Bayesian inference for stochastic kinetic models using data on proportions of cell death

Holly F. Ainsworth

Thesis submitted for the degree of

Doctor of Philosophy

School of Mathematics & Statistics

Newcastle University

Newcastle upon Tyne

United Kingdom

May 2014

Acknowledgements

I am deeply grateful to my supervisors, Richard Boys and Colin Gillespie. They have been generous with their time and unwavering with their support. Their enthusiasm has been infectious throughout and without their assistance, this thesis would never have been started, nor would it ever have been finished. Thank you.

Discussions with Andy Golightly, Daniel Henderson and Carole Proctor have all been extremely helpful. Thank you for giving your time and advice so generously.

To my friends and colleagues in the School of Mathematics and Statistics who made my time there so thoroughly enjoyable – thank you all. Special thanks must go to Nina and Joy for their friendship and support throughout.

I would also like to express my gratitude to the Engineering and Physical Sciences Research Council for the financial support which made the writing of this thesis possible, and to the School of Mathematics and Statistics at Newcastle University who have provided a supportive and pleasant atmosphere throughout my time here.

I owe a very important debt to Tom for his constant love, patience and kindness and for being so understanding – I can't wait for the next chapter! Special thanks must go my family: Claire, Jules, Steve, Gordon, Ella, Rory, Tom, Phoebe and Rose – thank you all for the love and laughter. Finally, I would like to thank my Mum and Dad – your love, kindness and support has helped me more than you will ever know.

Abstract

The PolyQ model is a large stochastic kinetic model that describes protein aggregation within human cells as they undergo ageing. The presence of protein aggregates in cells is a known feature in many age-related diseases, such as Huntington's. Experimental data are available consisting of the proportions of cell death over time. This thesis is motivated by the need to make inference for the rate parameters of the PolyQ model.

Ideally observations would be obtained on all chemical species, observed continuously in time. More realistically, it would be hoped that partial observations were available on the chemical species observed discretely in time. However, current experimental techniques only allow noisy observations on the proportions of cell death at a few discrete time points. This presents an ambitious inference problem.

The model has a large state space and it is not possible to evaluate the data likelihood analytically. However, realisations from the model can be obtained using a stochastic simulator such as the Gillespie algorithm. The time evolution of a cell can be repeatedly simulated, giving an estimate of the proportion of cell death. Various MCMC schemes can be constructed targeting the posterior distribution of the rate parameters. Although evaluating the marginal likelihood is challenging, a pseudo-marginal approach can be used to replace the marginal likelihood with an easy to construct unbiased estimate. Another alternative which allows for the sampling error in the simulated proportions is also considered.

Unfortunately, in practice, simulation from the model is too slow to be used in an MCMC inference scheme. A fast Gaussian process emulator is used to approximate the simulator. This emulator produces fully probabilistic predictions of the simulator output and can be embedded into inference schemes for the rate parameters.

The methods developed are illustrated in two smaller models; the birth-death model and a medium sized model of mitochondrial DNA. Finally, inference on the large PolyQ model is considered.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

The aim of modelling of biological systems is to describe the state of the system and the relationships between components in the system. One motivating factor behind modelling is to test current scientific understanding of the system, by comparing it with data arising from an observed phenomenon. Models can also be used to facilitate *in silico* experiments, where virtual experiments are performed on a computer. The advantage over lab-based experiments is that *in silico* experiments have the potential to be much cheaper and faster. These experiments can then be used to guide and inform the design of future lab-based, *in vitro* experiments.

The work in this thesis is motivated by a large biological model, the PolyQ model, developed by Tang et al. (2010). The aim of this model is to capture biological processes at the molecular level within human cells as they undergo ageing. The accumulation of abnormal protein deposits within cells are hallmarks of neurodegenerative diseases affecting humans as they age. Specifically, interest lies in expanded polyglutaime (PolyQ) proteins which appear following a gene mutation and are known to feature in diseases such as Huntington's disease (Rubinsztein and Carmichael, 2004; Imarisio et al., 2008).

The effect PolyQ proteins have on the cell is not well understood. It could be that the presence of PolyQ proteins induces a sequence of biological processes which ultimately damage the cell and result in cell death. It has also been suggested that in the short-term the presence of PolyQ proteins has a protective effect on the cell; Tang

et al. (2010) state that there is "controversy over whether these entities are protective, detrimental, or relatively benign". The PolyQ model aims to explore the complex interactions of PolyQ proteins with other elements of the cell. Tang et al. (2010) use computer simulations from the model to suggest ways to reduce the toxicity of PolyQ proteins on cells.

A dynamical model describes a system which changes over time, the PolyQ model is an example of such a system. There are many other biological examples of dynamical models such as population dynamics and intracellular processes. A deterministic approach to dynamical modelling describes the state of the system by a set of ordinary differential equations (ODEs). In this setup, the components of the model are continuous by nature. This may be appropriate for some scenarios, for example, the average level of concentration of a protein in a population of cells. However, at the single cell level (as is the case in the PolyQ model), the number of different biochemical species is driven by Brownian motion and consequently they vary discretely and often with low copy numbers (Gillespie, 1977). In the PolyQ model and other such examples, stochasticity is inherently present. When the copy numbers of the chemical species are high, a deterministic approach to modelling may be appropriate. However, for low copy numbers, a deterministic approach fails to describe stochastic and discrete dynamics of the process.

Currently the parameters in the PolyQ model are fixed at the best guesses of the modellers. The modellers use their expert knowledge, along with information from the literature to adjust parameters such that simulations from the model match experimental data. However, Kitano (2001) states that to be able to analyse the model and simulate from it, it is necessary to obtain knowledge about all of the parameters. The aim of this thesis is to develop methods for using experimental data to formally calibrate models by inferring plausible regions for uncertain parameters.

The task of performing parameter inference for the PolyQ model is hindered by the fact that the experimental data only give a very partial insight into the system. There are no time-course experimental data available on the underlying components of cells –

only data on the proportion of cells which are alive at certain times after the start of the experiment. The data are noisy since they are subject to measurement error.

This thesis considers parameter inference for stochastic kinetic models when the data consist of noisy proportions of cell death which are observed discretely in time, rather than observations on the underlying chemical species.

1.1 Overview of thesis

The principles of stochastic modelling are outlined in Chapter 2. This includes the use of chemical reaction notation to formally describe stochastic kinetic models. Several exact and approximate algorithms for stochastic simulation are introduced.

In order to study properties of the PolyQ model, it is necessary to be able to simulate realisations from the model for different parameter choices. Since the experimental data are proportions of cell death, it is the probability of cell death over time that is modelled. Simulation from the stochastic kinetic model can be used to estimate these probabilities by considering the proportion of simulated cells which die over time. This is done by obtaining realisations from n simulated cells and observing how many cells die over time. The quality of the estimate improves as n gets larger and in the limit as $n \to \infty$ the estimate equals the true proportion of cell death; the standard deviation of the estimate is on the order of $\mathcal{O}(\sqrt{n})$.

Two further stochastic models used to illustrate methods throughout the thesis are introduced. The first of these is the birth-death model. This describes how the dynamics of a population vary over time given that individuals in the population can either reproduce or die. If the population level reaches zero, the population becomes extinct mimicking the cell death feature of the PolyQ model. The simplicity of this model means it is quick to simulate from and also has a tractable data likelihood. The tractable data likelihood allows the posterior distribution to be evaluated. This posterior distribution can then be compared to posterior distributions obtained when using various methods for inference which assume the likelihood is not available. This

provides a benchmark with which to assess the performance of methods.

The second model describes mitochondrial DNA (mtDNA) and is motivated by understanding the relationship between Parkinson's disease and the loss of neurons in the substantia nigra region of the human brain. The model contains two components which represent the number of healthy and unhealthy copies of mtDNA. When the number of copies of the unhealthy mtDNA reaches a certain threshold, the cell dies. This is a model of intermediate complexity; it is more complex than the birth-death model but much more manageable than the PolyQ model.

Since, for models of reasonable size and complexity, the observed data likelihood is intractable, the problem of parameter inference naturally lends itself towards the Bayesian framework. This also has the advantage of allowing expert prior knowledge to be incorporated into the analysis. Chapter 3 introduces Bayesian inference and presents simulation based algorithms for parameter inference for models such as the PolyQ model. These algorithms use Markov-chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods to learn about parameters. These algorithms are *likelihood-free* and rely on forward simulation(s) from the model at each iteration of the scheme.

Chapter 4 applies the algorithms introduced in Chapter 3 to a toy model and the birth-death model. For both of these models, the likelihood is tractable which would not be the case in more complex models. Comparing the results of simulation based methods with the exact methods for the simple birth-death model, will give an insight into how well the simulation based methods perform.

The inference schemes presented in Chapter 3 require a potentially large number of simulations from the model at each iteration of the scheme, to be able to provide an estimate of cell death. As illustrated in Chapter 4, for a small model such as the birth-death model, the computational burden of running theses algorithms is manageable. However, the size and complexity of the PolyQ model means that it is not feasible to use such slow algorithms and an an alternative approach must be found. Chapter 5 describes the construction of a Gaussian process emulator. The emulator is an approximation to the slow simulator, which is hopefully accurate and much faster. The emulator is

built using a set of training runs from the simulator and provides fully probabilistic predictions of what the simulator would produce for given inputs. The emulator can then be embedded into an inference scheme. These methods are compared and contrasted for the birth–death model in Chapter 6 before being applied to larger models in subsequent chapters.

Chapter 7 considers inference for the medium sized model of mitochondrial DNA, where experimental data are available on proportions of neuron survival. Previous attempts have been made to calibrate this model (Henderson et al., 2009, 2010) by incorporating data on the underlying chemical species in the model in the analysis. However, the focus of Chapter 7 is to perform parameter inference using only the data on proportions of neuron survival. Since the model is relatively slow to simulate from, Gaussian process emulators are built and used in an inference scheme for model parameters. Before attempting inference on the experimental data, a synthetic dataset is used where the true parameter values are known.

The PolyQ model is the focus of Chapter 8. The model is introduced in greater detail and Gaussian process emulators are built as a surrogate for the slow simulator. Inference is considered for model parameters using two synthetic datasets of different sizes. Finally, conclusions and suggestions for further work are given in Chapter 9.

Chapter 2

Stochastic Modelling

This chapter establishes the principles of stochastic modelling. Chemical reaction notation, a framework for formally describing stochastic kinetic models, is introduced along with various algorithms for simulation. Three stochastic kinetic models are described: the simple birth–death model, a medium sized model of mitochondria DNA and the large PolyQ model.

When implementing schemes for parameter inference, as described in Chapter 3, the ability to efficiently simulate from the model is crucial. This is because for each of the schemes, at each iteration, it is necessary to obtain a realisation from the model for a particular choice of parameters. The simulated data is compared to observed data and the proposed parameters are either accepted or rejected.

For a fuller discussion of the concepts introduced in this chapter see Wilkinson (2012) and Golightly and Gillespie (2013).

2.1 Chemical reaction notation

Consider a cellular model, where interest lies in the numbers of molecules of particular chemical species within the cell. The mechanisms in which molecules can interact is described by a series of chemical reactions. Reactions take place when the level of one or more of the chemical species is changed. For example, a molecule of type X_1 could

react with a molecule of type X_2 to produce a molecule of type X_3. The effect of this reaction taking place would be to decrease the number of molecules of type X_1 and X_2 by one and increase the number of molecules of type X_3 by one. A reaction of this type is denoted

$$X_1 + X_2 \to X_3.$$

The chemical species on the left of the reaction are known as the *reactants*, and those on the right as the *products*. A network of reactions with u chemical species $X_1, X_2, \ldots, X_u$ and v reactions $R_1, R_2, \ldots, R_v$ involved is

$$\begin{aligned}
R_1: &\quad \sum_{j=1}^{u} p_{1j} X_j \longrightarrow \sum_{j=1}^{u} q_{1j} X_j \\
&\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
R_i: &\quad \sum_{j=1}^{u} p_{ij} X_j \longrightarrow \sum_{j=1}^{u} q_{ij} X_j \\
&\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
R_v: &\quad \sum_{j=1}^{u} p_{vj} X_j \longrightarrow \sum_{j=1}^{u} q_{vj} X_j.
\end{aligned}$$

Each *reactant* and *product* have associated *stoichiometries* $P = (p_{ij})$ and $Q = (q_{ij})$, denoting the discrete number of molecules of type j which are involved in reaction i. The reaction matrix is defined to be $A = P - Q$ and describes the net effect of each reaction on the system, the ijth entry describes how reaction i changes the level of species j. The *stoichiometry* matrix is given by $S = A'$.

At a particular time, the number of molecules of type X_j in the system is given by x_j, hence the state of the full system can be described by $\boldsymbol{x} = (x_1, x_2, \ldots, x_u)'$.

2.2 Markov jump process

Assuming that the molecules are in a container with a fixed volume which is *well stirred* and in thermal equilibrium, then the movement of the molecules is random and driven

by Brownian motion. Gillespie (1992) showed that the rate of the reaction is constant over a small time interval δt. In general, each reaction has associated with it a *stochastic rate constant* denoted θ_i, and along with the current state of the system, this defines the hazard function $h_i(\boldsymbol{x}, \theta_i)$. For a given state of the system $\boldsymbol{x}$ at time t, reaction R_i will happen with rate $h_i(\boldsymbol{x}, \theta_i)\delta t$ in a small time interval δt.

An example of a first order reaction is

$$X \to 2X$$

where the hazard of a molecule of X undergoing the reaction is λ. For x molecules of this chemical species, the combined hazard is

$$h_1(x, \lambda) = \lambda x.$$

A second order reaction could take the form

$$X_1 + X_2 \to X_3.$$

This reaction occurs when a collision between a molecule of type X_1 and a molecule of type X_2 occurs. Denoting this rate θ then the hazard of a reaction happening in the interval δt is $\theta\,\delta t$. For x_1 molecules of X_1 and x_2 molecules of X_2, the overall hazard of this reaction is

$$h(\boldsymbol{x}, \theta) = \theta x_1 x_2.$$

Since the hazards are constructed by considering the number of ways in which the *reactants* on the left hand side of the reaction can react, in general the hazard is proportional to a product of binomial coefficients

$$h_i(\boldsymbol{x}, \theta_i) = \theta_i \prod_{j=1}^{u} \binom{x_j}{p_{ij}}.$$

Table 2.1 gives some example reactions and their associated hazards. The overall hazard

Order	Reaction	Hazard
0	$\emptyset \to X_1$	θ_1
1	$X_1 \to \emptyset$	$\theta_2 x_1$
2	$X_1 + X_2 \to X_3$	$\theta_3 x_1 x_2$
2	$2X_1 \to X_2$	$\theta_4 x_1(x_1 - 1)/2$
3	$3X_1 \to X_3$	$\theta_5 x_1(x_1 - 1)(x_1 - 2)/6$

Table 2.1: Example reactions and their associated hazards.

of any reaction happening is

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

and the time until the next reaction occurring has distribution $Exp(h_0(\boldsymbol{x}, \boldsymbol{\theta}))$.

2.2.1 Chemical master equation

Let $P_{\boldsymbol{x}}(t)$ denote the probability of the state of the system being $\boldsymbol{x} = (x_1, x_2, \ldots, x_u)'$ at time t. This is the transition kernel of the Markov jump process. After some time δt, the transition kernel is $P_{\boldsymbol{x}}(t + \delta t)$. This can be constructed by considering the possible events leading to this outcome given the state of the system at time t.

In a small time interval δt, for the birth–death model, a birth occurs with probability $\lambda x \delta t + o(\delta t)$ and a death occurs with probability $\mu x \delta t + o(\delta t)$. The probability of observing multiple events in this small time interval is $o(\delta t)$ and assumed to be negligible. The probability of having x individuals in the population at $t + \delta t$ is

$$P_x(t + \delta t) = \lambda(x - 1)\delta t P_{x-1}(t) + \mu(x + 1)\delta t P_{x+1}(t) + [1 - x(\lambda + \mu)\delta t]P_x(t).$$

Therefore

$$\frac{P_x(t + \delta t) - P_x(t)}{\delta t} = \lambda(x - 1)P_{x-1}(t) + \mu(x + 1)P_{x+1}(t) - x(\lambda + \mu)P_x(t)$$

and taking the limit as $\delta t \to 0$ gives the Chemical Master Equation (CME)

$$\frac{dP_x(t)}{dt} = \lambda(x-1)P(t)_{x-1} - x(\lambda+\mu)P(t)_x + \mu(x+1)P(t)_{x+1}.$$

For a general model, the CME is given by

$$\frac{dP_x(t)}{dt} = \sum_{i=1}^{v} \left\{ h_i(x - S^i, c_i)P_{x-S^i}(t) - h_i(x, c_i)P_x(t) \right\}$$

where S^i represents the ith column of the *stoichiometry* matrix.

2.2.2 The direct method

This algorithm, in the context of stochastic kinetic models was introduced by Gillespie (1977) and is often referred to as the Gillespie algorithm. It can be used to simulate exact trajectories from a stochastic model. The detailed algorithm is given in Algorithm 1.

At each iteration of the Gillespie algorithm, the time until the next reaction is simulated as an exponential random variable (Step 4). The rate is given by the sum of hazards of each reaction happening which have been calculated in Steps 2 and 3. Once the time until the next reaction has been simulated, the type of reaction must also be simulated. This is done by simulating from a discrete distribution with probabilities given by

$$\frac{h_i(\boldsymbol{x}, \theta_i)}{h_0(\boldsymbol{x}, \boldsymbol{\theta})} \quad \text{for} \quad i = 1, \ldots, v.$$

Three realisations from the birth–death model are given in Figure 2.1 for a particular choice of parameters and initial population level x_0. For these simulations, $\mu > \lambda$ hence the population will eventually die out. It can be seen that in these three simulations, the populations become extinct at times $t \approx (3.4, 5.2, 6.0)$.

Algorithm 1 Stochastic simulation: Gillespie's direct method

1. Set $t = 0$. Initialise the rate constants $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_v)'$ and the initial molecule numbers $\boldsymbol{x} = (x_1, \ldots, x_u)'$.
2. Calculate the hazard $h_i(\boldsymbol{x}, \theta_i)$ for each potential reaction $i = 1, \ldots, v$.
3. Calculate the *combined hazard* $h_0(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{v} h_i(\boldsymbol{x}, \theta_i)$.
4. Simulate the time until the next reaction, $t^* \sim Exp(h_0(\boldsymbol{x}, \boldsymbol{\theta}))$ and set $t := t + t^*$.
5. Simulate the reaction index, j, as a discrete random quantity with probabilities $h_i(\boldsymbol{x}, \theta_i)/h_0(\boldsymbol{x}, \boldsymbol{\theta})$, $i = 1, \ldots, v$.
6. Update the state $\boldsymbol{x}$ according to reaction j.
7. Output x and t. If $t < T_{max}$, return to step 2.

Label	Reaction	Hazard	Description
R_1	$X \to 2X$	$h_1(x, \lambda) = \lambda x$	Birth
R_2	$X \to \emptyset$	$h_2(x, \mu) = \mu x$	Death

Table 2.2: Reactions and their hazards for the birth–death model.

