
5. Discard $(1 - \alpha)N_{\text{ABC}}$ of the particles with the highest discrepancy. Note that $\epsilon = \rho^{\alpha N_{\text{ABC}}}$.

advance, for use in the ABC discrepancy function.

8.2.1 Finding the multivariate modal design

Müller (1999) advises that the boosting parameter J should be chosen so that the distributions of the designs have low variability, in which case the (trimmed) sample mean will be a good approximation to the optimal design. However, having J too large can cause problems in the MCMC scheme if the utility surface is multimodal as the scheme becomes stuck in a local mode (Hainy et al., 2013). Müller et al. (2004) suggest increasing J slowly during the MCMC scheme since, as J increases, the MCMC samples focus on the mode which corresponds to the highest utilities (Hainy et al., 2013). This strategy produces an inhomogeneous Markov chain of the design variables (Hainy et al., 2013) and the whole design space is explored by the MCMC sampler initially, ensuring that all modes are covered.

Drovandi and Pettitt (2013) restrict the range of J values they consider as large values lead to very long run times for their algorithm. This leads to their marginal modes (of the designs) being less accurate estimates of the multivariate modal design. In this thesis, we follow their approach to estimating the multivariate modes which employs the non-parametric approach described in Algorithm 19.

8.2.2 Example using the death model

We illustrate this method using the example given in Drovandi and Pettitt (2013) of the death model described in Chapter 7.3. We use Algorithm 16 and, in steps 2 and 3b) where the utility function is evaluated, we use Algorithm 18 to obtain an approximation of the posterior variance matrix for $\boldsymbol{\theta}$. This matrix is then used in Equation (8.2) to approximate the utility function. We replicate their example by taking their log–normal prior $\theta \sim LN(-0.005, 0.01)$ and discretising the time axis $t_{\min} = 0.01$, $t_{\max} = 10$ into steps of size $\Delta t = 0.01$. The MCMC scheme in Algorithm 17 is run for $n = 100k$ iterations. Also

Algorithm 19 Finding the multivariate mode (Drovandi and Pettitt, 2013)

Suppose the design takes the form $\boldsymbol{d} = (t_1, \ldots, t_{n_d})$.

1. For $i = 1, \ldots, n_d$ find the bandwidth, h_i, of the kernel density estimator for each timepoint, t_i, $i = 1, \ldots, n_d$ using the marginal samples of t_i.

2. For each of the smoothing factors $h = (1, 1.05, 1.1, 1.15, 1.2, 1.25, 1.3, 1.35, 1.4, 1.45, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.25, 2.5, 2.75, 3, 3.5, 4)$, use a multivariate Gaussian smoothing kernel with $h\,\mathrm{diag}(h_i)$ as the bandwidth matrix to find a set of points of highest density (one for each h).

3. Approximate the expected utility $u(\boldsymbol{d}^*)$ at each of the possible designs using

$$u(\boldsymbol{d}^*) = \frac{1}{N} \sum_{i=1}^{N} \hat{u}(\boldsymbol{d}^*, \boldsymbol{y}_j)$$

   where $\boldsymbol{y}_j \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}_j, \boldsymbol{d}^*)$ and $\boldsymbol{\theta}_j \sim \pi(\boldsymbol{\theta})$. The utility $\hat{u}$ is approximated using ABC as described in Algorithm 18.

4. Choose the optimal design to be the modal design that produces the highest approximated expected utility.

the ABC algorithm uses $N_{\mathrm{ABC}} = 200k$ pre–computed model simulations and keeps 200 values for the ABC posterior ($\alpha = 0.001$).

Figure 8.1 shows plots of the marginal distributions for the designs; the multivariate modes of these distributions are the optimal times to take observations. Estimates of the optimal designs have been calculated as the multivariate mode using Algorithm 19 and are given in Tables 8.1. These results are similar to those given in Drovandi and Pettitt (2013). It is likely that these differences are due to the dependence on the pre–computed datasets to be used in the ABC. This dependency is probably due to the way in which the 'closeness' of datasets is judged within the ABC algorithm. Usually, a maximum value ϵ for the discrepancy is specified. However this ABC algorithm simply selects the closest, say 200 datasets, some of which may not be particularly close to the simulated 'real' dataset. We study this aspect more fully in the next section. It is clear from these runs that the algorithm requires lots of memory to store the ABC datasets and so the method will not scale well to larger models with more species and designs with a large number of timepoints. In the next chapter we explore a method which does not require such large amounts of memory which is based on using Gaussian Processes.

# timepoints (d)	ABC1	ABC2	Exact	D&P
1	(1.623)	(1.571)	(1.60)	(1.60)
	0.005%	0.005%	0.000%	0.000%
2	(1.14, 2.85)	(1.07, 2.76)	(1.03, 2.65)	(1.15, 3.05)
	0.078%	0.018%	0.000%	0.198%
3	(0.90, 2.00, 3.83)	(0.86, 2.00, 3.76)	(0.76, 1.79, 3.42)	(0.75, 1.90, 3.90)
	0.116%	0.075%	0.000%	0.125%
4	(0.74,1.69,	(0.74,1.69,	(0.60,1.36,	(0.75,1.70,
	2.74,3.94)	2.74,3.94)	2.38,3.98)	2.75,4.35)
	0.214%	0.214%	0.000%	0.214%

Table 8.1: Optimal design results for the death model. Optimal designs, $\boldsymbol{d}^* = (t_1^*, \ldots, t_d^*)$ are given in brackets and the sub–optimalities of those designs are given as a percentage below each design. ABC1 and ABC2 are two repeats of the Drovandi and Pettitt (2013) methods, D&P are the Drovandi and Pettitt (2013) results and using numerical integration we have calculated the exact optimal design which is given in the Exact column.

8.2.3 Dependence of the expected utility on the pre–computed ABC datasets

The above results suggest that there is some variability in the optimal design due to the actual simulated ABC datasets used. We investigate this proposition by a series of analyses using different collections of ABC datasets. In each case, the ABC dataset collections contain $200k$ datasets. In the ABC algorithm, a simulated 'real' dataset is compared to those in an ABC dataset collection. We have investigated how the variability in the (estimated) expected utility depends on $m = 1000$ different simulated 'real' datasets and 20 different ABC dataset collections. Note that in each case we keep 200 parameter values (and ties) corresponding to those simulations in the ABC dataset with the lowest discrepancy. The additional tied values are those resulting from datasets which have equal discrepancy to that of the 200th ranked dataset.

The different sources of variability in the (estimated) expected utilities can be assessed by using a two–way analysis of variance (ANOVA). This model can be written as

$$u_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij},$$

where $\epsilon_{i,j} \sim N(0, \sigma^2)$. Here μ is the overall mean expected utility, α_i is the (additive mean) effect using the simulated 'real' dataset i, and β_j is the (additive mean) effect of making comparisons to ABC dataset collection j. The ANOVA table given in Table 8.2 shows that there are considerable differences in the (estimated) expected utility between different ABC dataset collections. The accuracy of these (estimated) expected utilities (in

	SS	Df	MS	F ratio
Within ABC dataset collection	5350806	999	5356.2	1210.359
Between ABC dataset collections	4795	19	252.3	57.023
Residual	83996	18981	4.4	

Table 8.2: ANOVA table for dataset collections containing $200k$ ABC datasets, keeping 200 parameter values (and ties) for the ABC posterior. Note the upper 0.1 percentiles $F_{999,18981} = 1.1483$ and $F_{19,18981} = 2.380$.

say collection j) is

$$s.e.(u_{.,j}) = \sqrt{\frac{4.4}{20000} + \frac{252.3}{20}} = \sqrt{12.61522} \simeq 3.552.$$

Figure 8.2(a) shows 95% confidence intervals for the (estimated) expected utility calculated for each ABC dataset collection. The figure highlights the high variability in these estimates, though we note that each interval contains the exact expected utility of 133.13.

If instead we keep 100 parameter values (and ties) for the ABC posterior so that the ABC datasets are closer to the simulated 'real' dataset, then we obtain the results shown in Table 8.3. Again we see that there are considerable differences between ABC dataset collections, suggesting that there are still many of the included 100 ABC datasets that are not particularly close the the simulated 'real' datasets. The accuracy of the utilities can be seen in their 95% confidence intervals, shown in Figure 8.2(b). These intervals have similar widths to those obtained when keeping 200 parameter values (and ties).

We now increase the number of datasets in the dataset collections to $1M$ in an attempt to ensure that the top 200 matching datasets are indeed close to the simulated 'real' datasets. The results are given in Table 8.4. Although there are still large differences between the dataset collections, this source of variability has reduced considerably: see the 95% confidence intervals for the expected utility in Figure 8.2(c). These intervals are much narrower. We repeated this analysis but kept only the top 100 parameter values for the ABC posterior and found this gave exactly the same results as in Table 8.4. This suggests that, in these runs, the discrepancies for the 100th closest ABC dataset and the 200th closest ABC dataset were the same (or very close).

This investigation has showed that a very large number of ABC datasets need to be used before the (estimated) expected utility can be determined accurately. Unfortunately this poses a very real problem for the memory usage and computing time for any ABC determination of optimal designs.

	SS	Df	MS	F ratio
Within ABC dataset collection	5351632	999	5357.0	1054.631
Between ABC dataset collections	4972	19	261.7	51.514
Residual	96414	18981	5.1	

Table 8.3: ANOVA table for dataset collections containing $200k$ ABC datasets, keeping 100 parameter values (and ties) for the ABC posterior. Note the upper 0.1 percentiles $F_{999,18981} = 1.1483$ and $F_{19,18981} = 2.380$.

	SS	Df	MS	F ratio
Within ABC dataset collection	5325218	999	5330.5	5536.194
Between ABC dataset collections	384	19	20.2	20.965
Residual	18276	18981	1.0	

Table 8.4: ANOVA table for dataset collections containing $1M$ ABC datasets, keeping 200 parameter values (and ties) for the ABC posterior. Note the upper 0.1 percentiles: $F_{999,18981} = 1.1483$ and $F_{19,18981} = 2.380$.

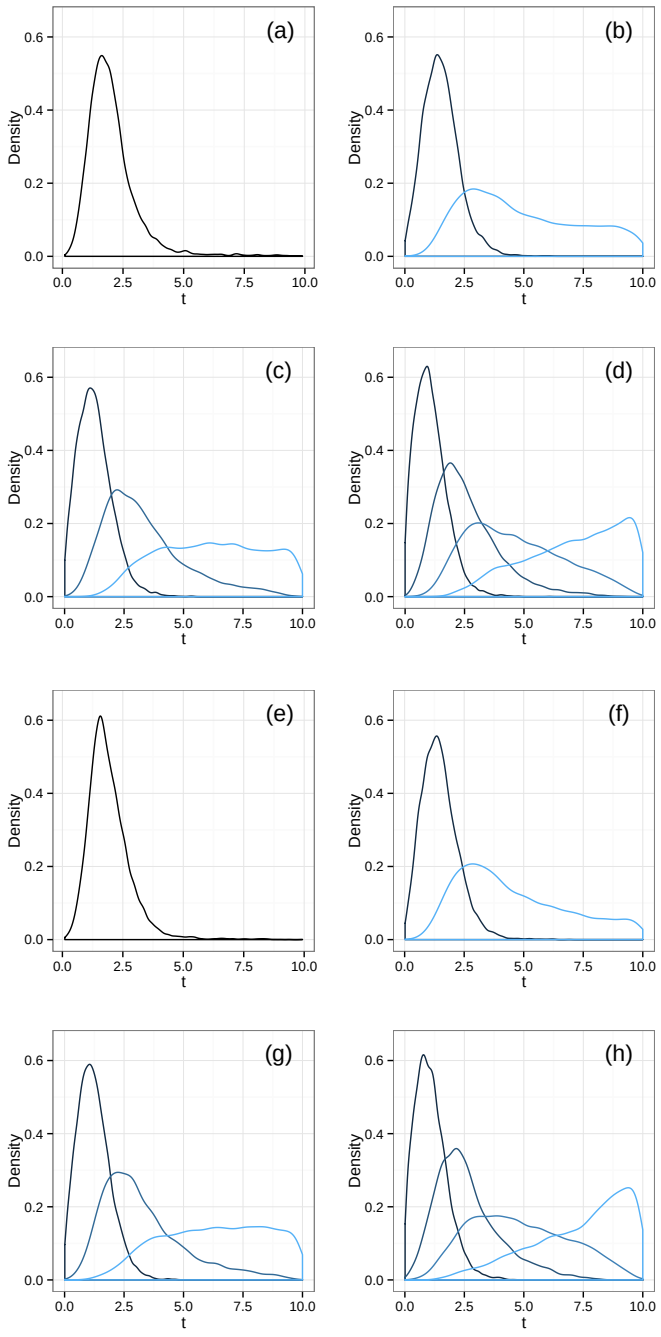


Figure 8.1: Marginal distributions for $\boldsymbol{d}$ for a (a) single, (b) two, (c) three and (d) four timepoint design for the death model for the first repeat and marginal distributions for $\boldsymbol{d}$ for a (e) single, (f) two, (g) three and (h) four timepoint design for the death model for the second repeat.

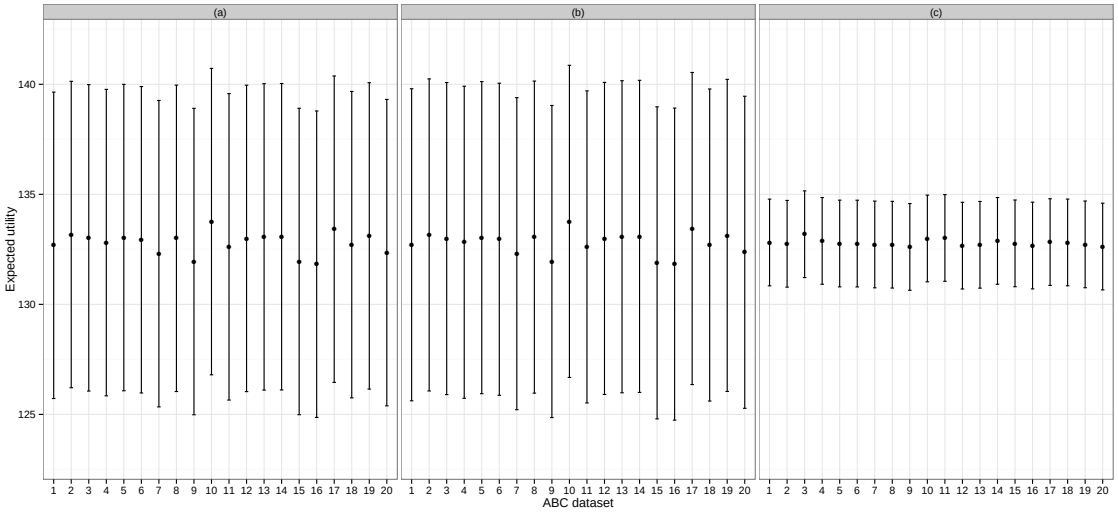


Figure 8.2: Plots showing the confidence intervals (black lines) for each ABC dataset collection, $j = 1, \ldots, 20$ with the mean as a black point for collections containing (a) $200k$ datasets in which 200 are kept, (b) $200k$ datasets in which 100 are kept and (c) $1M$ datasets in which 200 (or 100) are kept in the ABC posterior.

Chapter 9

Experimental design using Gaussian processes

In this chapter we consider how we might use Gaussian processes to reduce the computational cost of determining optimal experimental designs. We begin by introducing Gaussian processes, describe how a Gaussian process can be fitted to data, and then describe some diagnostic tools to assess fit. We also consider how to select the size and location of the training data which is used to fit the Gaussian process. Finally, we discuss how the fitted Gaussian process can be used to find the optimal design.

Recall that the objective of this part of the thesis is to determine the optimal times at which to observe a stochastic kinetic model. Using the same notation in Chapter 8, we write $\boldsymbol{d} = (t_1, \ldots, t_d)$ for the d-timepoint design at which we take observations $\boldsymbol{y}$ from the experiment. For example, in a single timepoint design, we have $\boldsymbol{d} = t_1$ and the observations taken are $\boldsymbol{y} = y_{t_1}$. We note that all observations (without noise) from a stochastic kinetic model are integers, that is, are discrete. Also the likelihood function when using design $\boldsymbol{d}$ is $\pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta})$.

The optimal design is chosen to maximise the expected utility

$$u(\boldsymbol{d}) = E_{\boldsymbol{y}}\left[u(\boldsymbol{d}, \boldsymbol{y})\right] \tag{9.1}$$

which can be rewritten as

$$u(\boldsymbol{d}) = E_{\boldsymbol{\theta}}\left[u(\boldsymbol{d}, \boldsymbol{\theta})\right] = \int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}) u(\boldsymbol{d}, \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{9.2}$$

where

$$u(\boldsymbol{d}, \boldsymbol{\theta}) = E_{\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{d}}\left[u(\boldsymbol{d}, \boldsymbol{y})\right] = \int_{\boldsymbol{y}} u(\boldsymbol{d}, \boldsymbol{y}) \pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta}) d\boldsymbol{y}. \tag{9.3}$$

Throughout this thesis we use the utility function

$$u(\boldsymbol{d}, \boldsymbol{y}) = \frac{1}{\det\{(Var(\boldsymbol{\theta}|\boldsymbol{d}, \boldsymbol{y})\}} \tag{9.4}$$

which is the posterior generalised precision as this utility function takes the optimal design as that which reduces posterior uncertainty about $\boldsymbol{\theta}$. Note that the utility function (9.4) does not depend explicitly on $\boldsymbol{\theta}$. Also, as $\boldsymbol{y}$ is discrete, Equation (9.3) simplifies to

$$u(\boldsymbol{d}, \boldsymbol{\theta}) = E_{\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{d}}\left[u(\boldsymbol{d}, \boldsymbol{y})\right] = \sum_{\boldsymbol{y}} u(\boldsymbol{d}, \boldsymbol{y})\pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta}). \tag{9.5}$$

We choose to fit a Gaussian processes to $u(\boldsymbol{d}, \boldsymbol{\theta})$ rather than to $u(\boldsymbol{d}, \boldsymbol{y})$ as this makes the calculation of the expected utility straightforward by using realisations from the prior for $\boldsymbol{\theta}$ using Equation (9.2) and avoids the need to average $u(\boldsymbol{d}, \boldsymbol{y})$ over the much large range of values for $\boldsymbol{y}$ (from its prior predictive distribution).

The main problem in determining the optimal design is the time it takes to evaluate the expected utility at a particular iteration in the MCMC scheme, that is, for a particular choice of $(\boldsymbol{d}, \boldsymbol{\theta})$. Therefore we seek a Gaussian process approximation to the expected utility $u(\boldsymbol{d}, \boldsymbol{\theta})$. Here the inputs to the process are the model parameters $\boldsymbol{\theta}$ and the design $\boldsymbol{d}$. In this chapter, we illustrate the general method using the death model as the likelihood can be determined exactly for this model, and hence we can determine the accuracy of using the Gaussian process approximation.

9.1 Introduction to Gaussian processes

Rasmussen and Williams (2006) define a Gaussian process as *a collection of random variables any finite number of which have a joint Gaussian distribution.* A Gaussian process is specified by its mean and covariance functions. The flexibility of a Gaussian process to fit a variety of response variable surfaces makes them a popular choice for emulation (Kaufman et al., 2011). It also satisfies the intuitively appealing property of relatively low uncertainty close to design points and increasing uncertainty as the distance from a design point increases (Kaufman et al., 2011).

In this chapter we will introduce Gaussian processes with training data inputs denoted as $\boldsymbol{X} = (\boldsymbol{x}_i)$ and outputs denoted as $\boldsymbol{y} = (y_i)$, $i = 1, \ldots, n_d$. Suppose we have data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, n_d\}$ where $y_i = f(\boldsymbol{x}_i)$, that is, we observe $\boldsymbol{y}$ exactly at a collection of input points. We need to specify a mean function $m(\boldsymbol{x})$ and usually this depends on parameters $\boldsymbol{\beta}$. For example, we might take the mean function to be the least squares regression of a function in $\boldsymbol{x}$. Using such a function we can then fit a zero mean

Gaussian process to the residuals

$$z(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) - m(\boldsymbol{x}_i) \quad \text{for} \quad i = 1, \ldots, n_d.$$

The covariance function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ describes the dependence that is believed to be between the function f at two input points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Essentially it describes the smoothness of the function and it usually depends on additional parameters, known as hyperparameters. It is this assumption of smoothness of the output space which is fundamental to the accuracy of a Gaussian process.

The purpose of fitting a Gaussian process is to be able to describe the values the function will take at input values other than those used for fitting the process (in the form of a distribution). Suppose we already have the data $\mathcal{D}$ but would like to know the value of the function at a set of test inputs $\boldsymbol{X}^* = (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{n^*}^*)'$, that is, learn the value of $\boldsymbol{f}^* \equiv f(\boldsymbol{X}^*) = (f(\boldsymbol{x}_1^*), \ldots, f(\boldsymbol{x}_{n^*}^*))'$. If we assume a Gaussian process prior for $\boldsymbol{y} = (f(\boldsymbol{x}_i))$ then we have

$$\boldsymbol{y} \sim N(m(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X})) \qquad \text{and} \qquad \boldsymbol{f}^* \sim N(m(\boldsymbol{X}^*), K(\boldsymbol{X}^*, \boldsymbol{X}^*)).$$

Now $\text{Cov}(\boldsymbol{f}^*, \boldsymbol{y}) = \text{Cov}(f(\boldsymbol{X}^*), f(\boldsymbol{X})) = K(\boldsymbol{X}^*, \boldsymbol{X})$ and similarly $\text{Cov}(\boldsymbol{y}, \boldsymbol{f}^*) = K(\boldsymbol{X}, \boldsymbol{X}^*)$. Also, as $(\boldsymbol{y}, \boldsymbol{f}^*)$ can be described by the Gaussian process we have that their joint distribution is normal with

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{f}^* \end{pmatrix} \sim N\left\{ \begin{pmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}^*) \end{pmatrix}, \begin{pmatrix} K(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}, \boldsymbol{X}^*) \\ K(\boldsymbol{X}^*, \boldsymbol{X}) & K(\boldsymbol{X}^*, \boldsymbol{X}^*) \end{pmatrix} \right\}. \tag{9.6}$$

Conditioning this joint density on the data $\boldsymbol{y}$ gives the 'posterior' distribution

$$\boldsymbol{f}^* | \boldsymbol{y} \sim N(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$$

where

$$\boldsymbol{\mu}^* = m(\boldsymbol{X}^*) + K(\boldsymbol{X}^*, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} (\boldsymbol{y} - m(\boldsymbol{X}))$$

and

$$\boldsymbol{\Sigma}^* = K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X}) K(\boldsymbol{X}, \boldsymbol{X})^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*).$$

Suppose now that the observations on $\boldsymbol{y}$ are noisy with $y_i = f(\boldsymbol{x}_i) + \epsilon$ and $\epsilon \sim N(0, \sigma^2)$. The aim is still to make statements about $\boldsymbol{f}^* = (f(\boldsymbol{x}_1^*), \ldots, f(\boldsymbol{x}_{n^*}^*))'$. Following a similar derivation to that above alters the posterior mean function to

$$\boldsymbol{\mu}^* = m(\boldsymbol{X}^*) + K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \mathbf{I} \right]^{-1} (\boldsymbol{y} - m(\boldsymbol{X}))$$

and variance matrix to

$$\boldsymbol{\Sigma}^* = K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \mathbf{I}\right]^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*).$$

Notice that the measurement error σ^2 appears in the mean function $\boldsymbol{\mu}^*$ and so the mean function will only pass through the training data outputs, $\boldsymbol{y}$ if $\sigma^2 = 0$, that is, if there is no measurement error. Put another way, the mean function will no longer pass though all of the training data outputs $\boldsymbol{y}$ when $\sigma^2 > 0$, though the smaller σ^2 is, the closer the mean function will be to the design points.

The so called *nugget* term σ^2 is important as it accounts for error in the training data output and can also help with computational problems when inverting the variance matrix. These problems occur when the condition number, the ratio of the biggest and smallest eigenvalues, becomes large. This difficulty is well known (Ababou et al., 1994) and adding the nugget term to the diagonal of the variance matrix reduces the numerical instability. Gramacy and Lee (2012) state that the addition of this jitter term can protect against small violations of the assumption of stationarity of the covariance function.

9.1.1 The mean function

The mean function $m(\cdot)$ is chosen to reflect prior knowledge of the structure of $f(\boldsymbol{x})$. Typically this will take the form

$$m(\boldsymbol{x}) = \sum_{i=1}^{p} \hat{\beta}_i h_i(\boldsymbol{x})$$

where $h(\cdot)$ is a specified function of inputs. A very simple mean function would take $p = 1$ and $h_1(\boldsymbol{x}) = 1$ so that $m(\boldsymbol{x}) = \hat{\beta}_1$. This means that $\hat{\beta}_1$ is an average value for the output $f(\boldsymbol{x})$ and would be appropriate if it was thought that the function was essentially constant. However, care must be taken in specifying the mean function as the fitted Gaussian process is mean reverting, that is, the process reverts to the mean function in areas not close to the training data. This is particularly a problem if there are large distances between training data inputs. That said, the constant mean function has been a popular choice; see, for example, Oakley and O'Hagan (2004) and Williams and Barber (1998).

We will use another commonly used mean function, namely a fitted regression function, that is, use

$$m(\boldsymbol{x}) = \sum_{i=1}^{p} \hat{\hat{\beta}}_i h_i(\boldsymbol{x}) \tag{9.7}$$

where the $\hat{\hat{\beta}}_i$ are the least squares estimates of the $\hat{\beta}_i$. In this case the residuals $z(\boldsymbol{X})$ have shorter range correlations and result in more efficient predictions (Kaufman et al.,

2011). More complex mean functions can be used; for example, Kaufman et al. (2011) use Legendre polynomials.

9.1.2 The covariance function

After specifying the mean function, the covariance function needs to be chosen before the Gaussian process prior is fully described. The covariance function is important as it describes the relationship of two outputs based on how close any two inputs are. Typically we would expect to observe similar outputs for inputs that are close to each other (Bastos, 2010). The covariance function describes the covariance between the corresponding outputs when the inputs are $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Therefore the choice of covariance function must be restricted to ones which result in a non-negative definite, symmetric and invertible variance matrix for all inputs. Often stationary covariance functions are used in which the covariance function depends simply on $\boldsymbol{x}_i - \boldsymbol{x}_j$, the distance between two inputs, rather than their actual position. In this case the covariance function does not depend on where the inputs are in the input space, just on the distance between them. These covariance functions have the property that they do not change under translation of the input space (Rasmussen and Williams, 2006). In cases where the assumption of stationarity is not appropriate, a suitable mean function is often fitted in order to remove large scale variation, with the hope that a stationary covariance function is more suitable for the residuals.

A widely used covariance function, and one we used repeatedly in this thesis, is the squared exponential covariance function which is of the form

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j | a, \boldsymbol{r}) = a \exp\left\{-(\boldsymbol{x}_i - \boldsymbol{x}_j)' \mathrm{diag}(r_1, r_2, \ldots, r_{n_p})^{-2} (\boldsymbol{x}_i - \boldsymbol{x}_j)/2\right\}. \tag{9.8}$$

This function depends on hyperparameters $(a, \boldsymbol{r})$. The hyperparameter a is a variance term as it describes the vertical scale of variation of the output. The hyperparameters $\boldsymbol{r}$ describe the length scale and dictate the smoothness the function f (Bazi and Melgani, 2010), that is, they describe how far we can move away from a particular input before the output is classed as uncorrelated. When $\boldsymbol{r}$ is large the covariance is essentially constant and so is independent of the input values (Rasmussen and Williams, 2006). Note that each dimension of the input has its own length scale so that the covariance function can depend on each input dimension differently.

One key difficulty in fitting a Gaussian process to data is that this involves inverting the $n_p \times n_p$ covariance matrix: an $\mathcal{O}(n_p^3)$ algorithm. However, we reduce the computational cost by avoiding directly inverting the matrix by using a Cholesky decomposition and solving a linear equation. This roughly halves the computation time.

Figure 9.1 shows some Gaussian processes that have been fitted to training data assuming a zero prior mean function. Here the input is $\boldsymbol{x} = t_1$ and the output is the utility

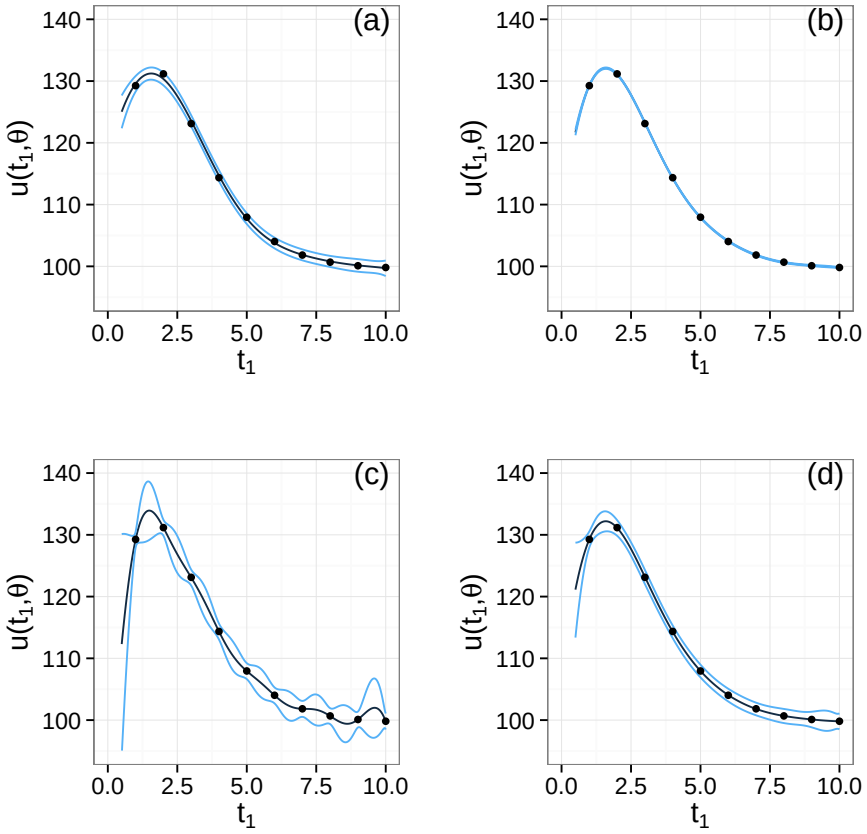


Figure 9.1: Examples of fitting a Gaussian process with input t_1 and output $u(t_1, \theta)$, where $\theta = 1$. The posterior mean is plotted as a black line and the posterior mean $\pm$ 2 standard deviations are plotted as blue lines. Plot (a) has the hyperparameters fitted at their posterior means $(r = 3.56, a = 1533, \sigma = 0.65)$, (b) has $(r = 3.56, a = 1533, \sigma = 0.0.05)$ to show the effect of reducing noise σ, (c) has $(r = 1.2, a = 1533, \sigma = 0.65)$ and (d) has $(r = 3.56, a = 1,000,000, \sigma = 0.65)$. The black points are the training data used.

$u(t_1, \theta)$ for a single timepoint design (therefore $\boldsymbol{d} = t_1$) for the death model assuming $\theta = 1$. The figure clearly shows how reducing σ leads to the mean function becoming closer to the training data; see Figure 9.1(b). Decreasing r decreases the influence of nearby training data; see Figure 9.1(c). Also increasing the hyperparameter a increases the variance and widens the prediction intervals of the Gaussian process; see Figure 9.1(d). Finally the figures show the effect of the prior mean function as the function is fitted to input values away from the training data.