2.3 Example: the birth–death model

The simple birth–death process is a well documented stochastic model. It was first introduced by Yule (1925) and Feller W. (1939) in the context of population growth. It has been widely used in biological applications, for example Kendall (1948) uses it to model the early stages of an epidemic. Novozhilov (2006) discusses the suitability of the birth–death model for modelling biological processes; the state space is discrete rather than continuous, ideal for describing counts such as cells or genes.

For the birth–death model, X represents a population and x represents the number of individuals present, rather than a number of molecules of a chemical species. In this model only two events can happen: either a birth or a death. In chemical kinetic notation this system is represented in Table 2.2, where R_1 denotes a birth event which happens with rate λ and R_2 denotes a death event which happens with rate μ.

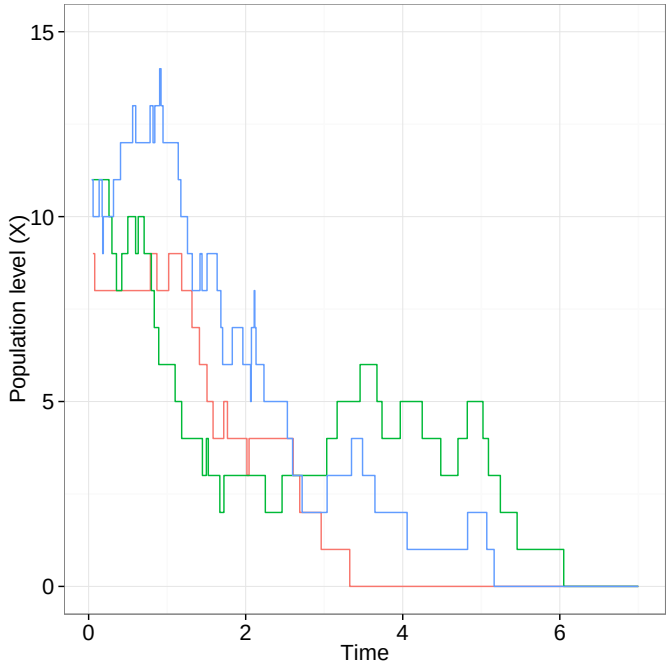

Figure 2.1: Three realisations from the birth–death model for $x_0 = 10$, $\lambda = 0.6$ and $\mu = 1$.

2.4 Example: mitochondrial DNA model

Neurons are specialised types of cell in the human body and are a fundamental aspect of the nervous system. The idea that the loss of neurons in the *substantia nigra* region of the human brain is implicated in Parkinson's disease dates back to the work of Hassler (1938) and Fearnley and Lees (1991). Bender et al. (2006) observed that patients with Parkinson's disease exhibited higher than average levels of mitochondrial DNA (mtDNA) deletions in the *substantia nigra* region of the brain. However, the mechanism which these mtDNA deletions play in disease is still poorly understood.

A model was developed by Henderson et al. (2009) to explore the relationship between mtDNA and cell death. The model is based on the ideas of Elson et al. (2001) who suggest that the relaxed replication of mtDNA is responsible for the the accumulation of mutant mtDNA through random genetic drift.

The model involves two chemical species $\boldsymbol{X} = (X_1, X_2)'$ where X_1 represents healthy mtDNA and X_2 represents unhealthy mtDNA (mtDNA with deletions). The total number of mtDNA present in the cell is given by $x_1 + x_2$.

Label	Reaction	Hazard	Description
R_1	$X_1 \to X_2$	$h_1(\boldsymbol{x}, \theta_1) = \theta_1 x_1$	Mutation
R_2	$X_1 \to 2X_1$	$h_2(\boldsymbol{x}, \theta_2) = \theta_2 \left(\frac{x_1}{x_1+x_2}\right)$	Synthesis
R_3	$X_1 \to \emptyset$	$h_3(\boldsymbol{x}, \theta_3) = \theta_3 x_1$	Degradation
R_4	$X_2 \to 2X_2$	$h_4(\boldsymbol{x}, \theta_4) = \theta_4 \left(\frac{x_2}{x_1+x_2}\right)$	Mutant Synthesis
R_5	$X_2 \to \emptyset$	$h_5(\boldsymbol{x}, \theta_5) = \theta_5 x_2$	Mutant Degradation

Table 2.3: Reactions and their hazards for the mtDNA model.

There are five possible reactions in the system which are given in Table 2.3. R_1 represents healthy mtDNA mutating into unhealthy mtDNA. R_2 and R_4 represent synthesis (birth) of the health and unhealthy mtDNA respectively. R_3 and R_5 represent degradation (death) of the healthy and unhealthy mtDNA respectively.

The model contains a mechanism for cell death. This is modelled by a deterministic process; when the proportion of unhealthy mtDNA reaches some critical threshold, the cell dies. The proportion of unhealthy mtDNA is given by

$$p = \frac{x_2}{x_1 + x_2}$$

and cell death occurs when $p \geq \tau \in (0, 1]$.

Three example simulations from the model are given in Figure 2.2 for parameter choices

$$\theta_1 = e^{-3.8}, \quad \theta_2 = 1000\theta_3, \quad \theta_3 = e^{-10.4}$$
$$\theta_4 = 1000\theta_3, \quad \theta_5 = e^{-10.4}, \quad \tau = 0.75$$

and initial species levels $x_1(0) = 1000$ and $x_2(0) = 0$. This choice of parameters was guided by the prior expectations of Henderson et al. (2009). For these simulations, it was assumed that the healthy and unhealthy mtDNA synthesise and degrade at the same rate. The rate laws for reactions R_2 and R_4 along with the parameter choices for θ_2 and θ_4 are constructed to ensure that the total number of mtDNA in the cell $x_1 + x_2$

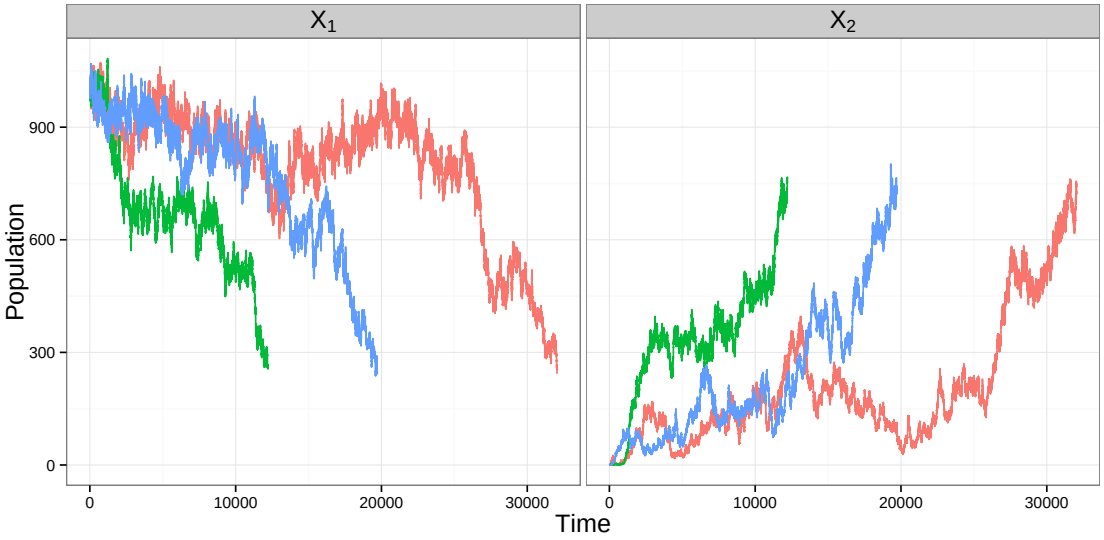

Figure 2.2: Three realisations from the mtDNA model.

remains approximately constant (at 1000) throughout the lifetime of the cell.

It can be seen that for this choice of parameters, the number of healthy mtDNA decrease over time and correspondingly the number of unhealthy mtDNA increase. When the proportion of unhealthy mtDNA reaches the lethal threshold ($\tau = 0.75$ in these simulations), the cell dies.

2.5 Example: the PolyQ model

The PolyQ model was briefly introduced in Chapter 1. It is a large model containing 25 chemical species and 69 reactions. The chemical species are listed in Table 2.4 and the reactions in Table 2.5.

Cell death can occur via two biological pathways, either via proteasome inhibition or p38MAPK activation. The dummy species `PIdeath` and `p38death` are included in the model to encode cell death via these pathways. These species are both binary variables which take the value zero while the cell is alive. When death occurs, either `PIdeath` $= 1$ or `p38death` $= 1$ depending on the death pathway.

An event in the PolyQ model is triggered when either `PIdeath` > 0 and `p38death` > 0, this changes the value of k_{alive} from 0 to 1. k_{alive} is a dummy rate parameter which is

present in every reaction (omitted from the rates in Table 2.5), when it is zero, this has the effect of preventing more reactions happening and inducing cell death in the model.

Name	Description	Initial amount
PolyQ	Polyglutamine-containing protein	1000
mRFPu	Red fluorescent protein	300
Proteasome	26S Proteasome	1000
PolyQProteasome	PolyQ bound to proteasome	0
mRFPuProteasome	mRFPu bound to proteasome	0
$\texttt{AggPolyQ}_i$	PolyQ aggregate of size i ($i = 1, \ldots, 5$)	0
SeqAggP	Inclusion	0
AggPProteasome	Aggregated protein bound to proteasome	0
ROS	Reactive oxygen species	10
p38P	Active P38MAPK	0
p38	Inactive p38MAPK	100
NatP	Generic pool of native protein	19500
MisP	Misfolded protein	0
MisPProteasome	Misfolded protein bound to proteasome	0
$\texttt{AggP}_i$	Small aggregate of size i ($i = 1, \ldots, 5$)	0
PIdeath	Dummy species to record cell death due to proteasome inhibition	0
p38death	Dummy species to record cell death due to p38MAPK activation	0

Table 2.4: Species involved in the PolyQ model and their initial amounts.

ID	Reaction name	Reaction	Rate law
1	PolyQ synthesis	$\texttt{Source} \rightarrow \texttt{PolyQ}$	$k_{synPolyQ}$
2	PolyQ/proteasome binding	$\texttt{PolyQ} + \texttt{Proteasome} \rightarrow \texttt{PolyQProteasome}$	$k_{binPolyQ}\,\texttt{[PolyQ]}\,\texttt{[Proteasome]}$
3	PolyQ/proteasome release	$\texttt{PolyQProteasome} \rightarrow \texttt{PolyQ} + \texttt{Proteasome}$	$k_{relPolyQ}\,\texttt{[PolyQProteasome]}$
4	PolyQ degradation	$\texttt{PolyQProteasome} \rightarrow \texttt{Proteasome}$	$k_{kdegPolyQ}k_{proteff}\,\texttt{[PolyQProteasome]}$
5	mRFPu synthesis	$\texttt{Source} \rightarrow \texttt{mRFPu}$	$k_{synmRFPu}$
6	mRFPu/proteasome binding	$\texttt{mRFPu} + \texttt{Proteasome} \rightarrow \texttt{mRFPuProteasome}$	$k_{binmRFPu}\,\texttt{[mRFPu]}\,\texttt{[Proteasome]}$
7	mRFPu/proteasome release	$\texttt{mRFPuProteasome} \rightarrow \texttt{mRFPu} + \texttt{Proteasome}$	$k_{relmRFPu}\,\texttt{[mRFPuProteasome]}$
8	mRFPu degradation	$\texttt{mRFPuProteasome} \rightarrow \texttt{Proteasome}$	$k_{degmRFPu}k_{proteff}\,\texttt{[mRFPuProteasome]}$
9	Aggregation	$\texttt{2PolyQ + ROS} \rightarrow \texttt{AggPolyQ}_1 + \texttt{ROS}$	$k_{aggPolyQ}\ \texttt{[PolyQ]}\,\texttt{[PolyQ-1]}\,\texttt{[ROS}^2\texttt{]}\ \texttt{0.5(}10^2 + \texttt{[ROS}^2\texttt{])}$
10	Aggregation growth $\texttt{AggPolyQ}_1$	$\texttt{AggPolyQ}_1 + \texttt{PolyQ} + \texttt{ROS} \rightarrow \texttt{AggPolyQ}_2 + \texttt{ROS}$	$k_{aggPolyQ}\,\texttt{[AggPolyQ}_1\texttt{]}\,\texttt{[PolyQ]}\,\texttt{[ROS}^2\texttt{]}\ \texttt{/(}10^2 + \texttt{[ROS}^2\texttt{])}$
11	Aggregation growth $\texttt{AggPolyQ}_2$	$\texttt{AggPolyQ}_2 + \texttt{PolyQ} + \texttt{ROS} \rightarrow \texttt{AggPolyQ}_3 + \texttt{ROS}$	$k_{aggPolyQ}\,\texttt{[AggPolyQ}_2\texttt{]}\,\texttt{[PolyQ]}\,\texttt{[ROS}^2\texttt{]}\ \texttt{/(}10^2 + \texttt{[ROS}^2\texttt{])}$
12	Aggregation growth $\texttt{AggPolyQ}_3$	$\texttt{AggPolyQ}_3 + \texttt{PolyQ} + \texttt{ROS} \rightarrow \texttt{AggPolyQ}_4 + \texttt{ROS}$	$k_{aggPolyQ}\,\texttt{[AggPolyQ}_3\texttt{]}\,\texttt{[PolyQ]}\,\texttt{[ROS}^2\texttt{]}\ \texttt{/(}10^2 + \texttt{[ROS}^2\texttt{])}$
13	Aggregation growth $\texttt{AggPolyQ}_4$	$\texttt{AggPolyQ}_4 + \texttt{PolyQ} + \texttt{ROS} \rightarrow \texttt{AggPolyQ}_5 + \texttt{ROS}$	$k_{aggPolyQ}\,\texttt{[AggPolyQ}_4\texttt{]}\,\texttt{[PolyQ]}\,\texttt{[ROS}^2\texttt{]}\ \texttt{/(}10^2 + \texttt{[ROS}^2\texttt{])}$
14a	PolyQ disaggregation 1	$\texttt{AggPolyQ}_1 \rightarrow \texttt{2AggPolyQ}$	$k_{dissaggPolyQ1}\,\texttt{[AggPolyQ}_1\texttt{]}$
14b	PolyQ disaggregation 2	$\texttt{AggPolyQ}_2 \rightarrow \texttt{AggPolyQ}_1 + \texttt{PolyQ}$	$k_{dissaggPolyQ2}\,\texttt{[AggPolyQ}_2\texttt{]}$
14c	PolyQ disaggregation 3	$\texttt{AggPolyQ}_3 \rightarrow \texttt{AggPolyQ}_2 + \texttt{PolyQ}$	$k_{dissaggPolyQ3}\,\texttt{[AggPolyQ}_3\texttt{]}$
14d	PolyQ disaggregation 4	$\texttt{AggPolyQ}_4 \rightarrow \texttt{AggPolyQ}_3 + \texttt{PolyQ}$	$k_{dissaggPolyQ4}\,\texttt{[AggPolyQ}_4\texttt{]}$

14e	PolyQ disaggregation 5	$\texttt{AggPolyQ}_5 \rightarrow \texttt{AggPolyQ}_4$ + PolyQ	$k_{dissaggPolyQ5}$ [$\texttt{AggPolyQ}_5$]
15	Inclusion formation	$\texttt{AggPolyQ}_5$ + PolyQ → 7SeqAggP	$k_{aggPolyQ}$ [AggPolyQ5] [PolyQ]
16	Inclusion growth	SegAggP + PolyQ → 2SeqAggP	$k_{seqPolyQ}$ [SeqAggP] [PolyQ]
17a	Proteasome inhibition by aggregates 1	$\texttt{AggPolyQ}_1$ + Proteasome → AggPProteasome	$k_{inhprot}$ [$\texttt{AggPPolyQ}_1$] [Proteasome]
17b	Proteasome inhibition by aggregates 2	$\texttt{AggPolyQ}_2$ + Proteasome → AggPProteasome	$k_{inhprot}$ [$\texttt{AggPPolyQ}_2$] [Proteasome]
17c	Proteasome inhibition by aggregates 3	$\texttt{AggPolyQ}_3$ + Proteasome → AggPProteasome	$k_{inhprot}$ [$\texttt{AggPPolyQ}_3$] [Proteasome]
17d	Proteasome inhibition by aggregates 4	$\texttt{AggPolyQ}_4$ + Proteasome → AggPProteasome	$k_{inhprot}$ [$\texttt{AggPPolyQ}_4$] [Proteasome]
17e	Proteasome inhibition by aggregates 5	$\texttt{AggPolyQ}_5$ + Proteasome → AggPProteasome	$k_{inhprot}$ [$\texttt{AggPPolyQ}_5$] [Proteasome]
18	Basal ROS production	Source → ROS	k_{genROS}
19	ROS removal	ROS → Sink	k_{remROS} [ROS]
20a	ROS generation AggPolyQ_1	$\texttt{AggPolyQ}_1 \rightarrow \texttt{AggPolyQ}_1$ + ROS	$k_{genROSAggP}$ [$\texttt{AggPolyQ}_1$]
20b	ROS generation AggPolyQ_2	$\texttt{AggPolyQ}_2 \rightarrow \texttt{AggPolyQ}_2$ + ROS	$k_{genROSAggP}$ [$\texttt{AggPolyQ}_2$]
20c	ROS generation AggPolyQ_3	$\texttt{AggPolyQ}_3 \rightarrow \texttt{AggPolyQ}_3$ + ROS	$k_{genROSAggP}$ [$\texttt{AggPolyQ}_3$]
20d	ROS generation AggPolyQ_4	$\texttt{AggPolyQ}_4 \rightarrow \texttt{AggPolyQ}_4$ + ROS	$k_{genROSAggP}$ [$\texttt{AggPolyQ}_4$]
20e	ROS generation AggPolyQ_5	$\texttt{AggPolyQ}_5 \rightarrow \texttt{AggPolyQ}_5$ + ROS	$k_{genROSAggP}$ [$\texttt{AggPolyQ}_5$]
21	ROS generation AggP Proteasome	AggPProteasome → AggPProteasome + ROS	$k_{genROSAggP}$ [AppPProteasome]
22	p38MAPK activation	ROS + p38 → ROS + p38P	k_{actp38} [ROS] [p38]
23	p38MAPK inactivation	p38P → p38	$k_{inactp38}$ [p38P]
24	AggPProteasome sequestering	AggPProteasome + SeqAggP → 2SeqAggP	$k_{seqAggPProt}$ [AggPProteasome] [SeqAggP]
25	PolyQProteasome sequestering	PolyQProteasome + SeqAggP → 2SeqAggP	$k_{seqPolyQProt}$ [PolyQProteasome] [SeqAggP]
26	Protein synthesis	Source → NatP	$k_{synNatP}$
27	Protein misfolding	NatP + ROS → MisP + ROS	$k_{misfold}$ [NatP] [ROS]
28	Protein refolding	MisP → NatP	k_{refold} [MisP]
29	MisP/Proteasome binding	MisP + Proteasome → MisPProteasome	$k_{binMisPProt}$ [MisP] [Proteasome]
30	MisPProteasome release	MisPProteasome → MisP + Proteasome	$k_{relMisPProt}$ [MisPProteasome]
31	Degradation of misfolded protein	MisPProteasome → Proteasome	$k_{degMisP}k_{proteff}$ [MisPProteasome]
32	Aggregation of misfolded protein	2MisP → $\texttt{AggP}_1$	$k_{aggMisP}$ [MisP] [MisP -1]/2
33a	Aggregation growth 1	$\texttt{AggP}_1$ + MisP → $\texttt{AggP}_2$	$k_{agg2MisP}$ [MisP] [$\texttt{AggP}_1$]
33b	Aggregation growth 2	$\texttt{AggP}_2$ + MisP → $\texttt{AggP}_3$	$k_{agg2MisP}$ [MisP] [$\texttt{AggP}_2$]
33c	Aggregation growth 3	$\texttt{AggP}_3$ + MisP → $\texttt{AggP}_4$	$k_{agg2MisP}$ [MisP] [$\texttt{AggP}_3$]
33d	Aggregation growth 4	$\texttt{AggP}_4$ + MisP → $\texttt{AggP}_5$	$k_{agg2MisP}$ [MisP] [$\texttt{AggP}_4$]
34a	Disaggregation 1	$\texttt{AggP}_1$ → 2MisP	$k_{diasaggMisP_1}$ [$\texttt{AggP}_1$]
34b	Disaggregation 2	$\texttt{AggP}_2$ → $\texttt{AggP}_1$ + MisP	$k_{diasaggMisP_2}$ [$\texttt{AggP}_2$]
34c	Disaggregation 3	$\texttt{AggP}_3$ → $\texttt{AggP}_2$ + MisP	$k_{diasaggMisP_3}$ [$\texttt{AggP}_3$]
34d	Disaggregation 4	$\texttt{AggP}_4$ → $\texttt{AggP}_3$ + MisP	$k_{diasaggMisP_4}$ [$\texttt{AggP}_4$]

34e	Disaggregation 5	$\texttt{AggP}_5 \rightarrow \texttt{AggP}_4 + \texttt{MisP}$	$k_{diasaggMisP_5}[\texttt{AggP}_5]$
35	MisP Inclusion formation	$\texttt{AggP}_5 + \texttt{MisP} \rightarrow \texttt{7SeqAggP}$	$k_{agg2MisP}[\texttt{AggP}_5][\texttt{MisP}]$
36	MisP Inclusion growth	$\texttt{SeqAggP} + \texttt{MisP} \rightarrow \texttt{2SeqAggP}$	$k_{seqMisP}[\texttt{SeqAggP}][\texttt{MisP}]$
37a	Proteasome inhibition AggP_1	$\texttt{AggP}_1 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}[\texttt{AggP}_1][\texttt{Proteasome}]$
37b	Proteasome inhibition AggP_2	$\texttt{AggP}_2 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}[\texttt{AggP}_2][\texttt{Proteasome}]$
37c	Proteasome inhibition AggP_3	$\texttt{AggP}_3 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}[\texttt{AggP}_3][\texttt{Proteasome}]$
37d	Proteasome inhibition AggP_4	$\texttt{AggP}_4 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}[\texttt{AggP}_4][\texttt{Proteasome}]$
37e	Proteasome inhibition AggP_5	$\texttt{AggP}_5 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}[\texttt{AggP}_5][\texttt{Proteasome}]$
38a	ROS generation AggP_1	$\texttt{AggP}_1 \rightarrow \texttt{AggP}_1 + \texttt{ROS}$	$k_{genROSAggP}[\texttt{AggP}_1]$
38b	ROS generation AggP_2	$\texttt{AggP}_2 \rightarrow \texttt{AggP}_2 + \texttt{ROS}$	$k_{genROSAggP}[\texttt{AggP}_2]$
38c	ROS generation AggP_3	$\texttt{AggP}_3 \rightarrow \texttt{AggP}_3 + \texttt{ROS}$	$k_{genROSAggP}[\texttt{AggP}_3]$
38d	ROS generation AggP_4	$\texttt{AggP}_4 \rightarrow \texttt{AggP}_4 + \texttt{ROS}$	$k_{genROSAggP}[\texttt{AggP}_4]$
38e	ROS generation AggP_5	$\texttt{AggP}_5 \rightarrow \texttt{AggP}_5 + \texttt{ROS}$	$k_{genROSAggP}[\texttt{AggP}_5]$
39	p38 ROS generation	$\texttt{p38P} \rightarrow \texttt{p38P} + \texttt{ROS}$	$k_{genROSp38}k_{kp38act}[\texttt{p38P}]$
40	SeqAggP ROS generation	$\texttt{SeqAggP} \rightarrow \texttt{SeqAggP} + \texttt{ROS}$	$k_{genROSseqAggP}[\texttt{SeqAggP}]$
41	p38 cell death	$\texttt{p38P} \rightarrow \texttt{p38P} + \texttt{p38death}$	$k_{p38death}k_{p38act}[\texttt{p38P}]$
42	PI cell death	$\texttt{AggPProteasome} \rightarrow \texttt{AggPProteasome} + \texttt{PIdeath}$	$k_{PIdeath}[\texttt{AggPProteasome}]$

Table 2.5: List of reactions and hazards for the PolyQ model

Three example realisations from the model are given in Figure 2.3. It can be seen that the cells represented in blue and green both die via the `p38death` pathway and the cell represented in red dies via the `PIdeath` pathway.

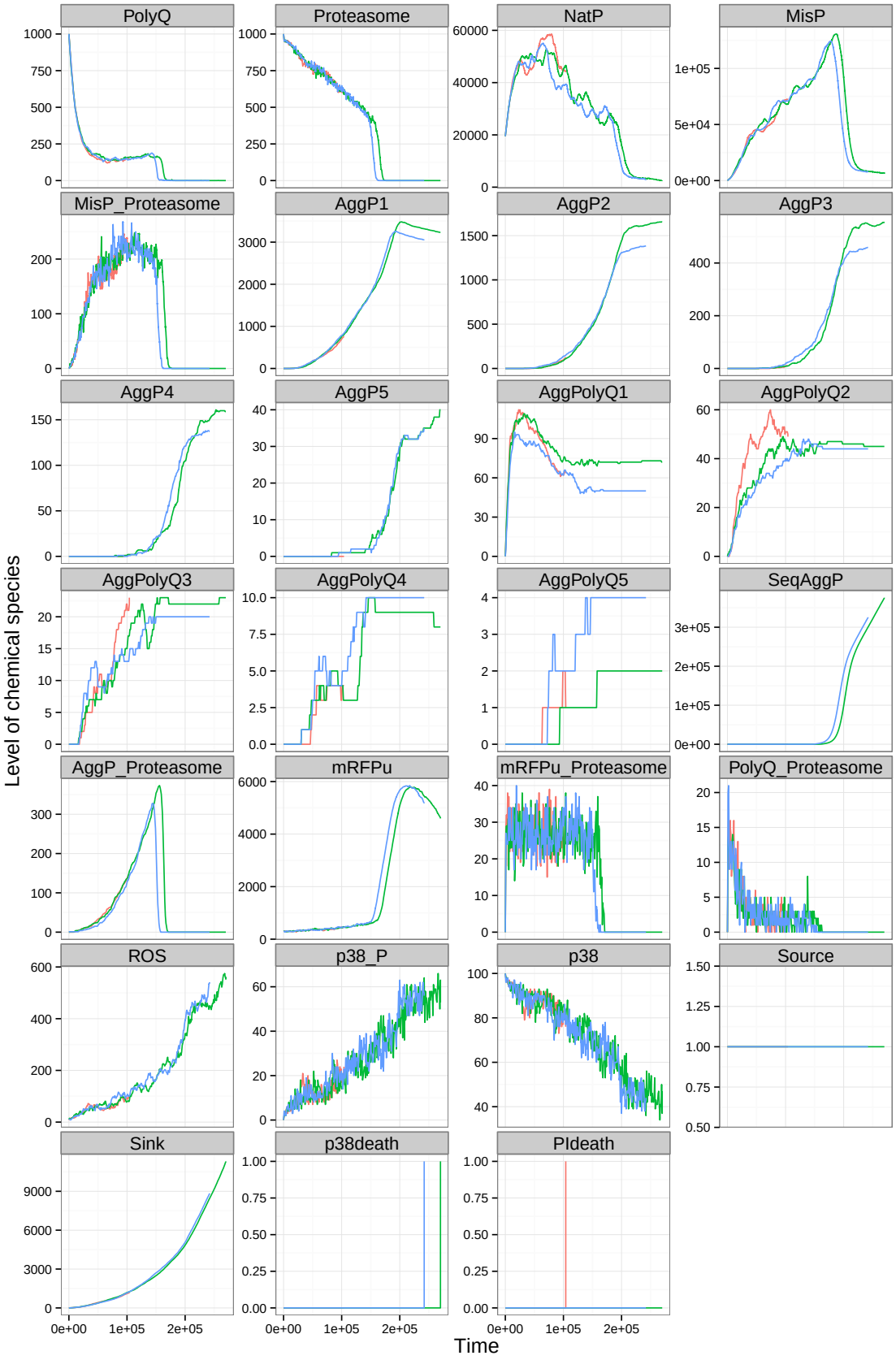

Figure 2.3: Three realisations from the PolyQ model

2.6 Other simulation strategies

For most models of interest, the Markov jump process is typically not analytically tractable. However, the simplicity of the birth–death means that it is possible to find an analytic expression for the transition probability; further details are given in Chapter 4. For the mtDNA and PolyQ models, no analytic results exist for the transition probabilities.

It is simple to simulate realisations from such models. The ability to simulate from a model allows it properties to be studied. Simulation strategies can broadly be divided into exact, approximate and hybrid methods.

2.6.1 Exact simulation strategies

Using an exact simulation strategy produces an exact realisation from the Markov jump process. The disadvantage of using exact simulations strategies is that they have the potential to be computationally expensive.

The Gibson-Bruck algorithm

The Gibson-Bruck algorithm (Gibson and Bruck, 2000) is another example of an exact simulation strategy. This is based on an alternative version of Gillespie's algorithm named the First Reaction Method (Gillespie, 1976) and is generally considered to be the fastest and most efficient exact method.

Gibson and Bruck (2000) use the notion of a *dependency graph* that has a vertex for each reaction. A directed edge is present between two vertices a and b if the occurrence of reaction a causes the state of the system to be altered in such a way that the hazard of reaction b is changed. This graph can then be used to update only the hazards which need to be updated after each reaction event, rather than all hazards.

2.6.2 Approximate simulation strategies

While exact methods for simulation such as the Gillespie algorithm are preferable, they have the potential to be very slow, especially when the model is complex. Large speed-ups can be gained by using an approximate method which still captures the vital kinetics of the model but does not necessarily simulate every reaction.

τ-leap method

The τ-leap method of Gillespie (2001) approximates the numbers of each type of reaction occurring in a small interval, by assuming they are independent Poisson random variables. Simulation proceeds by choosing a variable time interval τ and simulating the number of reactions of type i from a $Po(h_i(\boldsymbol{x}, \theta_i)\tau)$, for each reaction $i = 1, \ldots, v$. The time is then updated to $t := t + \tau$ and the states updated accordingly.

The accuracy of the algorithm depends on the size of τ chosen, smaller τ leads to a more accurate algorithm. For large τ, the assumption that the hazards are constant over the interval becomes less realistic. As a consequence, assuming that the number of occurrences of each type of reaction are independent Poisson random variables is less reliable. However, the algorithm will run faster for larger τ, thus τ represents a trade off between accuracy and speed.

For any interval τ, where at least one reaction has occurred, the assumption that the hazard is constant over the interval may not hold. This is because, the occurrence of any reaction changes the state of the system. Reactions with hazards above zero order depend on the state of the system. Consequently, a change in the state causes the hazard to change.

The size of τ chosen at each step is designed to ensure that the disruption to the assumption of constant hazard is within some acceptable tolerance which is a proportion of the cumulative hazard. The expected new states $\boldsymbol{x}^*$ after time τ can easily be calculated and the hazards at these expected new states evaluated. Gillespie (2001) suggests that the chosen τ should ensure that, for each reaction, the difference in hazard

over the interval τ is less than some fraction of the cumulative hazard

$$|h(\boldsymbol{x}^*, \theta_i) - h(\boldsymbol{x}, \theta_i)| \leq \epsilon h_0(\boldsymbol{x}, \boldsymbol{\theta}).$$

Several authors have proposed further methods for choosing τ including (Gillespie and Petzold, 2003; Cao et al., 2006). Sandmann (2009) gives a summary of the various extensions proposed to the basic τ-leap algorithm.

Chemical Langevin equation

The Chemical Langevin equation (CLE) approximates the Markov Jump process by an Itô stochastic differential equation

$$d\boldsymbol{x}_t = Sh(\boldsymbol{x}_t, \boldsymbol{\theta})dt + \sqrt{S\, diag\{h(\boldsymbol{x}_t, \boldsymbol{\theta})\}\, S'} dW_t.$$

In this equation, $\boldsymbol{x}_t$ represents the state of the system at time t, dW_t is an increment of standard Brownian motion, $Sh(\boldsymbol{x}_t, \boldsymbol{\theta})$ is the drift term and $h(\boldsymbol{x}_t, \boldsymbol{\theta})$ is a vector of the hazards $h(\boldsymbol{x}_t, \boldsymbol{\theta}) = [h_1(\boldsymbol{x}_t, \boldsymbol{\theta}), h_2(\boldsymbol{x}_t, \boldsymbol{\theta}), \ldots, h_v(\boldsymbol{x}_t, \boldsymbol{\theta})]'$. For details on the derivation of the CLE, see Gillespie (2000, 2001); Golightly and Wilkinson (2011).

For the birth–death model, the CLE is

$$dx_t = x(\lambda - \mu)dt + \sqrt{x(\lambda + \mu)}\, dW_t.$$

Although this assumes a continuous state approximation to the Markov process, the diffusion term ensures that the stochasticity of the system is retained. Simulating from the model can proceed by seeking a numerical solution such as the Euler-Maruyama approximation.

2.6.3 Hybrid simulation strategies

Hybrid simulation strategies are a compromise between exact and approximate algorithms. They acknowledge that for low copy numbers, a continuous approximation

which ignores the inherent discreteness is inappropriate. However they exploit the fact that for large copy numbers, a fast approximation is satisfactory.

The computational cost of running an exact simulation strategy is directly proportional to the number of reactions which happen in the system. If certain reactions happen very frequently then this can cause the algorithm to slow down significantly.

In general, hybrid algorithms class reactions as either *fast* or *slow*. This is typically done by partitioning the chemical species into those that must be modelled discretely and those which can be modelled by a continuous approximation. Any reaction involving at least one chemical species which is modelled discretely is labelled as a *slow* reaction and and all other reactions as *fast*.

The following gives an overview of a generic hybrid simulation scheme. At each iteration of the scheme, reactions are classified as *fast* or *slow* based on the current state of the system $\boldsymbol{x}_t$. For a chosen time-step δt, a path is sampled over $(t, t+\delta t)$ for the *fast* reactions using the fast approximation. The *slow* reaction hazards are then evaluated to decide whether or not a *slow* reaction happened in $(t, t+\delta t)$. If no *slow* reaction occurred in this time interval, then time is updated to $t := t + \delta t$ and the state of the system is updated according to the proposed values from the *fast* reactions. If one or more *slow* reaction does occur then let t' denote the time at which the first *slow* reaction happened. Update time to $t := t + t'$ and update the state of the system according to the first *slow* reaction.

In this setup, only time intervals in which *slow* reactions happen require exact simulation. Intervals where no slow reactions happen are simulated using the fast approximation, thus a saving in computational time is achieved.

One specific approach to hybrid simulation is to represent the *fast* reactions via an ODE which can be approximated numerically (Alfonsi et al., 2005; Kiehl et al., 2004). Other authors use the CLE to approximate the *fast* reactions (Salis and Kaznessis, 2005; Higham et al., 2011). Puchaka and Kierzek (2004) use a combination of the Gibson-Bruck algorithm for the *slow* reactions and the τ-leap algorithm for the *fast* reactions.

Chapter 3

Bayesian inference

The aim of this thesis is to perform statistical inference for the parameters of a large stochastic kinetic model, based on experimental data. Inference will be approached in the Bayesian framework. A particular problem with such methods is that the likelihood is intractable. The PolyQ model, which was introduced in Chapter 1, is an example of such a model. This chapter begins by introducing the notion of state–space models before giving an overview of the concepts behind Bayesian inference. Several schemes for implementing parameter inference are outlined.

3.1 State–space models

Consider a dynamical system consisting of states which change over time. State–space models describe the time evolution of such a system. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ represent states of the system where $\boldsymbol{x}_t$ is the value of the process at time t. In general, the states will depend on parameters $\boldsymbol{\theta}$. The states evolve according to the state equation

$$\pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \boldsymbol{\theta}) = \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{\theta})$$

and since $\boldsymbol{x}_{t+1}$ is dependent only on $\boldsymbol{x}_t$ and no other previous states, the evolution is Markovian.

Let $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$ be observations of unobserved latent states $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$. Con-

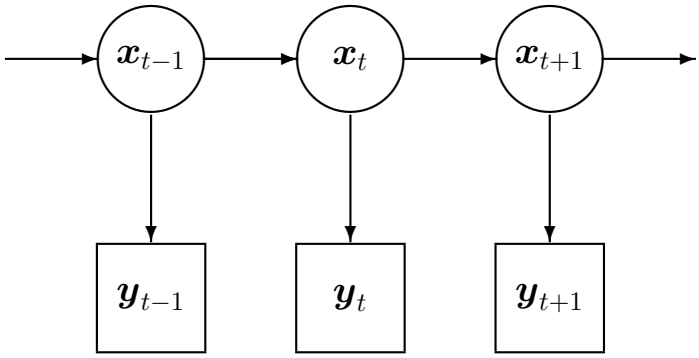

Figure 3.1: DAG representation of a state–space model.

sider the birth-death example – suppose that observing the exact population level is impossible. In this situation, the $\boldsymbol{x}_t$ remain unobserved, however the population level can be observed with some error; the noisy observations are the $\boldsymbol{y}_t$. For example, the error model could be Gaussian

$$\boldsymbol{y}_t|\boldsymbol{x}_t \sim N(\boldsymbol{x}_t, \sigma^2 I),$$

where I is the identity matrix of appropriate dimension. The observations $\boldsymbol{y}_t$ are conditionally independent given the states $\boldsymbol{x}_t$ and the parameters $\boldsymbol{\theta}$. The conditional distribution of $\boldsymbol{y}_t$ is

$$\pi(\boldsymbol{y}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{t-1}, \boldsymbol{\theta}) = \pi(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}).$$

That is, given the latent states, the observations are conditionally independent. This set up is pictured schematically in Figure 3.1.