9.1.3 Determining the hyperparameters

When fitting a Gaussian process, we need appropriate values for its unknown parameters. As we assume a zero prior mean function, these parameters are the hyperparameters $(a, \boldsymbol{r})'$

Algorithm 20 Inferring $(a, \boldsymbol{r}, \sigma)$

1. Initialise $(a^0, \boldsymbol{r}^0, \sigma^0)$ by sampling from their prior distributions and set $j = 1$.
2. While $j \leq m$
   (a) Propose $(a^c, \boldsymbol{r}^c, \sigma^c)$ from a symmetric random walk with independent components on the log scale.
   (b) Accept proposed states with probability

$$\frac{\pi(a^c)}{\pi(a^{j-1})}\frac{\pi(\boldsymbol{r}^c)}{\pi(\boldsymbol{r}^{j-1})}\frac{\pi(\sigma^c)}{\pi(\sigma^{j-1})}\frac{\pi(\boldsymbol{z}|a^c,\boldsymbol{r}^c,\sigma^c)}{\pi(\boldsymbol{z}|a^{j-1},\boldsymbol{r}^{j-1},\sigma^{j-1})}\frac{q(a^{j-1}|a^c)}{q(a^c|a^{j-1})}\frac{q(\sigma^{j-1}|\sigma^c)}{q(\sigma^c|\sigma^{j-1})}\frac{q(\boldsymbol{r}^{j-1}|\mathbf{r}^c)}{q(\boldsymbol{r}^c|\boldsymbol{r}^{j-1})}$$
$$=\frac{\pi(a^c)}{\pi(a^{j-1})}\frac{\pi(\boldsymbol{r}^c)}{\pi(\boldsymbol{r}^{j-1})}\frac{\pi(\sigma^c)}{\pi(\sigma^{j-1})}\frac{\pi(\boldsymbol{z}|a^c,\boldsymbol{r}^c,\sigma^c)}{\pi(\boldsymbol{z}|a^{j-1},\boldsymbol{r}^{j-1},\sigma^{j-1})}\frac{a^c}{a^{j-1}}\frac{\sigma^c}{\sigma^{j-1}}\prod_{i=1}^{n_p}\frac{r_i^c}{r_i^{j-1}}.$$

   If the r_i's are assumed to be independent *a priori* then the above becomes

$$\frac{\pi(a^c)}{\pi(a^{j-1})}\frac{\pi(\sigma^c)}{\pi(\sigma^{j-1})}\frac{\pi(\boldsymbol{z}|a^c,\boldsymbol{r}^c,\sigma^c)}{\pi(\boldsymbol{z}|a^{j-1},\boldsymbol{r}^{j-1},\sigma^{j-1})}\frac{a^c}{a^{j-1}}\frac{\sigma^c}{\sigma^{j-1}}\prod_{i=1}^{n_p}\frac{r_i^c\pi(r_i^c)}{r_i^{j-1}\pi(r_i^{j-1})}.$$

   (c) Set $j = j + 1$.

of the covariance function. If, in addition, the dataset is noisy then the hyperparameters are $(a, \boldsymbol{r}, \sigma)'$. These parameters can be estimated from the residuals $\boldsymbol{z}$ as follows. As $\boldsymbol{y} \sim N(m(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\mathbf{I})$, the likelihood function is

$$\pi(\boldsymbol{z}|a, \mathbf{r}, \sigma) = (2\pi)^{-\frac{n}{2}}|K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\mathbf{I}|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\boldsymbol{z}'(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\mathbf{I})^{-1}\boldsymbol{z}\right\} \tag{9.9}$$

where the covariance function $K(\boldsymbol{X}, \boldsymbol{X})$ also depends on the hyperparameters. To proceed with a Bayesian analysis, after a prior distribution is specified for $(a, \boldsymbol{r}, \sigma)$, this information can be combined with that in the likelihood function using Bayes Theorem to obtain the posterior distribution $\pi(a, \boldsymbol{r}, \sigma|\boldsymbol{z})$. Typically this posterior distribution is intractable and an MCMC scheme is needed to obtain posterior samples of the hyperparameters; we use Algorithm 20. We suggest using a prior distribution with independent log-normal components as these distributions are appropriate for positive quantities and are quite flexible: $r_i \sim LN(c_i, 1/d_i)$ for $i = 1, \ldots, n_p$, $a \sim LN(c_0, 1/d_0)$ and $\sigma \sim LN(e, 1/f)$. Using this prior gives the logged Metropolis-Hastings ratio as a difference between terms of the

form

$$\log\left\{\pi(y|a,\boldsymbol{r},\sigma)a\pi(a)\sigma\pi(\sigma)\prod_{i=1}^{n}r_i\pi(r_i)\right\} = k - \frac{1}{2}\log|K(\boldsymbol{X},\boldsymbol{X})+\sigma^2\mathbf{I}|$$
$$-\frac{1}{2}\boldsymbol{z}'\left[K(\boldsymbol{X},\boldsymbol{X})+\sigma^2\mathbf{I}\right]^{-1}\boldsymbol{z} - \frac{d_0}{2}(\log a - c_0)^2 - \frac{f}{2}(\log\sigma - e)^2 - \frac{1}{2}\sum_{i=1}^{n}d_i(\log r_i - c_i)^2.$$

We can use the posterior samples for $(a,\boldsymbol{r},\sigma)$ to fit a Gaussian process by fixing these hyperparameters at their posterior mean. Ideally, we would use a process which averaged over the posterior uncertainty in the hyperparameters but this can lead to a very slow evaluation of the fitted process.

9.1.4 Choice of training data

Before fitting a Gaussian process to training data we need to consider how best to select the input values in order to give a good coverage of the input space, as having a good coverage will reduce uncertainty in Gaussian process predictions. Of course, an accurate fitted Gaussian process can be almost guaranteed by taking a very large number of training points. However, in general it is not sensible for the training set to be too large as each training point requires a function evaluation and, in fitting the Gaussian process, the computational load of working with a large matrix can be high. We also want to develop generic methods that are scalable to larger models which require high dimensional Gaussian process approximations. Chapman et al. (1994) suggest that $n_d = 10n_p$ training points, where n_p is the dimension of the input space, is the minimum needed. We use $n_d = 100n_p$ training points throughout this thesis.

Consider a Gaussian process with a two dimensional input. It seems reasonable to equate 'good coverage' with the training points having roughly uniform distributions in each input dimension. To obtain such training points we could simply simulate uniform realisations in each dimension. These points would also form a uniform scatter over the input space. However, this does not guarantee an even spread of the training data throughout the input space – the stochastic element in the production of the training points can lead to areas of the input space that are not well represented.

Another option is to use a Latin hypercube which produces uniform marginal coverage of the whole design space (Bastos, 2010).Latin hypercube sampling was introduced by Mckay et al. (1979). Using Latin hypercube sampling produces a lower asymptotic variance of the mean simulator output when compared to simple random sampling (Stein, 1987). In two dimensions, Latin hypercube sampling works by splitting the input space (a square) into n_d rows and n_d columns and points are placed so that each row and column contains only one point. Note that such a scheme would produce points that were roughly uniform in

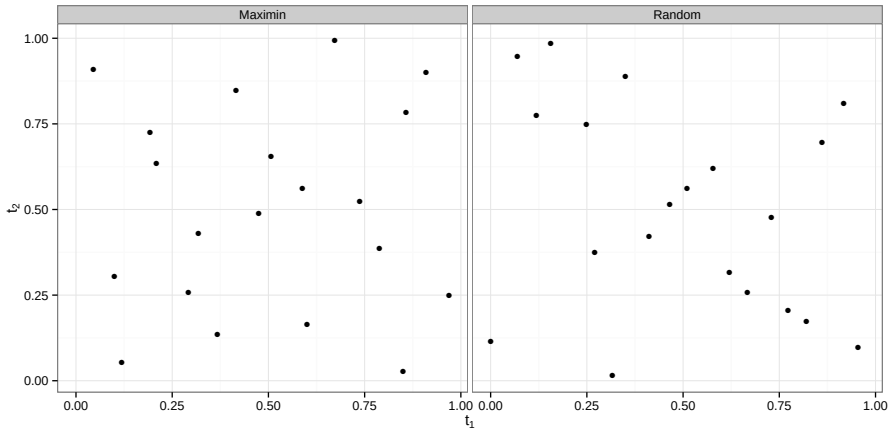


Figure 9.2: Comparison of maximin Latin hypercube sampling (left) and Latin hypercube sampling (right) with $n_d = 20$.

each input dimension. However, a drawback of this sampling method is that it also can lead to poor coverage of the input space; for example, having training data points along the diagonal would be a possibility.

To avoid such poor coverage in training data, Morris and Mitchell (1995) introduced the maximin Latin hypercube design. The maximin approach works by choosing the Latin hypercube design which maximises the minimum Euclidean distance between all of the points and so provides better coverage than Latin hypercube sampling. Figure 9.2 shows a 20 point maximin Latin hypercube design and an inferior randomly generated Latin hypercube design. It is clear that the maximin Latin hypercube design provides a better coverage of the input space.

In this work, we seek a Gaussian process which approximates the expected utility over both the model parameters $\boldsymbol{\theta}$ and the design $\boldsymbol{d} = (t_1, \ldots, t_{n_d})$. As the timepoints are ordered, this introduces an additional complication into the choice of training points. For example, in a two timepoint design, we must have $t_1 < t_2$. Figure 9.3 shows both suitable and unsuitable points from a maximin Latin hypercube design. There are many ways to circumvent this problem, the most simple of which is just to ignore the time ordering, evaluate the expected utility at the ordered version of unordered sequence of timepoints and then fit the Gaussian process to this training set. However, using such a training set is very wasteful as the Gaussian process will never need to be evaluated at an unordered sequence. Further, distances in this space cannot be adequately described using a standard metric such as the squared exponential; for example, in Figure 9.3, $u(1, 0) = u(0, 1)$ and $u(0.49, 0.51) = u(0.51, 0.49)$. Instead we consider three possible methods for creating the training data within the temporally ordered input space.

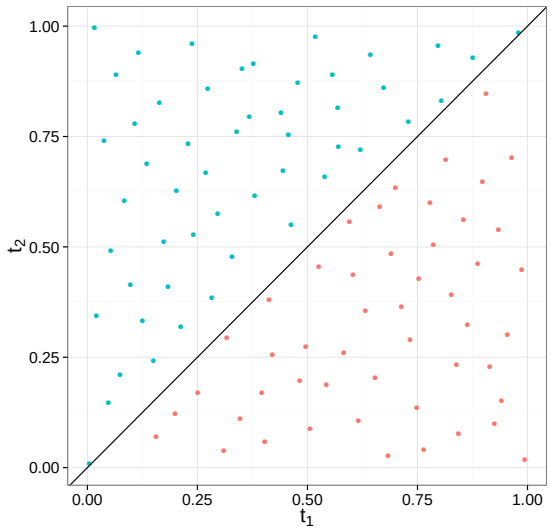


Figure 9.3: Plot showing unsuitable points (red) in a maximim Latin hypercube with the constraint $t_1 < t_2$.

The fold method

The fold method works by creating a Latin hypercube over the full input space and then ordering the timepoints. Essentially the points that are not temporally ordered are folded back onto that part of the space that is ordered. For example, in the case of a two timepoint design over time interval $(0, T)$, we simulate a Latin hypercube over $0 < t_1 < T$ and $0 < t_2 < T$ and any points in the region $t_1 > t_2$ are reflected into the correct region.

The uniform method

This is a rejection method which uniformly samples timepoints and then rejects any points that are not in the correct region. For a two timepoint design, we simulate $t_1 \sim U(0, T)$ and $t_2 \sim U(0, T)$ and reject points with $t_1 > t_2$. The algorithm runs until the correct number of training points have been produced.

The cut method

The third approach is essentially the same as the uniform method but without the repeated simulation of points until the correct number have been determined. Instead a larger number of points is generated from a Latin hypercube design and then any timepoints that are not ordered are simply cut. Any design which does not have the correct number of training points is rejected. This last step makes this algorithm quite slow as it is quite unlikely that a cut design will produce the required number of training points.

Method comparison

It is not clear which of these methods will give the best coverage of the input space as judged by the maximum minimum distance. Therefore for each method we determined the maximum minimum distance for 1M designs with $n_d = 100$. This was repeated for $2, 3$ and 4 timepoint designs; see Table 9.1. The cut method performs best for lower dimensional designs, although the coverage does not look appreciably better than those for the other methods; see Figures 9.4, 9.5 and 9.6. We note that, because of its rejection of any designs not having the correct number of points, the cut method required ten times as many designs to be simulated and therefore took ten times as long to generate. Another problem with the cut method is that when we require a good emulator design for the expected utility $u(\boldsymbol{d}, \boldsymbol{\theta})$, and hence need a design over both $(\boldsymbol{d}, \boldsymbol{\theta})$, cutting points which are temporally unordered can lead to a poor coverage of points within the $\boldsymbol{\theta}$ subspace.

The uniform method is also time consuming as a result of rejecting points that do not have the correct ordering. For example, in a two timepoint design, roughly half of the simulated designs will not satisfy the ordering constraint. Table 9.1 shows that the uniform method has the smaller maximum minimum distances, as we no longer have the space filling properties of the Latin hypercube. However, this effect becomes smaller as the number of timepoints in the design increases.

The fold method produces maximum minimum distances which are quite close to those of the other two methods. However, the greatest advantage of using this method is that it is relatively quick (as no designs are rejected) and there are no problems in the $\boldsymbol{\theta}$ subspace when using this method to cover a temporally ordered $(\boldsymbol{d}, \boldsymbol{\theta})$ input space. Therefore we will use the fold method to construct the designs of training points for the Gaussian process fits in this thesis.

To verify that the uniform and cut methods are not mathematically equivalent we found the maximum minimum distance over $100k$ designs and repeated this $2k$ times. The mean maximum minimum distance for the cut method and the uniform method were 0.5314 and 0.5681 with standard errors 0.0003 and 0.0004 respectively. The mean absolute difference between the two methods is 0.0367 with standard error 0.005. If we assume no difference between the two methods then only 3.6% would have differences as large as those we see in this sample.

9.1.5 Diagnostics

When using a fitted Gaussian process as an approximation to a function it is important to check that the approximation is accurate and that the underlying normality assumptions (that any finite collection of function evaluations have a multivariate normal distribution) of the process are plausible. One simple way of diagnosing deviations from the multivariate

Method	Two timepoints	Three timepoints	Four timepoints
Fold	0.024	0.052	0.080
Uniform	0.021	0.049	0.074
Cut	0.025	0.054	0.073

Table 9.1: Maximum minimum distances for the fold, uniform and cut methods for selecting training data for ordered timepoints.

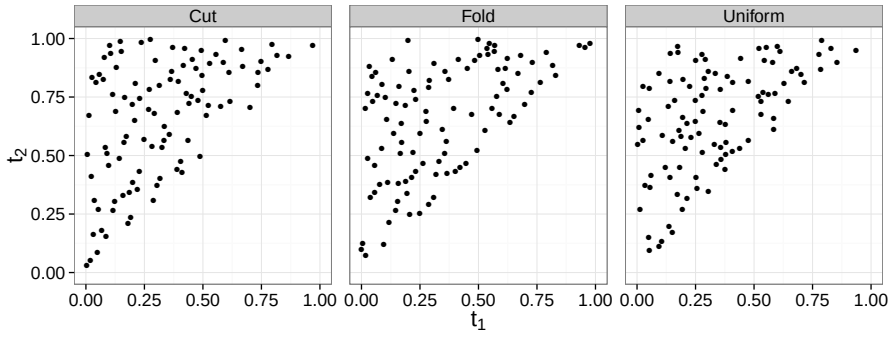


Figure 9.4: Comparison of methods to obtain ordered training data with $n_d = 100$ for a two timepoint design.

normality of the function values at any collection of inputs is to see how plausible a univariate normality assumption is for individual function evaluations. Such evaluations need to be made at inputs other than those used in the training data and comparisons are made between the predicted values from the Gaussian process and the actual value of the function. For example, standardised differences between the fitted and observed values should follow (roughly) a standard normal distribution and so (roughly) 5% of standardised differences should be greater or less than two. We note that problems with normality can sometimes be tackled using a transformation of the training data (Bastos and O'Hagan, 2009), that is, the Gaussian process is fitted to some function of the output function f.

Other problems that can occur in using a Gaussian process concerns the adequacy of the kernel and the representativeness of the training data. Here, for example, a stationary kernel might be used when a non-stationary kernel is needed. Also if the training data are not representative of the input space then hyperparameter estimates may not be very accurate, even if stationarity can be assumed (Bastos and O'Hagan, 2009). If the parameters σ^2 or a are underestimated then credible intervals for predictions from the Gaussian process are too small, whereas if σ^2 or a is overestimated, the intervals are too large. In addition, if $\boldsymbol{r}$ is misspecified then the relationships between outputs which have inputs the same distance apart will be inappropriate and this affects the accuracy of the credible intervals near training data. If an incorrect mean function is assumed, predictions

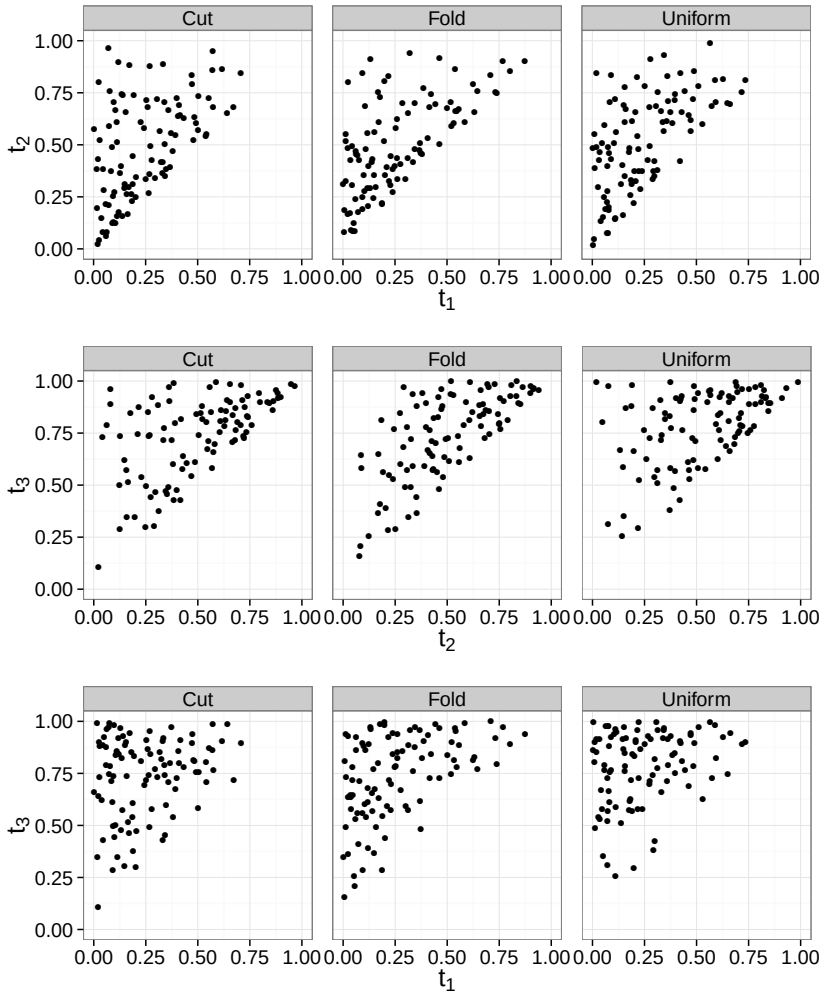


Figure 9.5: Comparison of methods to obtain ordered training data with $n_d = 100$ for a three timepoint design.

from the Gaussian process may be generally too high or too low (Bastos and O'Hagan, 2009).

Generally diagnostics used to validate the use of a Gaussian process involve using a new set of training data and looking at the predictions of the Gaussian process in comparison to these data. Other possible diagnostics include those based on a jackknife comparison in which each individual training point is compared with its predicted value from a Gaussian process fitted to the training data after excluding the individual training point (Rougier et al., 2009). Methods also exist which leave out more than one data point (Kennedy and O'Hagan, 2001).

In this work, we calculate the $u(\boldsymbol{d}, \boldsymbol{\theta})$ at a new training dataset which we call the validation data. Diagnostics are then produced which compare the validation data with the predictions made by the fitted Gaussian process.

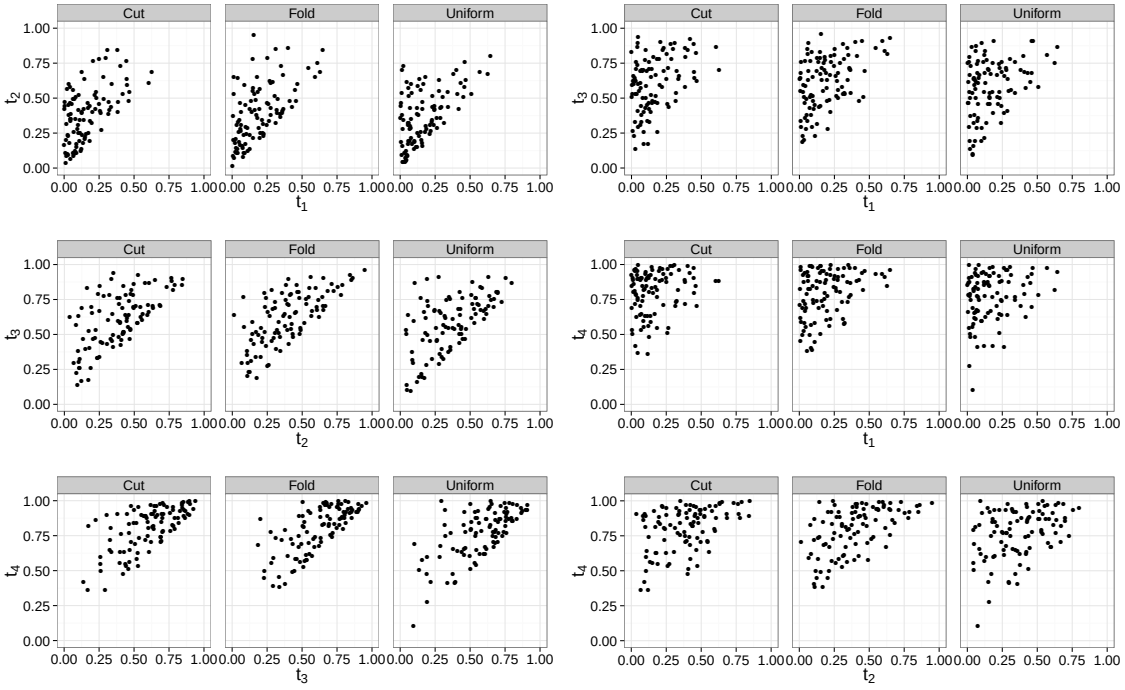


Figure 9.6: Comparison of methods to obtain ordered training data with $n_d = 100$ for a four timepoint design.

In general terms, the validation data will consist of a new set of inputs

$$\boldsymbol{X}^{\diamondsuit} = (\boldsymbol{x}_1^{\diamondsuit}, \ldots, \boldsymbol{x}_{n_d^{\diamondsuit}}^{\diamondsuit})$$

and their outputs

$$\boldsymbol{y}_i^{\diamondsuit} = (f(\boldsymbol{x}_1^{\diamondsuit}), \ldots, f(\boldsymbol{x}_{n_d^{\diamondsuit}}^{\diamondsuit})).$$

9.1.6 Individual prediction errors

The individual prediction error (IPE) for the validation data is given by

$$IPE(\boldsymbol{x}_i^{\diamondsuit}) = f(\boldsymbol{x}_i^{\diamondsuit}) - m^*(\boldsymbol{x}_i^{\diamondsuit}),$$

that is, the difference between the validation output data for the validation inputs and the Gaussian process predictive mean for the validation inputs. The IPE can be standardised to account for its uncertainty, giving the standardised prediction error (SPE) as

$$SPE(\boldsymbol{x}_i^{\diamondsuit}) = \frac{f(\boldsymbol{x}_i^{\diamondsuit}) - m^*(\boldsymbol{x}_i^{\diamondsuit})}{\sqrt{K(\boldsymbol{x}_i^{\diamondsuit}, \boldsymbol{x}_i^{\diamondsuit})}} \quad \text{for} \quad i = 1, \ldots, n_d^{\diamondsuit}.$$

If the fitted Gaussian process is a good representation of the utility function then the SPEs should have a standard normal distribution. This means that if more than 5% of the SPEs are outside the interval $[-2, 2]$ then this suggests that the Gaussian process is not a good fit. An example of how the SPEs should look when the assumptions are valid is given in Figure 9.7 (a). If there are a small number of outliers then these can either be ignored or investigated further by looking at new validation points close to the inputs of the outliers (Bastos and O'Hagan, 2009).

If many SPEs are outside the interval $[-2, 2]$, it could suggest a systematic problem. For example, if these errors are of the same sign, it may be that the mean function is not removing enough variability in the output, or $\boldsymbol{\beta}$ is not estimated well, or that stationarity should not be assumed. If large errors occur for validation points close to training data points then it could be that some or all of correlation length parameters r_i are too large and the Gaussian process predictions are affected too much by close by training data.

Another possible reason for too many errors outside of the range $[-2, 2]$ is that the σ^2 estimate is not accurate. This may occur when large errors have no systematic pattern. The opposite of the above would follow for SPEs that are too small (Bastos and O'Hagan, 2009).

9.1.7 Mahalanobis distance

The IPEs do not take into account the correlations between the outputs. One diagnostic which does account for this correlation (and variance) is the Mahalanobis distance

$$MD(\boldsymbol{X}^{\Diamond}) = \{f(\boldsymbol{X}^{\Diamond}) - m^*(\boldsymbol{X}^{\Diamond})\}' K^*(\boldsymbol{X}^{\Diamond}, \boldsymbol{X}^{\Diamond})^{-1} \{f(\boldsymbol{X}^{\Diamond}) - m^*(\boldsymbol{X}^{\Diamond})\}.$$

The MD has a χ^2 distribution with $n_d^{\Diamond}$ degrees of freedom, conditional on the training data and the hyperparameters, as

$$f^*(\boldsymbol{X}^{\Diamond}) \sim N_{n_d^*}(m^*(\boldsymbol{X}^{\Diamond}), K^*(\boldsymbol{X}^{\Diamond}, \boldsymbol{X}^{\Diamond})).$$

Particularly large or small values of the MD may indicate that the Gaussian process is not a good fit and this should be investigated.

9.1.8 Probability integral transform

Another diagnostic used to check the Gaussian process assumptions is the probability integral transform (PIT), described in Gneiting et al. (2007). This diagnostic focusses attention on the marginal standard normality of the SPE values and assesses the distributional fit by using

$$PIT(\boldsymbol{x}_i^{\Diamond}) = \Phi(SPE(\boldsymbol{x}_i^{\Diamond})) \quad \text{for} \quad i = 1, \ldots, n_d^{\Diamond}.$$

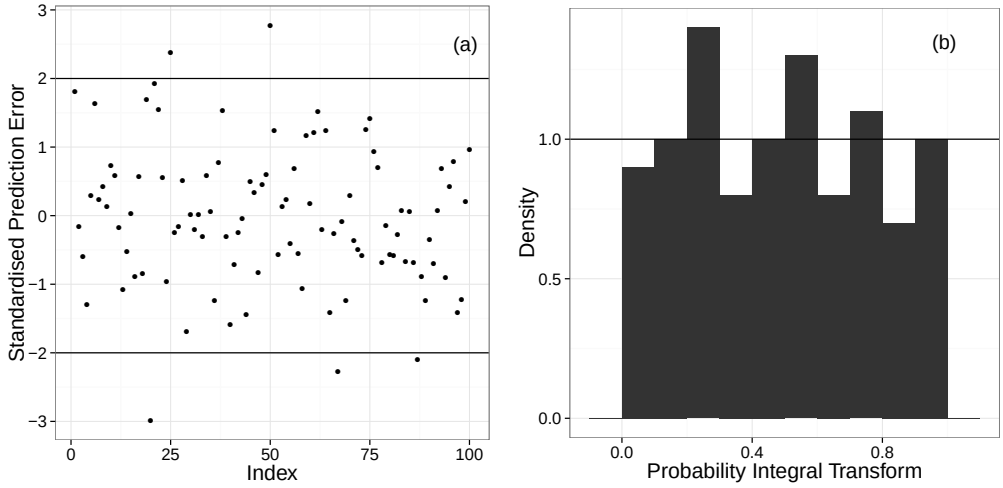


Figure 9.7: Example diagnostics using 100 training data points. Plot (a) shows the standardised prediction errors and plot (b) is the probability integral transform.

If the assumptions about a fitted Gaussian process are satisfied then the $PIT(\boldsymbol{x}_i^{\Diamond})$ values should follow a standard uniform distribution. This is easily checked using a histogram of the PIT values. If the assumptions are correct then this histogram should be uniform. An example of how the histogram should look when the assumptions are valid is given in Figure 9.7(b).

9.2 Experimental design

In the following sections we will describe how Gaussian processes can be used in experimental design but first we will consider how the expected utility can be calculated almost exactly when the likelihood is tractable and how a delta approximation can approximate the expected utility. We then compare the optimal designs found using all three methods to those found using the Gaussian process approach.

9.3 The exact method

The expected utility $u(\boldsymbol{d})$ can be calculated using numerical integration. In particular, it is straightforward to calculate for the death model, as the likelihood can be expressed analytically, and then used in the integral in Equation (9.1). Numerical integration is performed using the GNU Scientific Library (Galassi, 2013) in C using an adaptive Gaussian quadrature method (Laurie, 1997). We will refer to this method of calculation as the exact method as this can be done with good accuracy and will give us a standard against which to judge other approximate solutions.

9.4 The delta approximation method

A delta approximation can be used to approximate the expected utility using

$$u(\boldsymbol{d}) = E_{\boldsymbol{\theta},\boldsymbol{y}}\left[u(\boldsymbol{d},\boldsymbol{y})\right] \simeq u(\boldsymbol{d},\boldsymbol{y}^*_{\boldsymbol{d}})$$

where

$$\boldsymbol{y}^*_{\boldsymbol{d}} = E\left[\boldsymbol{y}|\boldsymbol{d}, E(\boldsymbol{\theta})\right]$$

is the expected experimental realisation if all prior mass is located at the prior mean. This expected realisation can be determined analytically for simple models. However for larger models we might need to use the linear noise approximation mean or even the solution of the deterministic (ODE system) model equivalent to the stochastic model.

The delta method ignores variation in both $\boldsymbol{\theta}$ and $\boldsymbol{y}$ and so might only provide a crude approximation to $u(\boldsymbol{d})$. However, a delta method based optimal design should provide a good initial guess to the actual optimal design. In particular, it should give insight into the appropriate part of design space containing the actual optimal design.

9.5 The Gaussian process method

We now describe how a Gaussian process approximation to $u(\boldsymbol{d},\boldsymbol{\theta})$ can be used to determine the optimal design $\boldsymbol{d}^*$. The first task is to decide on an appropriate training set for $(\boldsymbol{d},\boldsymbol{\theta})$ to use to fit the Gaussian process. It makes sense to focus this training set in an area of design space that contains the optimal design; we do this using delta approximation method. We restrict the input space to this reduced design space and take $\boldsymbol{\theta}$ values with 99% central coverage with respect to its prior distribution. The training points are then determined via the fold method described in Section 9.1.4. The utility $u(\boldsymbol{d},\boldsymbol{\theta})$ is then calculated at each training point either by an 'exact' evaluation using numerical integration or by using the LNA approximation to the likelihood function.

The evaluation of $u(\boldsymbol{d},\boldsymbol{\theta})$ at each training point proceeds a follows. First a realisation $\boldsymbol{y}$ is simulated for the particular $\boldsymbol{\theta}$ and design $\boldsymbol{d}$ using the Gillespie algorithm as given in Algorithm 14. Then $u(\boldsymbol{d},\boldsymbol{y})$ is approximated by using the LNA approximation to the likelihood $\pi_{LNA}(\boldsymbol{y}|\boldsymbol{d},\boldsymbol{\theta})$ within an MCMC scheme described in Algorithm 21. Details of the LNA method are given in Chapter 7. Finally $u(\boldsymbol{d},\boldsymbol{\theta})$ is approximated using m_1 repeats of the above, giving

$$u(\boldsymbol{d},\boldsymbol{\theta}) \simeq \frac{1}{m_1}\sum_{i=1}^{m_1} u(\boldsymbol{d},\boldsymbol{y}_i). \tag{9.10}$$

This approximation becomes more accurate as m_1 increases.

A Gaussian process is then fitted to the training data using an appropriate mean

function that represents the training data well. We choose to use a fitted regression function as the mean function as described in Section 9.1.1. The resulting (fitted) Gaussian process is then used to approximate the $u(\boldsymbol{d}, \boldsymbol{\theta})$ for other data inputs. Note that a nugget term σ^2 is required to account for the approximation of $u(\boldsymbol{d}, \boldsymbol{\theta})$.

For low dimensional designs (d small) and simple models, the optimal design can then be calculated by approximating the integral in Equation (9.2) to estimate $u(\boldsymbol{d})$ by averaging $u(\boldsymbol{d}, \boldsymbol{\theta})$ over the prior, using

$$u(\boldsymbol{d}) \simeq \frac{1}{m_2} \sum_{i=1}^{m_2} u(\boldsymbol{d}, \boldsymbol{\theta}_i), \tag{9.11}$$

where the $\boldsymbol{\theta}_i$ are a random sample of size m_2 from the prior $\pi(\boldsymbol{\theta})$. We then choose the design which maximises $u(\boldsymbol{d})$.

For higher dimensional designs, the above method does not give an efficient way of determining the optimal design. Therefore we adopt the strategy of $\boldsymbol{d}$ (Müller, 1999) and use an MCMC scheme to obtain the marginal distribution of $\boldsymbol{d}$ by targeting the joint distribution

$$g(\boldsymbol{d}, \boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta}) u(\boldsymbol{d}, \boldsymbol{\theta}).$$

This MCMC scheme is given in Algorithm 22. Note that we use an independent proposal for $\boldsymbol{\theta}$ which is the prior distribution.

For easier identification of the mode, we also follow Müller (1999) and sample from the joint distribution

$$g(\boldsymbol{d}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) \propto \prod_{j=1}^{J} \pi(\boldsymbol{\theta}_j) u(\boldsymbol{d}, \boldsymbol{\theta}_j)$$

as this amplifies the signal for the location of the mode. This changes the MCMC algorithm scheme to that in Algorithm 23. The optimal design $\boldsymbol{d}^*$ is then estimated by the multivariate mode of $g(\boldsymbol{d}, \boldsymbol{\theta})$ using marginal posterior samples of $\boldsymbol{d}$ as described in Algorithm 19. As Algorithm 23 is not computationally expensive, it is feasible to use high values of J, in which case, the trimmed mean of the $\boldsymbol{d}$ iterates is a good estimate of the multivariate mode (Müller, 1999). A step by step description of the procedure to find the optimal design using Gaussian processes is given in Algorithm 24.

9.6 Application of the Gaussian process method to the death model

The death model has a single species Y and reaction $Y \xrightarrow{\theta} \emptyset$, where θ is the rate parameter for the reaction. Consider the d timepoint design $\boldsymbol{d} = (t_1, \ldots, t_d)$. Suppose that y_k is the observation at time t_k. We will assume that the value of the process at $t_0 = 0$ is known to

Algorithm 21 MCMC scheme to approximate $u(\boldsymbol{d}, \boldsymbol{\theta})$ using the LNA.