3.2 Introduction to Bayesian inference

The goal is to quantify uncertainty about parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_p)'$ using observed data $\boldsymbol{y}$. Suppose $\boldsymbol{y}$ is modelled by some probability density function $f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta})$, then the

likelihood is defined as

$$L(\boldsymbol{\theta}|\boldsymbol{y}) = f_{\boldsymbol{y}}(\boldsymbol{y}|\boldsymbol{\theta}).$$

The likelihood represents the probability distribution of the data $\boldsymbol{y}$ as a function of the parameters $\boldsymbol{\theta}$. Prior beliefs about $\boldsymbol{\theta}$ are represented by the density $\pi(\boldsymbol{\theta})$ and Bayes' Theorem provides a way of updating these beliefs based upon observed data. The posterior distribution

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{y})}{\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{y})\ d\boldsymbol{\theta}}, \tag{3.1}$$

represents the updated beliefs about $\boldsymbol{\theta}$ after observing the data $\boldsymbol{y}$. Since the denominator of Equation 3.1 is not a function of $\boldsymbol{\theta}$, it can be regarded as a constant of proportionality and thus

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

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}.$$

3.3 Markov chain Monte Carlo (MCMC)

In general, obtaining the constant of proportionality (the denominator of Equation 3.1) is a non–trivial problem for anything but the very simplest of cases. It involves the integral

$$\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{y})\ d\boldsymbol{\theta}$$

which is often non–standard and multidimensional. Also of interest is calculating moments of the posterior distribution such as means, variances and marginal and conditional distributions. Markov chain Monte Carlo (MCMC) algorithms draw samples from the desired density, without knowledge of the normalising constant.

Metropolis–Hastings

The Metropolis–Hastings algorithm can be used to sample from the density of interest $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. The algorithm was developed by Metropolis et al. (1953) and later generalised

Algorithm 2 Metropolis–Hastings algorithm

Initialise the state of the chain $\boldsymbol{\theta}^{(0)}$.

For each iteration of the scheme:

1. Sample $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta})$ where q is some proposal distribution.
2. Compute the acceptance probability
$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\frac{\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{y}|\boldsymbol{\theta})}\frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}.\right\}$$
3. Set $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta})$, otherwise retain $\boldsymbol{\theta}$.

by Hastings (1970). The idea is to construct a Markov chain with stationary distribution equal to the target distribution; the details are given in Algorithm 2.

Step 1 of the algorithm generates a proposed parameter value denoted $\boldsymbol{\theta}^*$ from an easy to simulate from transition kernel, $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i-1)})$, known as the *proposal distribution*, which should have the same support as the target. At each iteration, a new value $\boldsymbol{\theta}^*$ is generated from the proposal distribution, this new value depends on the previous state of the chain $\boldsymbol{\theta}^{(i-1)}$. The proposed value is either accepted or rejected resulting in the chain either moving to $\boldsymbol{\theta}^*$ or staying at its current value. The accept/reject move depends on the acceptance probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i-1)})$, which in turn depends on the proposal distribution and $\pi(\cdot|\boldsymbol{y})$. Crucially, the dependence on $\pi(\cdot|\boldsymbol{y})$ is only in the form of a ratio, and therefore the target distribution only needs to be known up to a constant of proportionality.

Choice of proposal distribution

A special case of the Metropolis-Hastings algorithm arises when a symmetric proposal distribution is used

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*).$$

Here the ratio of proposal densities cancels and the acceptance probability simplifies to

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

In this case, whenever a $\boldsymbol{\theta}^*$ is proposed which moves the chain to an area of higher posterior density than previously, it will be accepted with certainty.

Another special case of the Metropolis–Hastings algorithm is the scenario in which the proposal distribution is chosen so that it does not depend on the current value of the chain, i.e. $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = f(\boldsymbol{\theta}^*)$ for some density f. Here the acceptance probability simplifies to

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

It can be seen that the acceptance ratio can be controlled by the similarity of $f(\cdot)$ and $\pi(\cdot|\boldsymbol{y})$. Choosing an $f(\cdot)$ that is very close to $\pi(\cdot)\pi(y|\cdot)$ will ensure a high acceptance probability.

If the proposal distribution q takes the form

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

where $\boldsymbol{\omega}$ are independent identically distributed random variates (known as innovations), then this special case of the Metropolis algorithm is known as a *random walk sampler*. Common choices of distribution for $\boldsymbol{\omega}$ are uniform and Gaussian, with mean zero.

Choice of tuning parameters

The *mixing* of the MCMC scheme refers to how well the chain moves around the space and consequently how long it takes for the chain to converge. The parameters that govern the distribution of ω will determine how well the chain mixes. Suppose ω follows

a multivariate normal distribution such that

$$\omega \sim N(\mathbf{0}, V),$$

then V must be carefully chosen to ensure good mixing. If the variance is too low then small moves will be proposed and the chain will explore the space too slowly. If the variance is too big then large moves will be proposed, most of which will be rejected meaning that chain will move too little. Taking into account the correlation in $\boldsymbol{\theta}$ by allowing V to have non–zero off–diagonal elements is important to ensure the space is explored efficiently.

It has been suggested that when the target distribution is Gaussian, an acceptance probability of 0.234 is optimal (Roberts and Rosenthal, 2001). However, this result has been extended to elliptically symmetric targets (Sherlock and Roberts, 2009) and more recently Sherlock (2013) give a general set of sufficient conditions under which an acceptance probability of 0.234 is optimal. Gelman et al. (1996); Roberts et al. (1997); Roberts and Rosenthal (2001) suggest that the random walk should be tuned such that

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

where Σ_π is the covariance matrix of the target distribution π and p is the dimension of $\boldsymbol{\theta}$. Although Σ_π is not typically available, an estimate can be obtained from one or more pilot runs of the scheme.

Analysis of MCMC output

To ensure a scheme such as Metropolis Hastings samples from the target probability distribution, convergence must be carefully monitored. Samples obtained before the chain has converged are known as *burn–in* and should be discarded. Convergence can be checked informally using graphical methods. These include viewing trace plots of the output to check for irregularities and using an autocorrelation plot to monitor the

autocorrelation in the chain at different lags.

More formal ideas for detecting convergence have been proposed by several authors including Gelman and Rubin (1992), who suggest initialising multiple chains in different places and checking they become indistinguishable after some time. There are a series of more formal checks for assessing convergence by other authors, see Heidelberger and Welch (1983) and Geweke (1992). If the autocorrelation is high then the chains can be *thinned* which involves keeping only every *ith* iteration, ensuring subsequent samples are independent. Raferty and Lewis (1992) give guidelines on how to pick the length of the *burn–in* to discard and by how much the chain should be *thinned* based on the user specifying how accurate they would like the posterior summaries to be.

Once the chain has reached convergence and it is sampling from the distribution of interest, the output can be analysed. It is trivial to compute estimates of summary statistics such as marginal means and variances. Plotting histograms or density plots gives an idea of the shape of marginal distributions.

3.4 Likelihood free inference

The motivation for likelihood free inference are models which have intractable likelihoods, for example the complex PolyQ model described in Chapter 1. In such cases, it is typically not possible to evaluate the likelihood although it is possible to simulate from the model. Recent interest has turned to methods for inference which do not require the likelihood to be evaluated and the ideas behind these methods date back to Diggle and Gratton (1984).

The general idea behind likelihood free inference involves utilising the ability to simulate from the model for the latent states, that is, obtain realisations from $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ for different choices of $\boldsymbol{\theta}$, without needing to evaluate this density. For example, using the Gillespie algorithm described in Section 2.2.2 of Chapter 2 it is simple to obtain realisations from the model for a given set of parameters and initial conditions. The simulated datasets are then compared to the observed data.

Approximate Bayesian computation (ABC) is an example of a likelihood-free method for inference and it was first applied (in its current form) to problems in population genetics by Tavaré et al. (1997) and Pritchard et al. (1999). The idea is to generate many synthetic datasets from the model for different parameter choices and compare the simulated data to the observed data. Typically, ABC schemes will compare summary statistics of the simulated datasets with summary statistics from the observed data. Crucially ABC methods do not generate samples from the true posterior, rather an approximate posterior which is believed to be similar.

The synthetic likelihood method is a non-Bayesian method introduced by Wood (2010) and has many similarities to ABC. This approach works by generating many datasets conditional on the proposed $\boldsymbol{\theta}$, then computing summary statistics for these datasets. The *synthetic* likelihood is defined as being a normal distribution with mean and variance equal to the mean and variance of the summary statistics. A Metropolis-Hastings scheme can be used to explore the *synthetic* likelihood.

3.4.1 Likelihood free MCMC

Consider the set up of Section 3.1, noisy observed data $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T)'$ arise from some process which depends on model parameters $\boldsymbol{\theta}$. The joint density of the model parameters and data is

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

where $\pi(\boldsymbol{\theta})$ is the prior distribution for $\boldsymbol{\theta}$ and $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ is the likelihood. Suppose interest lies in the posterior distribution

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

A marginal Metropolis-Hastings scheme to target the posterior $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ proceeds by proposing values from some proposal distribution $q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$ and accepting with probability

α where

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\frac{\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{y}|\boldsymbol{\theta})}\right\}. \tag{3.2}$$

Here, $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ is known as the *marginal likelihood* term since latent states $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)'$ have been integrated out

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}) = \int \pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\, d\boldsymbol{x}.$$

The problem with this approach is that the marginal likelihood term is often not available analytically. However, an alternative approach considers the augmented sample space which also includes the latent states $\boldsymbol{x}$. The joint distribution of the data, latent states and parameters is

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

Algorithm 3 gives a simple likelihood free MCMC approach which targets the joint posterior of the parameters and latent states. This approach will be referred to as the *naive* scheme when compared to schemes introduced later in the chapter.

Here $\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ describes the relationship between the latent unobserved states and the observed data. This term will be referred to as the observational error (or measurement error) term which is typically simple to evaluate. For the scheme to work efficiently, it is required that there is sufficient *noise* on the data such that this term is not so small that it leads to a negligible acceptance probability.

The proposal is constructed in two parts, first $\boldsymbol{\theta}^*$ is sampled from some proposal distribution q, and this is used to simulate $\boldsymbol{x}^*$ from the model $\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)$. This ensures that the $\boldsymbol{x}^*$ is consistent with $\boldsymbol{\theta}^*$ although not necessarily consistent with $\boldsymbol{y}$.

Note that the likelihood term in the acceptance ratio of Algorithm 3 takes the form

$$\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{t=1}^{T} \pi(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta})$$

for time course data. If T is large then $\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ can get very small, due to an $\boldsymbol{x}$

Algorithm 3 Likelihood free MCMC

For each iteration of the scheme:

1. Propose $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$.
2. Simulate $\boldsymbol{x}^* \sim \pi(\boldsymbol{x^*}|\boldsymbol{\theta}^*)$.
3. Compute the acceptance probability
$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\frac{\pi(\boldsymbol{y}|\boldsymbol{x}^*,\boldsymbol{\theta}^*)}{\pi(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}\frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$
4. Set $\boldsymbol{\theta}^* = \boldsymbol{\theta}$ with probability α, otherwise retain $\boldsymbol{\theta}$.

generated from a proposal mechanism independent of $\boldsymbol{y}$. This leads to a very low acceptance probability, and consequently, a badly mixing scheme. A way to avoid this problem is to use a sequential scheme which introduces a series of intermediate distributions, this will be explored in Section 3.4.3.

3.4.2 Pseudo–marginal approach

Consider the acceptance probability given in Equation 3.2

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

Evaluating $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ is typically difficult for models of reasonable complexity. Suppose a Monte Carlo estimate of $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ can be computed

$$\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\pi(\boldsymbol{y}|\boldsymbol{x}^{(i)},\boldsymbol{\theta})$$

where $\boldsymbol{x}^{(i)}$ are realisations of the latent states. This estimate could replace the intractable likelihood giving the new acceptance ratio

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

Algorithm 4 Pseudo–marginal approach

For each iteration of the scheme:

1. Propose $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$.
2. Calculate a suitable approximation $\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}^*)$ to the marginal likelihood $\pi(\boldsymbol{y}|\boldsymbol{\theta}^*)$.
3. Compute the acceptance probability
$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\frac{\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta})}\frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$
4. Set $\boldsymbol{\theta}^* = \boldsymbol{\theta}$ with probability α, otherwise retain $\boldsymbol{\theta}$.

The algorithm was first suggested by Beaumont (2003) and refined by Andrieu and Roberts (2009). It can be shown that this leads to a Markov–chain with stationary distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ as required, provided that $\widehat{\pi}$ has constant multiplicative bias which is independent of $\boldsymbol{\theta}$. In fact, when using a Monte Carlo estimate, showing that $\widehat{\pi}$ is unbiased is straightforward.

A potential drawback of this approach is that the approximation of the marginal likelihood can have large variance leading to a poorly mixing chain. The choice of N affects the variance of the approximation, with a larger value of N leading to a better estimate. Note that when $N = 1$, this scheme reduces to the *naive* scheme of Section 3.4.1 and when $N \to \infty$, the idealised scheme is obtained.

An alternative approach is to estimate the marginal likelihood using a particle filter. Since this produces an unbiased estimator, the scheme still samples from the true posterior. This is described in the next section.

The rational behind why the pseudo–marginal scheme works, begins by considering the augmented state space which includes all of the random variables $\boldsymbol{u}$ which are generated in the constructions of the estimate. The above acceptance probability can be rewritten by considering a Metropolis–Hastings scheme with a two–stage proposal:

(a) Propose $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$

(b) Propose $\boldsymbol{u}^* \sim \pi(\boldsymbol{u}^*|\boldsymbol{\theta}^*)$

and the acceptance probability becomes

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

It can be seen that the target distribution of this scheme is $\widehat{\pi}(\boldsymbol{\theta},\boldsymbol{u}|\boldsymbol{y})$ and marginalising over $\boldsymbol{u}$ gives

$$\begin{aligned}\int \widehat{\pi}(\boldsymbol{\theta},\boldsymbol{u}|\boldsymbol{y})\,du &\propto \int \pi(\boldsymbol{\theta})\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{u})\pi(\boldsymbol{u}|\boldsymbol{\theta})\,d\boldsymbol{u} \\ &\propto \pi(\boldsymbol{\theta})\int \widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{u})\pi(\boldsymbol{u}|\boldsymbol{\theta})\,d\boldsymbol{u} \\ &\propto \pi(\boldsymbol{\theta})\int E_{\boldsymbol{u}|\boldsymbol{\theta}}(\pi(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{u})) \\ &\propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{\theta}) \\ &\propto \pi(\boldsymbol{\theta}|\boldsymbol{y}).\end{aligned}$$

The scheme is targeting a joint density with marginal $\pi(\boldsymbol{\theta}|\boldsymbol{y})$, as required.

3.4.3 Particle filtering

A particle filter is a type of Sequential Monte Carlo (SMC) method which is a sequential analogue of the MCMC techniques previously discussed. These techniques are particularly useful for inference on state–space models which have a Markov structure. One major advantage of SMC methods is that the analysis does not need to be restarted for each new observation. This is particularly useful in applications where data arrive in real time.

Suppose observations $\boldsymbol{y}_{1:T} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T)$ are currently available, then the posterior distribution of interest, assuming fixed $\boldsymbol{\theta}$, is

$$\pi(\boldsymbol{x}_{1:T}|\boldsymbol{y}_{1:T}).$$

If a new observation $\boldsymbol{y}_{T+1}$ becomes available, the posterior distribution of interest is now $\pi(\boldsymbol{x}_{T+1}|\boldsymbol{y}_{1:T+1})$. Bayes' Theorem can be used to incorporate this new observation by updating the posterior distribution

$$\begin{aligned}\pi(\boldsymbol{x}_{T+1}|\boldsymbol{y}_{1:T+1}) &\propto \pi(\boldsymbol{x}_{T+1}|\boldsymbol{y}_{1:T})\pi(\boldsymbol{y}_{T+1}|\boldsymbol{x}_{T+1}) \\ &= \pi(\boldsymbol{y}_{T+1}|\boldsymbol{x}_{T+1}) \int \pi(\boldsymbol{x}_{T+1}|\boldsymbol{x}_T)\pi(\boldsymbol{x}_T|\boldsymbol{y}_{1:T})\, d\boldsymbol{x}_T.\end{aligned}$$

Notice that the final term in the integral is the posterior distribution at time T.

A particle filter can be used to target $\pi(\boldsymbol{x}_{T+1}|\boldsymbol{y}_{1:T+1})$. Before introducing particle filters it is necessary to be familiar with the concept of importance resampling, which is typically used in the implementation of the particle filtering algorithm.

Importance sampling and resampling

Importance resampling uses the principles of importance sampling: a technique for performing Monte Carlo integration. Suppose the integral of interest is

$$I = \int f(\boldsymbol{x})g(\boldsymbol{x})\, d\boldsymbol{x}.$$

Assume it is not possible to sample from $g(\boldsymbol{x})$, but it is possible to sample from a different distribution $q(\boldsymbol{x})$ which has the same support as $g(\boldsymbol{x})$. Now multiply the numerator and denominator by $q(\boldsymbol{x})$

$$I = \int f(\boldsymbol{x})g(\boldsymbol{x})\frac{q(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x} = \int \frac{f(\boldsymbol{x})g(\boldsymbol{x})}{q(\boldsymbol{x})} \times q(\boldsymbol{x})\, d\boldsymbol{x}.$$

Suppose a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is drawn from $q(\boldsymbol{x})$, then a Monte Carlo estimate of I can be constructed where

$$\widehat{I} = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\frac{g(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}.$$

The *importance weights* are defined to be

$$w_i = \frac{g(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$$

and these can be normalised to give

$$\tilde{w}_i = \frac{w_i}{\sum_{j=1}^{N} w_j}.$$

In importance resampling, these importance weights are used in a resampling step to construct a weighted sample from the original sample. This idea can be applied sequentially in a scheme known as Sequential Importance Resampling (SIR) and is the focus of the next section.

Sequential importance resampling (SIR) filter

The original particle filter was proposed by Gordon et al. (1993) and uses SIR. This algorithm is a recursive version of importance resampling and the full details are given in Algorithm 5.

Particle filters represent the posterior distribution at time t as a collection of points, $\{\boldsymbol{x}_t^{(i)}\}$ for $i = 1, \ldots, N$, known as *particles*. Sampling from these particles using a technique such as multinomial resampling results in an approximate sample, equally weighted from $\pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$. In general, when a new observation becomes available, the new target posterior distribution is

$$\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1}) \propto \pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t})\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1})$$

which is typically intractable. The particle filters approximation is given by

$$\widehat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1}) \propto \widehat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t})\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1})$$

Algorithm 5 Sequential importance resampling (SIR) filter

For time $t = 1, \ldots, T$
For each particle $\{\boldsymbol{x}_t^{(i)}\}_{i=1}^N$

1. Simulate $\boldsymbol{x}_{t+1}^{(i)}$ from $\pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)})$.
2. Calculate importance weights
$$w_i = \frac{\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(i)})}{\sum_{j=1}^N \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(j)})}.$$
3. Resample N times with replacement from the particles $\{\boldsymbol{x}_t^{(i)}\}_{i=1}^N$ using weights $\{w_i\}_{i=1}^N$.

where

$$\widehat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^N \pi\left(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}\right).$$

It is possible to sample from $\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t})$, by first taking the sample of particles approximately distributed according to $\pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ and *propagating forward*. This is done by forward simulating from the model i.e. using the Gillespie algorithm with parameters $\boldsymbol{\theta}$ and initial conditions $\boldsymbol{x}_t^{(i)}$.

Using the principles of importance resampling, the importance weights are $\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(i)})$ and the normalised weights are given by

$$\tilde{w}_i = \frac{\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(i)})}{\sum_{j=1}^N \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(j)})}.$$

To generate an approximate, equally weighted sample from $\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1})$, the particles $\{\boldsymbol{x}_{t+1}^{(i)}\}$ must be resampled such that they are weighted by the importance weights $\tilde{w}_i$.

The full algorithm is given in Algorithm 5 and is initialised by drawing an equally weighted sample of particles $\{\boldsymbol{x}_1^{(i)}\}$ from the prior distribution.

3.4.4 Application to pseudo–marginal approach

The particle marginal Metropolis-Hastings (PMMH) scheme of Andrieu et al. (2009, 2010) creates an SMC approximation to the marginal likelihood $\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$. At each iteration

of the MCMC scheme, the approximation is constructed by running a sequential scheme such as the SIR filter (Algorithm 5). This scheme targets the posterior distribution of the parameters and the latent states $\pi(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$.

A special case of the PMMH scheme is where the marginal posterior $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is targeted. In this case, the estimate of the marginal likelihood is constructed by noting that the average weight at time t gives an estimate of $\pi(\boldsymbol{y}_{t+1}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$,

$$\widehat{\pi}(\boldsymbol{y}_{t+1}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N} \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(j)}, \boldsymbol{\theta}),$$

which is the marginal likelihood given data up to time t. An estimate of the marginal likelihood given all data is the product of the average unnormalised weights

$$\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_{t=0}^{T-1} \left[\frac{1}{N}\sum_{j=1}^{N} \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}^{(j)}, \boldsymbol{\theta}) \right]. \tag{3.3}$$

It is shown by Del Moral (2004) that Equation 3.3 provides a consistent and unbiased estimate of the marginal likelihood. Given this property, a pseudo–marginal approach can be used with Equation 3.3 as an estimate of the marginal likelihood in Algorithm 4.

This scheme also reduces to the *naive* scheme in Section 3.4.1 when $N = 1$. Andrieu and Roberts (2009) note as the variance of the estimate of the marginal likelihood decreases, the mixing improves. The more particles used, the lower the variance of the estimate of marginal likelihood. Pitt et al. (2012) give guidelines on choosing the number of particles, they suggest that N should be chosen such that the variance of the log–posterior is 0.8464, although they suggest that anywhere in the range 0.25 - 2.25 will give only a small penalty. Doucet et al. (2012) suggest that the optimal value is approximately 1. **?**

3.4.5 Algorithm performance

When deciding to implement one of the methods for inference presented in this chapter, the efficiency of the scheme is of major importance. One method of quantifying the

respective efficiencies of MCMC schemes is to consider the *effective sample size* (ESS) of the resulting samples.

Consider the output of an MCMC run containing T samples, the ESS estimates the number of independent samples in the chain via

$$ESS = \frac{T}{1 + 2\sum_k \rho_k}$$

where ρ_k is the autocorrelation at lag k and the infinite sum will typically be truncated at some cutoff, such as $\rho_k < 0.05$. If there is no autocorrelation in the chain then the ESS will be equal to the length of the chain, T, this would be the optimal scenario. The computational time (CPU time) taken to run each scheme must also be considered. If one scheme takes twice as long to run as another, but produces double the number of independent samples, then the two schemes could be considered to be of equal efficiency. For this reason, algorithm performance is compared using the ESS normalised for CPU time.

Chapter 4

Numerical examples

This chapter considers two toy models, a model with no time dependence (constant model) and the birth–death process. These models will be used to illustrate the methods for inference outlined in Chapter 3. The advantage of testing methods on toy models before looking at more complicated models is that they are quick to simulate from and easier to work with. There are also certain analytic results which can be derived for these models, which are not available in larger models. This property can be exploited to compare approximate results to exact results, hence assess the performance of methods.

4.1 Constant model

Consider observed data $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)^T$ which are noisy proportions of cell death. The data have no time dependence; they represent a discretely observed, noisy version of a constant. When modelling proportions, a sensible choice is to work with the logit transformed data

$$\operatorname{logit} x = \log\left(\frac{x}{1-x}\right),$$

which take values in $\mathbb{R}$. Let $\mu = \operatorname{logit}\theta$ and assume the data model

$$y_i \equiv \operatorname{logit} x_i = \mu + \epsilon_i \quad i = 1, \ldots, T$$

where the ϵ_i are independent, $\epsilon_i \sim N(0, 1/\tau)$ and (μ, τ) are fixed but unknown parameters of interest. Note, in this section, the measurement error is defined via the precision $\tau = 1/\sigma^2$. While the error structure is normal on the logit scale, this translates to a logistic–normal distribution on the original scale. There exists no analytic solutions for the mean and variance of the logistic–normal distribution, see Aitchison (1986) for details.

The aim is to make inferences on (μ, τ) given observed data $\boldsymbol{y}$. This information is summarised by the posterior distribution

$$\pi(\mu, \tau|\boldsymbol{y}) \propto \pi(\mu)\pi(\tau)\pi(\boldsymbol{y}|\mu, \tau)$$

where

$$\pi(\boldsymbol{y}|\mu, \tau) = \prod_{i=1}^{T} \pi(y_i|\mu, \tau).$$

The likelihood is

$$\pi(\boldsymbol{y}|\mu, \tau) = \left(\frac{\tau}{2\pi}\right)^{n/2} \exp\left\{-\frac{n\tau}{2}[s^2 + (\bar{y} - \mu)^2]\right\}$$

where

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

Suppose *a priori*, beliefs about (μ, τ) are

$$\mu \sim N\left(b, \frac{1}{c}\right), \quad e < \mu < f \quad \text{and} \quad \tau \sim \Gamma(g, h), \quad \tau < k \tag{4.1}$$

where b, c, e, f, g, h and k are known constants, and μ and τ are independent. Here, the general case is considered, whereby prior beliefs may dictate that truncating either

μ or τ is necessary. The joint prior distribution is, for $e < \mu < f, \tau < k$

$$\begin{aligned}\pi(\mu,\tau) &= \pi(\mu)\pi(\tau) \\ &= \frac{\left(\frac{c}{2\pi}\right)^{1/2}\exp\left(\frac{-c(\mu-b)^2}{2}\right)}{\Phi(\sqrt{c}(f-b))-\Phi(\sqrt{c}(e-b))} \times \frac{h^g\tau^{g-1}e^{-h\tau}}{\Gamma(g)\int_0^k \frac{h^g\tau^{g-1}e^{-ht}}{\Gamma(g)}d\tau} \\ &\propto \tau^{g-1}\exp\left(-\frac{1}{2}[c(\mu-b)^2+2h\tau]\right).\end{aligned}$$

Bayes' Theorem (Equation 3.1) can be used to update beliefs about (μ, τ) after observing the data. For the particular form of the prior distribution chosen in Equation 4.1, the posterior is semi–conjugate and an explicit form for the conditional posteriors of μ and τ can be obtained.

The posterior density is, for $e < \mu < f, \tau < k$

$$\begin{aligned}\pi(\mu,\tau|\boldsymbol{y}) &\propto \pi(\mu,\tau)\pi(\boldsymbol{y}|\mu,\tau) \\ &\propto \tau^{g-1}\exp\left(-\frac{1}{2}[c(\mu-b)^2+2h\tau]\right)\times\left(\frac{\tau}{2\pi}\right)^{n/2}\exp\left[\frac{-n\tau}{2}\left\{s^2+(\bar{y}-\mu)^2\right\}\right] \\ &\propto \tau^{G-1}\exp\left(-\frac{1}{2}\left\{c(\mu-b)^2+\tau(2h+ns^2+n(\bar{y}-\mu)^2)\right\}\right).\end{aligned}$$

It may be the case that interest lies in the conditional distributions. The posterior for $\mu|\tau$ is

$$\mu|\tau,\boldsymbol{y} \sim N\left(\frac{A_\tau}{B_\tau},\frac{1}{B_\tau}\right) \qquad e<\mu<f,\ \ \tau<k$$

and for $\tau|\mu$ is

$$\tau|\mu,\boldsymbol{y} \sim \Gamma\left(G,H_\mu\right) \qquad e<\mu<f,\ \ \tau<k$$

where

$$A_\tau = cb + n\bar{y}\tau, \qquad B_\tau = c + n\tau$$
$$G = g + \frac{n}{2}, \qquad H_\mu = h + \frac{n(\bar{y}-\mu)^2}{2} + \frac{ns^2}{2}.$$

Interest may also lie in marginal posterior distributions. The marginal posterior for τ is, where $\tau < k$,

$$\begin{aligned}
\pi(\tau|\boldsymbol{y}) &= \int_e^f \pi(\mu,\tau|\boldsymbol{y})\,d\mu \\
&\propto \int_e^f \tau^{G-1}\exp\left(-\frac{1}{2}\left\{c(\mu-b)^2 + \tau(2h+ns^2+n(\bar{y}-\mu)^2)\right\}\right)d\mu \\
&\propto \frac{\tau^{G-1}}{\sqrt{B_\tau}}\exp\left(-\frac{1}{2}\left\{\tau(2h+ns^2+n\bar{y}^2) - \frac{A_\tau^2}{B_\tau}\right\}\right)\left[\Phi\left(\sqrt{B_\tau}\left(f-\frac{A_\tau}{B_\tau}\right)\right) - \Phi\left(\sqrt{B_\tau}\left(e-\frac{A_\tau}{B_\tau}\right)\right)\right]
\end{aligned}$$

and for μ is, where $e < \mu < f$,

$$\begin{aligned}
\pi(\mu|\boldsymbol{y}) &= \int_0^k \pi(\mu,\tau|\boldsymbol{y})\,d\tau \\
&\propto \int_0^k \tau^{G-1}\exp\left(-\frac{1}{2}\left\{c(\mu-b)^2 + \tau(2h+ns^2+n(\bar{y}-\mu)^2)\right\}\right)d\tau \\
&\propto \exp\left(-\frac{c(\mu-b)^2}{2}\right)H_\mu^{-G}\mathcal{G}(k|G,H_\mu)
\end{aligned}$$

where $\mathcal{G}(x|a,b) = Pr(\Gamma(a,b) \leq x)$.

The marginal posterior distributions for μ and τ are only known up to a constant of proportionality. However, a numerical integration technique such as the *composite trapezoidal rule* can be used to find this constant. For simulated data and particular choices of the prior parameters, marginal posterior distributions are given in the bottom plot of Figure 4.1 (dashed line).

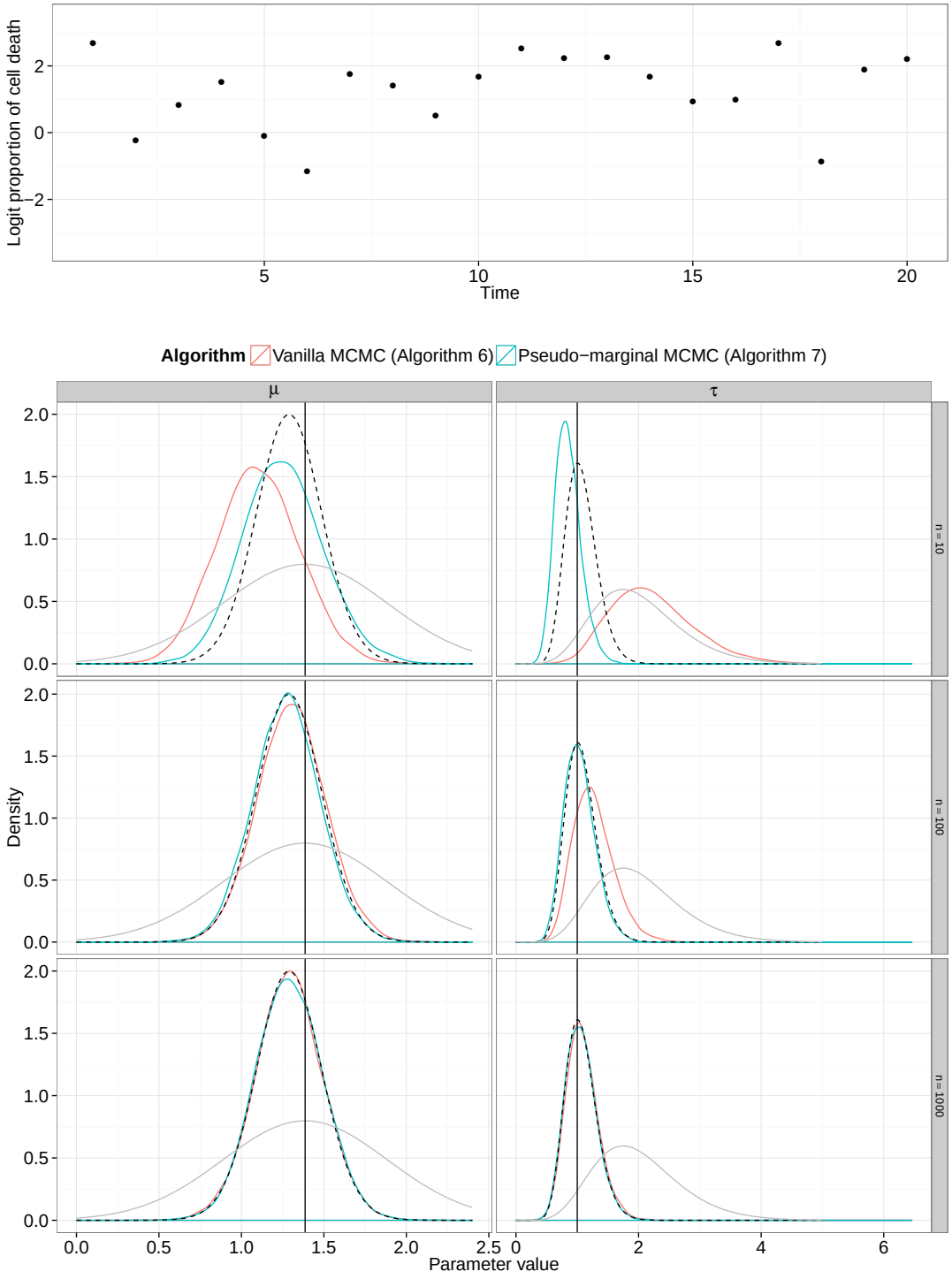

Figure 4.1: **Top**: simulated observed data for the constant model. **Bottom**: marginal posterior distributions for $\mu = \operatorname{logit}\theta$ and τ obtained using the *exact* (dashed), *vanilla* (red) and *pseudo-marginal* (blue) schemes. Light grey lines represent the prior distribution.

4.1.1 Constant model (with approximation)

Suppose the data model is now

$$y_i = \text{logit}\, x_i = \text{logit}\, p(\theta) + \epsilon_i \quad i = 1, \ldots, T \qquad \text{(independent)}$$

where the ϵ_i are independent, $\epsilon_i \sim N(0, 1/\tau)$ and (θ, τ) are fixed but unknown parameters of interest. The probability of cell death $p(\theta)$ is now a function of θ. It is important to consider this artificial scenario since it mimics the situation which arises when a more complicated model underpins the cell death process and it is not possible to find an analytic expression for the probability of cell death for given θ.

Of interest is inference on (θ, τ) where the posterior distribution is

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

where

$$\pi(\boldsymbol{y}|\theta, \tau) = \prod_{i=1}^{T} \pi(y_i|\theta, \tau).$$

The $\boldsymbol{p}(\theta) = [p_1(\theta), p_2(\theta), \ldots, p_T(\theta)]^T$ are unobserved latent states which can be approximated for any given θ by making T independent draws from the binomial distribution,

$$n\widehat{p}_{i,n}|p \sim \text{Bin}[n, p(\theta)] \quad i = 1, \ldots, T.$$

The simulated latent states are denoted

$$\widehat{\boldsymbol{p}}_n(\theta) = [\widehat{p}_{1,n}(\theta), \widehat{p}_{2,n}(\theta), \ldots, \widehat{p}_{T,n}(\theta)]^T,$$

where the n subscript describes how many independent draws from the Bernoulli distribution were used to obtain the approximation. The quality of this approximation depends on the value of n chosen. For small n, the approximation will be poor and as $n \to \infty$, the approximation tends towards the true proportion. Note, in a more

Algorithm 6 Constant model: inference using the *vanilla* scheme

Initialise the iteration counter and the state of the chain (θ, τ).

For each iteration of the scheme:

1. Sample $(\theta^*, \tau^*) \sim q(\cdot|\theta, \tau)$ from a symmetric proposal distribution q, on the log scale.
2. Simulate the path $\widehat{\boldsymbol{p}}_n^*(\theta^*)$ for some choice of n.
3. Compute the acceptance probability
$$\alpha = \min\left\{1, \frac{\pi(\theta^*)}{\pi(\theta)}\frac{\pi(\tau^*)}{\pi(\tau)}\frac{\tilde{\pi}(\boldsymbol{y}|\theta^*, \tau^*)}{\tilde{\pi}(\boldsymbol{y}|\theta, \tau)}\frac{\theta^*}{\theta}\frac{\tau^*}{\tau}\right\}.$$
4. Set $(\theta, \tau) = (\theta^*, \tau^*)$ with probability α, otherwise retain (θ, τ).

complicated model where $p(\theta)$ is unobserved, the $\widehat{p}_n(\theta)$ would be generated using n forward simulations from the model.

Algorithms 6 and 7 present two schemes for inference on (θ, τ). These are MCMC schemes which are based on the Metropolis-Hastings algorithm outlined in Algorithm 2 of Chapter 3. In both schemes, values of (θ^*, τ^*) are drawn from a symmetric proposal distribution and used to simulate a realisation $\widehat{\boldsymbol{p}}_n(\theta^*)$. To calculate the marginal likelihood for the data, it is necessary to marginalise over the simulated unobserved latent states

$$\tilde{\pi}(\boldsymbol{y}|\theta, \tau) \simeq \int \pi(\boldsymbol{y}|\widehat{\boldsymbol{p}}_n, \tau)\pi(\widehat{\boldsymbol{p}}_n|\theta)\, d\widehat{\boldsymbol{p}}_n$$

Also, as the components of $\widehat{\boldsymbol{p}}_n$ have independent normal measurement errors, its joint density can be written as

$$\pi(\widehat{\boldsymbol{p}}_n|\theta) = \prod_{i=1}^{T} \pi(\widehat{p}_{i,n}|\theta). \tag{4.2}$$

Note that Algorithm 6 uses the normal approximation to the distribution of the empirical logit of the $\widehat{\boldsymbol{p}}_n$ where

$$\operatorname{elogit} \widehat{p}_{i,n}(\theta) = \log\left(\frac{n\widehat{p}_{i,n}(\theta) + 1/2}{n - n\widehat{p}_{i,n}(\theta) + 1/2}\right) \quad i = 1, \ldots, T$$

Algorithm 7 Constant model: inference using the *pseudo-marginal* scheme

Initialise the iteration counter and the state of the chain (θ, τ).