For a particular $\boldsymbol{d}^\dagger, \boldsymbol{\theta}^\dagger$ and for $i = 1, \ldots, m_1$:

1. Simulate $\boldsymbol{y}_i \sim \pi(\boldsymbol{y}|\boldsymbol{d}^\dagger, \boldsymbol{\theta}^\dagger)$.
2. Initialise $\boldsymbol{\theta}^0 \sim \pi(\boldsymbol{\theta})$.
3. For $j = 1, \ldots, n$:
   (a) Propose $\boldsymbol{\theta}^c \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}^{j-1})$ where q is a symmetric random walk on the log scale.
   (b) Calculate the MH ratio $\alpha = \min(1, A)$ where
   $$A = \frac{\pi(\boldsymbol{\theta}^c)\hat{\pi}_{LNA}(\boldsymbol{y}_i|\boldsymbol{\theta}^c, \boldsymbol{d}^\dagger)q(\boldsymbol{\theta}^{j-1}|\boldsymbol{\theta}^c)}{\pi(\boldsymbol{\theta}^{j-1})\hat{\pi}_{LNA}(\boldsymbol{y}_i|\boldsymbol{\theta}^{j-1}, \boldsymbol{d}^\dagger)q(\boldsymbol{\theta}^c|\boldsymbol{\theta}^{j-1})}$$
   and accept proposed values with probability α.
   (c) Set $j = j + 1$ and return to 3(a).
4. Calculate
   $$u(\boldsymbol{d}^\dagger, \boldsymbol{y}_i) \simeq \frac{1}{\det\{\widehat{\mathrm{Var}}_{LNA}(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{d}^\dagger)\}}.$$
5. Set $i = i + 1$ and return to step 1.
6. Calculate
   $$u(\boldsymbol{d}^\dagger, \boldsymbol{\theta}^\dagger) \simeq \frac{1}{m_1}\sum_{i=1}^{m_1} u(\boldsymbol{d}^\dagger, \boldsymbol{y}_i).$$

be $y_0 = 50$. For the death model, we can solve the CME to obtain the likelihood to obtain

$$\pi(\boldsymbol{y}|\boldsymbol{d}, \boldsymbol{\theta}) = \prod_{k=1}^{d} \binom{y_{k-1}}{y_k} \exp\{-\theta y_k(t_k - t_{k-1})\}\left(1 - \exp\{-\theta(t_k - t_{k-1})\}\right)^{y_{k-1}-y_k}.$$

Values for the training data for the Gaussian process can be found almost exactly (within 2×10^{-2}) using this likelihood function via numerical integration. Since such calculations are almost exact, the nugget term σ^2 will be very small. Unfortunately this can cause numerical instabilities when inverting the covariance matrix. However such problems can be avoided using a Cholesky decomposition and solving a linear equation instead of inverting the matrix.

9.6.1 Delta approximation to the death model

The delta approximation evaluates the utility function at the mean experimental outcome

$$\boldsymbol{y}_{\boldsymbol{d}}^* = E\left[\boldsymbol{y}|\boldsymbol{d}, E(\boldsymbol{\theta})\right].$$

Algorithm 22 MCMC scheme to find the optimal design using the Gaussian process approximation to $u(\boldsymbol{d}, \boldsymbol{\theta})$.

1. Set $i = 1$. Initialise $\boldsymbol{d}^0$ and $\boldsymbol{\theta}^0 \sim \pi(\boldsymbol{\theta})$.
2. Estimate $u(\boldsymbol{d}^0, \boldsymbol{\theta}^0)$ from the Gaussian process.
3. Propose a new $\boldsymbol{d}^c \sim q(\boldsymbol{d}|\boldsymbol{d}^{i-1})$ and a new $\boldsymbol{\theta}^c \sim \pi(\boldsymbol{\theta})$.
4. Estimate $u^c = u(\boldsymbol{d}^c, \boldsymbol{\theta}^c)$ from the Gaussian process.
5. Calculate the MH ratio $\alpha = \min(1, A)$ where
$$A = \frac{u(\boldsymbol{d}^c, \boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c)q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)\pi(\boldsymbol{\theta}^{i-1})}{u(\boldsymbol{d}^{i-1}, \boldsymbol{\theta}^{i-1})\pi(\boldsymbol{\theta}^{i-1})q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})\pi(\boldsymbol{\theta}^c)} = \frac{u(\boldsymbol{d}^c, \boldsymbol{\theta}^c)q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)}{u(\boldsymbol{d}^{i-1}, \boldsymbol{\theta}^{i-1})q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})}.$$
and accept proposed values with probability α.
6. Set $i = i + 1$ and return to step 3.

Algorithm 23 MCMC scheme to find the optimal design using the Gaussian process approximation to $u(\boldsymbol{d}, \boldsymbol{\theta})$ amplifying the mode.

1. Set $i = 1$. Initialise $\boldsymbol{d}^0$ and $\boldsymbol{\theta}_j^0 \sim \pi(\boldsymbol{\theta})$ for $j = 1, \ldots, J$.
2. Estimate $u(\boldsymbol{d}^0, \boldsymbol{\theta}_j^0)$ from the Gaussian process for $j = 1, \ldots, J$.
3. Calculate $u_0 = \prod_{j=1}^{J} u(\boldsymbol{d}^0, \boldsymbol{\theta}_j^0)$.
4. Propose a new $\boldsymbol{d}^c \sim q(\boldsymbol{d}|\boldsymbol{d}^{i-1})$ and a new $\boldsymbol{\theta}_j^c \sim \pi(\boldsymbol{\theta})$ for $j = 1, \ldots, J$.
5. Estimate $u(\boldsymbol{d}^c, \boldsymbol{\theta}_j^c)$ from the Gaussian process for $j = 1, \ldots, J$.
6. Calculate $u^c = \prod_{j=1}^{J} u(\boldsymbol{d}^c, \boldsymbol{\theta}_j^c)$.
7. Calculate the MH acceptance probability $\alpha = \min(1, A)$, where
$$A = \frac{u^c\pi(\boldsymbol{\theta}^c)q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)\pi(\boldsymbol{\theta}^{i-1})}{u^{i-1}\pi(\boldsymbol{\theta}^{i-1})q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})\pi(\boldsymbol{\theta}^c)} = \frac{u^c q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)}{u^{i-1}q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})}, \quad \text{where} \quad \pi(\boldsymbol{\theta}) = \prod_{j=1}^{J}\pi(\boldsymbol{\theta}_j),$$
and accept proposed values with probability α.
8. Set $i = i + 1$ and return to step 4.

In the case of the death model, $\boldsymbol{y}_{\boldsymbol{d}}^* = (y_1^*, \ldots, y_d^*)'$ and has components

$$y_t^* = E[y_t|\boldsymbol{d}, E(\theta)] = y_0 \exp(-E(\theta)t). \tag{9.12}$$

Algorithm 24 Algorithm for experimental design using Gaussian processes.

1. Sample n_d training data inputs using a folded Latin hypercube over $\boldsymbol{d} \times \boldsymbol{\theta}$.
2. Estimate $u(\boldsymbol{d}, \boldsymbol{\theta})$ for each of the n_d training data inputs.
3. Fit a Gaussian process to the training data using Algorithm 20.
4. Fix hyperparameters at their posterior mean.
5. (a) For small models and small dimensional designs, determine the optimal design $\boldsymbol{d}^*$ by approximating $u(\boldsymbol{d})$ for all possible $\boldsymbol{d}$, where $u(\boldsymbol{d})$ is calculated by averaging over realisations of the fitted Gaussian process for $u(\boldsymbol{d}, \boldsymbol{\theta})$ evaluated at a random sample from the prior for $\boldsymbol{\theta}$:

$$u(\boldsymbol{d}) \simeq \frac{1}{m_2} \sum_{i=1}^{m_2} u(\boldsymbol{d}, \boldsymbol{\theta}_i).$$

   (b) If it is not possible to evaluate $u(\boldsymbol{d}, \boldsymbol{\theta})$ for all possible $\boldsymbol{d}$ within a reasonable computational time, use the MCMC scheme described in Algorithm 23 to obtain marginal posterior distributions for $\boldsymbol{d}$ and then use Algorithm 19, the marginal modes or the trimmed sample mean of the marginal distributions for $\boldsymbol{d}$ to estimate the optimal design $\boldsymbol{d}^*$.

Under the delta approximation, the expected utility is calculated as

$$u(\boldsymbol{d}, \boldsymbol{y}_{\boldsymbol{d}}^*) = \frac{1}{\mathrm{Var}(\theta|\boldsymbol{d}, \boldsymbol{y}_{\boldsymbol{d}}^*)}.$$

9.6.2 Optimal single timepoint design

We begin by selecting $n_d = 100n_p = 200$ training data points $(t_1, \boldsymbol{\theta})_i'$ for the single timepoint design using the fold method. We aim to match the analysis of Drovandi and Pettitt (2013) described in Section 7.3. Therefore we use their prior $\theta \sim LN(-0.005, 0.01)$, with 99% central prior interval $0.7681 < \theta < 1.2873$, and restrict the time range to $(0, T = 10)$. At each training point, the utility $u(t_1, \theta)$ is calculated using numerical integration. The Gaussian process prior mean function we use takes the form (9.7) and so we need to investigate an appropriate set of functions on which to regress the utility values. Figure 9.8 shows that $u(t_1, \theta)$ is decreasing in θ for fixed t_1 and is first increasing and then decreasing in t_1 for fixed θ. This suggests fitting a linear function with inverse powers of θ and powers of t_1, and their interactions. Keeping only those terms with significant coefficients leads to the mean function

$$m(t_1, \theta) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 \frac{1}{\theta} + \hat{\beta}_3 t_1^2 + \hat{\beta}_4 \frac{1}{\theta^2} + \hat{\beta}_5 t_1^3 + \hat{\beta}_6 \frac{1}{\theta^3} + \hat{\beta}_7 \frac{t_1}{\theta}.$$

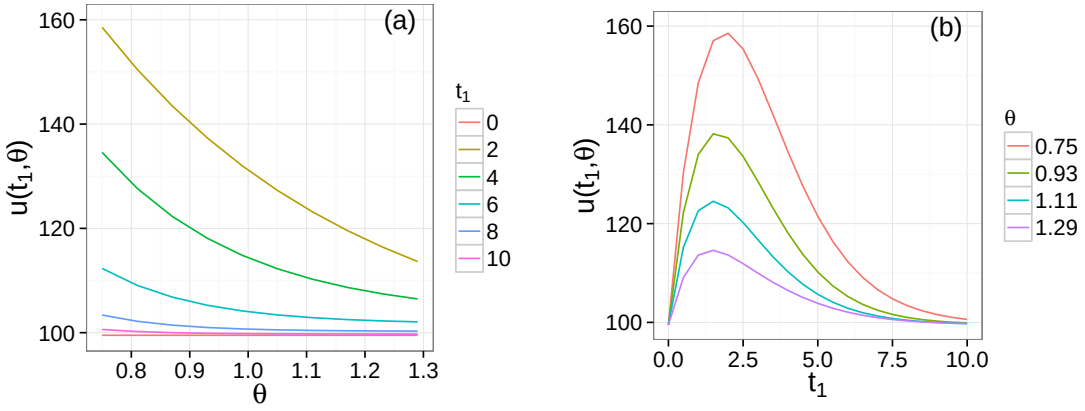


Figure 9.8: Graphs showing exact $u(t_1, \theta)$ plotted over (a) time and (b) θ.

This function fits the training data very well, with $R^2 = 0.8619$. The squared exponential covariance function (9.8) has four hyperparameters $(a, r_1, r_2, \sigma)'$. The hyperparameter r_1 represents the influence of θ on the output and r_2 represents the influence of t_1 on the output. Including the nugget term σ^2 means that the fitted mean function will not necessarily go through the points.

Hyperparameter estimation for the single timepoint design

In this and the next section, we fit Gaussian processes to the utility functions of designs of various sizes d. These Gaussian processes have input space $(t_1, \ldots, t_d, \theta)$ with dimension $n_p = d + 1$. Throughout we take the prior distribution for the Gaussian process hyperparameters $(a, \boldsymbol{r}, \sigma)$ to have independent log–normal components, with

$$r_i \sim LN(0, 1), \quad i = 1, \ldots, d + 1, \qquad a \sim LN(0, 1) \qquad \text{and} \qquad \sigma \sim LN(0, 0.5).$$

Note that these component priors are not particularly vague and, as we expect σ to be very small due to the utilities being estimated almost exactly, σ has a particularly concentrated prior near zero. The algorithm to estimate the hyperparameters is described in Section 9.1.3.

We first consider the case when we have a single timepoint design, that is, we need a Gaussian process approximation to $u(t_1, \theta)$. The marginal posterior distributions for the hyperparameters in this case are given in Figure 9.9. These distributions are strongly unimodal and we estimate the hyperparameters by their posterior means. Later in this section we investigate the sensitivity of the optimal design to using this posterior mean

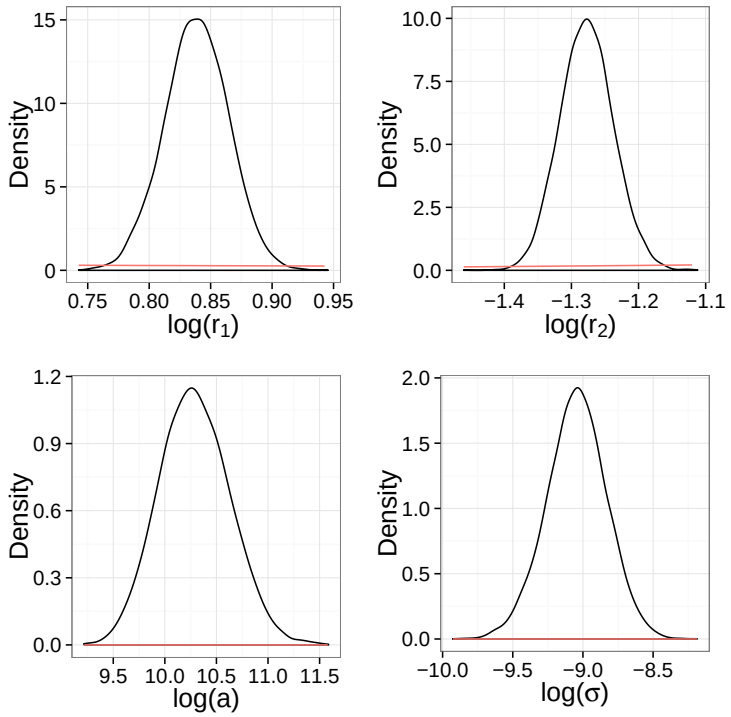


Figure 9.9: Marginal posterior distributions for the hyperparameters for a single timepoint design for the death model. Prior distributions are given in red.

rather than taking full account of its posterior uncertainty.

Diagnostics

The diagnostics for the single timepoint design are given in Figure 9.10. Looking at the standardised prediction errors plotted over θ, we can see that the largest errors occur near zero. This is due to $u(t_1, \theta)$ being constant over θ at time zero as we know the start value of the death process. This means $u(t_1, \theta)$ is one over the prior variance for θ. Therefore, using a constant value for σ is not suitable for timepoints near zero. As we know the value of the process at time zero, it is unlikely that the optimal design will be near zero. Therefore the expected utilities calculated for $t_1 < 1$ should be ignored. The second row of graphs gives the individual prediction errors. These show that the Gaussian process predictions are very accurate as the largest individual prediction error is 0.1, which is near zero. The probability integral transform looks roughly uniform. The MD for this Gaussian process is 331.63 which is large as $\chi^2_{100} = 135.81$ at the 99% point suggesting that the Gaussian process is not a good fit.

We have re–produced the diagnostics, but ignoring any validation data less than one, to see if the diagnostics improve, in Figure 9.11. We can see that the standardised prediction errors now have only a couple of the points outside the bounds $[-2, 2]$. Looking at the

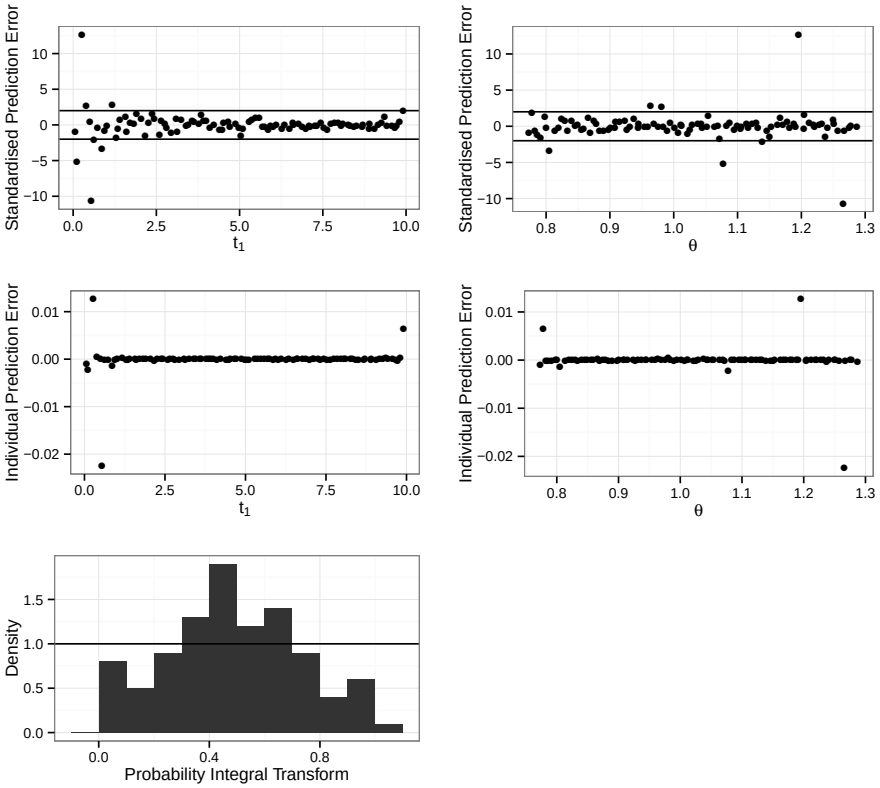


Figure 9.10: Diagnostics for the single timepoint design Gaussian process

individual prediction errors, the predictions of the Gaussian process are very accurate as the posterior mean function is not out by more than 0.007. The probability integral transform looks roughly uniform. The MD is 72.05 which is less than the critical value (χ^2_{90}=124.12 at the 99% point). This means that the Gaussian process is suitable for values greater than one. This problem should only be a feature of the single timepoint design as we intend to use the delta approximation to reduce the design space for more complicated designs and models.

Optimal design

A Gaussian process approximation has been produced for the death model. In Figure 9.12 we give a plot comparing three different methods of calculating the expected utility; the exact method, the delta method, and the Gaussian process approximation. For the death model the likelihood is known so the expected utility, $u(\boldsymbol{d})$, can be calculated exactly with only very small errors due to numerical integration. In this graph it can be seen that the three methods produce very similar results. It is not important whether the expected utility values are the same, the key is that the maximum is in the same position.

The exact method gives an optimal design of $\boldsymbol{d}^* = 1.60$, the delta approximation gives

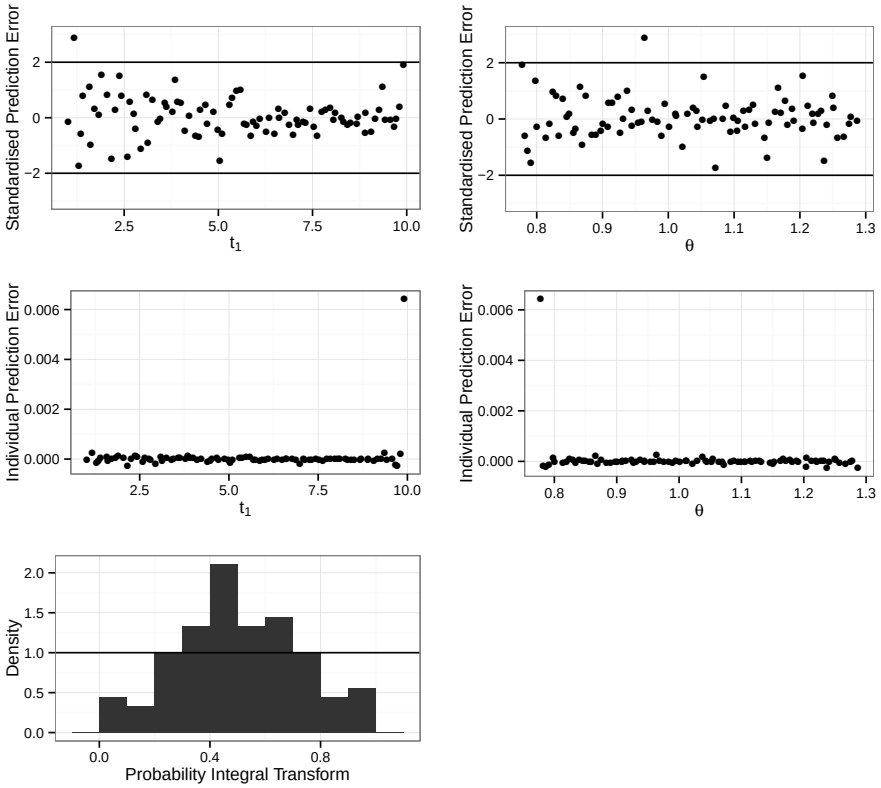


Figure 9.11: Diagnostics for the single timepoint design Gaussian process with $t_1 > 1$.

$\boldsymbol{d}^* = 1.58$ and the Gaussian process approximation gives $\boldsymbol{d}^* = 1.58$. Therefore the Gaussian process and delta approximations provide good approximations to the optimal design. This also matches the optimal design found in Drovandi and Pettitt (2013). A comparison of the optimal designs from the different methods is given in Table 9.2. We can see that both the delta and Gaussian process approximations only have a sub–optimality of 0.003%. In Table 9.2, ABC1 and ABC2 are repeats using the methods of Drovandi and Pettitt (2013), which we discussed in Chapter 8. We can see that each repeat has a sub–optimality of 0.005% which means the Gaussian process produces a design closer to the optimal design.

Uncertainty in hyperparameters

So far we have determined the optimal design by using a fitted Gaussian process in which the hyperparameters $\boldsymbol{\zeta} = (a, \boldsymbol{r}, \sigma)$ in the covariance function are fixed at their posterior mean. However, this ignores posterior uncertainty in these quantities. Here we examine the extent to which the expected utility is sensitive to ignoring this posterior uncertainty and, if it is not, how sensitive it is to various choices of $\boldsymbol{\zeta}$.

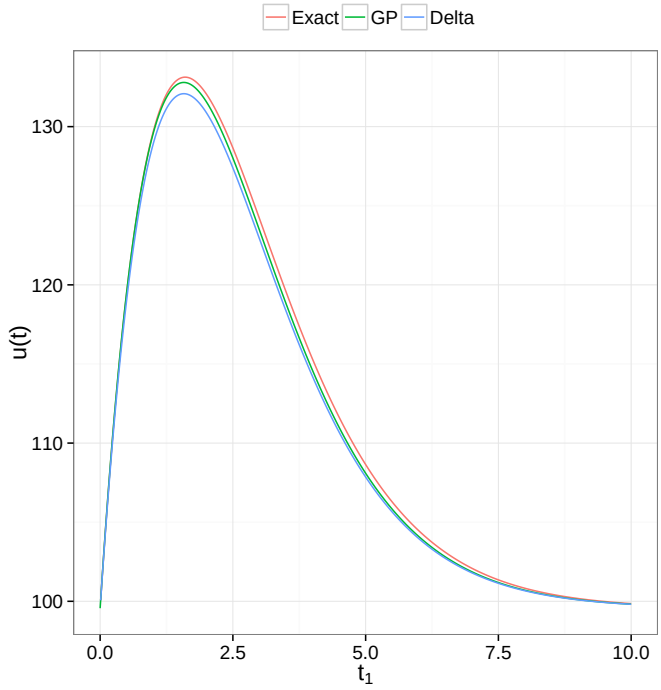


Figure 9.12: Graph comparing the exact, delta and Gaussian process (GP) approximation for a single timepoint design.

Method	Optimal design ($\boldsymbol{d}^*$)	Exact expected utility ($u(\boldsymbol{d}^*)$)	% sub–optimality
Exact	(1.60)	133.13	0.000
D&P	(1.60)	133.13	0.000
Delta	(1.58)	133.12	0.003
GP	(1.58)	133.12	0.003
ABC1	(1.57)	133.12	0.005
ABC2	(1.62)	133.12	0.005

Table 9.2: Comparison of optimal designs for different methods for a three timepoint design. Exact is the optimal design found using numerical integration, D&P are the Drovandi and Pettitt (2013) optimal designs, Delta is the optimal design found using the delta approximation, GP is the optimal design found using the Gaussian process methods and ABC1 and ABC2 are the optimal designs from two repeats of the Drovandi and Pettitt (2013) methods.

As the fitted Gaussian process is

$$f(\boldsymbol{X}^*|\boldsymbol{X},\boldsymbol{\zeta}) \sim N(\mu^*(\boldsymbol{X}^*|\boldsymbol{X},\boldsymbol{\zeta}), \Sigma^*(\boldsymbol{X}^*,\boldsymbol{X}^*|\boldsymbol{X},\boldsymbol{\zeta})),$$

we can allow for posterior uncertainty in $\boldsymbol{\zeta}$ by using posterior draws $\{\boldsymbol{\zeta}_i,\ i = 1, \ldots, n\}$ and

$$f(\boldsymbol{X}^*|\boldsymbol{X}) = E_{\boldsymbol{\zeta}|\boldsymbol{X}}[f(\boldsymbol{X}^*|\boldsymbol{X}, \boldsymbol{\zeta})] \simeq \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{X}^*|\boldsymbol{X}, \boldsymbol{\zeta}_i) \sim N\left(\mu^{**}, \Sigma^{**}\right),$$

where

$$\mu^{**} = \frac{1}{n}\sum_{i=1}^{n} \mu^*(\boldsymbol{X}^*|\boldsymbol{X}, \boldsymbol{\zeta}_i) \quad \text{and} \quad \Sigma^{**} = \frac{1}{n}\sum_{i=1}^{n} \Sigma^*(\boldsymbol{X}^*, \boldsymbol{X}^*|\boldsymbol{X}, \boldsymbol{\zeta}_i).$$

One potential problem in using this approximation is that it can be time consuming for large n. An alternative is to make a further approximation (using the delta method) in which we use $f(\boldsymbol{X}^*|\boldsymbol{X}, \boldsymbol{\zeta} = \mathrm{E}(\boldsymbol{\zeta}|\boldsymbol{X}))$. Figure 9.13 shows the expected utility determined by averaging over posterior realisations of the hyperparameters and also by fixing them at various values, namely, their posterior mean, posterior mode, posterior median, lower 2.5% and lower 97.5% values (determined from their posterior realisations). The plot appears to show only one curve and this is because all these methods for determining the expected utility give extremely close approximations of the expected utility – so close that they cannot be distinguished on the plot. This shows that, in this case, evaluating the expected utility is quite insensitive to the choice of hyperparameter used (but within posterior support) in fitting the Gaussian process. In general, we cannot rely on such a high level of insensitivity and so we will use a Gaussian approximation with the hyperparameter fixed at its posterior mean. This has the advantage of being quicker to evaluate than when averaging the Gaussian process over hyperparameter uncertainty but is still a principled evaluation in that it is a delta approximation.

Number of training points

We now consider how many training points should be used when fitting the Gaussian process. Figure 9.14(a) shows the fitted Gaussian process for the expected utility when using 25, 50, 100 and 200 training data points in a folded Latin hypercube design for (t_1, θ), where $t_1 \in (0, 10)$ and $\theta \in (0.7681, 1.2873)$, the central 95% prior interval. Some of these curves are difficult to distinguish and so Figure 9.14(b) provides a more focussed plot around the peak of the expected utility. We note that the optimal single timepoint designs $\boldsymbol{d}^*$ are at 1.68, 1.53, 1.57 and 1.58 for the Gaussian processes fitted using 25, 50, 100 and 200 training points respectively and that the exact optimal design is $\boldsymbol{d}^* = 1.60$. Therefore, not surprisingly, larger numbers of training points give more accurate estimates of the optimal design. Clearly focusing in on a smaller region within the design space (using the delta approximation method) will yield a more efficient use of the n_d training points.

Chapman et al. (1994) suggest that using $n_d = 10n_p$ training points, where n_p is the number of inputs for the Gaussian process, is sufficient. However the above results show

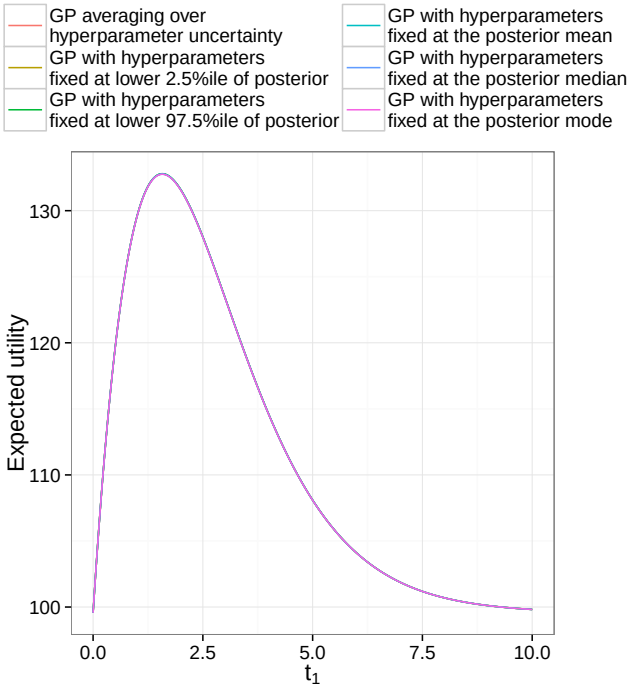


Figure 9.13: Graph showing the Gaussian process approximation to $u(t_1)$, determined by fixing the hyperparameters at their posterior mean, posterior mode, posterior median, lower 2.5%ile and lower 97.5%ile. Also shown is a curve in which the Gaussian process has been averaged over hyperparameter uncertainty. These methods produce very similar approximations of the expected utility which is why the curves overlap.

that even using $25n_p$ training points does not give a sufficiently accurate approximation and so we will fit the Gaussian process using $n_d = 100n_p$ training points.

9.6.3 Optimal two timepoint design

We now determine the optimal two timepoint design by using the fold method to construct the training points used to fit the Gaussian process with input $\boldsymbol{x} = (t_1, t_2, \theta)'$. However, before doing this we will reduce the design space by evaluating a delta approximation to the expected utility over a grid of feasible points with mesh size 0.01. This approximation gives the (approximate) expected utility shown in Figure 9.17(a). Here the optimal design is $\boldsymbol{d}^* = (1.01, 2.59)$ and so we need to reduce the design space to a region around this point. There are many ways for doing this but it is perhaps helpful to use a formal (rather than informal) approach. We will make use of the approximate utilities calculated using the delta approximation. Here there are two simple ways of using this information to reduce the design space: one is to only include designs which have approximate utilities within say 5% of the optimal approximate utility and another is to take the feasible region within a square defined by say the central 95% posterior univariate intervals of $\boldsymbol{d}$. In this

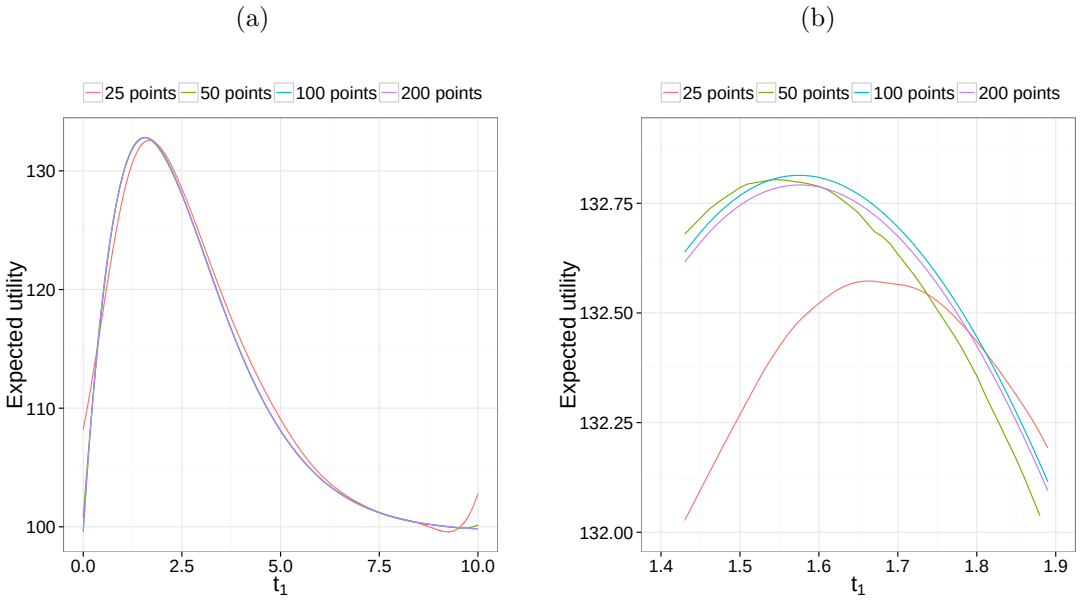


Figure 9.14: (*a*) Graph showing the Gaussian process approximation to $u(t_1)$, determined using $t_1 \in (0, 10)$ and $\theta \in (0.7681, 1.2873)$, the central 95% prior interval, and different training data sets with either 25, 50, 100 or 200 points in the folded Latin hypercube over (t_1, θ). (*b*) As (*a*) but focused in around the modes.

example, the optimal approximate utility is 140.85 and so we could take those designs with approximate utility larger than 133.81. Unfortunately this region is rather complex and not amiable to the generation of training points along the lines we have discussed (which use a folded Latin hypercube design). That said, we could take the smallest rectangle containing approximate utilities larger than 133.81. However, the second strategy is much more straightforward to apply and so we will adopt this method to reduce the design space before attempting to determine the fully optimal design.