For each iteration of the scheme:

1. Sample $(\theta^*, \tau^*) \sim q(\cdot|\theta, \tau)$ from a symmetric proposal distribution q, on the log scale.
2. For each particle $1, \ldots, N$, simulate a path $\widehat{\boldsymbol{p}}_n^{(1)}(\theta^*), \widehat{\boldsymbol{p}}_n^{(2)}(\theta^*), \ldots, \widehat{\boldsymbol{p}}_n^{(N)}(\theta^*)$ for some choice of n.
3. Construct a Monte Carlo, unbiased estimate of $\pi(\boldsymbol{y}|\theta^*, \tau^*)$

$$\widehat{\pi}(\boldsymbol{y}|\theta^*, \tau^*) = \frac{1}{N}\sum_{j=1}^{N} \pi(\boldsymbol{y}|\widehat{\boldsymbol{p}}_n^{(j)}(\theta^*), \tau^*)$$

   where

$$\pi(\boldsymbol{y}|\widehat{\boldsymbol{p}}_n^{(j)}(\theta^*), \tau^*) = \prod_{i=1}^{T} \pi(y_t|\widehat{p}_{i,n}^{(j)}(\theta^*), \tau^*)$$

4. Compute the acceptance probability

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

5. Set $(\theta, \tau) = (\theta^*, \tau^*)$ with probability α, otherwise retain (θ, τ).

and, for large n

$$\text{elogit}\,\widehat{p}_{i,n}|\theta \sim N\left(\text{logit}\,p_{i,n}, \frac{1}{np_{i,n}[1-p_{i,n}]}\right). \tag{4.3}$$

This result can be shown as follows. Suppose $\widehat{p}_n$ is the proportion of successes out of n independent trials. Then $n\widehat{p}|p \sim \text{Bin}(n, p)$ and let $n\widehat{p}_n = np + \sqrt{np(1-p)}U_n$, with $E(U_n) = 0$, $Var(U_n) = 1$, $E(U_n^2) = 1$. Also, $U_n \to N(0,1)$ as $n \to \infty$. Now

$$\begin{aligned}\frac{n\widehat{p}_n + \frac{1}{2}}{n - n\widehat{p}_n + \frac{1}{2}} &= \frac{np + \sqrt{np(1-p)}U_n + \frac{1}{2}}{n - np - \sqrt{np(1-p)}U_n + \frac{1}{2}} \\ &= \frac{p}{1-p} \times \frac{1 + \sqrt{\frac{(1-p)}{np}}\,U_n + \frac{1}{2np}}{1 - \sqrt{\frac{p}{n(1-p)}}\,U_n + \frac{1}{2n(1-p)}}\end{aligned}$$

and so

$$\begin{aligned}
\operatorname{elogit}\widehat{p}_n &= \log\left(\frac{n\widehat{p}_n+\frac{1}{2}}{n-n\widehat{p}_n+\frac{1}{2}}\right)\\
&= \operatorname{logit} p + \log\left(1+\sqrt{\frac{(1-p)}{np}}\,U_n+\frac{1}{2np}\right)-\log\left(1-\sqrt{\frac{p}{n(1-p)}}\,U_n+\frac{1}{2n(1-p)}\right)\\
&= \operatorname{logit} p + \sqrt{\frac{(1-p)}{np}}\,U_n+\frac{1}{2np}-\frac{(1-p)}{2np}\,U_n^2\\
&\qquad -\left\{-\sqrt{\frac{p}{n(1-p)}}\,U_n+\frac{1}{2n(1-p)}-\frac{p}{n(1-p)}\,U_n^2\right\}+O(n^{-3/2})\\
&= \operatorname{logit} p + \frac{U_n}{\sqrt{p(1-p)}}\,n^{-1/2}+\frac{(1-2p)(1-U_n^2)}{2p(1-p)}\,n^{-1}+O(n^{-3/2}).
\end{aligned}$$

Therefore

$$E(\operatorname{elogit}\widehat{p}_n)=\operatorname{logit} p+O(n^{-3/2}) \qquad \text{and} \qquad Var(\operatorname{elogit}\widehat{p}_n)=\frac{1}{np(1-p)}+O(n^{-3/2}).$$

Also, as $U_n \to N(0,1)$ as $n\to\infty$, for large n

$$\operatorname{elogit}\widehat{p}_n \sim N\left(\operatorname{logit} p,\ \frac{1}{np(1-p)}\right) \qquad \text{approximately.}$$

The scheme in Algorithm 6 will be referred to as the *vanilla* scheme throughout. For the *vanilla* scheme, the marginal likelihood is evaluated directly using the results in Equation 4.3 and 4.2. Since this result is approximate, this scheme is not exact and the success of the scheme depends on the strength of the approximation in Equation 4.3.

Algorithm 7 is a *pseudo-marginal* scheme and is based on Algorithm 4 of Chapter 3. At each iteration of the scheme, N realisations (*particles*) of the $\widehat{\boldsymbol{p}}_n$ are simulated

$$\begin{bmatrix} \widehat{\boldsymbol{p}}_n^{(1)}(\theta^*) \\ \vdots \\ \widehat{\boldsymbol{p}}_n^{(j)}(\theta^*) \\ \vdots \\ \widehat{\boldsymbol{p}}_n^{(N)}(\theta^*) \end{bmatrix} = \begin{bmatrix} \widehat{p}_{t_1,n}^{(1)}(\theta^*) & \dots & \widehat{p}_{t_i,n}^{(1)}(\theta^*) & \dots & \widehat{p}_{t_T,n}^{(1)}(\theta^*) \\ \vdots & & \vdots & & \vdots \\ \widehat{p}_{t_1,n}^{(i)}(\theta^*) & \dots & \widehat{p}_{t_i,n}^{(i)}(\theta^*) & \dots & \widehat{p}_{t_T,n}^{(i)}(\theta^*) \\ \vdots & & \vdots & & \vdots \\ \widehat{p}_{t_1,n}^{(N)}(\theta^*) & \dots & \widehat{p}_{t_i,n}^{(N)}(\theta^*) & \dots & \widehat{p}_{t_T,n}^{(N)}(\theta^*) \end{bmatrix}.$$

These *particles* are used to construct an unbiased estimate of the marginal likelihood

$$\widehat{\pi}(\boldsymbol{y}|\theta,\tau) = \frac{1}{N}\sum_{j=1}^{N}\pi(\boldsymbol{y}|\boldsymbol{p}_n^{(j)},\tau).$$

This estimate of the marginal likelihood is used in place of $\pi(\boldsymbol{y}|\theta,\tau)$ in the acceptance ratio and since approximation is unbiased, the exact posterior is targeted.

For the *vanilla* scheme, at each iteration T draws from the binomial distribution are required, one for each data point. In the *pseudo-marginal* scheme, $N \times T$ draws are required from the binomial distribution for each iteration of the scheme. For this model, the simulation strategy is very simple and quick. However, a more complex model will be slower to simulate from; hence, keeping the number of simulations to a minimum will be of greater importance.

4.1.2 Constant model: results

Data were simulated from the model with $\theta = 0.8$, $\tau = 1$ and $T = 20$; the data are presented in the top plot of Figure 4.1 (page 45). The results of inference on this data using Algorithms 6 and 7 with prior distributions

$$\mu = \operatorname{logit}\theta \sim N\left(\operatorname{logit}0.8, \frac{1}{4}\right) \quad 0 < \mu < 5 \quad \text{and} \quad \tau \sim \Gamma(8,4) \quad \tau < 10$$

can be seen in the bottom plot.

The exact marginal posterior distributions of $\mu = \operatorname{logit}\theta$ and τ can both be obtained up to a constant of proportionality using the expressions calculated in Section 4.1. Numerical integration was used to obtain these constants. These marginal posterior distributions are shown in dotted lines in Figure 4.1 and will be known as the *exact* case; also shown are the results of running Algorithms 6 and 7 for different choices of n.

The performance of the algorithms can be assessed informally by observing how close the resulting posterior distributions are to the *exact* case in Figure 4.1. Firstly, it can be noted that as n increases, both the *vanilla* and *pseudo-marginal* schemes become

indistinguishable from the exact case, this happens when $n \geq 1000$. The *pseudo-marginal* approach seems to do better than the *vanilla* approach, especially when n is small. This is as expected, since the approximation used in the *vanilla* approach (Equation 4.3) is dependent on n being large.

4.2 Birth–death model

The simple birth–death process was introduced in Chapter 2. The model is more complex than the constant model, although it is still possible to obtain an analytic expression for the transition probabilities. The chemical master equations (CME) is

$$\frac{dP_x(t)}{dt} = \lambda(x-1)P_{x-1}(t) + \mu(x+1)P_{x+1}(t) - x(\lambda+\mu)P_x(t). \tag{4.4}$$

This probability generating function is given by

$$Q(z;t) = \sum_{x=0}^{\infty} P_x(t)z^x$$

and it follows that

$$\frac{\partial}{\partial t}Q(z;t) = \sum_{x=0}^{\infty} \frac{\partial}{\partial t}P_x(t)z^x. \tag{4.5}$$

Equation 4.4 can be written in terms of the probability generating function by multiplying both sides by z^x and summing over x, giving

$$\frac{\partial}{\partial t}\sum_{x=0}^{\infty} z^x P_x(t) = \sum_{x=0}^{\infty} z^x \left\{\lambda(x-1)P_{x-1}(t) + \mu(x+1)P_{x+1}(t) - x(\lambda+\mu)P_x(t)\right\}$$

which satisfies the partial differential equation (p.d.e.)

$$\frac{\partial Q}{\partial t} = (\lambda z - \mu)(z-1)\frac{\partial Q}{\partial z}.$$

Using Lagrange's method to solve this p.d.e. gives

$$Q(z;t) = \left\{ \frac{\mu(1-z) - (\mu - \lambda z)e^{(\mu-\lambda)t}}{\lambda(1-z) - (\mu - \lambda z)e^{(\mu-\lambda)t}} \right\}^{x_0}$$

where x_0 is the initial population level. From this p.g.f, the mean and variance of the population size at time t can be determined

$$\begin{aligned} E[x(t)] &= Q'(1;t) = x_0 e^{(\mu-\lambda)t} \\ \text{Var}[x(t)] &= Q''(1;t) + Q'(1;t)[1 - Q'(1;t)] = \frac{x_0(\lambda+\mu)}{\lambda-\mu} e^{(\lambda-\mu)t}(e^{(\lambda-\mu)t} - 1). \end{aligned} \tag{4.6}$$

The probability of x individuals at time t, $P_x(t)$ can be obtained by expanding $Q(z;t)$ in powers of z^n, see Bailey (1964) and Renshaw (1993) for a fuller discussion. The full expression is omitted as it is cumbersome. However, for the special case when $x(t) = 0$ and the population becomes extinct, it simplifies to

$$P_0(t) = \begin{cases} \left(\frac{\mu - \mu e^{(\mu-\lambda)t}}{\lambda - \mu e^{(\mu-\lambda)t)}} \right)^{x_0}, & \text{for } \lambda \neq \mu \\ \left(\frac{\lambda t}{1+\lambda t} \right)^{x_0}, & \text{for } \lambda = \mu. \end{cases} \tag{4.7}$$

$P_0(t)$ can be thought of as being the probability of extinction happening in the interval $[0, t]$. Letting $t \to \infty$ gives the overall probability of extinction

$$P_0(\infty) = \begin{cases} (\mu/\lambda)^{x_0}, & \text{for } \lambda > \mu \\ 1, & \text{for } \lambda \leq \mu. \end{cases} \tag{4.8}$$

Using Equation 4.7, the probability density of extinction can be determined

$$p_0(t) = \begin{cases} \frac{x_0(\lambda-\mu)^2 e^{t(\mu-\lambda)}(\lambda - \mu e^{t(\mu-\lambda)})^{-(x_0+1)}(\mu - \mu e^{t(\mu-\lambda)})^{x_0}}{1 - e^{t(\mu-\lambda)}}, & \text{for } \lambda \neq \mu \\ \frac{x_0 \lambda^{x_0} t^{x_0-1}}{(1+\lambda t)^{x_0+1}}, & \text{for } \lambda = \mu. \end{cases} \tag{4.9}$$

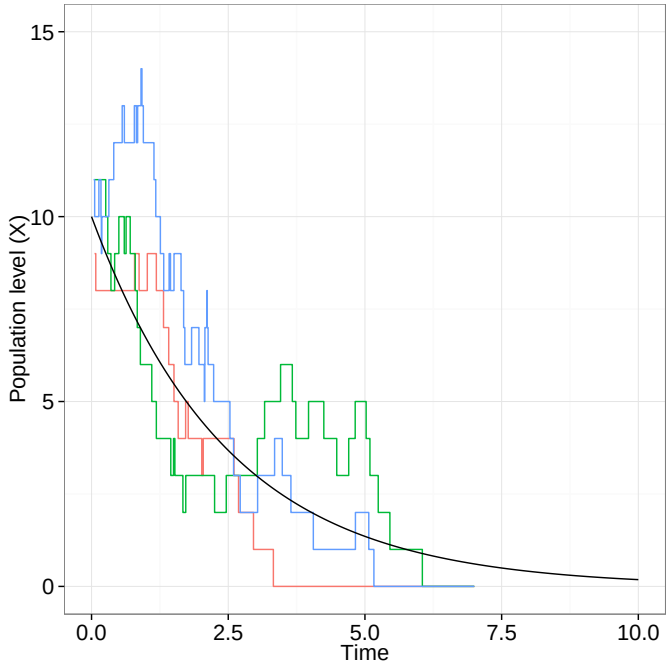

Figure 4.2: Three simulations of the birth–death process with $\lambda = 0.6, \mu = 1$ and $x_0 = 10$ along with the mean (black line).

4.2.1 Simulating from the model

The time evolution of a population governed by a birth–death process (with a particular choice of λ, μ and x_0) can be simulated using a stochastic simulation algorithm such as the Gillespie algorithm (see Algorithm 1 of Chapter 2). Since the model is stochastic, each simulation will be different; this is illustrated in Figure 4.2 where three typical trajectories are shown along with the mean (given in Equation 4.6).

A set of extinction times could be obtained by simulating the trajectories of several populations and recording the times at which they became extinct. However, since the cumulative distribution function (c.d.f.) for extinction times (Equation 4.7) is known and invertible, the *inversion sampling* method provides a more convenient method of doing this. To simulate a time of extinction, t:

1. Calculate the overall probability of extinction

$$P_0(\infty) = \begin{cases} (\mu/\lambda)^{x_0}, & \text{for } \lambda > \mu \\ 1, & \text{for } \lambda \leq \mu. \end{cases}$$

2. Generate a realisation $u \sim U(0,1)$.

3. If $u < P_0(\infty)$, compute

$$t = \begin{cases} \frac{1}{\mu-\lambda} \log\left(\frac{\lambda u^{1/x_0} - \mu}{\mu u^{1/x_0} - \mu}\right), & \text{for } \lambda \neq \mu \\ \frac{u^{1/x_0}}{\lambda(1-u^{1/x_0})}, & \text{for } \lambda = \mu, \end{cases}$$

   otherwise $t = \infty$.

Simulating n populations each governed by a birth–death process and noting when each population becomes extinct gives an estimate of the proportion of extinction through time. An example of this can be seen in Figure 4.3. In this plot, the green line is the exact proportion of extinction through time. The other lines represents the approximate proportion of extinction gained from using a simulator with $n = (10, 10^2, 10^3, 10^4)$. For low n, the approximation is very poor, however, when $n = 10^4$, the simulator output is virtually indistinguishable from the exact values.

In the subsequent sections, inference is described for data in which the underlying process is a birth–death process. Several situations for how the data are observed are discussed. In each scenario, it is assumed that there are several populations each governed by a birth–death process. In Section 4.2.2 is it assumed that the extinction time for each population is observed exactly. Section 4.2.3 assumes that extinction times are observed at discrete intervals, this represents a more plausible real life scenario. Of interest is the impact on the amount information learnt when data are observed on a finer grid. Section 4.2.4 considers the case in which the number of extinctions are observed with noise, thus the true proportions are unobserved latent states.

The MCMC schemes for inference are described in detail in Algorithms 8–12. For

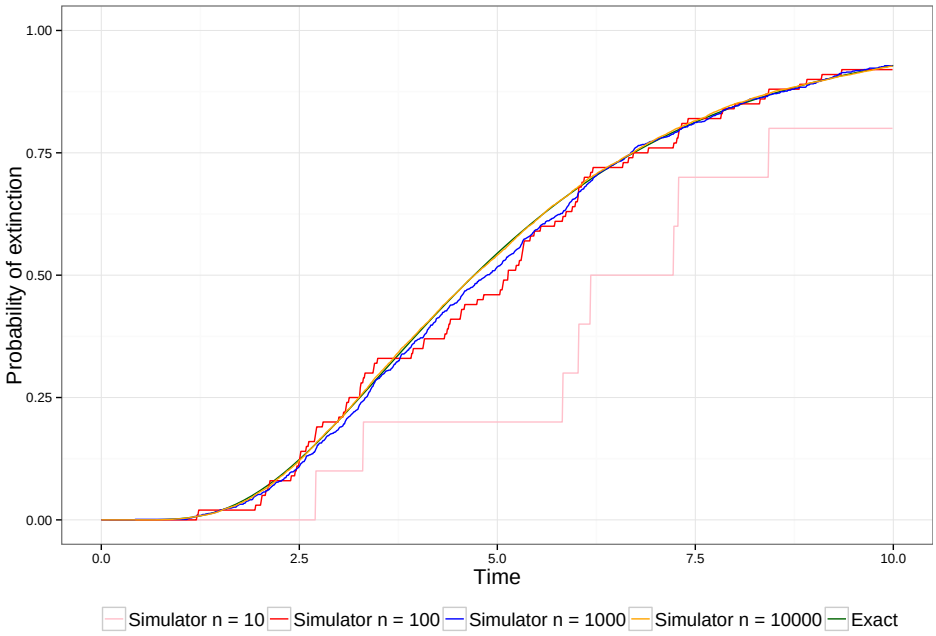

Figure 4.3: Realisations of the proportion of extinction in the birth–death process with $\lambda = 0.6, \mu = 1, x_0 = 10$ for different n.

each scheme, the proposal distribution q for the random walk is multivariate normal on the log scale. In each case, the random walk is tuned using the method described in Section 3.3 of Chapter 3. This will be the case for all subsequent MCMC schemes in this thesis, unless otherwise stated.

4.2.2 Inference for known extinction times

Suppose there are n populations, each of which is governed by a birth–death process with parameters $\boldsymbol{\theta} = (\lambda, \mu)$. The time each population becomes extinct is recorded (exactly) and denoted

$$\boldsymbol{t} = (t_1, t_2, t_3, \ldots, t_n)$$

where t_i is the time at which population i became extinct. Given that inference is required on $\boldsymbol{\theta}$, the likelihood is

$$\pi(\boldsymbol{t}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p_0(t_i, \boldsymbol{\theta}) \tag{4.10}$$

Algorithm 8 Birth–death model: inference using known extinction times

Initialise the iteration counter and the state of the chain $\boldsymbol{\theta}$.

For each iteration of the scheme:

1. Sample $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta})$ from a symmetric proposal distribution q, on the log scale.
2. Compute the acceptance probability

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \prod_{i=1}^{n} \frac{p_0(t_i, \boldsymbol{\theta^*})}{p_0(t_i, \boldsymbol{\theta})} \prod_{i=1}^{p} \frac{\theta_i^*}{\theta_i}\right\}.$$

3. Set $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ with probability α, otherwise retain $\boldsymbol{\theta}$.

where p_0 is given in Equation 4.9. The posterior distribution of interest is

$$\pi(\boldsymbol{\theta}|\boldsymbol{t}) \propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{t}|\boldsymbol{\theta}) \tag{4.11}$$

where $\pi(\boldsymbol{\theta})$ is the prior distribution for $\boldsymbol{\theta}$.