Using this second strategy leads us to take training points from the folded Latin hypercube restricted to the ranges

$$0.11 < t_1 < 2.14 \quad \text{and} \quad 1.2 < t_2 < 6.59.$$

Again we take $\theta \in (0.7681, 1.2873)$ as this is the central 95% region for our prior distribution. The training points $(\boldsymbol{d}, \theta)_i$ are constructed via the fold method as described in Section 9.1.4.

The Gaussian process for $\boldsymbol{x} = (t_1, t_2, \theta)'$ is fitted with mean function

$$\begin{aligned} m(t_1, t_2, \theta) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 \frac{1}{\theta} + \hat{\beta}_4 t_1^2 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 t_1^3 + \hat{\beta}_7 t_2^3 \\ + \hat{\beta}_8 \frac{t_1}{\theta} + \hat{\beta}_9 t_2 \theta + \hat{\beta}_{10} t_1 t_2 + \hat{\beta}_{11} \frac{t_2^2}{\theta} + \hat{\beta}_{12} \frac{t_2}{\theta^2} \end{aligned}$$

and squared exponential covariance function as in Equation (9.8) which has five hyper-

parameters $(a, r_1, r_2, r_3, \sigma)'$. Here the mean function has been determined by fitting a full linear model which includes up to cubic terms and two way interactions and then removing any terms that were not significant. This fit has $R^2 = 0.9816$ and this very high value shows that this prior mean function is a good fit to the training data. Note that the coefficients in the mean function are the least squares estimates obtained through the fitting process.

We now attempt to get an even better fitting model to the training data by fitting a Gaussian process to the residuals of this mean function. The marginal distributions for the hyperparameters are given in Figure 9.15. We now look at the fit of this Gaussian process when fixing the hyperparameters at their posterior mean. The diagnostics of this fitted Gaussian process are given in Figure 9.16. The validation points that produce large standardised prediction errors tend to be when $t_1 < 0.5$. We believe this is due to the assumption of a constant σ being unsuitable near to the initial timepoint: the initial value for the death process is known and so the utility $u(\boldsymbol{d}, \boldsymbol{\theta})$ is close to constant over $\boldsymbol{\theta}$ near the initial timepoint. The overall goodness-of-fit (MD) for this Gaussian process is 312.46 which is again large (upper $\chi^2_{100}(0.01) = 135.81$) and mainly due to large contributions from points with $t_1 < 0.5$. The probability integral transform plot looks roughly uniform, as it should if a Gaussian process is appropriate.

A contour plot of the expected utility calculated using this Gaussian process is shown in Figure 9.17(c). This looks very similar to those resulting from using the delta method (Figure 9.17(a)) and the exact method (Figure 9.17(b)). The optimal design using this Gaussian process method is $\boldsymbol{d}^* = (1.01, 2.60)$ and this is very close to the exact optimal design; see Table 9.3. Also the optimal design from the Gaussian process is only 0.003% sub–optimal and is better than the other approximations listed in the table. By contrast the Drovandi and Pettitt (2013) method is the worst performing, giving a solution which is 0.198% sub–optimal. However, it must be noted that we use the exact utilities in the training data for the Gaussian process.

9.6.4 Optimal three and four timepoint designs

In this section we determine the optimal three timepoint design and the optimal four timepoint design. As before, $n_d = 100 n_p$ training points from a folded Latin hypercube are used to fit a Gaussian process to expected utilities calculated via the delta approximation. This fitted Gaussian process is then used to determine the smallest cube containing expected utilities within 5% of the maximum expected utility. The optimal design is then determined using a Gaussian process fitted to a set of training points within this reduced space. This space has $\theta \in (0.7681, 1.2873)$, the central 95% region for our prior distribution and, for

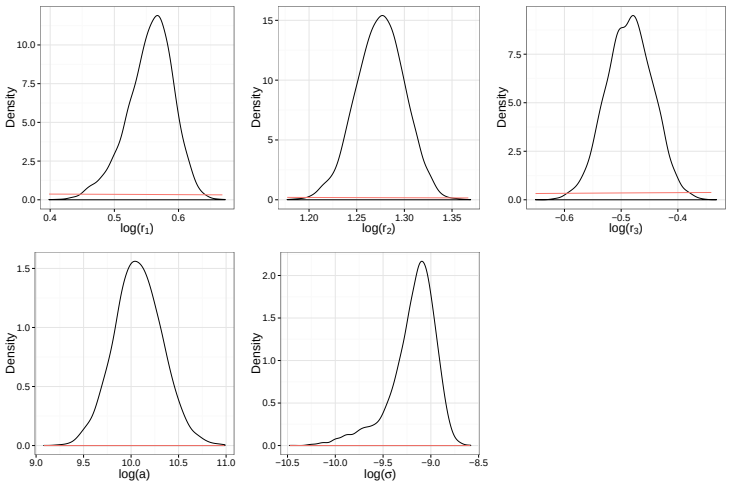


Figure 9.15: Marginal posterior distributions for the hyperparameters for the two timepoint design for the death model.

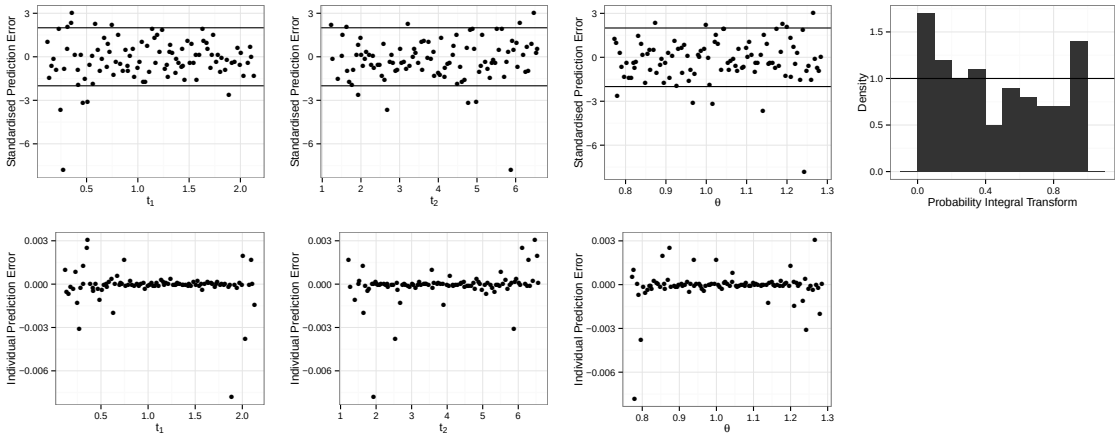


Figure 9.16: Diagnostics for the two timepoint design for the death model.

the three timepoint design

$$0 < t_1 < 1.9, \quad 0.4 < t_2 < 4.6 \quad \text{and} \quad 1.5 < t_3 < 10.0$$

and for the four timepoint design

$$0 < t_1 < 1.8, \quad 0 < t_2 < 4.1, \quad 0.5 < t_3 < 9.9, \quad \text{and} \quad 1.6 < t_4 < 10.0.$$

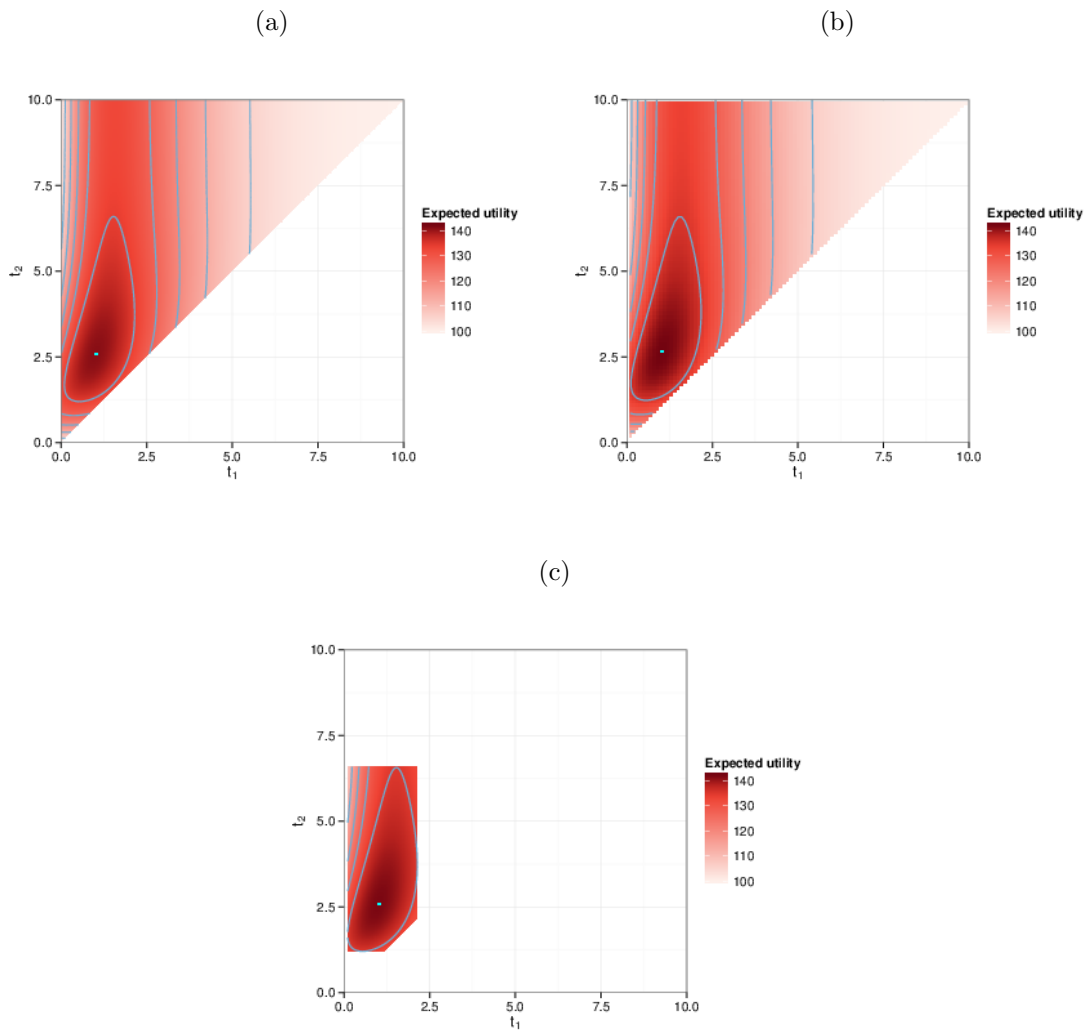


Figure 9.17: Contour plots of the expected utility calculated using (*a*) the delta method; (*b*) the exact method and (*c*) the GP method.

In the case of a three timepoint design, we take the mean function for the Gaussian process with inputs $\boldsymbol{x} = (t_1, t_2, t_3, \theta)'$ as

$$m(t_1, t_2, t_3, \theta) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 \frac{1}{\theta} + \hat{\beta}_5 t_1^2 + \hat{\beta}_6 t_2^2 + \hat{\beta}_7 t_3^2 + \hat{\beta}_8 \frac{1}{\theta^2} + \hat{\beta}_9 t_2^3 + \hat{\beta}_{10} t_3^3 + \hat{\beta}_{11} \frac{1}{\theta^3} + \hat{\beta}_{12} t_1 t_2 + \hat{\beta}_{13} t_2 t_3 + \hat{\beta}_{14} t_1 t_3 + \hat{\beta}_{15} \frac{t_1}{\theta} + \hat{\beta}_{16} \frac{t_3}{\theta}$$

and use the squared exponential covariance function in Equation (9.8) which has six hyperparameters $(a, r_1, r_2, r_3, r_4, \sigma)'$. For the four timepoint design we fit a Gaussian

Method	Optimal design ($\boldsymbol{d}^*$)	Exact expected utility ($u(\boldsymbol{d}^*)$)	% sub–optimality
Exact	(1.03, 2.65)	142.29	0.000
D&P	(1.15, 3.05)	142.01	0.198
Delta	(1.01, 2.59)	142.28	0.004
GP	(1.01, 2.60)	142.28	0.003
ABC1	(1.07, 2.76)	142.26	0.018
ABC2	(1.14, 2.85)	142.18	0.078

Table 9.3: Comparison of optimal designs for different methods for a three timepoint design. Exact is the optimal design found using numerical integration, D&P are the Drovandi and Pettitt (2013) optimal designs, Delta is the optimal design found using the delta approximation, GP is the optimal design found using the Gaussian process methods and ABC1 and ABC2 are the optimal designs from two repeats of the Drovandi and Pettitt (2013) methods.

process with inputs $\boldsymbol{x} = (t_1, t_2, t_3, t_4, \theta)'$, mean function

$$\begin{aligned} m(t_1, t_2, t_3, t_4, \theta) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_4 + \hat{\beta}_5 \frac{1}{\theta} + \hat{\beta}_6 t_1^2 + \hat{\beta}_7 t_2^2 + \hat{\beta}_8 t_3^2 + \hat{\beta}_9 \frac{1}{\theta^2} \\ + \hat{\beta}_{10} t_2^3 + \hat{\beta}_{11} t_3^3 + \hat{\beta}_{12} t_1 t_2 + \hat{\beta}_{13} t_2 t_3 + \hat{\beta}_{14} t_2 t_4 + \hat{\beta}_{15} \frac{t_1}{\theta} + \hat{\beta}_{16} \frac{t_2}{\theta} \\ + \hat{\beta}_{17} \frac{t_3}{\theta} + \hat{\beta}_{18} \frac{t_4}{\theta} \end{aligned}$$

and use a squared exponential covariance function in Equation (9.8) which has seven hyperparameters $(a, r_1, r_2, r_3, r_4, r_5, \sigma)'$. The linear models with these fitted mean functions have $R^2 = 0.9831$ and $R^2 = 0.9803$ respectively and these high values again show that these prior mean functions provide a good fit to the training data.

Fitting a Gaussian process to the residuals of the linear model in each case, the marginal posterior distributions for the hyperparameters for the three and four timepoint designs are as in Figure 9.18 and Figure 9.20 respectively. We see fairly symmetric unimodal densities in all cases and again examine the diagnostics of the Gaussian processes with the hyperparameters fixed at their posterior mean. The graphical diagnostics are given in Figures 9.19 and 9.21. Just over 5% of the individual prediction errors are outside of the range $(-2, 2)$, with the problematic points occurring when the first timepoint is near zero; see the plot of standardised and individual prediction errors against t_1 in Figures 9.19 and 9.21. We note that the standardised error plots suggest are some potential problems in the fit of the Gaussian processes. This is confirmed by the large values for the goodness-of-fit measures (MD) of 180.92 and 238.00 respectively (upper $\chi^2_{100}(0.01) = 135.81$). However, the individual error plots suggest that, despite these statistical diagnostics indicating poor fit, the Gaussian processes do actually produce quite accurate predictions. Therefore, we will proceed by using these fitted Gaussian processes to determine the optimal designs.

Tables 9.4 and 9.5 give the optimal three and four timepoint designs respectively

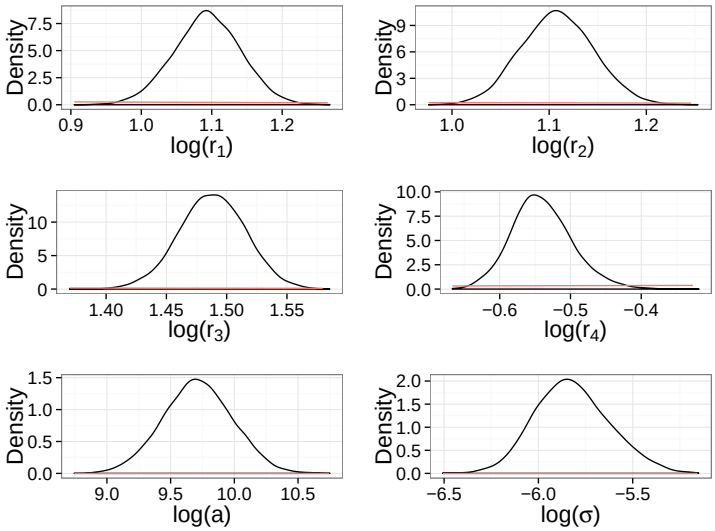


Figure 9.18: Marginal posterior distributions for the hyperparameters for a three timepoint design for the death model. Prior distributions are given in red.

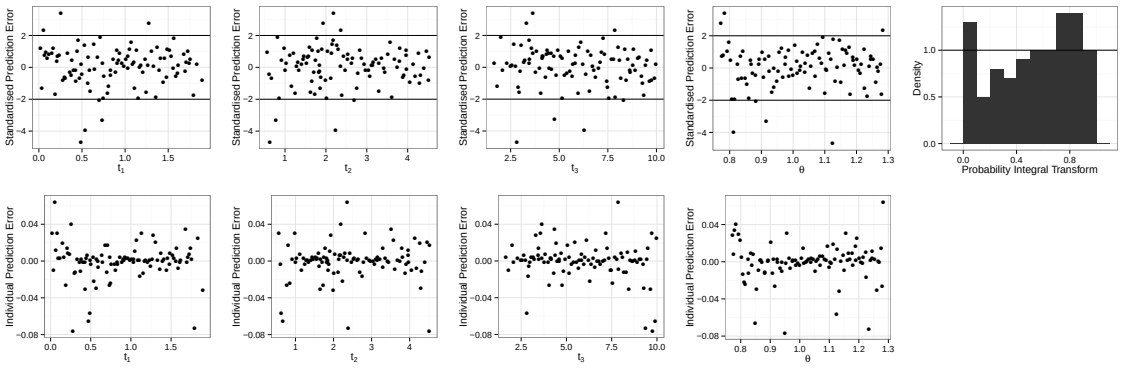


Figure 9.19: Diagnostics for the three timepoint design Gaussian process for the death model.

determined when using several methods. For the three timepoint design, the Gaussian process method gives an optimal design which is only 0.003% sub–optimal and better than the other non-exact methods. Similarly the Gaussian process method gives a four timepoint optimal design which is very close to the exact optimal design, being only 0.002% sub–optimal. We note that, for both three and four timepoint designs, the Drovandi and Pettitt (2013) and our implementation of their methods have higher sub–optimalities.

9.6.5 Summary

Table 9.6 contains a summary of the optimal one to four timepoint designs and their expected utilities. Designs with more timepoints have higher expected utilities and so with no other restrictions, we would also choose to use a design with more timepoints. However,

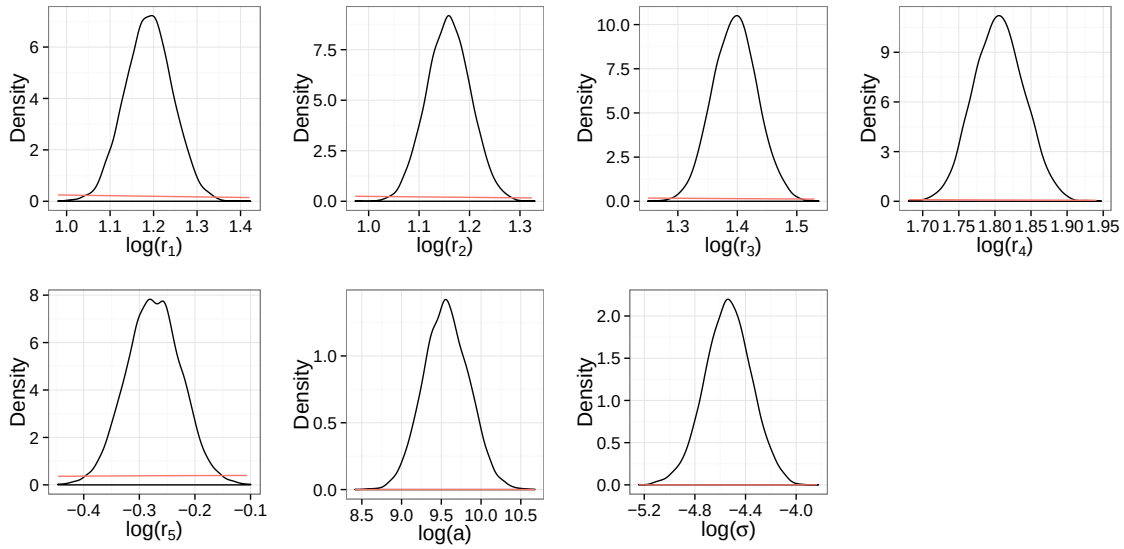


Figure 9.20: Marginal posterior distributions for the hyperparameters for a four timepoint design for the death model. Prior distributions are given in red.

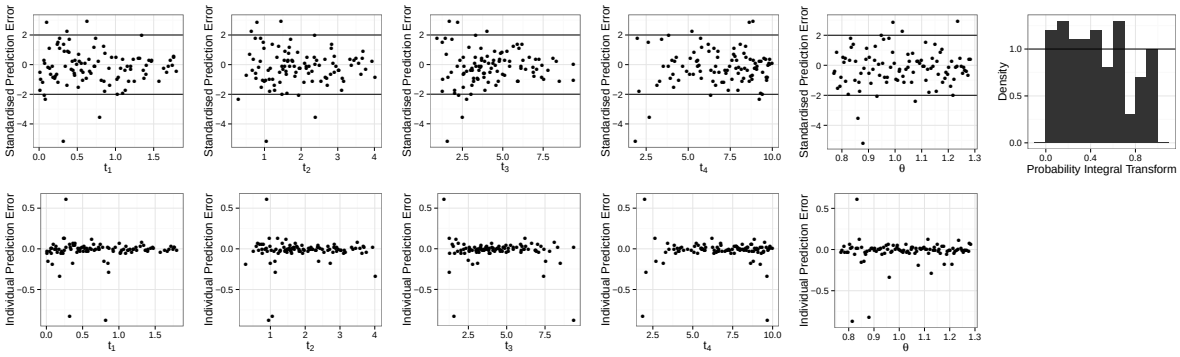


Figure 9.21: Diagnostics for the four timepoint design Gaussian process for the death model.

it is plausible that in practice taking measurements has a cost of its own. Indeed this is likely to take the form of a (constant) cost for each timepoint. Such considerations allow us to determine both the optimal number of timepoints to use in a design and also the actual times to use. For example, the increase in the expected utility from one to two timepoint designs, two to three timepoint designs and three to four timepoint designs is 9.16, 3.80 and 1.92 respectively. Here, the marginal benefit of adding another timepoint to the design decreases with each additional timepoint. This is to be expected as the amount of remaining uncertainty decreases as the number of timepoints in the design increases. Therefore, if the cost per timepoint was 2 units then the optimal number of timepoints would be three: adding a fourth timepoint adds a further 1.92 units but costs 2 units, giving a negative benefit.

It is interesting to see the optimal timing of the observations in the death model when

Method	Optimal design ($\boldsymbol{d}^*$)	Exact expected utility ($u(\boldsymbol{d}^*)$)	% sub–optimality
Exact	(0.76, 1.79, 3.42)	146.08	0.000
D&P	(0.75, 1.90, 3.90)	145.90	0.125
Delta	(0.75, 1.76, 3.34)	146.07	0.005
GP	(0.75, 1.76, 3.35)	146.08	0.003
ABC1	(0.86, 2.00, 3.76)	145.97	0.075
ABC2	(0.90, 2.00, 3.83)	145.91	0.116

Table 9.4: Comparison of optimal designs for different methods for a three timepoint design. Exact is the optimal design found using numerical integration, D&P is the Drovandi and Pettitt (2013) optimal design, Delta is the optimal design found using the delta approximation, GP is the optimal design found using the Gaussian process method and ABC1 and ABC2 are the optimal designs from two repeats of the Drovandi and Pettitt (2013) methods.

Method	Optimal design ($\boldsymbol{d}^*$)	Exact expected utility ($u(\boldsymbol{d}^*)$)	% sub–optimality
Exact	(0.60, 1.36, 2.38, 3.98)	148.01	0.000
D&P	(0.75, 1.70, 2.75, 4.35)	147.80	0.214
Delta	(0.60, 1.35, 2.36, 3.94)	148.01	0.002
GP	(0.60, 1.39, 2.38, 3.98)	148.00	0.002
ABC1	(0.74, 1.69, 2.74, 3.94)	147.70	0.214
ABC2	(0.74, 1.69, 2.74, 3.94)	147.70	0.214

Table 9.5: Comparison of optimal designs for different methods for a four timepoint design. Exact is the optimal design found using numerical integration, D&P is the Drovandi and Pettitt (2013) optimal design, Delta is the optimal design found using the delta approximation, GP is the optimal design found using the Gaussian process method and ABC1 and ABC2 are the optimal designs from two repeats of the Drovandi and Pettitt (2013) methods.

increasing the number of timepoints. Figure 9.22 shows these timings as the joins between blocks of grey and white. The figure also shows the stochastic mean of the death model, fixing the death rate at its prior mean ($\theta = 1$). It is clear that the optimal two timepoint design is roughly evenly spaced around the optimal single timepoint design. The optimal three timepoint design has its second timepoint close to the optimal single timepoint design and then the first and third timepoints either side. The optimal four timepoint design has its second and third timepoints roughly evenly spaced around the optimal single timepoint design and the first and fourth timepoints are either side of the second and third.

It is not surprising that none of the optimal designs have timepoints very close to zero as the initial number of individuals in the population is known and a very high death rate is pretty implausible (from the prior for θ). Also none of the optimal designs have timepoints after time 5. This is probably because it is likely that population would be extinct beyond this point; for example, the probability that the population is extinct by

# timepoints (d)	Optimal design ($\boldsymbol{d}^*$)	Exact expected utility ($u(\boldsymbol{d}^*)$)
1	(1.58)	133.12
2	(1.01, 2.60)	142.28
3	(0.75, 1.76, 3.35)	146.08
4	(0.60, 1.39, 2.38, 3.98)	148.00

Table 9.6: Optimal designs for one to four timepoint designs found using the Gaussian process methods and their exact expected utility.

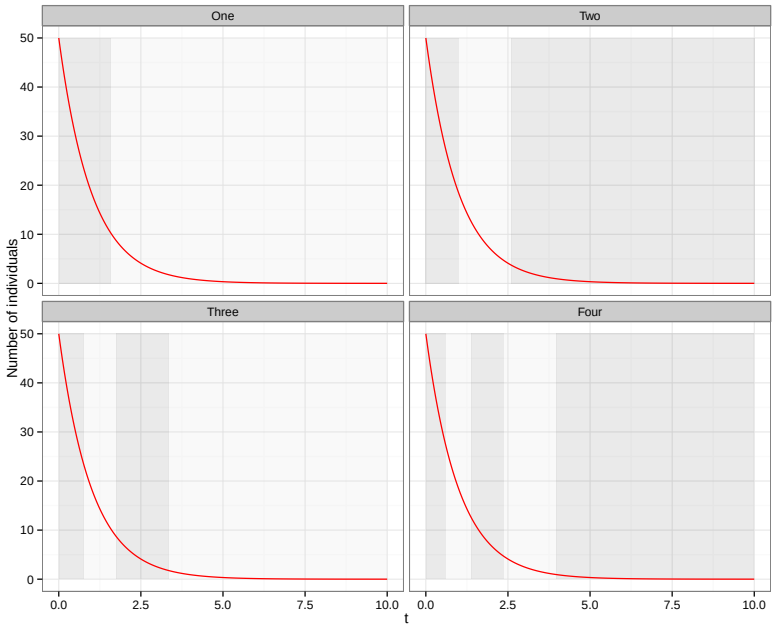


Figure 9.22: The red line is the stochastic mean of the death model when the parameter is fixed at the prior mean ($\theta = 1$). The edges of the grey and white blocks represent the optimal designs for one, two, three and four timepoint designs.

time 5 is over 70% if $\theta = 1$, with similar sized extinction probabilities for other plausible values from the prior distribution for θ. In such cases, taking further observations would not yield additional information about the death rate θ and so it would not be sensible (or indeed optimal) to take observations in the time interval $(5, 10)$.

Chapter 10

Experimental design for the Lotka–Volterra model

In this chapter our goal is to determine the optimal timepoints for observations to take place in the Lotka–Volterra (LV) system (Lotka, 1925; Volterra, 1926). This model is fully described in Section 7.3.2 but briefly it is governed by three reactions

$$\mathcal{Y}_1 \xrightarrow{\theta_1} 2\mathcal{Y}_1, \qquad \mathcal{Y}_1 + \mathcal{Y}_2 \xrightarrow{\theta_2} 2\mathcal{Y}_2 \qquad \text{and} \qquad \mathcal{Y}_2 \xrightarrow{\theta_3} \emptyset,$$

where $\mathcal{Y}_1$ are predators and $\mathcal{Y}_2$ are prey. The parameters in the model are θ_1, prey reproduction rate, θ_2, the prey death/predator reproduction rate and θ_3, the predator death rate.

As in the previous chapter, we solve this experimental design problem by taking a two step approach. First, we approximate the expected utility function using the delta approximation and use this to reduce the design space. We then focus on this reduced space, incorporate parameter uncertainty described by the prior distribution for the rate constants, and finally estimate the optimal design using the mode of the expected utility. We use the same utility function as before, namely, the posterior generalised precision and, as the likelihood for this model is intractable, we use the linear noise approximation to estimate the posterior variance matrix.

10.1 Design space reduction

In Chapter 9 we illustrated that the delta approximation can be used effectively to 'zoom' into the mode of the utility function. However since the design space of the Lotka–Volterra system is larger than that of the death model, instead of constructing a regular grid of design points, we will fit a Gaussian process to the hypercube generated training data.

For each scenario we consider we use $n_d = 100n_p$ points generated via the fold method,

where n_p is the dimension of the point (see Chapter 9). At each design point, we use the delta approximation to estimate the expected utility, that is, we fix the parameters at their prior means and then generate an 'average' realisation $\boldsymbol{y}_{\boldsymbol{d}}^*$ using the mean equations from the linear noise approximation. Therefore the expected utility is

$$u(\boldsymbol{d}, \boldsymbol{y}_{\boldsymbol{d}}^*) = \frac{1}{\det\{\mathrm{Var}_{\mathrm{LNA}}(\boldsymbol{\theta} \mid \boldsymbol{d}, \boldsymbol{y}_{\boldsymbol{d}}^*)\}}.$$

A Gaussian process is then fitted to the training data using the techniques described in Section 9.1.3. The prior mean function is chosen by fitting a multiple linear regression model up to and including cubic terms in $\boldsymbol{d}$ (including all two way interactions) and removing any non significant terms. Throughout this chapter we take the prior distribution for the Gaussian process hyperparameters $(a, \boldsymbol{r}, \sigma)$ to have independent diffuse log–normal components, with

$$r_i \sim LN(0, 10), \quad i = 1, \ldots, n_p, \qquad a \sim LN(0, 10) \qquad \text{and} \qquad \sigma \sim LN(0, 20).$$

In the fitted Gaussian process approximation to the expected utility, the hyperparameters are fixed at their posterior mean.

Next, we embed this fitted Gaussian process within an MCMC scheme to find the marginal distribution of $\boldsymbol{d}$. The scheme used is given in Algorithm 25. To aid the multivariate mode estimation, we amplify the utility function using the method suggested by Müller (1999). For the delta approximation this means we approximate J expected utilities for each proposed design and then set the expected utility to be a product of the J approximated expected utilities. This results in a more peaked multivariate expected utility surface, from which the mode is easier to identify. Note that the marginal modes are not necessarily the same for different values of J but the multivariate mode is the same for different values of J as, for each J, the multivariate mode is the mode of the Jth power of the expected utility. Again following Müller (1999), to select a value for J we start with $J = 1$ and slowly increase J until simulated designs are tightly clustered. In this study, we use $J = 1$, 2, 5, 10 and 20. If J is increased too quickly, local modes can become isolated resulting in the MCMC scheme having poor or indeed no mixing.

Finally, the design space is reduced by taking the cube formed from the central 95% interval of the marginal samples for $\boldsymbol{d}$.

Algorithm 25 MCMC scheme to find the optimal design using the Gaussian process approximation to $u(\boldsymbol{d}, \boldsymbol{y}_{\boldsymbol{d}}^*)$ amplifying the mode.

1. Set $i = 1$ and initialise $\boldsymbol{d}^0 \sim \pi(\boldsymbol{d})$ for $j = 1, \ldots, J$.
2. Estimate $u_j(\boldsymbol{d}^0, \boldsymbol{y}_{\boldsymbol{d}}^*)$ from the Gaussian process for $j = 1, \ldots, J$.
3. Calculate $u_0 = \prod_{j=1}^{J} u(\boldsymbol{d}^0, \boldsymbol{y}_{\boldsymbol{d}}^*)$.
4. Propose a new $d^c \sim q(\boldsymbol{d}|\boldsymbol{d}^{i-1})$ for $j = 1, \ldots, J$.
5. Estimate $u(\boldsymbol{d}^c, \boldsymbol{y}_{\boldsymbol{d}}^*)$ from the Gaussian process for $j = 1, \ldots, J$.
6. Calculate $u^c = \prod_{j=1}^{J} u(\boldsymbol{d}^c, \boldsymbol{y}_{\boldsymbol{d}}^*)$
7. Calculate the MH ratio $\alpha = \min(1, A)$ where
$$A = \frac{u^c q(\boldsymbol{d}^{i-1}|\boldsymbol{d}^c)}{u^{i-1} q(\boldsymbol{d}^c|\boldsymbol{d}^{i-1})}.$$
8. Accept proposed values with probability α.
9. Set $i = i + 1$ and return to step 4.