Implementation

It was assumed *a priori* that $x_0 = 10$, λ and μ were independent and

$$\lambda \sim \text{Log-Normal}(\log 0.6, 2) \qquad \text{and} \qquad \mu \sim \text{Log-Normal}(\log 1, 2). \tag{4.12}$$

These posterior distributions were chosen to represent vague prior information about parameters. The posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{t})$ does not have recognisable form but can be targeted with an MCMC scheme outlined in Algorithm 8. This scheme is based on the Metropolis-Hastings scheme which was first introduced in Chapter 3 (Algorithm 2).

Three different datasets $\boldsymbol{t}$ were simulated using the algorithm in Section 4.2.1 and using $n = (10, 100, 1000)$ forward simulations from the model, each with $\lambda = 0.6$ and $\mu = 1$. Inference was performed for each dataset using Algorithm 8.

The marginal posterior distributions (pink) along with the joint posterior (blue) are shown in Figure 4.4, where each row corresponds to a different dataset. It appears that

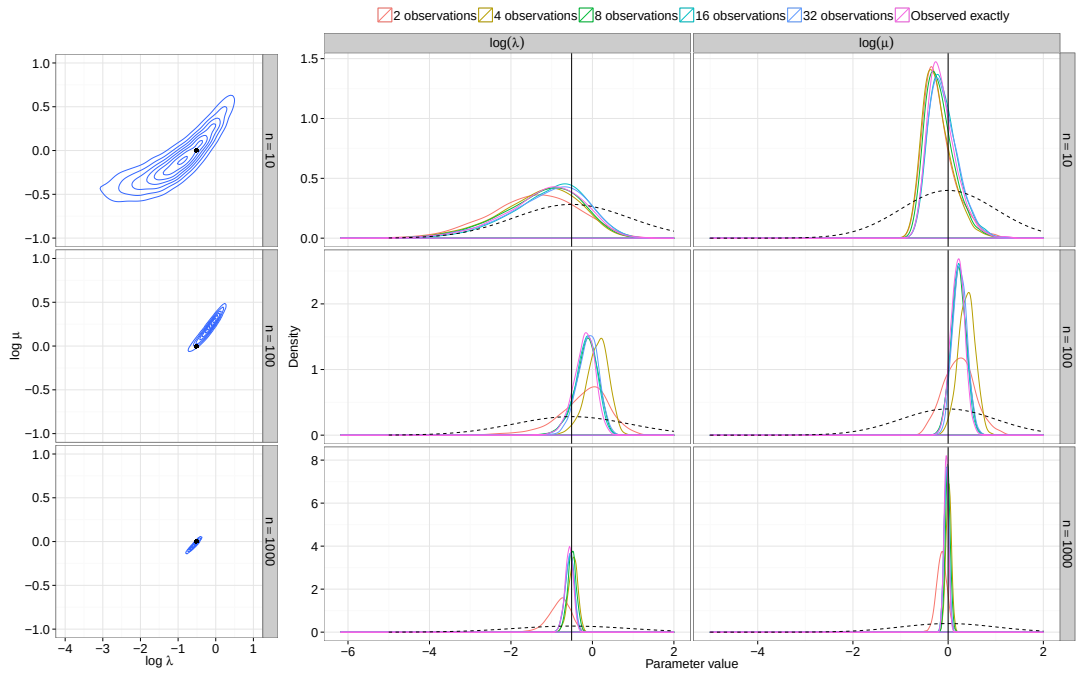

Figure 4.4: Left: joint posterior distribution for (log λ, log μ). Black points represent true values used to simulate data. Right: marginal posterior distributions for log λ and log μ. Vertical black lines represent the true values used to simulate the data and the dashed black lines represent the prior distribution.

for each dataset, it is possible to recover the true parameter values with the posterior means being being very close to the true value even when only 10 populations are observed. As would be expected, as the number of populations increases, more is learnt and the posterior become more concentrated.

4.2.3 Inference for discretised extinction times

Suppose now that the exact time of extinction is not observed, rather extinction status is observed at discrete time points $t_1, t_2, \ldots, t_T$ where $t_1 < t_2 < \ldots < t_T$. When an extinction time t_i is recorded, this corresponds to a population which became extinct sometime in the interval $(t_{i-1}, t_i]$. The set of observed data is denoted

$$\boldsymbol{t} = (n_1^d, n_2^d, n_3^d, \ldots, n_T^d)$$

Algorithm 9 Birth–death model: inference using discretised extinction times

Initialise the iteration counter and the state of the chain $\boldsymbol{\theta}$.

For each iteration of the scheme:

1. Sample $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta})$ from a symmetric proposal distribution q, on the log scale.
2. Compute the acceptance probability

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\prod_{i=1}^{T}\left[\frac{P_0(t_i,\boldsymbol{\theta^*}) - P_0(t_{i-1},\boldsymbol{\theta^*})}{P_0(t_i,\boldsymbol{\theta}) - P_0(t_{i-1},\boldsymbol{\theta})}\right]^{n_i^d}\prod_{i=1}^{p}\frac{\theta_i^*}{\theta_i}\right\}$$

3. Set $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ with probability α, otherwise retain $\boldsymbol{\theta}$.

where n_i^d represents the number of populations that become extinct in the period $(t_{i-1}, t_i]$. The likelihood is now

$$\pi_d(\boldsymbol{t}|\boldsymbol{\theta}) = \prod_{i=1}^{T}[P_0(t_i) - P_0(t_{i-1})]^{n_i^d}. \tag{4.13}$$

where P_0 is given in Equation 4.7.

Implementation

The same prior distributions for parameters $\boldsymbol{\theta} = (\lambda, \mu)$ are used as Section 4.2.2. The posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{t})$ is again intractable thus an MCMC algorithm the same as that of Section 4.2.2 is used with the exception that the likelihood is now Equation 4.13 (see Algorithm 9).

The datasets in Section 4.2.2 are discretised to produce data of this form. It was assumed the observation times were in $t = (0, 10]$ with five different discretisations, which corresponded to time steps of $(5, 2.5, 1.25, 0.625, 0.3125)$. This resulted in datasets with $T = (2, 2^2, 2^3, 2^4, 2^5)$. These values were chosen such that the number of observations doubled each time.

The marginal posterior distributions are shown in Figure 4.4. The results suggest that only observing extinction times at discrete time points makes very little difference to the inference on λ and μ. The censoring of the observations has led to very little loss

of information about rate parameters of the process. Even with only two observations, posterior distributions of the rate parameters are centered around the correct region with variance only slightly larger than obtained when the extinction times are observed exactly. The nature of the process means that the proportion of extinct populations is monotonically increasing. As a result of this, observing the process at the beginning, middle and end captures most of the vital kinetics.

4.2.4 Inference for noisy proportions of extinction

Suppose that the observed data are noisy proportions of extinction. This equates to observing data as in Section 4.2.3 with measurement error. Denote the observed data by x_t and assume the following data model

$$y_t = \text{logit}\, x_t = \text{logit}\, p_t(\boldsymbol{\theta}) + \epsilon_t, \quad t = 1, \ldots, T \qquad \text{(independent)}$$

where $p_t(\boldsymbol{\theta})$ denotes the probability of extinction at time t and the ϵ_t are independent with $\epsilon_t \sim N(0, \sigma^2)$. It is of interest to learn about $(\boldsymbol{\theta}, \sigma)$ and assuming *a priori* that $\boldsymbol{\theta}$ and σ are independent, the posterior distribution is

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

4.2.5 Exact probability of extinction

For a given $\boldsymbol{\theta}$ and x_0, assuming that the probability of extinction, $p_t(\boldsymbol{\theta})$ is known in closed form, the likelihood is

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}, \sigma) = \prod_{t=1}^{T} \phi\left(y_t|\text{logit}\, p_t(\boldsymbol{\theta}), \sigma^2\right).$$

For the birth–death process, this probability is

$$p_t(\boldsymbol{\theta}) = \begin{cases} \left(\frac{\mu-\mu e^{(\mu-\lambda)t}}{\lambda-\mu e^{(\mu-\lambda)t)}}\right)^{x_0}, & \text{for } \lambda \neq \mu \\ \left(\frac{\lambda t}{1+\lambda t}\right)^{x_0}, & \text{for } \lambda = \mu. \end{cases} \tag{4.14}$$

4.2.6 Approximate probability of extinction

In the scenario where $p_t(\boldsymbol{\theta})$ is not known exactly, a simulation based approach can be used to estimate $p_t(\boldsymbol{\theta})$. The method was described in Section 4.2.1 and can be used to simulate the time evolution of a population for a particular choice of $\boldsymbol{\theta}$ and x_0. This can be converted to a binary time series denoting the extinction status of the population at discrete time points $t = 1, \ldots, T$. Repeating this for n populations gives

$$n\widehat{p}_{t,n}(\boldsymbol{\theta}) \sim Bin(n, p_t(\boldsymbol{\theta}))$$

where $\widehat{p}_{t,n}(\boldsymbol{\theta})$ is the observed proportion of extinction. $\widehat{p}_{t,n}(\boldsymbol{\theta})$ is an approximation of $p_t(\boldsymbol{\theta})$ and as $n \to \infty$, the approximation approaches the true proportion.

4.2.7 Approaches to inference

Three different approaches to inference will be considered, a *likelihood-free vanilla* MCMC scheme and two *pseudo-marginal* schemes. The first *pseudo-marginal scheme* has a Monte Carlo estimate of the marginal likelihood and the second has a Sequential Monte Carlo estimate. These methods will be known as *pseudo-marginal 1* and *pseudo-marginal 2*. The theory behind each of these methods was described in Chapter 3. The *vanilla* and *pseudo-marginal* schemes were previously applied to the constant model. The detailed algorithms are given for the birth–death model in Algorithms 10 and 11.

The third approach is to use a sequential Monte Carlo estimate of the marginal likelihood rather than a Monte Carlo estimate. The scheme outlined in Algorithm 12 uses a particle filter (SIR filter, Algorithm 5) at each iteration to construct an estimate of the marginal likelihood. This approach is discussed in detail in Section 3.4.4.

Algorithm 10 Birth–death model: inference using the *vanilla* scheme

Initialise the iteration counter and the state of the chain $(\boldsymbol{\theta}, \sigma)$.

For each iteration of the scheme:

1. Sample $(\boldsymbol{\theta}^*, \sigma^*) \sim q(\cdot|\boldsymbol{\theta}, \sigma)$ from a symmetric proposal distribution q, on the log scale.
2. Simulate the path $\widehat{\boldsymbol{p}}_n(\boldsymbol{\theta}^*) = [\widehat{p}_{1,n}(\boldsymbol{\theta}), \widehat{p}_{2,n}(\boldsymbol{\theta}), \ldots, \widehat{p}_{T,n}(\boldsymbol{\theta})]^T$ for some choice of n.
3. Compute the acceptance probability
$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\frac{\pi(\sigma^*)}{\pi(\sigma)}\frac{\tilde{\pi}(\boldsymbol{y}|\theta^*, \sigma^*)}{\tilde{\pi}(\boldsymbol{y}|\theta, \tau)}\frac{\theta^*}{\theta}\frac{\tau^*}{\tau}\right\}.$$
4. Set $(\boldsymbol{\theta}, \sigma) = (\boldsymbol{\theta}^*, \sigma^*)$ with probability α, otherwise retain $(\boldsymbol{\theta}, \sigma)$.

When using the SIR filter, it is necessary to store the states of the system. Let $z_{1:t}^{1:n}$ denote the state of the system for n populations up to time t, where

$$z_{1:t}^{1:n} = \begin{bmatrix} z_1^1 & \ldots & z_1^2 & \ldots & z_1^n \\ \vdots & & \vdots & & \vdots \\ z_i^1 & \ldots & z_i^2 & \ldots & z_i^n \\ \vdots & & \vdots & & \vdots \\ z_t^1 & \ldots & z_t^2 & \ldots & z_t^n \end{bmatrix}.$$

The $\widehat{p}_{t,n}(\boldsymbol{\theta})$ is deterministically related to $z_{1:t}^{1:n}$ via the function h, where $z_{1:t}^{1:n}$ represents n simulations of the process up to time t

$$\widehat{p}_{t,n}(\boldsymbol{\theta}) = h\left(z_{1:t}^{(1:n)}\right) = \frac{1}{n}\sum_{i=1}^{n} I_{(z_t^i \neq 0)}.$$

For the birth–death process there is only one *state* for each population, namely the level of the population. The $\widehat{p}_{t,n}$ is calculated by counting up the number of populations in which extinction has occurred at time t and dividing by n.

The estimate of the marginal likelihood is constructed by noting that an estimate of

Algorithm 11 Birth–death model: inference using *pseudo-marginal* scheme 1

Initialise the iteration counter and the state of the chain $(\boldsymbol{\theta}, \sigma)$.

For each iteration of the scheme:

1. Sample $(\boldsymbol{\theta}^*, \sigma^*) \sim q(\cdot|\boldsymbol{\theta}, \sigma)$ from a symmetric proposal distribution q, on the log scale.
2. For each particle $1, \ldots, N$, simulate a path $\widehat{\boldsymbol{p}}_n^{(1)}(\boldsymbol{\theta}^*), \widehat{\boldsymbol{p}}_n^{(2)}(\boldsymbol{\theta}^*), \ldots, \widehat{\boldsymbol{p}}_n^{(N)}(\boldsymbol{\theta}^*)$ for some choice of n.
3. Construct a Monte Carlo, unbiased estimate of $\pi(\boldsymbol{y}|\boldsymbol{\theta}^*, \sigma^*)$

$$\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}^*, \sigma^*) = \frac{1}{N} \sum_{j=1}^{N} \pi(\boldsymbol{y}|\widehat{\boldsymbol{p}}_n^{(j)}(\boldsymbol{\theta}^*), \sigma^*)$$

   where

$$\pi(\boldsymbol{y}|\widehat{\boldsymbol{p}}_n^{(j)}(\boldsymbol{\theta}^*), \sigma^*) = \prod_{i=1}^{T} \pi(y_t|\widehat{p}_{i,n}^{(j)}(\boldsymbol{\theta}^*), \sigma^*)$$

4. Compute the acceptance probability

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

5. Set $(\boldsymbol{\theta}, \sigma) = (\boldsymbol{\theta}^*, \sigma^*)$ with probability α, otherwise retain $(\boldsymbol{\theta}, \sigma)$.

$\pi(y_{t+1}|y_{1:t}, \boldsymbol{\theta}^*, \sigma^*)$ is given by

$$\frac{1}{N} \sum_{j=1}^{N} w_{t+1,j},$$

where the $w_{t+1,j}$ are defined in Algorithm 12. An estimate of the marginal likelihood is

$$\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}^*, \sigma^*) = \prod_{t=0}^{T-1} \left[\frac{1}{N} \sum_{j=1}^{N} w_{t+1,j}\right].$$

Algorithm 12 Birth–death model: inference using *pseudo-marginal* scheme 2

Initialise the iteration counter and the state of the chain $(\boldsymbol{\theta}, \sigma)$.

For each iteration of the scheme:

1. Sample $(\boldsymbol{\theta}^*, \sigma^*) \sim q(\cdot|\boldsymbol{\theta}, \sigma)$ from a symmetric proposal distribution q, on the log scale.
2. For each time $t = 1, 2, \ldots, T$:

   For each particle $j = 1, 2, \ldots, N$:

   (i) Draw $z_{t+1,j}^{(1:n)} \sim \pi\left(\cdot \middle| z_{t,j}^{(1:n)}, \boldsymbol{\theta}^*\right)$. Run the Gillespie algorithm from t to $t+1$ for $1{:}n$ paths.

   (ii) Construct weights

   $$w_{t+1,j} = \pi\left(y_{t+1} \middle| h\left(z_{t+1,j}^{(1:n)}\right), \sigma^*\right).$$

   (iii) Calculate normalised weights

   $$\widehat{w}_{t+1,j}^n = \frac{w_{t+1,j}}{\sum_{j=1}^N w_{t+1,j}}.$$

3. Resample N times amongst $z_{t+1,1}^{(1:n)}, z_{t+1,2}^{(1:n)}, \ldots, z_{t+1,N}^{(1:n)}$ using normalised weights as probabilities.
4. Calculate

   $$\widehat{\pi}(\boldsymbol{y}|\boldsymbol{\theta}^*, \sigma^*) = \prod_{t=0}^{T-1}\left[\frac{1}{N}\sum_{j=1}^N w_{t+1,j}\right].$$

5. Compute the acceptance probability

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

6. Set $(\boldsymbol{\theta}, \sigma) = (\boldsymbol{\theta}^*, \sigma^*)$ with probability α, otherwise retain $(\boldsymbol{\theta}, \sigma)$.

Implementation

It was assumed *a priori* that λ, μ and σ were independent and

$$\lambda \sim \text{Log-Normal}(\log 0.6, 0.5)$$
$$\mu \sim \text{Log-Normal}(\log 1, 0.5)$$
$$\sigma \sim \text{Log-Normal}(\log 0.3, 0.5).$$

Three different datasets were simulated, each of which was observed at the same start and end time but had different discretisations such that $T = (10, 25, 50)$. Each dataset had true parameter values

$$\lambda = 0.6, \quad \mu = 1, \quad \sigma = 0.3 \quad \text{and} \quad x_0 = 10. \tag{4.15}$$

The results of inference using Algorithms 10–12 are shown in Figure 4.5 for three different choices of n.

4.2.8 Comparing algorithm performance

The marginal posterior distributions for λ, μ and σ can be seen in Figure 4.5 for the synthetic dataset shown in Figure 4.6. As expected, the simulator with the highest n produces posterior distributions which are most like those obtained using the exact probabilities of extinction. It would appear that using $n = 1000$ is significantly better than $n = 100$, and when $n = 10$ all schemes perform badly.

For the *vanilla* scheme, the distributional result is an approximation and only holds for large n. Although from a computational speed perspective, using as small an n as possible is advantageous, this must be balanced with using a large enough n to give a good approximation.