10.2 Fitting a Gaussian process to $u(\boldsymbol{d}, \boldsymbol{\theta})$

In this chapter we use a prior distribution for the rate constants $\boldsymbol{\theta}$ which has independent log-normal components, with

$$\theta_1 \sim LN(-0.69, 0.01), \qquad \theta_2 \sim LN(-6, 0.01) \qquad \text{and} \qquad \theta_3 \sim LN(-1.21, 0.01).$$

These have been chosen to represent a similar level of prior knowledge to that used in Drovandi and Pettitt (2013) for the death model. Note that the prior mean rate is $E(\boldsymbol{\theta}) = (0.5, 0.0025, 0.3)'$.

The Gaussian process is built by first generating a set of training points within the reduced design space and the central 99% prior intervals for the rate parameters. At each input $\boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{\theta})$, we simulate m_1 realisations of the LV process using Gillespie's Direct method and then estimate $u(\boldsymbol{d}, \boldsymbol{y})$ for each realisation. At a particular design point, we calculate the expected utility as

$$u(\boldsymbol{d}, \boldsymbol{\theta}) \simeq \frac{1}{m_1} \sum_{i=1}^{m_1} u(\boldsymbol{d}, \boldsymbol{y}_i). \tag{10.1}$$

The prior mean function for the Gaussian process is chosen by fitting a multiple linear

regression model up to cubic terms in $\boldsymbol{d}$ and $\boldsymbol{\theta}$ including all two way interactions and removing any non significant terms. The prior distribution for the hyperparameters is that given in Section 10.1. The Gaussian process approximation to $u(\boldsymbol{d}, \boldsymbol{\theta})$ is then incorporated into an MCMC scheme (Algorithm 23) and used to find the marginal distribution of $\boldsymbol{d}$. The mode is amplified using a value of J, chosen by slowly increasing J.

10.3 Estimating the mode

We approximate the multivariate mode of $\boldsymbol{d}$ by using marginal samples of $\boldsymbol{d}$ obtained from the MCMC scheme (Algorithm 23) in three ways:

- approximate the multivariate mode using Algorithm 19 as suggested by Drovandi and Pettitt (2013);
- use a 10% trimmed sample mean of the marginal samples of $\boldsymbol{d}$ (Müller, 1999);
- use the modes of the marginal samples of $\boldsymbol{d}$.

Once we obtain estimates to the multivariate mode from each of the three methods, we use the Gaussian process of the utility function to estimate the expected utility at each of the three estimates. We then choose to use the the estimated multivariate mode which has the highest expected utility as a guide to where the optimal design is.

To further improve the estimate of the multivariate mode, we evaluate the expected utility using the Gaussian process for a set of designs around the chosen multivariate mode. This is done by selecting a small range of values for each timepoint around the chosen multivariate mode, and evaluating the expected utility using the Gaussian process at all possible combinations of design in the range to two decimal places. For example, if the approximate multivariate mode chosen from the three methods for a two timepoint design is $\boldsymbol{d}^* = (3, 6)$ then we evaluate $u(\boldsymbol{d})$ using the Gaussian process at all possible combinations of designs with $2.90 < t_1 < 3.10$ and $5.90 < t_2 < 6.10$ to two decimal places. The design with the highest expected utility is chosen as the optimal design. The whole process is summarised in Algorithm 24.

10.4 Optimal design with only one unknown rate constant

We begin our search for optimal designs by considering first the case in which two of the three rate constants are known. Specifically we fix $\theta_2 = 0.0025$ and $\theta_3 = 0.3$ at their prior mean values. This reduces the problem to be one with only one unknown parameter θ_1.

10.4.1 Design space reduction

The first step is to reduce the design space. The prior distribution is $\theta_1 \sim LN(-0.69, 0.01)$ and so the (delta) mean path $\boldsymbol{y}_{\boldsymbol{d}}^*$ is obtained by setting θ_1 to its prior mean and then calculating the mean of the linear noise approximation.

Single timepoint design

Since the design space for the single timepoint design is relatively compact, we use a regular grid to generate the training data inputs $t_1 = 0, 0.2, 0.4, \ldots, 20$. For the delta approximation, all parameters are fixed at their prior means and so we take the prior mean function as

$$m(t_1) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_1^2 + \hat{\beta}_3 t_1^3,$$

that is, it does not depend on θ_1. We take the covariance function as the squared exponential function in Equation (9.8), where $\boldsymbol{x} = t_1$. This covariance function has three hyperparameters $(a, r_1, \sigma)'$.

The marginal posterior distributions for the hyperparameters are given in Figure 10.1. The posterior distributions for $\log r_1$ and $\log a$ are both positively skewed whereas that of $\log \sigma$ is symmetric. The associated diagnostics are shown in Figure 10.2. Although the diagnostic plots indicate that the Gaussian process may have issues, Figure 10.3(a) shows that the posterior mean function captures the training data surprising well. That said, there appears to be more noise in the training data around the peak of $u(t_1)$.

The marginal distribution of t_1, for $J = 1$, 5 and 20, is given in Figure 10.3(b). The mode of the marginal distribution occurs at exactly the same place for each value of J but the density becomes tighter around the mode with increasing J, as expected since the design space is one dimensional. To obtain the reduced design space for t_1, we extract the central 95% interval of the $J = 20$ marginal distribution and obtain

$$5.02 < t_1 < 7.88.$$

Multiple timepoints

Training data for the two to four timepoint designs are generated from a folded Latin hypercube with $n_d = 200$, 300 and 400 points respectively. The prior mean functions used

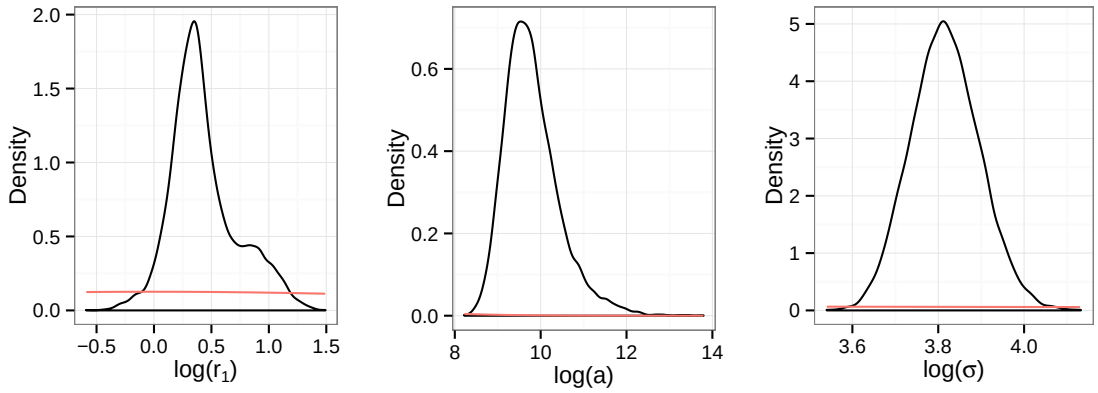


Figure 10.1: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a single timepoint design. Prior distributions are given in red.

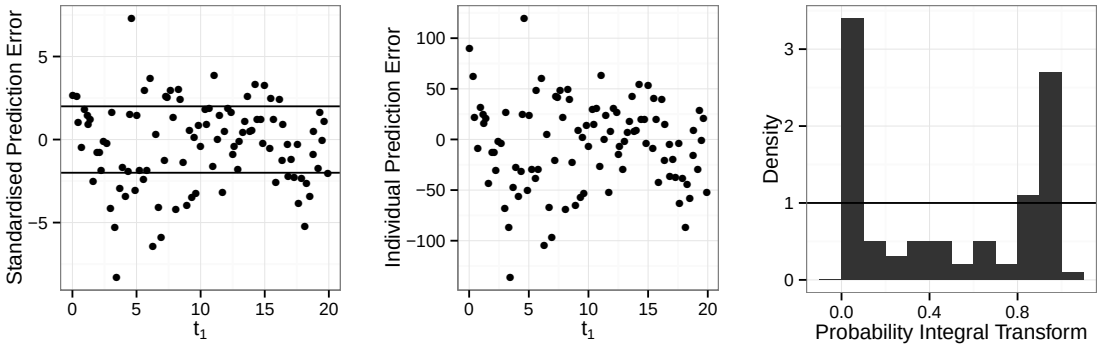


Figure 10.2: Diagnostics for the Gaussian process used to reduce the design space for the single timepoint design.

are

$$m(t_1, t_2) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_1^2 + \hat{\beta}_4 t_2^2 + \hat{\beta}_5 t_1^3 + \hat{\beta}_6 t_2^3,$$

$$\begin{aligned} m(t_1, t_2, t_3) = \hat{\beta}_0 &+ \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_1^2 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 t_3^2 + \hat{\beta}_7 t_1^3 + \hat{\beta}_8 t_2^3 + \hat{\beta}_9 t_3^3 \\ &+ \hat{\beta}_{10} t_1 t_2 + \hat{\beta}_{11} t_2 t_3, \end{aligned}$$

$$\begin{aligned} m(t_1, t_2, t_3, t_4) = \hat{\beta}_0 &+ \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_4 + \hat{\beta}_5 t_1^2 + \hat{\beta}_6 t_2^2 + \hat{\beta}_7 t_3^2 + \hat{\beta}_8 t_2^2 + \hat{\beta}_9 t_3^2 \\ &+ \hat{\beta}_{10} t_2^3 + \hat{\beta}_{11} t_3^3 + \hat{\beta}_{12} t_1 t_2 + \hat{\beta}_{13} t_2 t_3 + \hat{\beta}_{14} t_3 t_4 \end{aligned}$$

after non–significant terms have been removed. We again used the squared exponential covariance function. The marginal posterior distributions of the hyperparameters and the diagnostic plots are given in appendix A.1.1. They indicate a similar fit to the single timepoint design.

Using the resulting fitted Gaussian processes, we can then estimate the marginal

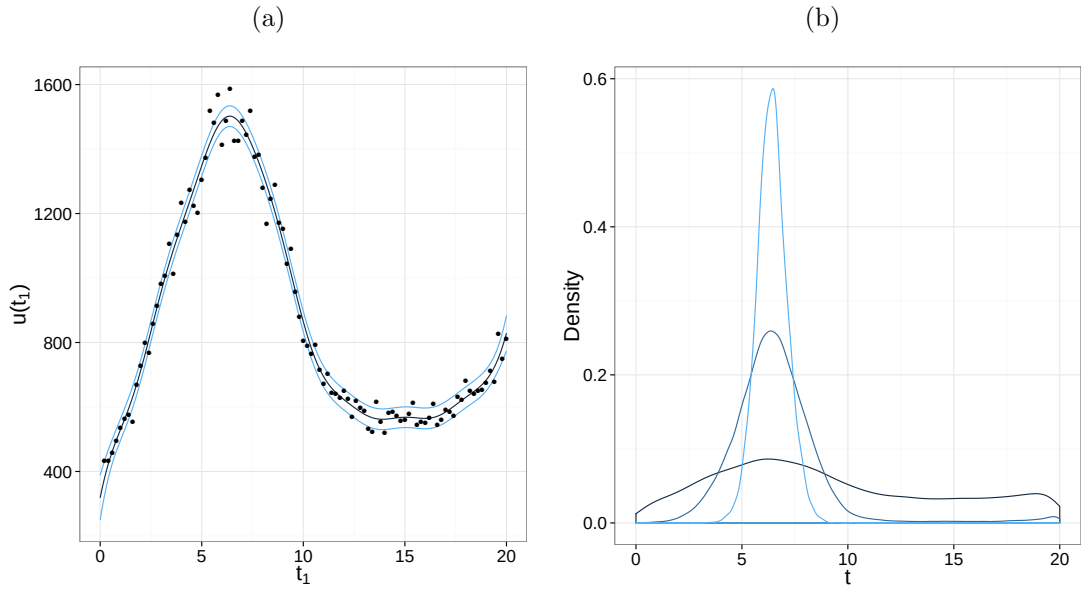


Figure 10.3: (a) The Gaussian process mean function ± 1.96 sd (blue lines) with the training data (black points). (b) Marginal distribution of t_1 with $J = 1$ (dark blue line), $J = 5$ (medium blue line) and $J = 20$ (light blue line).

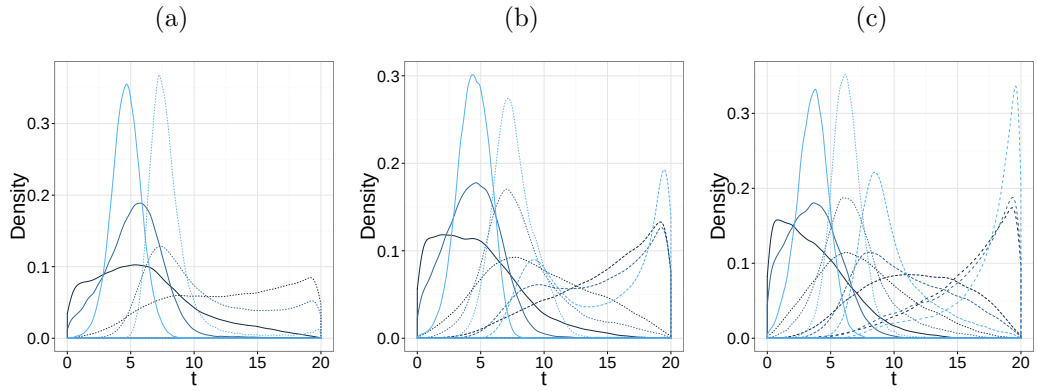


Figure 10.4: Marginal distributions of $\boldsymbol{d}$ with $J = 1$ (dark blue line), $J = 5$ (medium blue line) and $J = 20$ (light blue line). Plotted are the (a) two, (b) three and (c) four time point designs. The first to fourth time points are solid, small dashed, medium dashed and large dashed lines, respectively.

distribution of $\boldsymbol{d}$ for $J = 1, 5$ and 20; see Figure 10.4. Notice that as J is increased the marginal distributions become more peaked indicating the optimal timepoints more clearly. The resulting central 95% intervals which form the reduced space are

2–dim design: $2.33 < t_1 < 7.00$, $5.82 < t_2 < 15.92$
3–dim design: $1.86 < t_1 < 6.76$, $4.74 < t_2 < 11.37$, $6.29 < t_3 < 19.93$,
4–dim design: $1.09 < t_1 < 5.95$, $3.94 < t_2 < 9.02$, $6.27 < t_3 < 17.05$, $9.14 < t_4 < 19.94$.

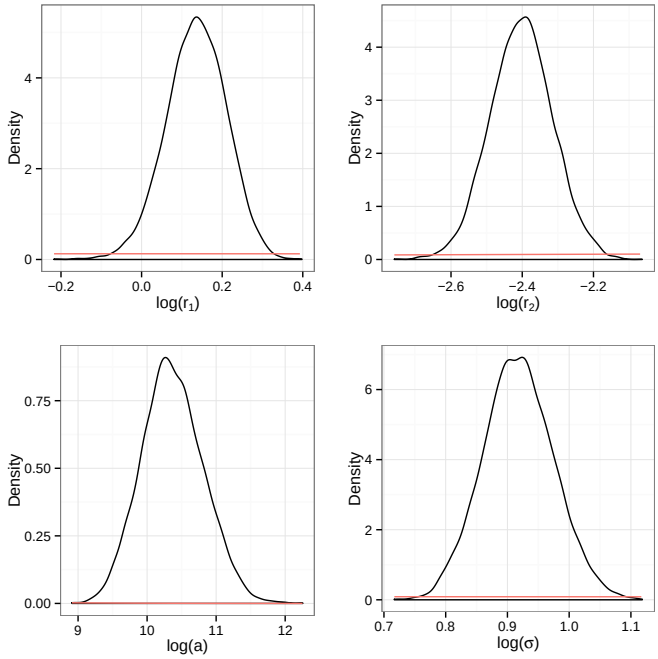


Figure 10.5: Marginal posterior distributions for the hyperparameters for a single timepoint design. Prior distributions are given in red.

10.4.2 Optimal designs using a Gaussian process fitted to $u(d, \theta_1)$ in the reduced space

Single timepoint design

Training data for the single timepoint design are generated using a maximin Latin hypercube on the truncated design space obtained from the delta approximation in the previous section. We take the mean function as

$$m(t_1, \theta_1) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 \theta_1 + \hat{\beta}_3 t_1^2 + \hat{\beta}_4 t_1 \theta_1 + \hat{\beta}_5 t_1^3 + \hat{\beta}_6 \theta_1^3,$$

and use the squared exponential covariance function in Equation (9.8). This prior mean function provides an excellent overall fit to the data ($R^2 = 0.932$). The hyperparameter marginal posterior distributions are given in Figure 10.5. All of these posterior distributions are fairly symmetric. The diagnostics are given in Figure 10.6. Again the diagnostic plots indicate potential issues with standardised prediction errors outside the range $(-2, 2)$. However, the individual prediction errors are very low, with the largest error being 0.7%.

Using this fitted Gaussian process to estimate the marginal distribution of t_1 with $J = 20$ gives the distribution in Figure 10.7. The optimal design here is $\boldsymbol{d}^* = 6.12$ with expected utility 1402. The location of this optimal timepoint is just after the typical prey peak as shown in Figure 10.9.

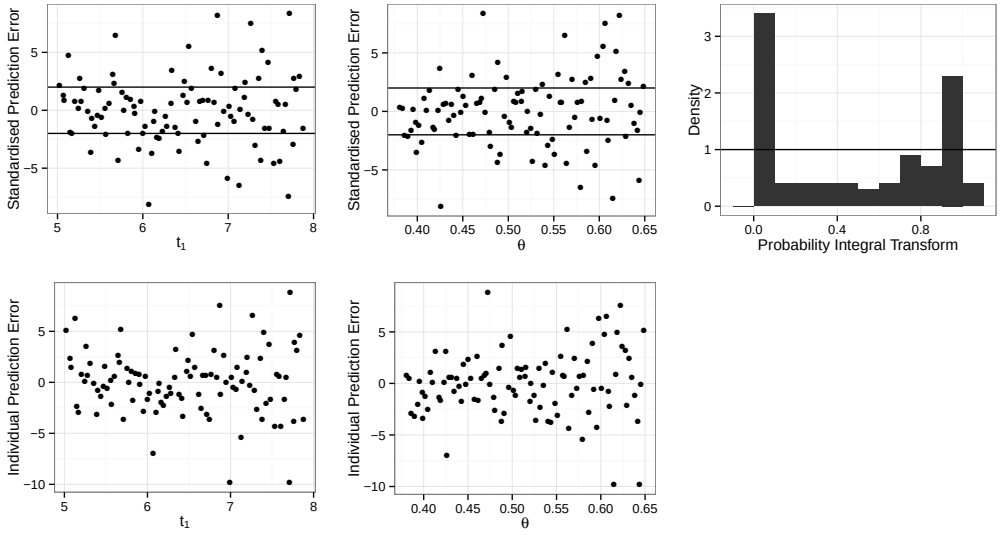


Figure 10.6: Diagnostics for the single timepoint design Gaussian process.

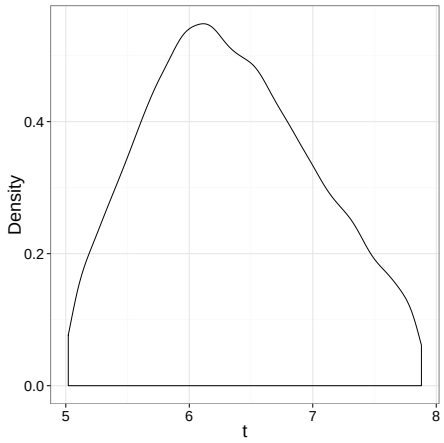


Figure 10.7: Marginal distribution of t_1 with $J = 20$.

Multiple timepoints

Training data for the two to four timepoint designs was generated using the fold method with $n_d = 400$, 500 and 600 respectively (see Appendix A.1.2). The prior mean functions

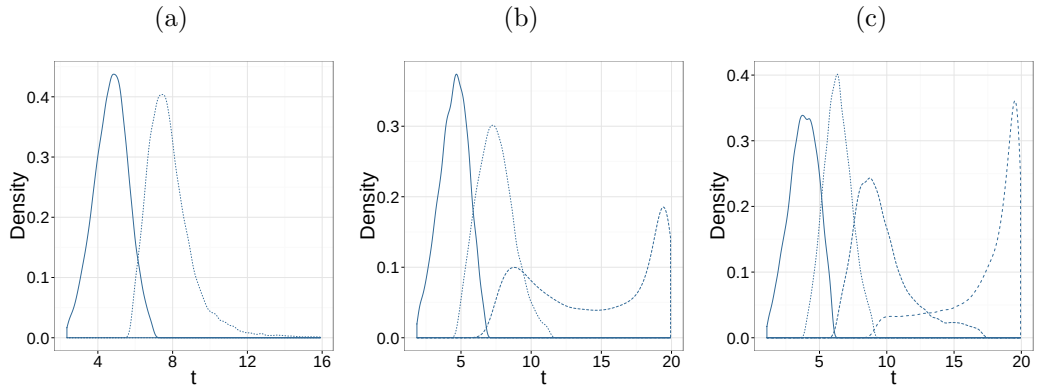


Figure 10.8: Marginal distributions of $\boldsymbol{d}$ with $J = 20$, for the (a) two time point, (b) three timepoint and (c) four timepoint design. The first to fourth timepoints within a design are shown using solid, small dashed, medium dashed lines and large dashed lines respectively.

used in the following analyses are

$$\begin{aligned} m(t_1, t_2, \theta_1) = \hat{\beta}_0 &+ \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 \theta_1 + \hat{\beta}_4 t_1^2 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 t_1^3 + \hat{\beta}_7 t_2^3 \\ &+ \hat{\beta}_8 t_2 \theta_1 + \hat{\beta}_9 t_1 t_2, \\ m(t_1, t_2, t_3, \theta_1) = \hat{\beta}_0 &+ \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 \theta_1 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 t_3^2 + \hat{\beta}_7 \theta_1^2 + \hat{\beta}_8 t_1^3 + \hat{\beta}_9 t_2^3 \\ &+ \hat{\beta}_{10} t_3^3 + \hat{\beta}_{11} t_1 t_2 + \hat{\beta}_{12} t_2 t_3 + \hat{\beta}_{13} t_2 \theta_1 + \hat{\beta}_{14} t_3 \theta_1, \\ m(t_1, t_2, t_3, t_4, \theta_1) = \hat{\beta}_0 &+ \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_4 + \hat{\beta}_5 \theta_1 + \hat{\beta}_6 t_2^2 + \hat{\beta}_7 t_3^2 + \hat{\beta}_8 t_4^2 + \hat{\beta}_9 \theta_1^2 \\ &+ \hat{\beta}_{10} t_1^3 + \hat{\beta}_{11} t_2^3 + \hat{\beta}_{12} t_3^3 + \hat{\beta}_{13} t_4^3 + \hat{\beta}_{15} t_1 t_2 + \hat{\beta}_{16} t_2 t_3 + \hat{\beta}_{17} t_3 t_4 \\ &+ \hat{\beta}_{18} t_2 \theta_1 + \hat{\beta}_{19} t_3 \theta_1 + \hat{\beta}_{20} t_4 \theta_1, \end{aligned}$$

having removed all non–significant terms. The associated R^2 values for these prior mean functions are 0.8429, 0.8775 and 0.9279 respectively, all of which show that the prior mean functions are a good fit. The marginal posterior distributions of the hyperparameters and associated diagnostics are given in Appendix A.1.2. Again, although standardised diagnostic plots indicate potential problems, the individual prediction errors are relatively small.

Figure 10.8 shows the marginal distribution of a design $\boldsymbol{d}$ with $J = 20$. Interestingly, for the three timepoint design in (b), the distribution of the third timepoint is bi–modal with peaks close to both $t_3 = 20$ and $t_3 = 8$. These then become the third and fourth timepoints in the four timepoint design. We estimate the multivariate mode using samples from the posterior distribution of $\boldsymbol{d}$ via the methods described in Section 10.3. The modes are given in Table 10.1. We note that the multivariate mode method usually produces a design with higher expected utilities than the trimmed mean and the marginal mode methods (apart from the four timepoint case). The optimal designs found using all methods are similar

Method	Optimal design $\boldsymbol{d}^*$	Expected utility $u(\boldsymbol{d})$
Multivariate mode	(4.72, 7.47)	2084.96
	(5.16, 8.37, 19.29)	2507.92
	(3.94, 6.26, 8.80, 19.4)	2956.55
Trimmed mean	(4.25, 7.63)	2077.15
	(4.50, 7.45, 18.25)	2385.94
	(3.83, 6.34, 9.42, 17.70)	2773.42
Marginal mode	(4.92, 7.26)	2075.73
	(4.67, 7.26, 19.43)	2478.29
	(3.74, 6.33, 8.74, 19.52)	2964.41
Gaussian process	(4.72, 7.70)	2085.41
	(5.18, 8.30, 19.35)	2518.08
	(3.79, 6.28, 8.79, 19.57)	2983.97

Table 10.1: Estimates of the optimal design using various techniques for one to four timepoints (see Section 10.3 for details). Multivariate mode uses Algorithm 19. Trimmed mean uses the mean of a 10% trim of the marginal samples of $\boldsymbol{d}$. Marginal mode uses the marginal modes of the marginal distributions of $\boldsymbol{d}$. Gaussian process uses the Gaussian process to evaluate a small range of designs around the optimal design from the other three methods with the highest expected utility. The expected utility is calculated using Equation 10.1 with the Gaussian process approximation to the utility.

# timepoints (d)	Optimal design ($\boldsymbol{d}^*$)	Estimated expected utility ($u(\boldsymbol{d}^*)$)
1	(6.13)	1402
2	(4.72, 7.38)	2085
3	(5.18, 8.30, 19.35)	2518
4	(3.79, 6.28, 8.79, 19.57)	2983

Table 10.2: Optimal designs for one to four timepoint designs with the estimated expected utility.

and their expected utilities differ by no more than 0.4%.

To ensure that we have indeed found the optimal designs, we have explored regions around the putative optimal designs and gauged their expected utility by direct evaluation of the fitted Gaussian processes. The optimal designs found using this direct local exploration can also be found in Table 10.1. We note that the designs with the highest expected utilities in this summary table are $\boldsymbol{d}^* = (4.72, 7.38)$, $\boldsymbol{d}^* = (5.18, 8.30, 19.35)$ and $\boldsymbol{d}^* = (3.79, 6.28, 8.79, 19.57)$ respectively. These optimal timepoints are mostly focused on the first prey cycle with, in the three and four timepoint designs, the final timepoint at the beginning of the next prey cycle.

Summary

Table 10.2 gives a summary of the results for one to four timepoint designs. As each additional timepoint is incorporated into the design, parameter inference becomes more accurate and so the expected utility increases. We could determine an optimal number of timepoints to use by including a cost in the utility function for each new design timepoint. The increase in expected utility from a single to a two timepoint design, a two to a three timepoint design and a three to a four timepoint design are 682.48, 432.67 and 465.89 respectively. It is interesting to see that the increase from two to three timepoints is smaller than three to four timepoints. This probably reflects the bimodality in the distribution of the third timepoint (for a three timepoint design) with this indecision being resolved when moving to a four timepoint deign in which the bimodal peaks are the locations of the third and fourth timepoints. Figure 10.9 shows the optimal designs as the vertical edges of the blocks of grey and white over the stochastic mean of the LV model with the parameters fixed at the prior mean. This graph shows that the timepoints in a two timepoint optimal design are either side of the one timepoint optimal design. This is also the case for the first two timepoints of the three timepoint design and the third optimal point is at the start of the next cycle of the prey. For the four timepoint optimal design, the second optimal point is very close to the single timepoint optimal design, the first and third timepoints are either side of the single timepoint optimal design and the forth timepoint is at the start of the next cycle. It appears that the optimal designs focus on the prey cycle which is perhaps not surprising as the utility function depends on knowledge of θ_1, the rate governing prey reproduction, and not on predator reproduction or death (as these rates are known).

10.5 Fully optimal design

We now consider the case where all three rate constants are unknown and the prior distribution has independent log–normal components, with

$$\theta_1 \sim LN(-0.69, 0.01), \qquad \theta_2 \sim LN(-6, 0.01) \qquad \text{and} \qquad \theta_3 \sim LN(-1.21, 0.01),$$

and prior mean rate $E(\boldsymbol{\theta}) = (0.5, 0.0025, 0.3)'$. Note that now we have three unknown parameters, the capacity to learn about parameters is increased and will probably lead to large changes in the expected utility over choices in design and number of timepoints.

10.5.1 Design space reduction

We now use the delta approximation, in which $\boldsymbol{\theta}$ is fixed at its prior mean, and estimate the expected utility using the average realisation $\boldsymbol{y}^*_{\boldsymbol{d}}$. We then fit a Gaussian process to this (approximate) expected utility and then reduce the design space to the product of the

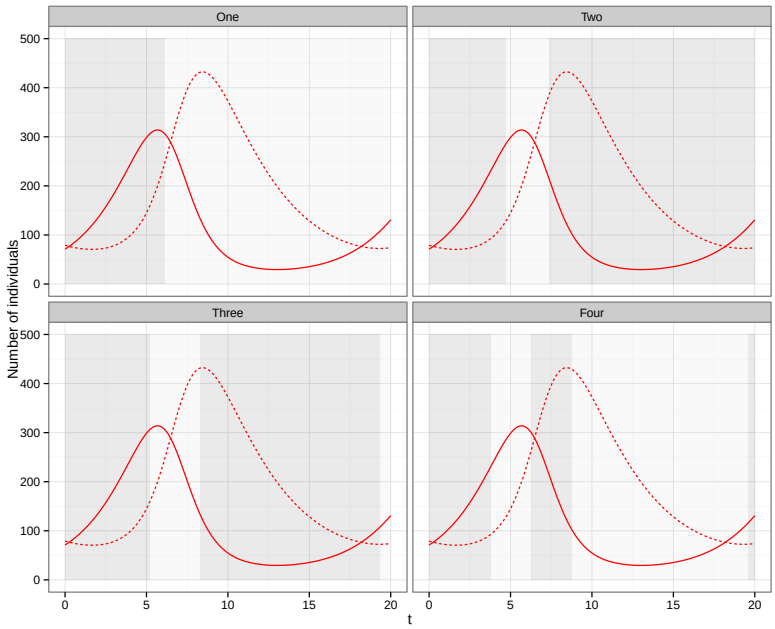


Figure 10.9: The red solid and dashed lines are the stochastic mean of the prey and predators, respectively. The parameters are fixed at their prior means ($E(\theta_1) = 0.5$, $E(\theta_2) = 0.0025$ and $E(\theta_3) = 0.3$). The edges of the blocks of grey and white represent the optimal designs for when θ_1 is unknown for the one, two, three and four timepoint designs.

central 95% posterior intervals for the timepoints.

Single timepoint design

Here the design is univariate ($\boldsymbol{d} = t_1$) and so we use a regular grid $t_1 = 0, 0.2, 0.4, \ldots, 20$ for the training data inputs. A Gaussian process is fitted to the training data which has mean function

$$m(t_1) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_1^2 + \hat{\beta}_3 t_1^3,$$

and squared exponential covariance function (as in Equation (9.8)); the Gaussian process has three hyperparameters $(a, r_1, \sigma)'$.

The marginal distributions of the hyperparameters are given in Figure 10.10. The posterior density of $\log r_1$ is negatively skewed and the other two are fairly symmetric. The Gaussian process diagnostics are given in Figure 10.11. These plots and that in Figure 10.12(a) again indicate a problem of fit around the peak expected utility. However the Gaussian process does fit the data pretty well as almost all of the points lie between the upper and lower 95% limits. Figure 10.12(b) gives the estimated marginal distribution of the design (t_1) for $J = 1$, 5 and 20. We again see the focusing effect around the mode of t_1 when J is increased. For $J = 20$ this results in truncated training data ranges of $6.16 < t_1 < 7.91$.

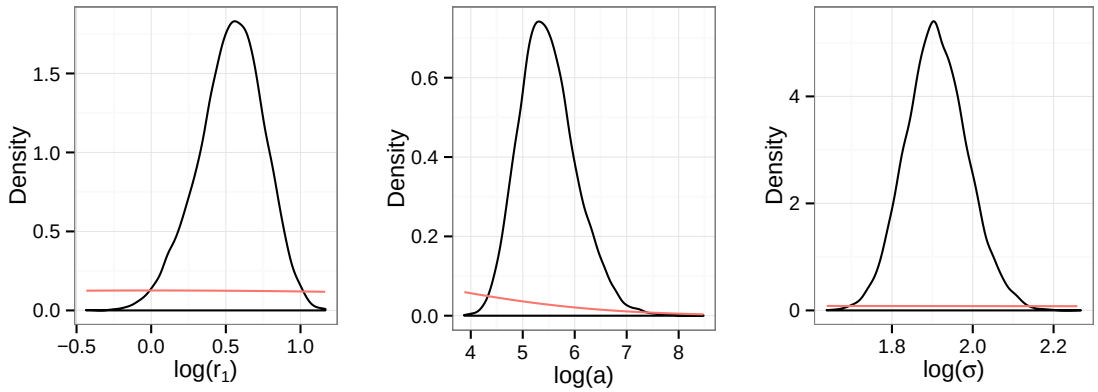


Figure 10.10: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a single timepoint design. Prior distributions are given in red.

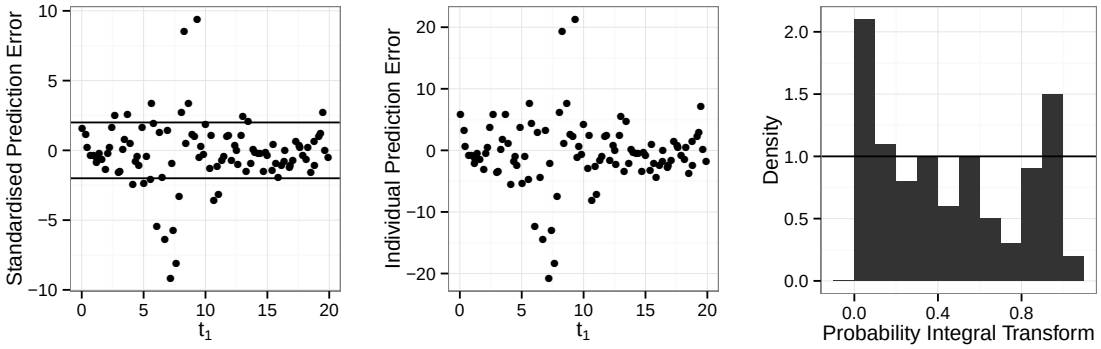


Figure 10.11: Diagnostics for the Gaussian process used to reduce the design space for the single timepoint design.

Multiple timepoints

The training data used for fitting a Gaussian process to designs with two to four timepoint is chosen using the fold method with $n_d = 200$, 300 and 400 respectively. The prior mean functions used are

$$\begin{aligned} m(t_1, t_2) &= \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_1^2 + \hat{\beta}_4 t_2^2 + \hat{\beta}_5 t_1^3 + \hat{\beta}_6 t_1 t_2, \\ m(t_1, t_2, t_3) &= \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_1^2 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 t_3^2 + \hat{\beta}_7 t_1^3 + \hat{\beta}_8 t_2^3 \\ &\quad + \hat{\beta}_9 t_3^3 + \hat{\beta}_{10} t_1 t_2 + \hat{\beta}_{11} t_2 t_3, \\ m(t_1, t_2, t_3, t_4) &= \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 t_2 + \hat{\beta}_3 t_3 + \hat{\beta}_4 t_4 + \hat{\beta}_5 t_1^2 + \hat{\beta}_6 t_2^2 + \hat{\beta}_7 t_3^2 + \hat{\beta}_8 t_4^2 + \hat{\beta}_9 t_2^3 \\ &\quad + \hat{\beta}_{10} t_3^3 + \hat{\beta}_{11} t_4^3 + \hat{\beta}_{12} t_1 t_2 + \hat{\beta}_{13} t_2 t_3 + \hat{\beta}_{14} t_3 t_4 \end{aligned}$$

and we use the squared exponential covariance function in Equation (9.8) with $\boldsymbol{x} = (t_1, \ldots, t_d)'$ where $d = 2$, 3 and 4 respectively.

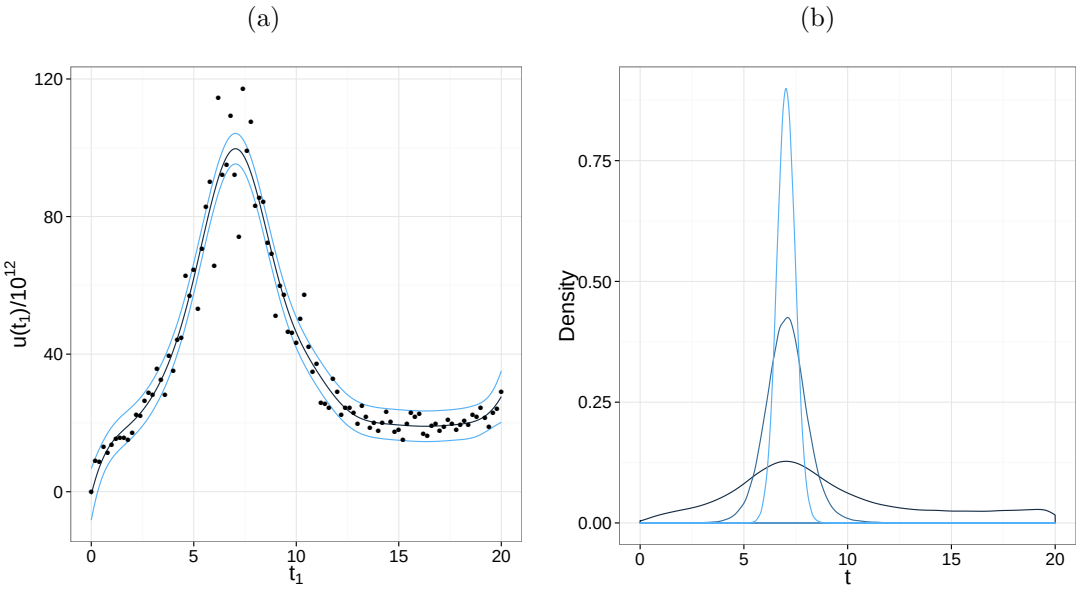


Figure 10.12: Figure (a) Gaussian process mean function ± 1.96sd (blue lines) with the training data (black points). Note that utilities have been divided by 10^{12}. (b) the marginal distribution of t_1 with $J = 1$ (dark blue line), $J = 5$ (medium blue line) and $J = 20$ (light blue line).

The marginal posterior distributions for the hyperparameters and diagnostic plots are given in Appendix A.2.1. Although the standardised errors are problematic, the individual prediction errors are small, and so we proceed with inference based on these fitted Gaussian processes.

Figure 10.13 shows the marginal distribution of $\boldsymbol{d}$, for $J = 1$, 5 and 20. Note that the marginal modes changes as J is increased as the marginal modes are not equivalent to the multivariate mode. However as J is increased the marginal mode becomes a better approximation to the multivariate mode. We now reduce the design space to the (folded) cubes formed by the 95% univariate interval for the timepoints, namely

2–dim design: $5.70 < t_1 < 7.63$ $10.03 < t_2 < 14.78$
3–dim design: $3.89 < t_1 < 6.58$ $7.22 < t_2 < 10.48$ $11.63 < t_3 < 18.98$
4–dim design: $2.59 < t_1 < 6.03$ $5.83 < t_2 < 8.77$ $8.42 < t_3 < 13.28$ $14.36 < t_4 < 19.81$.

10.5.2 Optimal designs using a Gaussian process fitted to $u(d, \theta)$ in the reduced space

Single timepoint design

The analysis using the delta approximation in section 10.5.1 has reduced the design space to $6.16 < t_1 < 7.91$. We also restrict the training data in the parameter space to the cube

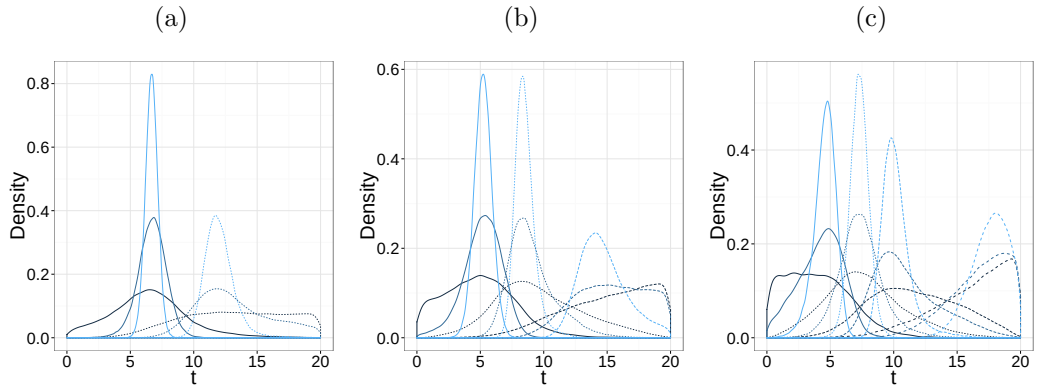


Figure 10.13: Marginal distributions of $\boldsymbol{d}$ (with $J = 1$,5 and 20) for the (a) two, (b) three , and (c) four timepoint design. The first to fourth timepoints are shown as the solid, small dashed, medium dashed and large dashed lines, respectively.

defined by the central univariate 99% prior intervals, that is

$$0.3877 < \theta_1 < 0.6489, \quad 0.0019 < \theta_2 < 0.0032 \quad \text{and} \quad 0.2305 < \theta_3 < 0.385\,.$$

A Gaussian process, which has inputs $\boldsymbol{x} = (t_1, \theta_1, \theta_2, \theta_3)'$, is then fitted to training data generated from a maximin Latin hypercube over this space. We use the mean function

$$m(t_1, \theta_1, \theta_2, \theta_3) = \hat{\beta}_0 + \hat{\beta}_1 t_1 + \hat{\beta}_2 \theta_1 + \hat{\beta}_3 \theta_2 + \hat{\beta}_4 \theta_3 + \hat{\beta}_5 t_1^2 + \hat{\beta}_6 \theta_1^2 + \hat{\beta}_7 \theta_2^2 + \hat{\beta}_8 t_1^3 + \hat{\beta}_9 t_1 \theta_1 + \hat{\beta}_{10} t_1 \theta_2 + \hat{\beta}_{11} t_1 \theta_3 + \hat{\beta}_{12} \theta_1 \theta_2 + \hat{\beta}_{13} \theta_2 \theta_3 + \hat{\beta}_{14} \theta_1 \theta_3$$

and the squared exponential covariance function in Equation (9.8), which has six hyperparameters $(a, r_1, r_2, r_3, r_4, \sigma)'$. We note that this prior mean function is an excellent fit to the data ($R^2 = 0.9239$). The hyperparameter marginal posterior distributions are given in Figure 10.14 and the diagnostics are given in Figure 10.15. Overall, the marginal posterior distributions are symmetric and uni-modal and the individual prediction errors are small. Figure 10.16 gives the marginal distribution of $\boldsymbol{d}$ for $J = 20$. From this we estimate the optimal design to be $\boldsymbol{d}^* = (7.431)$, with an estimated expected utility of 9.8458×10^{13}. We note that this marginal mode is a higher value of t_1 compared to the optimal design when only one parameter (θ_1) is unknown. We discuss this point further in Section 10.5.2.

Multiple timepoints

We fit Gaussian processes, with inputs $\boldsymbol{x} = (t_1, \ldots, t_d, \theta_1, \theta_2, \theta_3)'$, where $d = 2$, 3 and 4 respectively, for each of two to four timepoint designs to training data generated using the fold method within the reduced design space determined via the delta approximation in

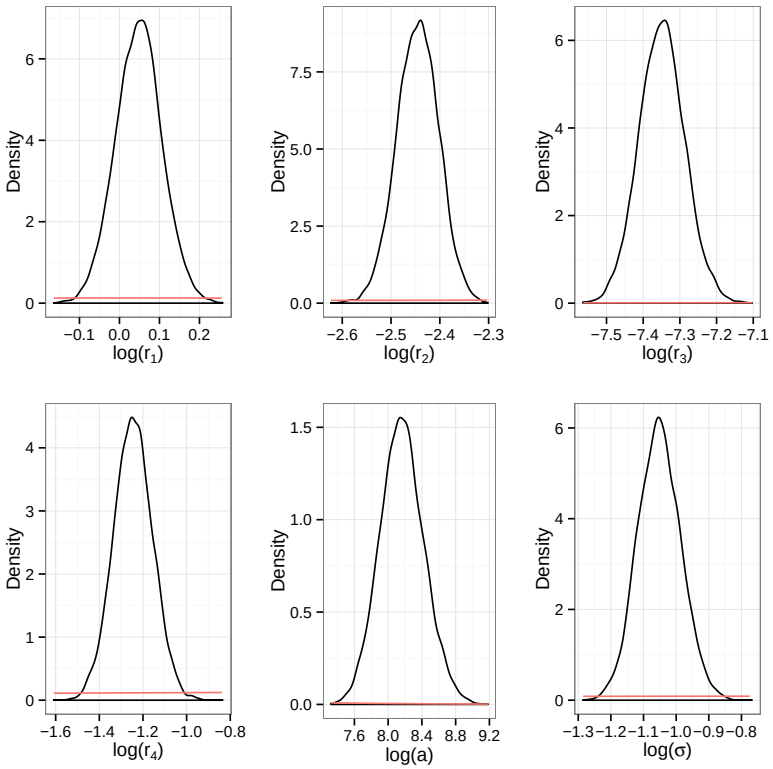


Figure 10.14: Marginal posterior distributions for the hyperparameters for a single timepoint design. Prior distributions are given in red.

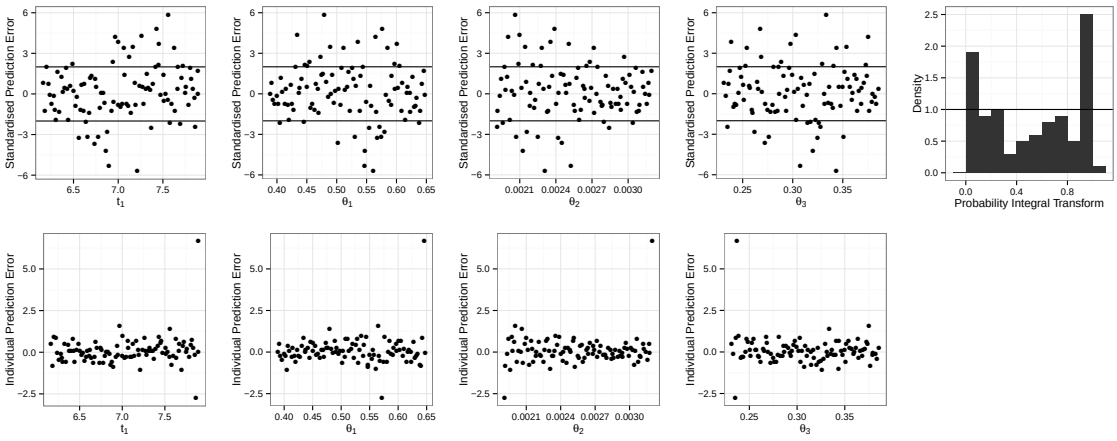


Figure 10.15: Gaussian process diagnostics for the single timepoint design.

Section 10.5.1. We use prior mean functions

$$\begin{aligned} m(t_1, t_2, \theta_1, \theta_2, \theta_3) = \hat{\beta}_0 t_1 + \hat{\beta}_1 t_2 + \hat{\beta}_2 \theta_1 + \hat{\beta}_3 \theta_2 + \hat{\beta}_4 \theta_3 + \hat{\beta}_5 t_2^2 + \hat{\beta}_6 \theta_1^2 + \hat{\beta}_7 \theta_2^2 \\ + \hat{\beta}_8 \theta_3^2 + \hat{\beta}_9 t_1^3 + \hat{\beta}_{10} \theta_1^3 + \hat{\beta}_{11} \theta_2^3 + \hat{\beta}_{12} \theta_3^3 + \hat{\beta}_{13} t_1 \theta_1 + \hat{\beta}_{14} t_2 \theta_2 \\ + \hat{\beta}_{15} t_2 \theta_3 + \hat{\beta}_{16} t_1 t_2 + \hat{\beta}_{17} t_1 \theta_2 + \hat{\beta}_{18} \theta_2 \theta_3 + \hat{\beta}_{19} \theta_1 \theta_3 \end{aligned}$$

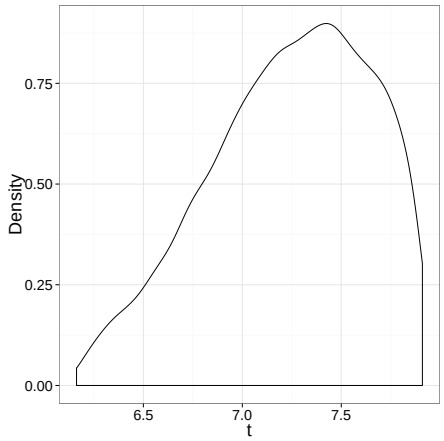


Figure 10.16: Marginal distribution of t_1 with $J = 20$.

$$\begin{aligned} m(t_1, t_2, t_3, \theta_1, \theta_2, \theta_3) = \hat{\beta}_0 t_1 + \hat{\beta}_1 t_2 + \hat{\beta}_2 t_3 + \hat{\beta}_3 \theta_1 + \hat{\beta}_4 \theta_2 + \hat{\beta}_5 \theta_3 + \hat{\beta}_6 t_2^2 + \hat{\beta}_7 t_3^2 \\ + \hat{\beta}_8 \theta_1^2 + \hat{\beta}_9 t_2^2 + \hat{\beta}_{10} \theta_3^2 + \hat{\beta}_{11} t_1^3 + \hat{\beta}_{12} \theta_1^3 + \hat{\beta}_{13} \theta_2^3 + \hat{\beta}_{14} \theta_3^3 \\ + \hat{\beta}_{15} t_1 \theta_3 + \hat{\beta}_{16} t_2 \theta_1 + \hat{\beta}_{17} t_2 \theta_2 + \hat{\beta}_{18} t_1 t_2 + \hat{\beta}_{19} t_2 t_3 \\ + \hat{\beta}_{20} \theta_1 \theta_2 + \hat{\beta}_{21} \theta_1 \theta_3 \end{aligned}$$

$$\begin{aligned} m(t_1, t_2, t_3, t_4, \theta_1, \theta_2, \theta_3) = \hat{\beta}_0 t_1 + \hat{\beta}_1 t_2 + \hat{\beta}_2 t_3 + \hat{\beta}_3 t_4 + \hat{\beta}_4 \theta_1 + \hat{\beta}_5 \theta_2 + \hat{\beta}_6 \theta_3 + \hat{\beta}_7 t_2^2 \\ + \hat{\beta}_8 t_3^2 + \hat{\beta}_9 \theta_2^2 + \hat{\beta}_{10} t_1^3 + \hat{\beta}_{11} \theta_2^3 + \hat{\beta}_{12} t_1 \theta_1 + \hat{\beta}_{13} t_1 \theta_2 + \hat{\beta}_{14} t_2 \theta_1 \\ + \hat{\beta}_{15} t_4 \theta_1 + \hat{\beta}_{16} t_1 t_2 + \hat{\beta}_{17} t_2 t_3 + \hat{\beta}_{18} t_3 t_4 + \hat{\beta}_{19} \theta_1 \theta_2 \\ + \hat{\beta}_{20} \theta_2 \theta_3 + \hat{\beta}_{21} \theta_1 \theta_3 \end{aligned}$$

and the squared exponential covariance function in Equation (9.8).

Again, the prior mean functions provide an excellent fit to the data, with $R^2 = 0.9450$, 0.9629 and 0.9478, for $d = 2, 3, 4$ respectively. The Gaussian process diagnostics (given in Appendix A.2.2) show that although the standardised prediction errors appear problematic, the individual prediction errors are relatively small. The univariate marginal distributions of the design $\boldsymbol{d}$, for $J = 1$, 5 and 20, is given in Figure 10.17. As before, we estimated the mode of the distribution using four different techniques; the results given in Table 10.3. Each of the methods yields similar designs. Comparing Figures 10.18 and 10.9, it is interesting to note that the timepoints in these optimal designs tend to occur later than those determined when θ_2 and θ_3 are fixed at their prior mean values.

Summary of the optimal designs

A summary of the fully optimal one to four timepoint designs is given in Table 10.4. Designs with more timepoints have higher expected utilities, since observing the process at additional timepoints reduces parameter uncertainty and so the expected utility $u(\boldsymbol{d})$ increases. The

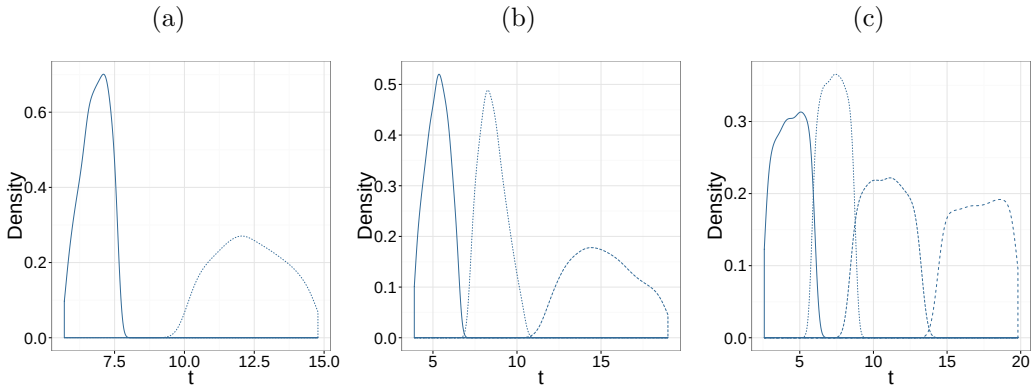


Figure 10.17: Marginal distributions of $\boldsymbol{d}$ with $J = 1, 5$ and 20. Plot (a) is for the two timepoint design, (b) is for the three timepoint design, (c) is for the four timepoint design. The first timepoints within a design are the solid lines, the second timepoints are the small dashed lines, the third time points are the medium dashed lines and the forth timepoints are the large dashed lines.

Method	Optimal design ($\boldsymbol{d}^*$)	Estimated expected utility ($u(\boldsymbol{d})$)
Multivariate mode	(6.86, 12.13)	5.7382×10^{14}
	(5.23, 8.40, 14.00)	1.2482×10^{15}
	(4.54, 7.25, 11.13, 17.69)	1.8440×10^{15}
Trimmed mean	(6.82, 12.34)	5.7311×10^{14}
	(5.28, 8.51, 15.10)	1.2485×10^{15}
	(4.38, 7.31, 10.83, 17.15)	1.8674×10^{15}
Marginal mode	(7.10, 12.07)	5.7397×10^{14}
	(5.31, 8.22, 14.16)	1.2623×10^{15}
	(5.06, 7.33, 11.04, 18.89)	1.8255×10^{15}
Gaussian process	(6.96,12.23)	5.7527×10^{14}
	(5.20, 8.12, 14.07)	1.2694×10^{15}
	(5.04, 7.35, 11.02, 19.90)	1.9033×10^{15}

Table 10.3: Estimates of the optimal design using various techniques for one to four timepoints. These techniques are discussed in Section 10.3. Multivariate mode uses Algorithm 19. Trimmed mean uses the mean of a 10% trim of the marginal samples of $\boldsymbol{d}$. Marginal mode uses the marginal modes of the marginal distributions of $\boldsymbol{d}$. Gaussian process uses the Gaussian process to evaluate a small range of designs around the multivariate mode from the other three methods (trimmed mean, marginal mode, multivariate mode) with the highest expected utility. The expected utility is calculated using the Gaussian process as shown in Equation 10.1.

largest increase in expected utility is from two to three timepoints. Figure 10.18 shows the location of the optimal designs – these are indicated by the change between grey and white blocks – and the stochastic mean of the LV model. All two to four timepoint optimal designs have one of their timepoints near the single timepoint optimal design, which is just before the predator peak. The second timepoint in the two timepoint optimal design is after the predator peak. The three timepoint design has its first and third timepoints

# timepoints (d)	Optimal design ($\boldsymbol{d}^*$)	Estimated expected utility ($u(\boldsymbol{d}^*)$)
1	(7.43)	9.8458×10^{13}
2	(6.96, 12.23)	5.7527×10^{14}
3	(5.20, 8.12, 14.07)	1.2694×10^{15}
4	(5.04, 7.35, 11.02, 19.90)	1.9033×10^{15}

Table 10.4: Optimal designs for one to four timepoint designs with the estimated expected utility.

either side of the predator peak and the four timepoint design is similar but has the fourth timepoint near the very end of the time interval (near the start of the next cycle).

It is interesting to see the differences between the optimal designs determined using all prior uncertainty (as in this section) and when fixing θ_2 and θ_3 at their prior mean values. This comparison involves comparing Figure 10.9 and Figure 10.18. For the single timepoint design in Figure 10.9 the optimal design is just after the peak of the prey curve, whereas in Figure 10.18 the optimal design is after the prey peak and just before the predator peak. This is not surprising because the unknown parameter in Figure 10.9 is θ_1 which represents prey reproduction and so we expect the optimal designs to be at timepoints where there are prey numbers change by larger amounts and therefore focussed around the prey cycle.

Looking at the optimal timepoints in the two cases studied we see that in general when all three parameters are unknown, the optimal timepoints take higher values. That said, the first and second timepoints in the three timepoint design are roughly in the same position. Overall, it is clear that increasing overall parameter uncertainty (by not fixing θ_2 and θ_3) leads to optimal designs which balance observations more evenly around the predator and prey cycles.

Table 10.4 shows as each additional timepoint is incorporated into the design, parameter inference becomes more accurate and so the expected utility increases. We could determine an optimal number of timepoints to use by including a cost in the utility function for each new design timepoint. The increase in expected utility from a single to a two timepoint design, a two to a three timepoint design and a three to a four timepoint design are 4.7681×10^{14}, 6.9313×10^{14} and 6.3490×10^{14} respectively. The increase from two to three timepoints is the largest. This probably reflects that the three timepoint optimal design can provide better coverage of the predator and prey cycles providing much more information about $\boldsymbol{\theta}$ than the two timepoint design. It is interesting to see the considerable change in the scale of the difference of the expected utility between adding additional timepoints, compared to when only the prey reproduction rate (θ_1) is unknown. This is due to the large difference in prior information between these scenarios and hence the very large benefit in taking additional observations when all three rates are unknown.

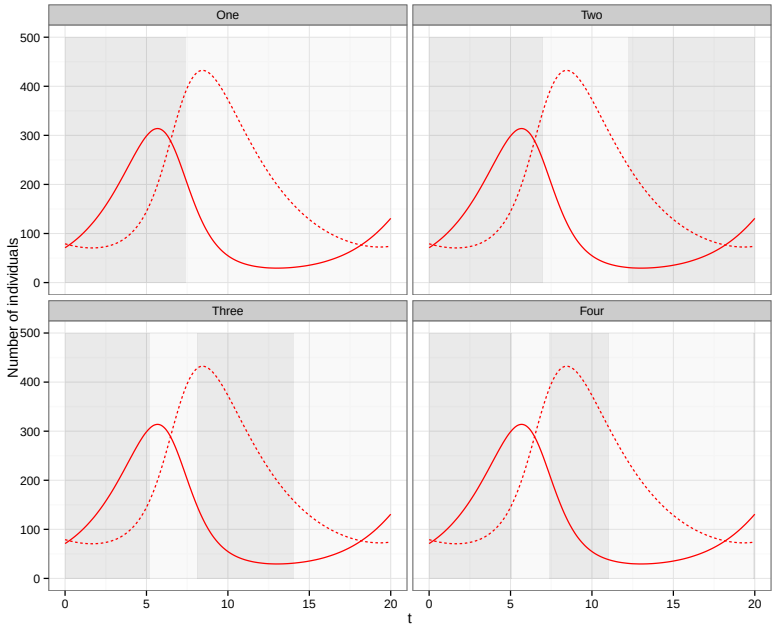


Figure 10.18: The red solid line is the stochastic mean of the prey and the red dashed lines represent the stochastic mean of the predators when the parameters are fixed at their prior means ($E(\theta_1) = 0.5$, $E(\theta_2) = 0.0025$ and $E(\theta_3) = 0.3$). The edges of the blocks of grey and white represent the optimal designs when $\boldsymbol{\theta}$ is unknown for the one, two, three and four timepoint designs.

Chapter 11

Conclusions and future work

11.1 Conclusion

In this second part of the thesis we have shown for that Gaussian processes can be used in Bayesian experimental design to provide a fast and accurate approximation of the expected utility and thereby determine accurate optimal designs. We showed that optimal designs determined using the Gaussian process approximation of the expected utility were close to the exact optimal designs for the pure death model. We then described how to use Gaussian processes in Bayesian experimental design for models with intractable likelihoods and illustrated this using the Lotka–Volterra model.

Using a Gaussian process has the advantage that the utility only needs to be evaluated at a certain number of inputs for the training data. This has the advantage that we no longer have to perform parameter inference at every iteration of an MCMC scheme in order to evaluate the utility (as we can use the Gaussian process instead). A further advantage of using a Gaussian process is that all training data is calculated in advance and, in particular, can be calculated in parallel, which can save considerable computing time. Our analysis took advantage of the HT-condor system to evaluate the training data for the Gaussian processes in parallel.

Drovandi and Pettitt (2013) reduce the time taken to perform ABC at each iteration of the Müller (1999) MCMC algorithm by pre–computing $200k$ simulations from the stochastic kinetic models. However these simulations must be stored, and this creates memory issues as model complexity increases. Clearly their optimal designs depend on these pre–computed datasets. We showed that their method can be quite inaccurate in approximating the utility for a particular (unobserved) dataset $\boldsymbol{y}$ as they do not require any absolute measure of closeness of $\boldsymbol{y}$ to the pre-computed dataset (they just use those that are closest). In contrast, using the Gaussian process approximations to the utility does not have this particular memory problem. However, there can be problems if they are

constructed using a very large number of training points due to the need to invert large square matrices.

A recurring problem when fitting the Gaussian process to the utility function was that the diagnostics indicated a poor fit. While the individual prediction errors were very small, the standardised errors suggested an issue with the assumption of constant variance. Future work could investigate more appropriate covariance functions that would ameliorate the diagnostics.

Estimating the hyperparameters of the Gaussian process can be a time consuming process as evaluation of the likelihood involves an inversion of a $n_d \times n_d$ matrix. Therefore, choosing the number of training points is a balance between computing time and accuracy of the Gaussian process. We have considered designs with up to four timepoints and so we were able to construct Gaussian processes using $100n_p$ training points, where n_p is the dimension of the Gaussian process inputs. This led to very accurate Gaussian process predictions. Unlike the ABC method to determine optimal designs, using Gaussian processes scales well to more complex problems. However, for large dimensional designs, to increase the speed of parameter estimation, some accuracy of the Gaussian process would need to be sacrificed.

It is interesting to measure the efficiency of the ABC method for determining the optimal design with that of our Gaussian process method. The following calculations refer to the analysis of the pure death model. For the Drovandi and Pettitt (2013) ABC method with $J = 20$, running the MCMC scheme for $100k$ iterations took 28.7, 39.8, 51.4 and 57.1 hours for the one to four timepoint designs respectively. However, the time taken by our MCMC was less (per iteration) and in some cases much less. Most of the time was spent fitting the Gaussian process hyperparameters: running the MCMC scheme for $500k$ iterations took 5.8, 14.6, 32.2 and 52.7 hours respectively. Some additional was spent calculating the training data and then finding the optimal design using the Gaussian process approximation. However, this time was very small (less than two minutes) as we use numerical integration to calculate the utility and take advantage in evaluating the Gaussian process in parallel. We note that our Gaussian process method appears to scale roughly as badly as the ABC method, with the ratio of the ABC and GP times being 4.9, 2.7, 1.6, 1.08 for designs with one to four timepoints. However, in practise the timings for the ABC method should be much larger as, if they were to provide the same accuracy in the optimal design as our Gaussian process method, they would need to use a much larger number of pre-simulated datasets. Further, should the Gaussian process timings be prohibitive, we could decide to determine our optimal design by using a less accurate fitted process determined via fewer points in our training data.

Numerical instabilities can occur when inverting the covariance matrix of the Gaussian process. However, this well known problem can be addressed by introducing a nugget term

into the matrix (along its diagonal); see, for example, Ababou et al. (1994) and Gramacy and Lee (2012). In our methods we attempt to alleviate this problem by working with the Cholesky decomposition of the matrix and solving a set of linear equations. Adopting this solution is also more efficient: it halves the computation time of a Gaussian process prediction. We also encountered problems when the training data outputs were large, for example, when then the utilities were $O(10^{12})$. The problem concerned the evaluation of the quadratic form in the log-likelihood when inferring the hyperparameters. Here evaluating a quadratic form with very large values in the vector but very small values in the inverse matrix induced numerical inaccuracies. However we found that scaling the utility function so that it is $O(1)$ reduced this problem.

11.2 Future work

One possible extension of the experimental design methodology is to consider designs in which not all species are observed. For example, in the LV model, we could consider designs in which only the prey are observed (and not the predators) at some or all timepoints. Calculation of the utility function is still possible as we can adapt our LNA analysis to take account of unobserved species at particular timepoints. In more complex scenarios/models it might be that the design needs to take account of the fact that certain species cannot be observed experimentally or cannot be observed at particular times.

In this work we have considered a simple utility function. However, we could consider other utility functions and examine whether the choice of optimal design is sensitive to the choice of utility function. One alternative utility function is based on the Kullbeck–Leibler divergence; this is particularly appropriate when the aim of the experiment is to perform parameter inference. We could also include more sophisticated costs in the utility function so that, for example, measuring the system at later timepoints is more expensive.

It would be interesting to see the sensitivity of the optimal design to having a well fitting Gaussian process approximation. In our current analyses, the Gaussian processes have not been a particularly good fit, with the main discrepancy appearing to be due to non-constant noise in the training data output. Another aspect might be to consider a process approximation which has heavier tails than the Gaussian process, such as a Student–t process. Such a process has the benefit of reduce the influence of outliers and improve predictions (Jylnki et al., 2011). However a disadvantage of this approach is that the posterior distribution for the hyperparameters is intractable. That said, a Laplace approximation (Lindley, 1980) or data augmentation can be used to fit the process; here data augmentation consists of rewriting the Student–t process as a continuous (inverse χ^2) mixture of Gaussian processes. MCMC techniques for the Gaussian process model can be extended to include auxiliary variables and thereby obtain posterior samples of the

hyperparameters from the Student–t process fit.

Finally, and quite importantly, the optimal design should be examined to determine its sensitivity to the chosen prior distribution. Hopefully the optimal design will not be too sensitive to the prior because, if it is, then much more effort is needed to ensure that the prior distribution used really does quantify true prior beliefs about the parameters. We have not studied this aspect in this thesis due to the considerable computational cost of examining such sensitivities.

Appendix A

Appendix

A.1 Optimal design for one unknown parameter

A.1.1 Design space reduction

Figures A.1, A.3 and A.5 are the marginal posterior distributions for the hyperparameters for the Gaussian processes fitted to the delta approximation to the expected utility for the two, three and four timepoint designs which are referred to in Subsection 10.4.1. Figures A.2, A.4 and A.6 are the associated diagnostic plots.

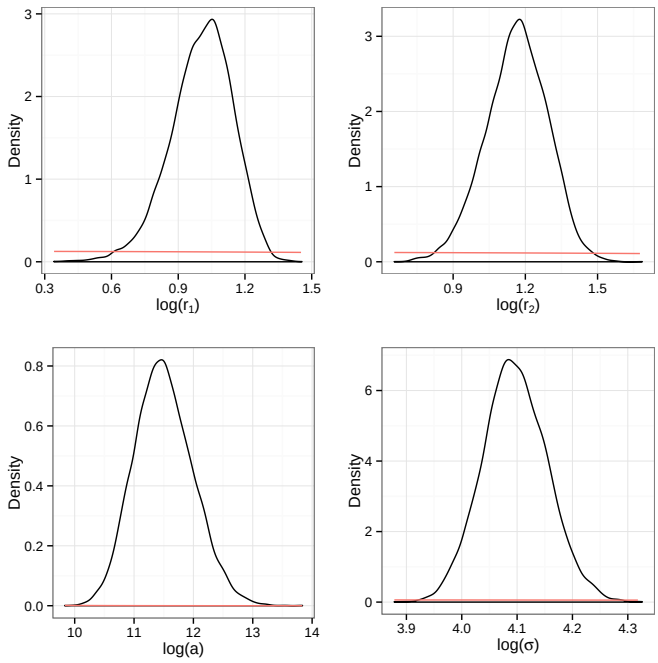


Figure A.1: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a two timepoint design.

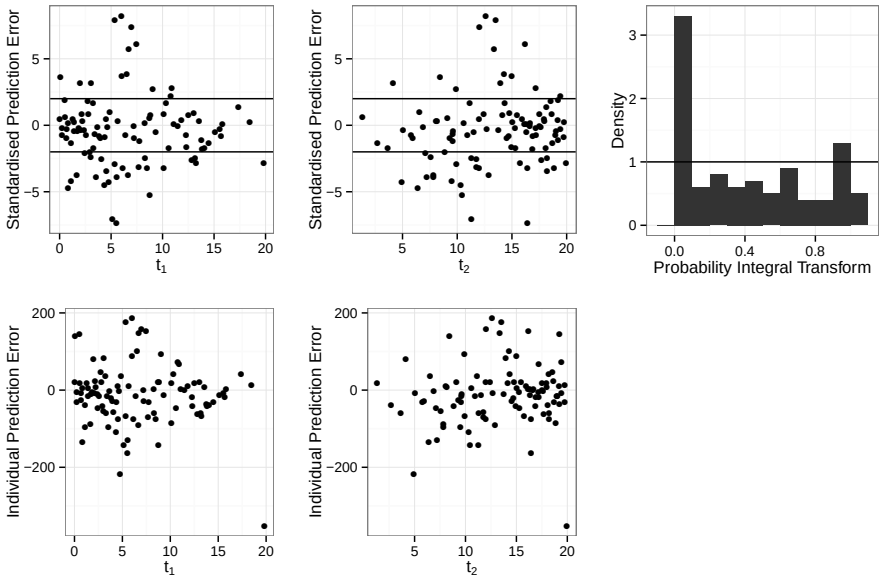


Figure A.2: Diagnostics for the Gaussian process used to reduce the design space for the two timepoint design.

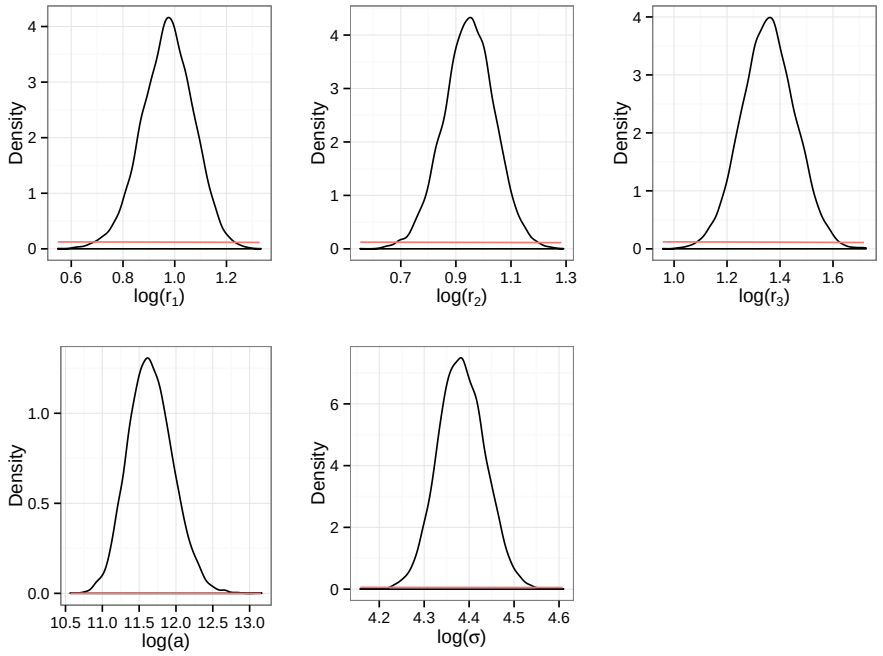


Figure A.3: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a three timepoint design.

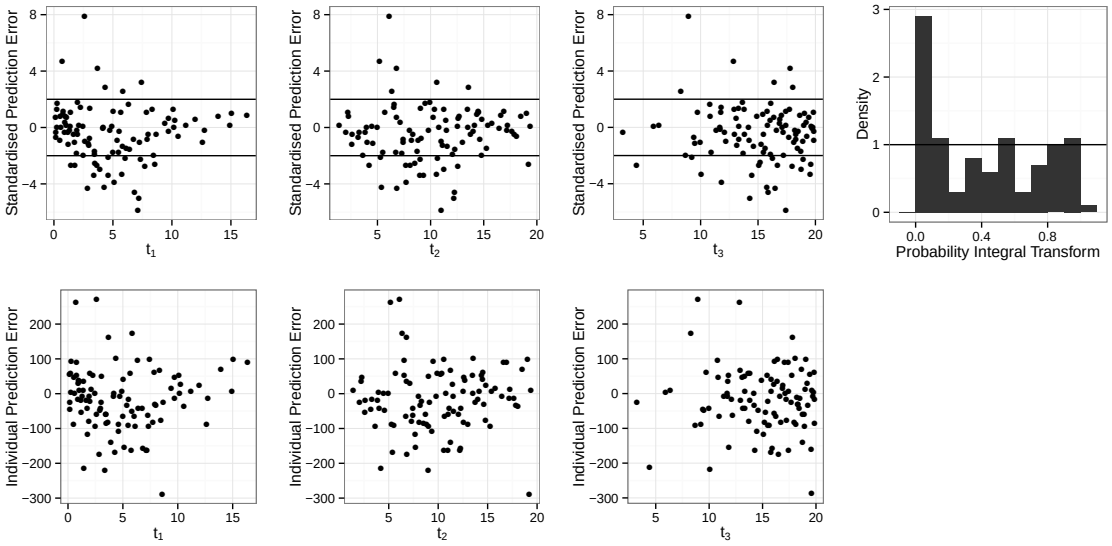


Figure A.4: Diagnostics for the Gaussian process used to reduce the design space for the three timepoint design.

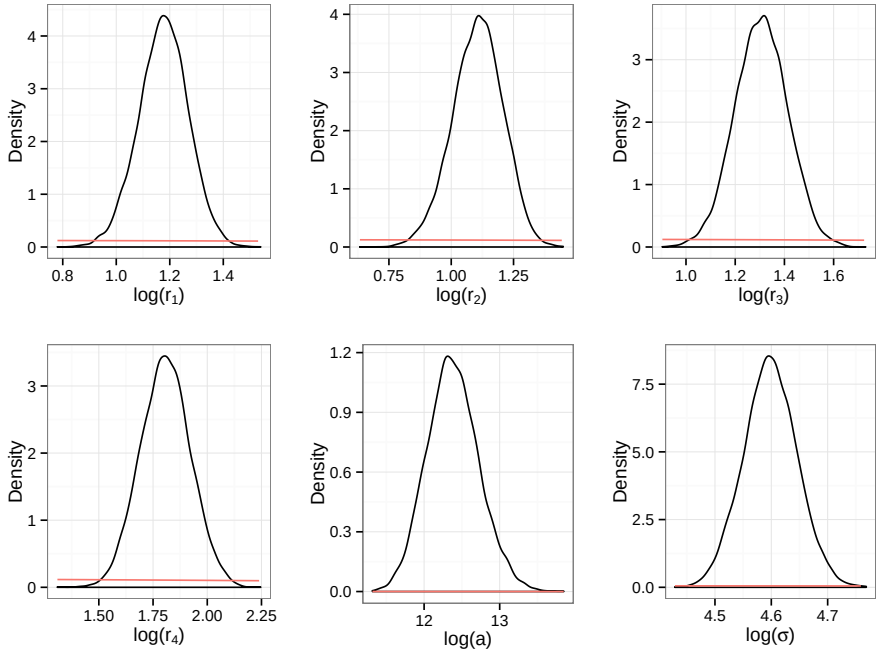


Figure A.5: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a four timepoint design.

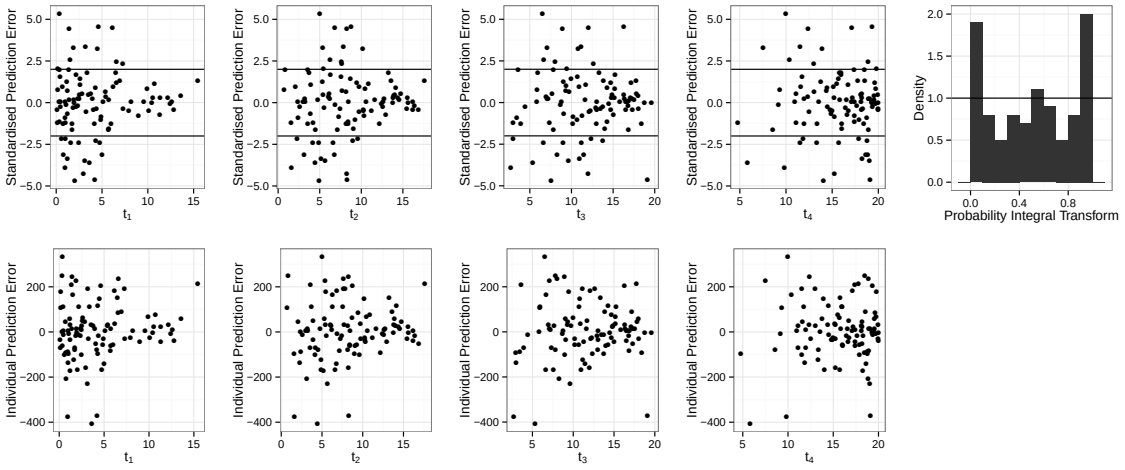


Figure A.6: Diagnostics for the Gaussian process used to reduce the design space for the four timepoint design.

A.1.2 Gaussian process fitted to $u(d, \theta_1)$ using the reduced space

Figures A.7, A.9 and A.11 are the marginal posterior distributions for the hyperparameters for the Gaussian processes fitted to $u(\boldsymbol{d}, \theta_1)$ for the two, three and four timepoint designs which are referred to in Subsection 10.4.2. Figures A.8, A.10 and A.12 are the associated diagnostic plots.

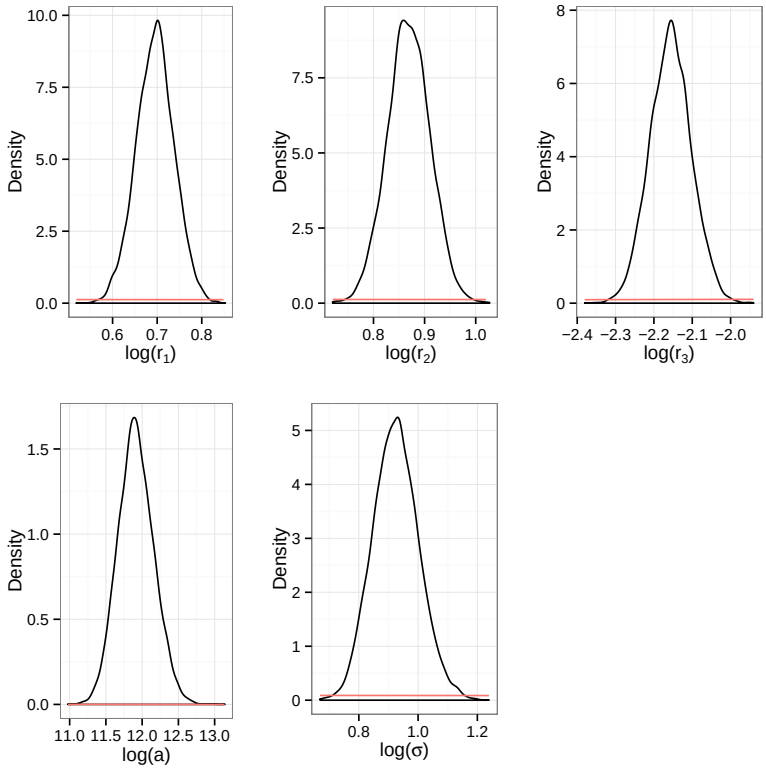


Figure A.7: Marginal posterior distributions for the hyperparameters for the two timepoint design Gaussian process.

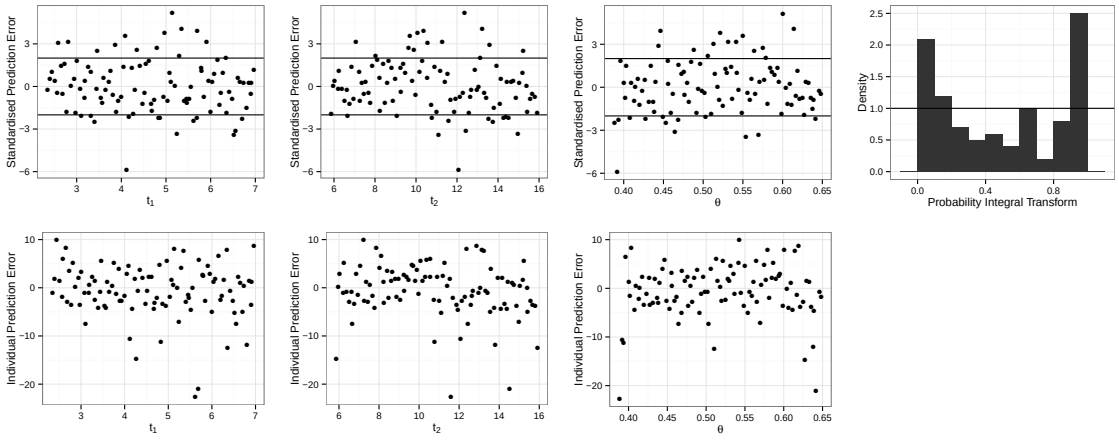


Figure A.8: Diagnostics for the two timepoint design Gaussian process.

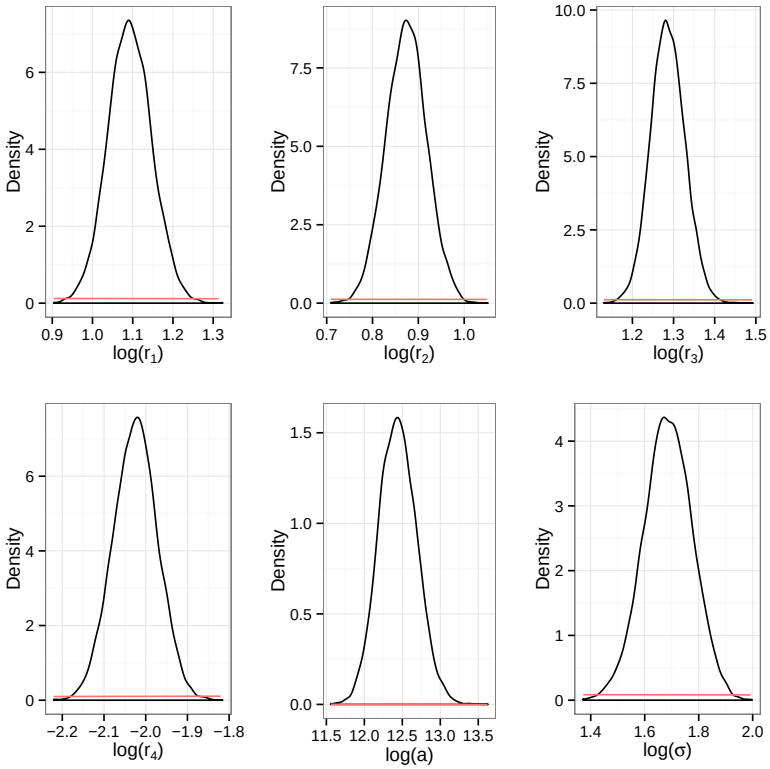


Figure A.9: Marginal posterior distributions for the hyperparameters for the three timepoint design Gaussian process.

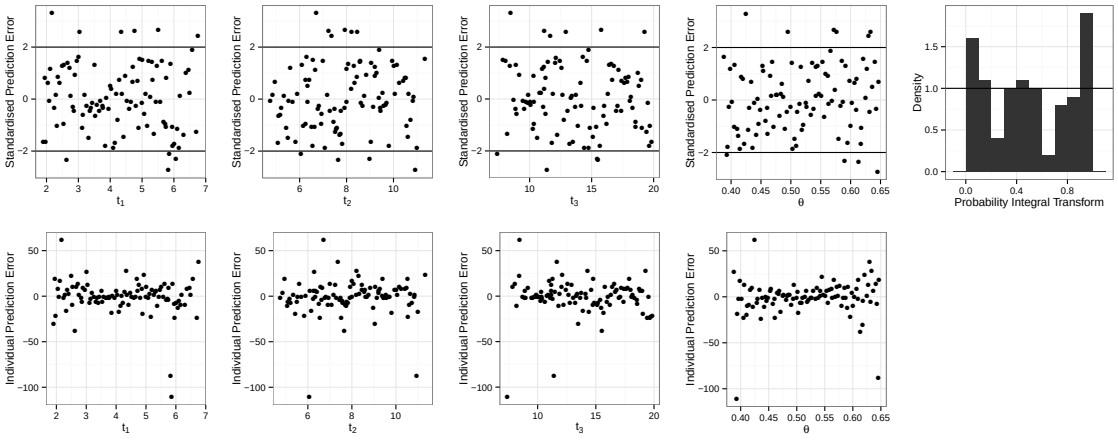


Figure A.10: Diagnostics for the three timepoint design Gaussian process.

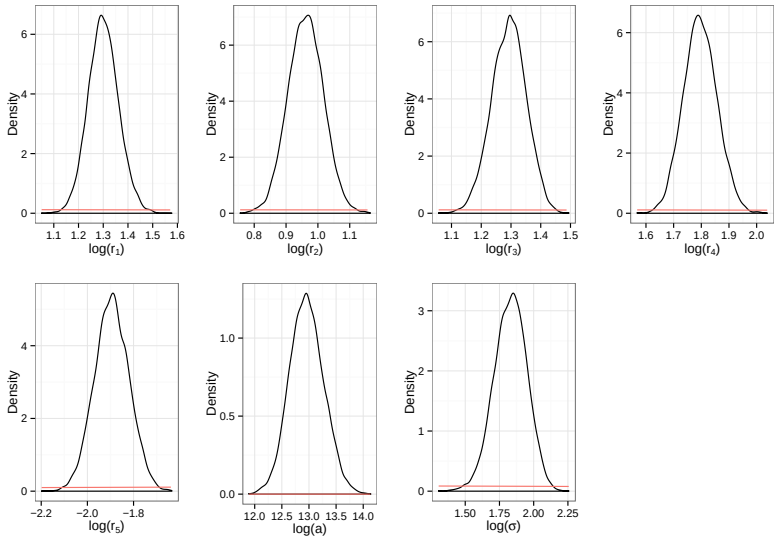


Figure A.11: Marginal posterior distributions for the hyperparameters for the four timepoint design Gaussian process.

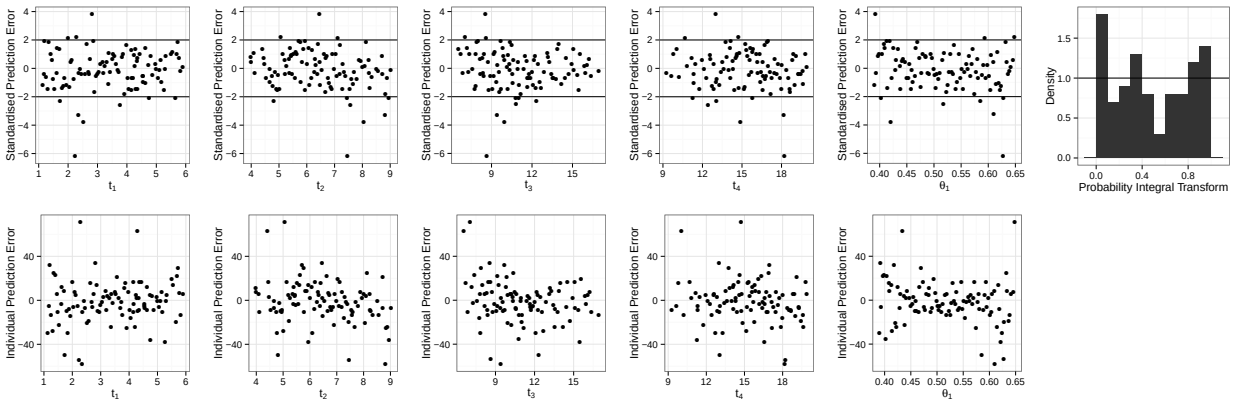


Figure A.12: Diagnostics for the for the four timepoint design Gaussian process.

A.2 Optimal design when all parameters are unknown

A.2.1 Design space reduction

Figures A.13, A.15 and A.17 are the marginal posterior distributions for the hyperparameters for the Gaussian processes fitted to the delta approximation to the expected utility for the two, three and four timepoint designs which are referred to in Subsection 10.5.1. Figures A.14, A.16 and A.18 are the associated diagnostic plots.

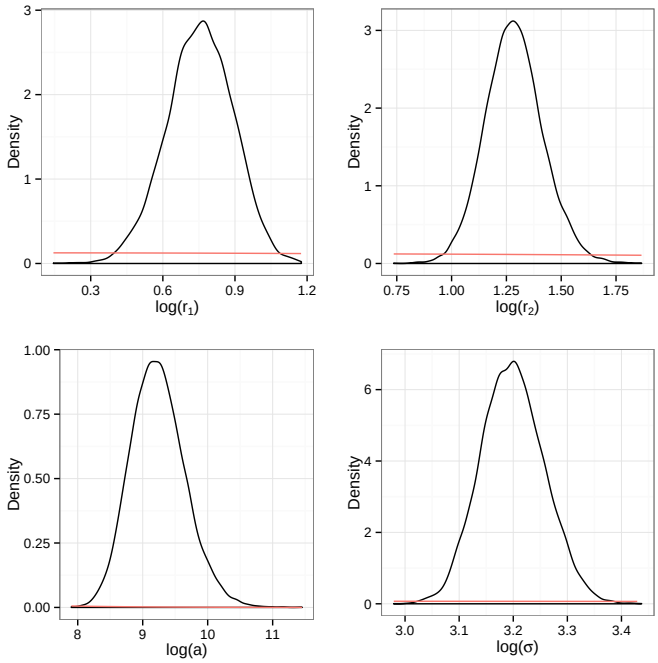


Figure A.13: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a two timepoint design.

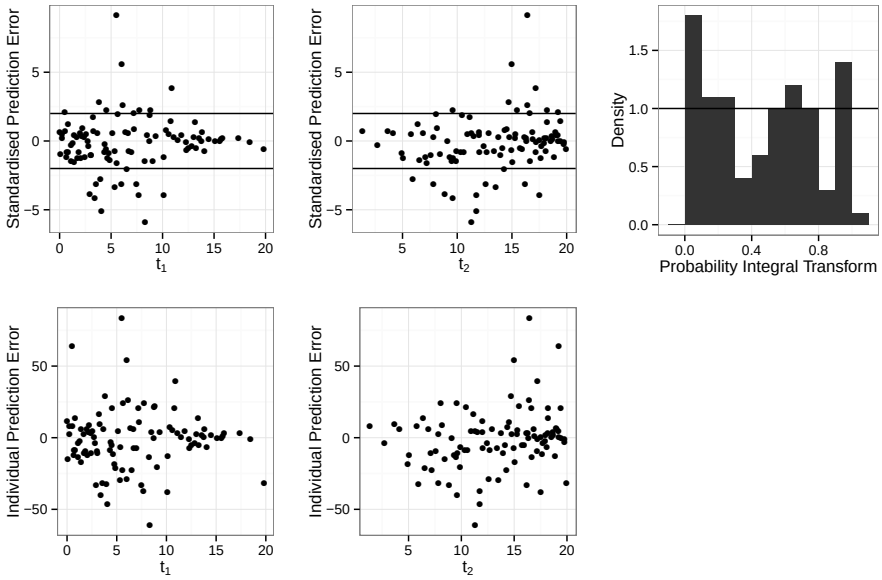


Figure A.14: Diagnostics for the Gaussian process used to reduce the design space for the two timepoint design.

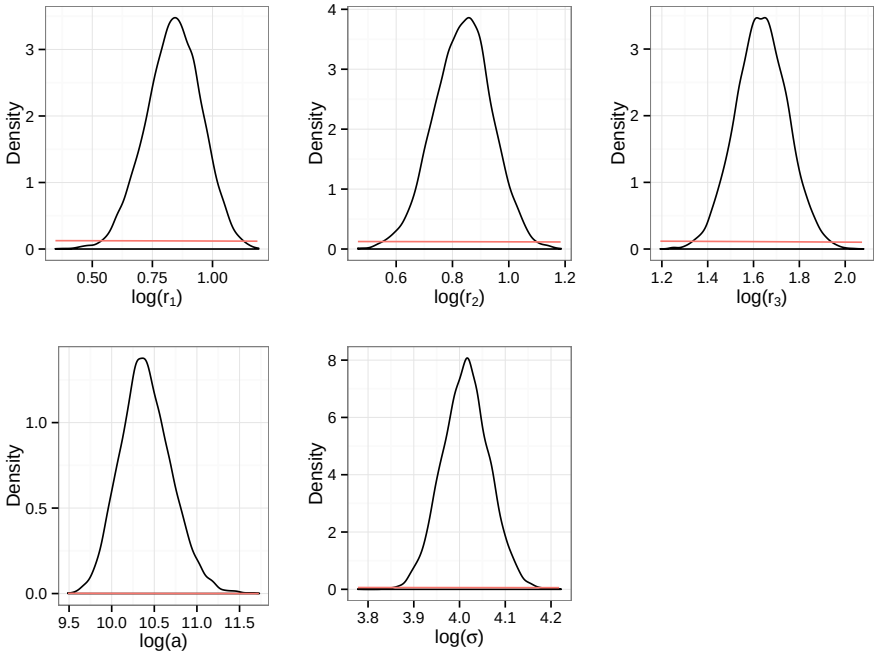


Figure A.15: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a three timepoint design.

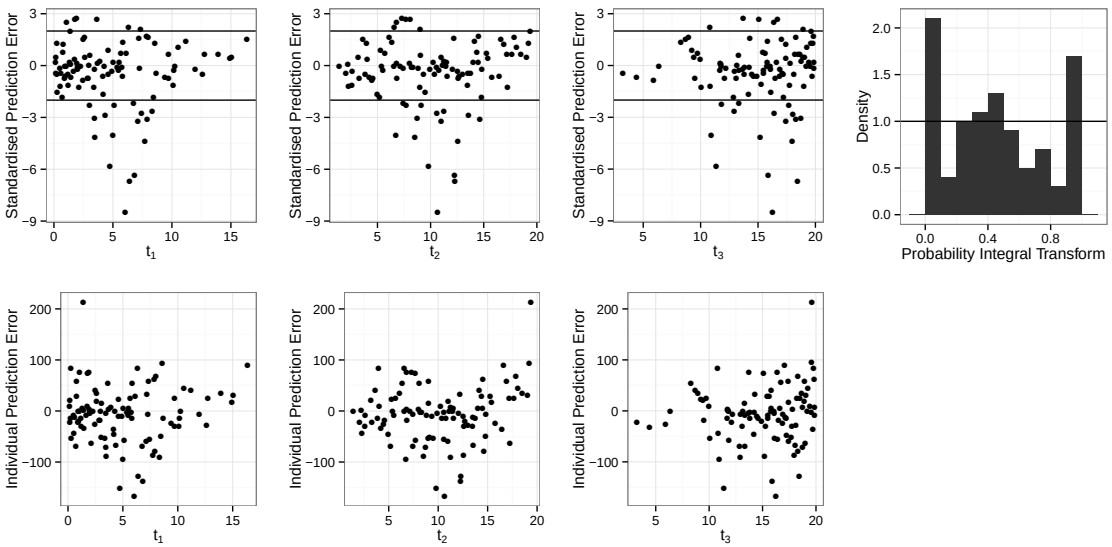


Figure A.16: Diagnostics for the Gaussian process used to reduce the design space for the three timepoint design.

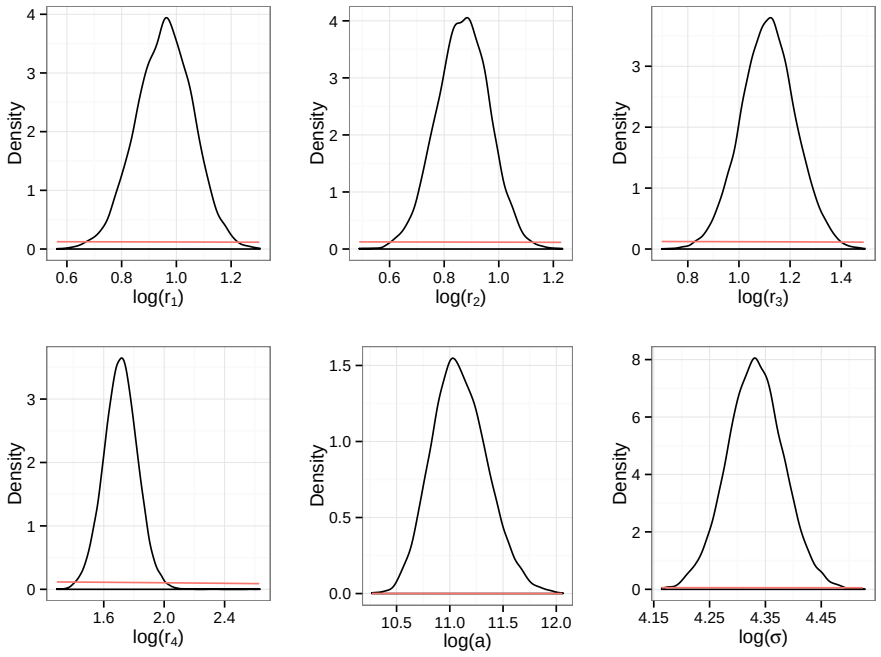


Figure A.17: Marginal posterior distributions for the hyperparameters for the Gaussian process used to reduce the design space for a four timepoint design.

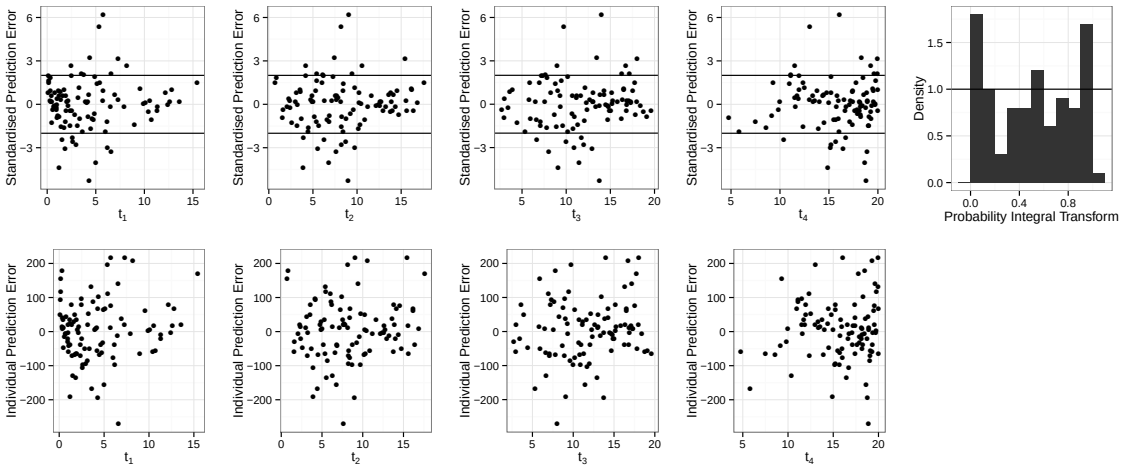


Figure A.18: Diagnostics for the Gaussian process used to reduce the design space for the four timepoint design.

A.2.2 Gaussian process fitted to $u(d, \theta)$ using the reduced space

Figures A.19, A.21 and A.23 are the marginal posterior distributions for the hyperparameters for the Gaussian processes fitted to $u(\boldsymbol{d}, \boldsymbol{\theta})$ for the two, three and four timepoint designs which are referred to in Subsection 10.5.2. Figures A.20, A.22 and A.24 are the associated

diagnostic plots.

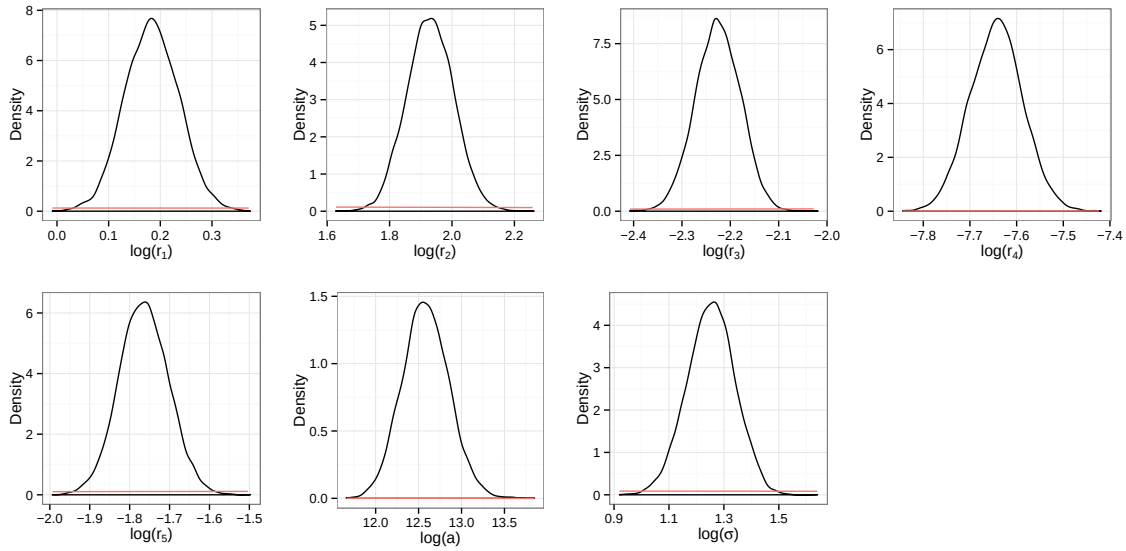


Figure A.19: Marginal posterior distributions for the hyperparameters for a two timepoint design.

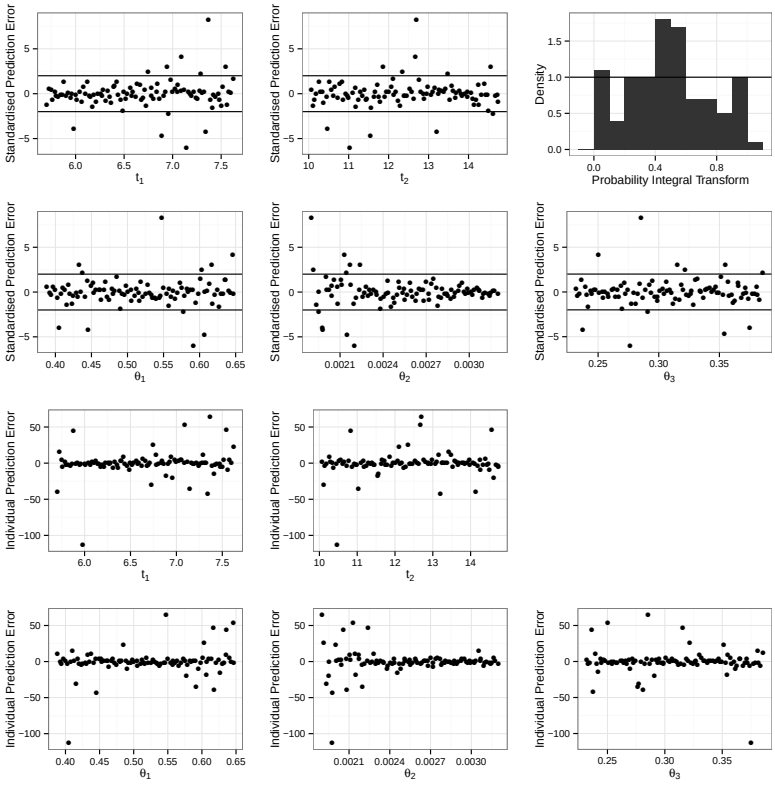


Figure A.20: Diagnostics for the two timepoint design Gaussian process.

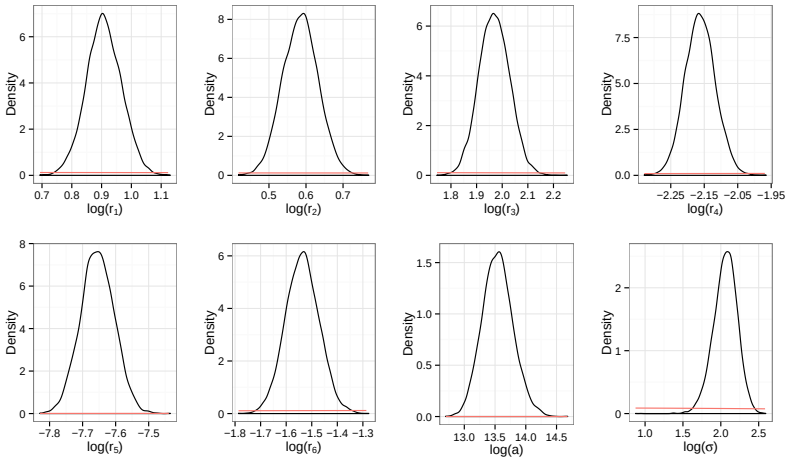


Figure A.21: Marginal posterior distributions for the hyperparameters for a three timepoint design.

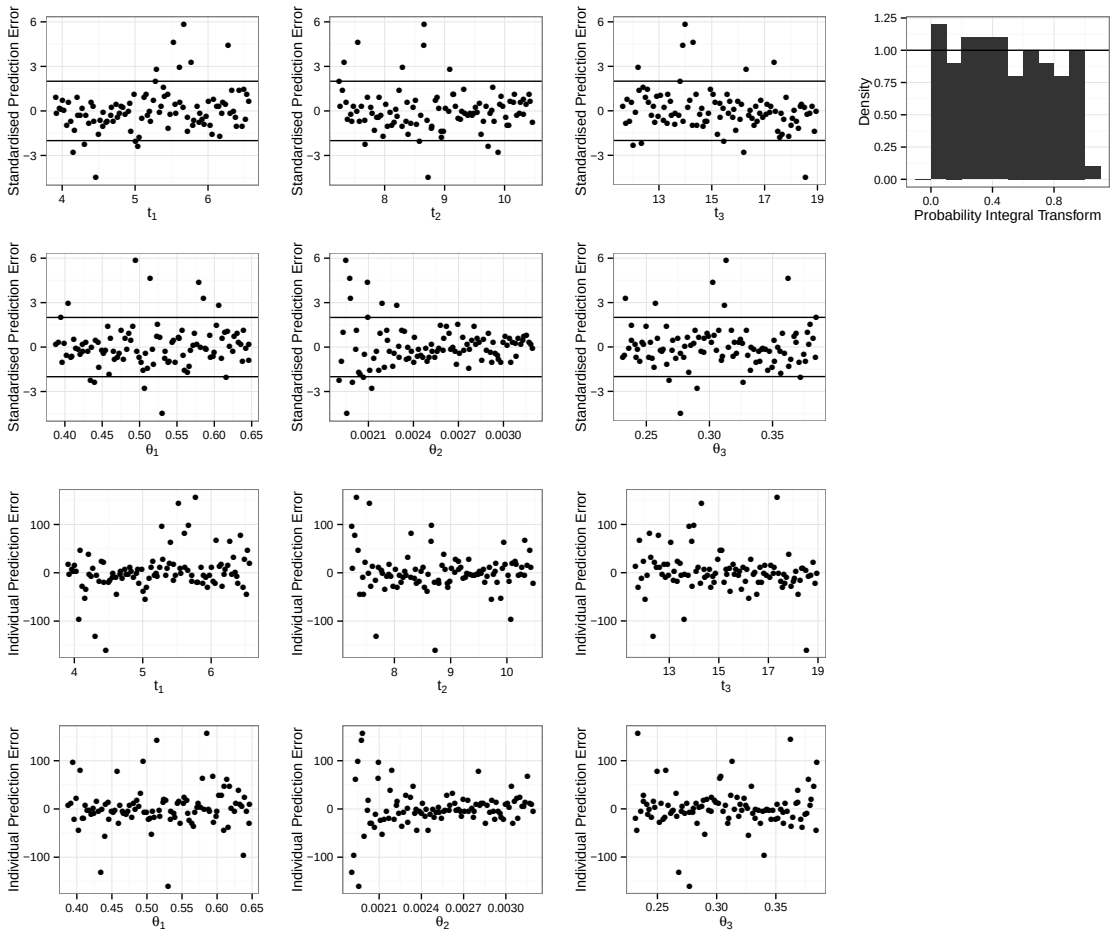


Figure A.22: Diagnostics for the three timepoint design Gaussian process.

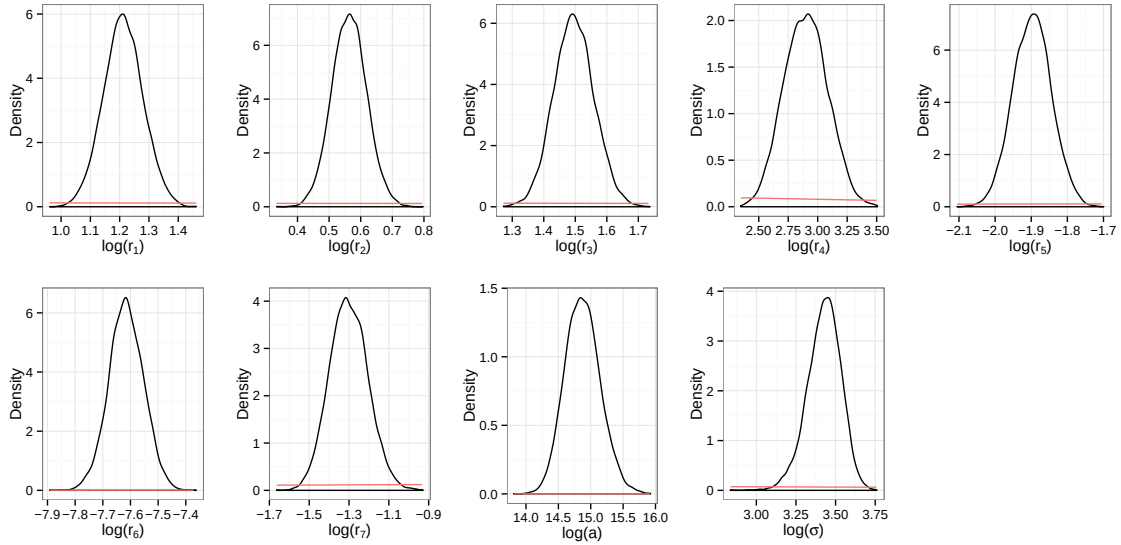


Figure A.23: Marginal posterior distributions for the hyperparameters for a four timepoint design.

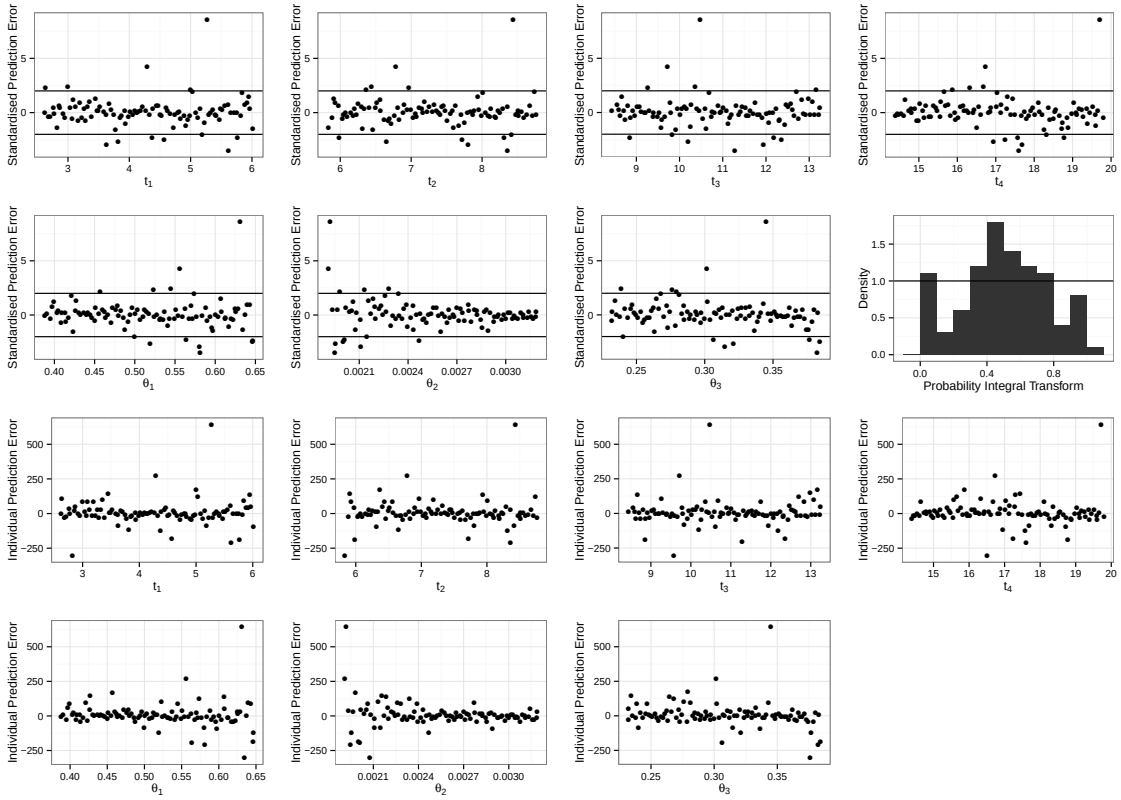


Figure A.24: Diagnostics for the four timepoint design Gaussian process.

Bibliography

Ababou, R., Bagtzoglou, A. C., and Wood, E. F. (1994). On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Mathematical Geology*, 26(1):99–133.

Albert, J. H. and Chib, S. (1993). Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association*, 88(422):669–679.

Alexopoulos, C., Fishman, G. S., and Seila, A. F. (1997). Computational experience with the batch means method. In *Winter Simulation Conference*, pages 194–201.

Alfonsi, A., Cancs, E., Turinici, G., Di Ventura, B., and Huisinga, W. (2005). Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proceedings*, 14:1–13.

Arvand, A. (2001). Biology of EWS/ETS fusions in Ewing's family tumors. *Oncogene*, 20:5747–54.

Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M. A. (1994). Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the United States of America*, 91(3):1059–1063.

Bastos, L. S. (2010). *Validating Gaussian process models in computer experiments*. PhD thesis, The University of Sheffield, UK.

Bastos, L. S. and O'Hagan, A. (2009). Diagnostics for Gaussian Process Emulators. *Technometrics*, 51(4):425–438.

Baum, L. E. and Eagon, J. A. (1966). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model in ecology. *Bulletin of the American Mathematical Society*, 73:360–363.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probablisitic functions of Markov chains. *The Annals of Mathematical Statistics*, 41:164–171.

Bazi, Y. and Melgani, F. (2010). Gaussian process approach to remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(1):186–197.

Beaumont, M. A., Wenyang, Z., and Balding, D. J. (2002). Approximate Bayesian Computation in population genetics. *Genetics*, 162:2025–2035.

Boys, R. J. and Henderson, D. A. (2001a). A comparison of reversible jump MCMC algorithms for DNA sequence segmentation using hidden Markov models. *Computing Science and Statistics*, 33(1989):1–15.

Boys, R. J. and Henderson, D. A. (2001b). A comparison of reversible jump MCMC algorithms for DNA sequence segmentation using hidden Markov models. *Computing Science and Statistics*, 33(1989):1–15.

Boys, R. J. and Henderson, D. A. (2002). On determining the order of Markov dependence of an observed process governed by a hidden Markov model. *Scientific Programming*, 10(3):241–251.

Boys, R. J. and Henderson, D. A. (2004). A Bayesian approach to DNA sequence segmentation. *Biometrics*, 60(3):573–581; discussion 581–588.

Boys, R. J., Henderson, D. A., and Wilkinson, D. J. (2000). Detecting homogeneous segments in DNA sequences by using hidden Markov models. *Applied Statistics*, 49:269–285.

Braun, J. V., Braun, R., and Müller, H.-G. (2000). Multiple changepoint fitting via quasilikelihood, with application to DNA sequence segmentation. *Biometrika*, 87(2):301–314.

Braun, J. V. and Müller, H.-G. (1998). Statistical methods for DNA sequence segmentation. *Statistical Science*, 13(2):142–162.

Breydo, L. and Uversky, V. N. (2011). Role of metal ions in aggregation of intrinsically disordered proteins in neurodegenerative diseases. *Metallomics*, 3:1163–1180.

Cao, Y., Gillespie, D. T., and Petzold, L. R. (2006). Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124(4):044109.

Celeux, G., Hurn, M., and Robert, C. P. (2000). Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, 95:957–970.

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: a review. *Statistical Science*, 10(3):273–304.

Chapman, W. L., Welch, W. J., Bowman, K. P., Sacks, J., and Walsh, J. E. (1994). Arctic sea ice variability: Model sensitivities and a multidecadal simulation. *Journal of Geophysical Research: Oceans*, 99(C1):919–935.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313.

Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51(1):79–94.

Clyde, M., Müller, P., and Parmigiani, G. (1995a). Inference and design strategies for a hierarchical logistic regression model. In Berry, D. A. and Stangl, D., editors, *Bayesian Biostatistics*, pages 297–320. CRC Press.

Clyde, M., Müller, P., and Parmigiani, G. (1995b). Optimal design for heart defibrillators. In Gatsonis, C., Hodges, J., Kass, R., and Singpurwalla, N., editors, *Case Studies in Bayesian Statistics, Volume II*, volume 105 of *Lecture Notes in Statistics*, pages 278–292. Springer New York.

Clyde, M. A. (2001). Experimental design: a Bayesian perspective.

Cook, A. R., Gibson, G. J., and Gilligan, C. A. (2008). Optimal observation times in experimental epidemic processes. *Biometrics*, 64(3):860–868.

Cumberworth, A., Lamour, G. M., Babu, M., and Gsponer, J. (2013). Promiscuity as a functional trait: intrinsically disordered regions as central players of interactomes. *Biochemical Journal*, 454(3):361–369.

Da Cruz, S. and Cleveland, D. W. (2011). Understanding the role of TDP-43 and FUS/TLS in ALS and beyond. *Current Opinion in Neurobiology*, 21(6):904 – 919.

Delattre, O., Zucman, J., Plougastel, B., Desmaze, C., Melot, T., Peter, M., Kovar, H., Joubert, I., de Jong, P., and Rouleau, G. (1992). Gene fusion with an ETS DNA-binding domain caused by chromosome translocation in human tumours. *Nature*, 359(6391):162–5.

Dosztányi, Z., Mészáros, B., and Simon, I. (2010). Bioinformatical approaches to characterize intrinsically disordered/unstructured proteins. *Briefings in Bioinformatics*, 11(2):225–43.

Drovandi, C. C. and Pettitt (2013). Bayesian experimental design for models with intractable likelihoods. *Biometrics*, 69(4):937–948.

Dunker, A. K. and Kriwacki, R. W. (2011). The orderly chaos of proteins. *Scientific American*, 304:68–73.

Dunker, A. K., Lawson, J. D., Brown, C. J., Williams, R. M., Romero, P., Oh, J. S., Oldfield, C. J., Campen, A. M., Ratliff, C. M., Hipps, K. W., Ausio, J., Nissen, M. S., Reeves, R., Kang, C., Kissinger, C. R., Bailey, R. W., Griswold, M. D., Chiu, W., Garner, E. C., and Obradovic, Z. (2001). Intrinsically disordered protein. *Journal of Molecular Graphics and Modelling*, 19(1):26–59.

Elf, J. and Ehrenberg, M. (2003). Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome Research*, 13(11):2475–2484.

Farrow, M. (2013). Optimal Experiment Design, Bayesian. *Encyclopedia of Systems Biology, Springer*.

Fearnhead, P., Giagos, V., and Sherlock, C. (2012). Inference for reaction networks using the linear noise approximation. Available from `http://arXiv:1205.6920`.

Ferm, L., Lotstedt, P., and Hellander, A. (2008). A hierarchy of approximations of the master equation scaled by a size parameter. *Journal of Scientific Computing*, 34(2):127–151.

Ferron, F., Longhi, S., Canard, B., and Karlin, D. (2006). A practical overview of protein disorder prediction methods. *Proteins*, 65(1):1–14.

Flegal, J. M. (2008). *Monte Carlo Standard Errors for Markov Chain Monte Carlo.* PhD thesis, University of Minnesota, US.

Flournoy, N. (1993). A clinical experiment in bone marrow transplantation: estimating a percentage point of a quantal response curve. In Gatsonis, C., Hodges, J. S., Kass, R. E., and Singpurwalla, N. D., editors, *Case Studies in Bayesian Statistics*, volume 83 of *Lecture Notes in Statistics*, pages 324–336. Springer New York.

Friel, N. and Pettitt, A. N. (2008). Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3):589–607.

Fuxreiter, M., Tóth-Petróczy, F., Kraut, D. A., Matouschek, A. T., Lim, R. Y. H., Xue, B., Kurgan, L., and Uversky, V. N. (2014). Disordered proteinaceous machines. *Chemical Reviews*, 114(13):6806–6843.

Galassi, M. (2013). *GNU Scientific Library Reference Manual (3rd Ed.)*.

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, 3rd edition.

Gelman, A. and Meng, X. L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457–472.

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(6):721–741.

Germain, S. E. (2010). *Bayesian spatiotemporal modelling of rainfall through nonhomogeneous hidden Markov models.* PhD thesis, Newcastle University, UK.

Geweke, J. (1991). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. Staff Report 148, Federal Reserve Bank of Minneapolis.

Giagos, V. (2011). *Inference for autoregulatory genetic networks using diffusion process approximations.* PhD thesis, Lancaster University, UK.

Gibson, M. A. and Bruck, J. (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104(9):1876–1889.

Giles, P. (2001). Probabilistic models for annotating DNA sequences. MMathStat Project, Newcastle University.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361.

Gillespie, D. T. (1992). A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404–425.

Gillespie, D. T. (2000). The chemical Langevin equation. *The Journal of Chemical Physics*, 113(1):297–306.

Gillespie, D. T. (2001). Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733.

Gillespie, D. T. and Petzold, L. R. (2003). Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 119(16):8229–8234.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 69(2):243–268.

Golightly, A. and Gillespie, C. S. (2013). Simulation of stochastic kinetic models. In *In-silico Systems Biology: A systems-based approach to understanding biological processes*. Humana Press.

Golightly, A. and Wilkinson, D. J. (2011). Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820.

Gramacy, R. B. and Lee, H. K. H. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3, SI):713–722.

Gray, D. A. (2014). Computational and experimental analysis of the disordered transforming domain of FUS. Introduction to a paper in preparation.

Hainy, M., Müller, W. G., and Wagner, H. (2013). Likelihood-free simulation-based optimal design. Available from `http://arXiv:1305.4273`.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Hayot, F. and Jayaprakash, C. (2004). The linear noise approximation for molecular fluctuations within cells. *Physical Biology*, 1(4):205–210.

Heidelberger, P. and Welch, P. D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, 31:1109–1145.

Higham, D. J., Intep, S., Mao, X., and Szpruch, L. (2011). Hybrid simulation of autoregulation within transcription and translation. *BIT Numerical Mathematics*, 51(1):177–196.

Joerger, A. C. and Fersht, A. R. (2008). Structural biology of the tumor suppressor p53. *Annual Review of Biochemistry*, 77:557–82.

Jylnki, P., Vanhatalo, J., and Vehtari, A. (2011). Robust Gaussian process regression with a Student-t likelihood. *Journal of Machine Learning Research*, 12:3227–3257.

Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011). Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *Annals of Applied Statistics*, 5(4):2470–2492.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(3):425–450.

Kiehl, T. R., Mattheyses, R. M., and Simmons, M. K. (2004). Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322.

Komorowski, M., Costa, M. J., Rand, D. A., and Stumpf, M. P. H. (2011). Sensitivity, robustness, and identifiability in stochastic chemical kinetics models. *Proceedings of the National Academy of Sciences*, 108(21):8645–8650.

Komorowski, M., Finkenstadt, B., Harper, C., and Rand, D. (2009). Bayesian inference of biochemical kinetic parameters using the linear noise approximation. *BMC Bioinformatics*, 10(1):343.

Kovar, H. (2011). Dr. Jekyll and Mr. Hyde: the two faces of the FUS/EWS/TAF15 protein family. *Sarcoma*, 2011:837474.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Kurtz, T. G. (1970). Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, 7(1):pp. 49–58.

Kurtz, T. G. (1971). Limit theorems for sequences of jump Markov processes approximating ordinary differential processes. *Journal of Applied Probability*, 8(2):344–356.

Laurie, D. P. (1997). Calculation of Gauss-Kronrod quadrature rules. *Mathematics of Computation*, 66(219):1133–1145.

Liepe, J., Filippi, S., Komorowski, M., and Stumpf, M. P. H. (2013). Maximizing the information content of experiments in systems biology. *PLoS Computational Biology*, 9(1):e1002888.

Linding, R., Schymkowitz, J., Rousseau, F., Diella, F., and Serrano, L. (2004). A comparative study of the relationship between protein structure and β-aggregation in globular and intrinsically disordered proteins. *Journal of Molecular Biology*, 342(1):345–353.

Lindley, D. V. (1980). Approximate Bayesian methods. *Trabajos de Estadstica e Investigacin Operativa*, 31(1):223–237.

Liu, J. F. and Rost, B. (2003). NORSp: predictions of long regions without regular secondary structure. *Nucleic Acids Research*, 31(13).

Liu, J. S. and Lawrence, C. E. (1999). Bayesian inference on biopolymer models. *Bioinformatics*, 15(1):38–52.

Lotka, A. J. (1925). *Elements of Physical Biology.* (Williams & Wilkins), Baltimore.

Marin, J. M., Pudlo, P., Robert, C. P., and Ryder, R. (2012). Approximate Bayesian Computation methods. *Statistics and Computing*, 22(6):1167–1180.

Marjoram, P., Molitor, J., Plagnol, V., and Tavar, S. (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328.

Martini, A., La Starza, R., and Janssen, H. (2002). Recurrent rearrangement of the Ewing's sarcoma gene, EWSR1, or its homologue, TAF15, with the transcription factor CIZ/NMP4 in acute leukemia. *Cancer Research*, 62:5408–12.

McCampbell, A. (2000). CREB-binding protein sequestration by expanded polyglutamine. *Human Molecular Genetics*, 9(14):2197–2202.

Mckay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–145.

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).

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402.

Müller, P. (1999). Simulation-based optimal design. *Bayesian Statistics 6: Proceedings of Sixth Valencia International Meeting*, 6:459.

Müller, P., Sansó, B., and De Iorio, M. (2004). Optimal Bayesian design by inhomogeneous Markov chain simulation. *Journal of the American Statistical Association*, 99(467):788–798.

Oakley, J. E. and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769.

Pagendam, D. and Pollett, P. (2009). Optimal sampling and problematic likelihood functions in a simple population model. *Environmental Modeling & Assessment*, 14(6):759–767.

Pagendam, D. and Pollett, P. (2010). Robust optimal observation of a metapopulation. *Ecological Modelling*, 221(21):2521 – 2525.

Pagendam, D. and Ross, J. (2013). Optimal use of GPS transmitter for estimating species migration rate. *Ecological Modelling*, 249(0):37 – 41.

Pareek, C. S., Smoczynski, R., and Tretyn, A. (2011). Sequencing technologies and genome sequencing. *Journal of Applied Genetics*, 52(4):413–435.

Podlich, H. M., Faddy, M. J., Smyth, G. K., and Statistics, C. F. (1999). Likelihood computations for extended poisson process models.

Prilusky, J., Felder, C. E., Zeev-Ben-Mordehai, T., Rydberg, E. H., Man, O., Beckmann, J. S., Silman, I., and Sussman, J. L. (2005). FoldIndex: a simple tool to predict whether a given protein sequence is intrinsically unfolded. *Bioinformatics*, 21(16):3435–3438.

Puchalka, J. and Kierzek, A. M. (2004). Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks. *Biophysical Journal*, 86(3):1357–1372.

Raftery, A. E. and Lewis, S. (1992). How Many Iterations in the Gibbs Sampler? In *In Bayesian Statistics 4*, pages 763–773. Oxford University Press.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

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

Robert, C., Ryden, T., and Titterington, D. (2000). Bayesian inference in hidden Markov models through the reversible jump Markov chain Monte Carlo method. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(1):57–75.

Roberts, G. . (1996). Markov chain concepts related to sampling algorithms. In Gilks, W. R., Richardson, S., and Spiegelhalter, D., editors, *Markov Chain Monte Carlo in Practice*, pages 45–57. London: Chapman & Hall.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367.

Rougier, J., Sexton, D. M. H., Murphy, J. M., and Stainforth, D. (2009). Analyzing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments. *Journal of Climate*, 22(13):3540–3557.

Salis, H. and Kaznessis, Y. (2005). Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *Journal of Chemical Physics*, 122(5).

Sandmann, W. (2009). Streamlined formulation of adaptive explicit-implicit tau-leaping with automatic tau selection. In *Winter Simulation Conference*, WSC '09, pages 1104–1112. Winter Simulation Conference.

Schmidler, S. C., Liu, J. S., and Brutlag, D. L. (2000). Bayesian segmentation of protein secondary structure. *Journal of Computational Biology*, 7(1-2):233–248.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

Sisson, S. A. and Fan, Y. (2010). Likelihood-free Markov chain Monte Carlo. Available from `http://arXiv:1001.2058`.

Sorensen, P. H., Lessnick, S. L., Lopez-Terrada, D., Liu, X. F., Triche, T. J., and Denny, C. T. (1994). A second Ewing's sarcoma translocation, t(21;22), fuses the EWS gene to another ETS-family transcription factor, ERG. *Nature Genetics*, 6(2):146–51.

Spiegelhalter, D. J. (1998). Bayesian graphical modelling: a case-study in monitoring health outcomes. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 47(1):115–133.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151.

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(4):795–809.

Tan, A. Y. and Manley, J. L. (2009). The TET family of proteins: functions and roles in disease. *Journal of Molecular Cell Biology*, 1(2):82–92.

Tavare, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. (1997). Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518.

Turchin, V. F. (1971). On the Computation of Multidimensional Integrals by the Monte Carlo Method. *Theory of Probablility and its Applications*, 16(4):720–724.

Turoverov, K. K., Kuznetsova, I. M., and Uversky, V. N. (2010). The protein kingdom extended: ordered and intrinsically disordered proteins, their folding, supramolecular complex formation, and aggregation. *Progress in Biophysics and Molecular Biology*, 102(2-3):73–84.

Urano, F., Umezawa, A., Hong, W., Kikuchi, H., and Hata, J.-i. (1996). A novel chimera gene between EWS and E1A-F, encoding the adenovirus E1A enhancer-binding protein, in extraosseous Ewing's sarcoma. *Biochemical and Biophysical Research Communications*, 219(2):608–612.

Uversky, V. N. (2009). Intrinsic disorder in proteins associated with neurodegenerative diseases. *Frontiers in Bioscience (Landmark Edition)*, 1(14):5188–238.

Uversky, V. N. (2011). Intrinsically disordered proteins from A to Z. *The International Journal of Biochemistry & Cell Biology*, 43(8):1090–103.

Uversky, V. N., Dave, V., Iakoucheva, L. M., Malaney, P., Metallo, S. J., Pathak, R. R., and Joerger, A. C. (2014). Pathological unfoldomics of uncontrolled chaos: intrinsically disordered proteins and human diseases. *Chemical Reviews*, 114(13):6844–6879.

Uversky, V. N., Oldfield, C. J., and Dunker, A. K. (2008). Intrinsically disordered proteins in human diseases: introducing the D(2) concept. *Annual Review of Biophysics*, 37:215–246.

Uversky, V. N., Oldfield, C. J., Midic, U., Xie, H., Xue, B., Vucetic, S., Iakoucheva, L. M., Obradovic, Z., and Dunker, A. K. (2009). Unfoldomics of human diseases: linking protein intrinsic disorder with diseases. *BMC Genomics*, 10(Suppl 1):S7.

Van Kampen, N. G. (2007). *Stochastic Processes in Physics and Chemistry*. North Holland, 3rd edition.

Volterra, V. (1926). Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560.

Waage, P. and Gulberg, C. M. (1986). Studies concerning affinity. *Journal of Chemical Education*, 63(12):1044–1047.

Wallace, E. W. J., Gillespie, D. T., Sanft, K. R., and Petzold, L. R. (2012). Linear noise approximation is valid over limited times for any chemical system that is sufficiently large. *IET Systems Biology*, 6(4):102–115.

Ward, J. J., McGuffin, L. J., Bryson, K., Buxton, B. F., and Jones, D. T. (2004). The DISOPRED server for the prediction of protein disorder. *Bioinformatics*, 20(13):2138–9.

Weathers, E. A., Paulaitis, M. E., Woolf, T. B., and Hoh, J. H. (2004). Reduced amino acid alphabet is sufficient to accurately recognize intrinsically disordered protein. *FEBS letters*, 576(3):348–52.

Wilkinson, D. J. (2011). *Stochastic Modelling for Systems Biology*. Chapman & Hall/CRC, second edition.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Yamada, J., Phillips, J. L., Patel, S., Goldfien, G., Calestagne-Morelli, A., Huang, H., Reza, R., Acheson, J., Krishnan, V. V., Newsam, S., Gopinathan, A., Lau, E. Y., Colvin, M. E., Uversky, V. N., and Rexach, M. F. (2010). A bimodal distribution of two distinct categories of intrinsically disordered structures with separate functions in FG nucleoporins. *Molecular & Cellular Proteomics*, 9(10):2205–24.