Using a *pseudo-marginal* will necessarily be more computationally intensive than the *vanilla* scheme, since the time taken to run this algorithm scales linearly with the number of particles. However, this scheme has the advantage that it targets the true

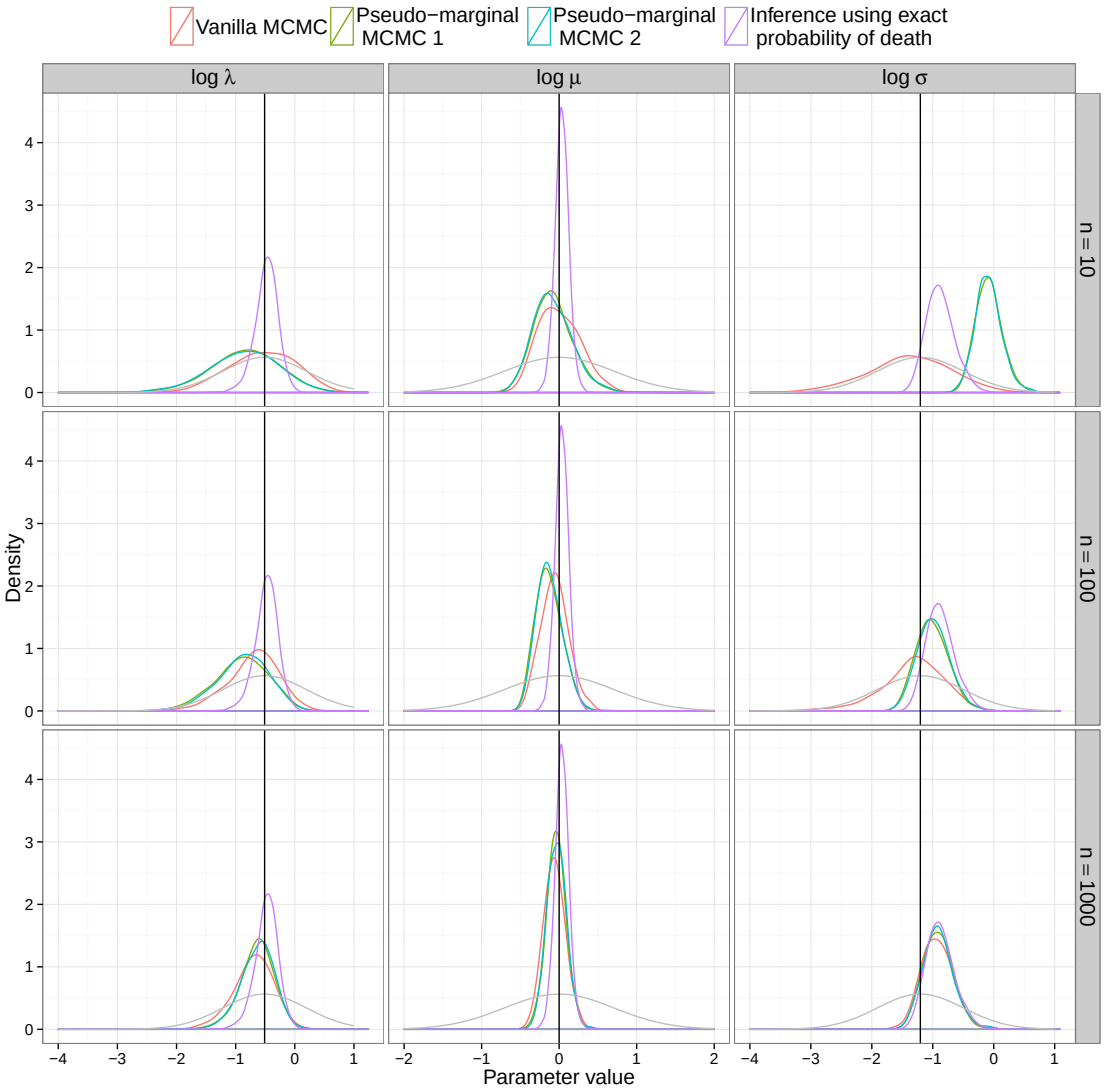

Figure 4.5: Marginal posterior distributions for log λ, log μ and log σ. Black vertical lines represent true values used to simulate data, grey lines represent the prior distribution.

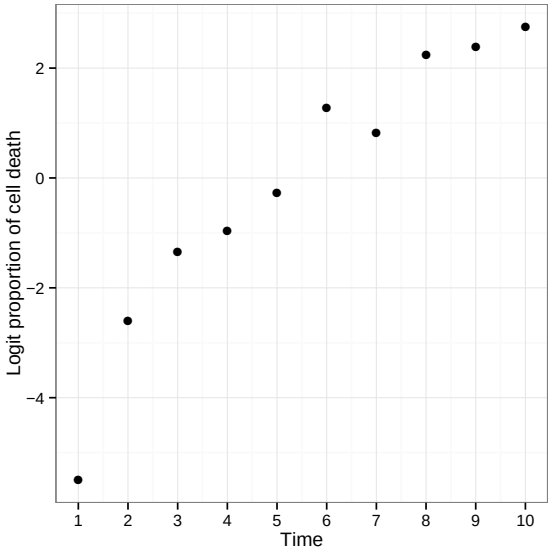

Figure 4.6: Noisy logit proportions of extinction from birth–death model.

posterior (conditional on n) and means an approximation is no longer required. It may well be the case that a smaller n than that used in the *vanilla* scheme is passable.

One potential problem with the scheme outlined in Algorithm 11 is that the variability of $\widehat{\pi}(\boldsymbol{y}|\theta,\sigma)$ could lead to poor mixing. This would lead to an inefficient scheme which could require extensive thinning and would consequently take a long time.

The efficiency of each of the algorithms can be assessed by considering the effective sample size (ESS), which was defined in Section 3.4.5 of Chapter 3. The ESS for the three approaches to inference are given in Figure 4.7. The left hand plot gives the ESS per iteration of the scheme and the right hand plot standardises the ESS for CPU time. As expected, both *pseudo–marginal* schemes give a larger ESS per iteration than the *vanilla* scheme, for all choices of n. It can also be seen that *pseudo–marginal* scheme 2 slightly outperforms *pseudo–marginal* scheme 1. For the *pseudo–marginal* schemes, using a larger number of particles gives small improvements in efficiency but comes at significant computational cost. After standardising the ESS for CPU time, the *vanilla* scheme far outperforms both *pseudo–marginal* scheme due to its speed.

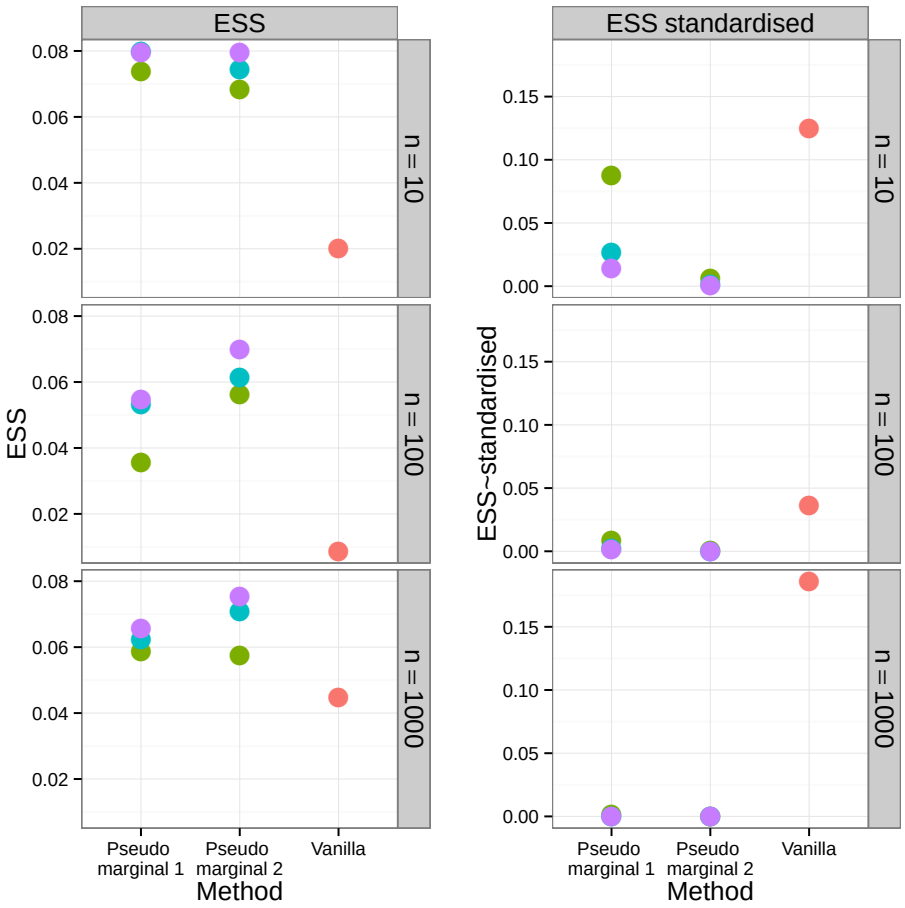

Figure 4.7: Effective sample sizes (ESS). Left: ESS per iteration. Right: ESS standardised for CPU time. Colours represent different numbers of particles in the pseudo marginal schemes: 10 particles (green), 50 particles (blue), 100 particles (purple).

Chapter 5

Gaussian process emulation

The aim of this thesis is to make inferences about parameters in the PolyQ model, a large stochastic kinetic model. One crucial component of the problem is that the experimental data are proportions of cell death, not observations on the underlying chemical species. The inferential task is not straightforward since the data likelihood is intractable.

Chapter 3 describes several simulation based, likelihood-free inference strategies which can be used to make inference on the parameters of the PolyQ model. These algorithms all require the use of simulation to estimate the proportion of cell death, for a proposed set of parameters. This involves obtaining n forward simulations from the stochastic model to construct an estimate of the proportion of cell death. This must be done at each iteration of the scheme.

Chapter 4 implements the algorithms described in Chapter 3 on much smaller models. In these cases, the computational time taken to run the algorithms is not large, taking in the order of minutes or hours, rather than days or weeks. However, the computational time does not scale well with the complexity of the model and, for the PolyQ model, simulating from the model is slow, and the inference schemes described in Chapter 3 become infeasible.

Chapter 2 describes methods for approximating the stochastic kinetic model, such as the τ-leap method and approximating the Markov jump process with the Chemical

Langevin Equation. Unfortunately the increased speed gained from using these strategies is not sufficient to facilitate inference from the PolyQ model, and hence, an alternative approach must be sought.

The problem of having a model that is prohibitively slow to simulate from is a generic one in the context of computer experiments, which are typically used to explore real life phenomena. One area where computer models are used extensively is in the climate sciences. For example, the HadOCC (Hadley Centre Ocean Carbon Cycle model) model (Palmer and Totterdell, 2001; Hemmings et al., 2008) models the ocean carbon cycle. When computer experiments are computationally expensive to run, a much faster approximation is sought.

The use of Gaussian processes to create a statistical model of a computer model, known as an emulator is described in detail by Sacks et al. (1989); Currin et al. (1991); Kennedy and O'Hagan (2001); Santner et al. (2003); O'Hagan (2006). Note, in the geostatistics literature, *kriging* is used as a synonym for emulation. The idea is to run the expensive simulator at only a limited number of input values and to infer the output at new input values. These input values along with the simulator output are known as the *training data*. The emulator can be thought of as a fast surrogate for the true simulator which makes probabilistic predictions of the simulator output.

A naive approach to prediction would be to fit a regression model to the known simulator outputs, and use this to predict the output for unknown inputs. However, this ignores the fact that the uncertainty in the the prediction depends on how close the prediction inputs are to inputs in the training dataset. Emulators are typically constructed to produce fully probabilistic predictions of what the simulator would produce. However, emulators that are constructed under the Bayes linear framework only predict the mean and variance (Craig and Goldstein, 2001; Goldstein and Rougier, 2006).

Emulators are most commonly fitted to deterministic processes, where running the simulator at the same inputs will produce identical outputs. There has been recent interest in building emulators for stochastic models (Bates and Kenett, 2006; Henderson

et al., 2009; Kleijnen, 2009). Despite the PolyQ model being a stochastic kinetic model, the proportion of cell death through time is a deterministic process. This is because given a very large number of cells governed by the PolyQ model, the proportion of cells which die at time t is fixed. In what follows, a deterministic emulator will be constructed to emulate proportions of cell death.

5.1 Building an emulator

Emulators are built using a set of *training data* which is obtained by running the simulator for a particular set of inputs. These training input values are denoted

$$\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_{n_d}]^T$$

which results in output values

$$\boldsymbol{y} = [y(\boldsymbol{\theta}_1), \ldots, y(\boldsymbol{\theta}_i), \ldots, y(\boldsymbol{\theta}_{n_d})]^T.$$

In the context of cell death models, the $y(\boldsymbol{\theta}_i)$ represent the logit proportions of cell death. The dimension of $\boldsymbol{\theta}_i$ is $(n_p \times 1)$; hence, this leads to a n_d–point n_p–dimensional design. More information on how choose the training design are given in Section 5.1.4.

It is necessary to determine a mean function which is denoted $m(\cdot)$ and typically depends on parameters $\boldsymbol{\beta}$. This could be, for example, a least squares fit to some suitable functions of $\boldsymbol{\theta}$. It is also necessary to specify a covariance function $K(\cdot, \cdot)$, which describes the relationship between the simulator output for inputs which are some *distance* apart. The covariance function typically depends on unknown hyperparameters.

The aim is to learn about the simulator output at new inputs

$$\boldsymbol{\Theta}^* = [\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_i^*, \ldots, \boldsymbol{\theta}_{n_{d^*}}^*]^T$$

not featured in the training dataset. The corresponding test outputs about which

inference is required are

$$\boldsymbol{y}^* = [y(\boldsymbol{\theta}_1^*), \ldots, y(\boldsymbol{\theta}_i^*), \ldots, y(\boldsymbol{\theta}_{n_d^*}^*)]^T.$$

Assuming a Gaussian Process prior

$$\boldsymbol{y} \sim \mathcal{N}(m(\boldsymbol{\Theta}), K(\boldsymbol{\Theta}, \boldsymbol{\Theta}))$$

and

$$\boldsymbol{y}^* \sim \mathcal{N}(m(\boldsymbol{\Theta^*}), K(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*)),$$

the joint distribution of $\boldsymbol{y}$ and $\boldsymbol{y}^*$ is

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

Conditioning on the training outputs, the posterior for the test outputs is

$$\boldsymbol{y}^* | \boldsymbol{y} \sim \mathcal{N}(m^*, K^*)$$

where

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

and

$$K^* = K(\boldsymbol{\Theta^*}, \boldsymbol{\Theta^*}) - K(\boldsymbol{\Theta}^*, \boldsymbol{\Theta})^T K(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} K(\boldsymbol{\Theta}, \boldsymbol{\Theta}^*).$$

Note in the above, the explicit dependence of the hyperparameters on the mean function $m(\cdot|\cdot)$ and covariance function $K(\cdot|\cdot)$ is dropped.

5.1.1 Choice of mean function

It is necessary to define a mean function $m(\cdot)$ for the inputs, which is a function of $\boldsymbol{\theta}$. The mean function must be carefully specified to ensure an accurate emulator. The most simple choice is a zero order (constant) mean function, such that the regression surface is flat. This amounts to expressing no prior knowledge about the likely output of the simulator for a particular choice of $\boldsymbol{\theta}$. This approach is often used in practice, see Sacks et al. (1989) and Oakley and O'Hagan (2004).

Caution must be taken when using a constant mean function as can be seen in Figure 5.1. In the fourth plot, the hyperparameters of the covariance function are chosen (for illustration purposes) so that the correlation between the outputs, at inputs which are a small distance apart, is low. It can be seen that the mean function attempts to revert to the prior (zero in this case) the further away from a training point it is. This undesired effect is seen more generally in emulators when there is large distance between training inputs, thus highlighting the need for a sensible choice of mean function.

Another common choice of mean function is a linear combination of the inputs

$$m(\boldsymbol{\theta}) = \beta_0 + \sum_{k=1}^{n_p} \beta_k \theta_k.$$

Examples of the use of this mean function can be seen in Oakley and O'Hagan (2004); Conti and O'Hagan (2010). Other authors use more complex mean functions, such as Kaufman et al. (2011) who constructed a mean function using Legendre polynomials.

A fully Bayesian approach to estimating the parameters of the mean function would obtain a posterior distribution for the parameters. The marginal posterior distribution of these parameters is a multivariate t-distribution. In general the number of points in the Latin hypercube design n_d will be large and so posterior uncertainty about these parameters will be low. Therefore, in this thesis (unless stated otherwise), this level of uncertainty is ignored and these parameters are fixed at their maximum likelihood (least squares) estimates. This provides a simpler approach to fitting emulators.

5.1.2 Choice of covariance function

The covariance function describes assumptions about the unknown function; it specifies the covariance between the simulator output for a pair of inputs $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$. It can be thought of as being a *similarity* measure, since it is expected that when the simulator is run at pairs of inputs which are *close* together, the output will be similar.

The covariance function must generate a covariance matrix which is symmetric, invertible and non-negative definite. It is typically the case that the covariance function is chosen to be stationary. The stationarity property says that the covariance depends only on the distance between two inputs $|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j|$ rather than the actual values of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$. It may be unrealistic to assume from the outset that the output varies similarly in all areas of input space. However, it is hoped that after fitting an appropriate mean function to the data, any large scale variation would be removed and a stationary covariance function is suitable for the residuals.

The most common choice of covariance function is the *squared exponential* covariance function and it has the form

$$\begin{aligned} K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) &= a \exp\left\{ -\sum_{k=1}^{n_p} (\theta_{ik} - \theta_{jk})^2 / r_k^2 \right\} \\ &= aR(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | \boldsymbol{r}). \end{aligned}$$

Chapter 4 of Rasmussen and Williams (2006) gives an overview of further choices of covariance functions and their properties.

This function depends on *hyperparameters*, $(a, \boldsymbol{r})$. The physical interpretation of these parameters is not immediately obvious. The vector $\boldsymbol{r}$ are known as the length scale parameters; r_k can be thought of as being a measure of how correlated outputs will be at inputs $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ which are a distance $|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j|$ apart. Since the dimension of $\boldsymbol{r}$ is equal to the dimension of the input, this covariance function allows different inputs to impact the response differently.

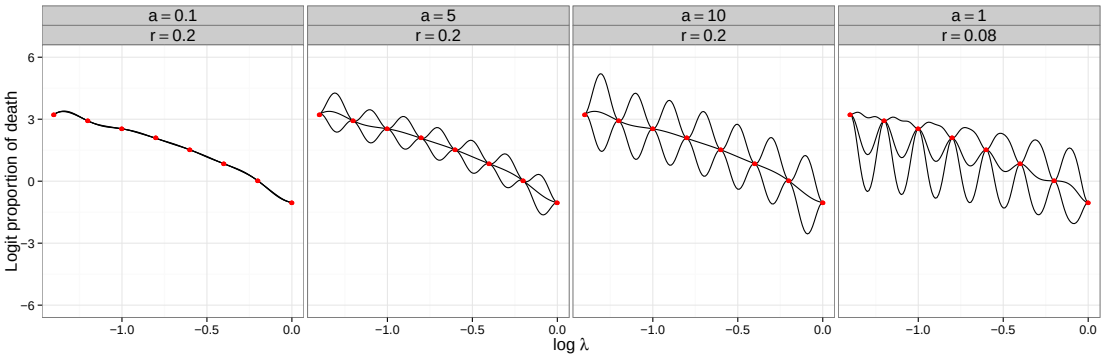

Figure 5.1: Emulation for logit proportions of death from the birth–death model (for fixed $\mu = 0.6$). Training data shown in red. Mean and 2 standard deviations from the fitted Gaussian Process emulator shown in black. Different panels represent different choices of hyperparameters.

5.1.3 1-D birth–death example

Figure 5.1 depicts several fitted Gaussian process emulators to data obtained from the birth–death model. In this example, the output (y-axis) is the logit probability that a population becomes extinct and the input (x-axis) is the birth rate λ. Here the input θ is one dimensional since the death rate μ is fixed at 0.6.

The red points denote the training data used to fit the emulator, the black lines represent the fitted mean function and ± 2 standard deviations. The different panels represent emulators fitted with different choices of hyperparameters.

In the first three panels, the r parameter is kept constant and the a parameter increases. It can be seen that effect of increasing a is to increase the emulator variance, giving larger prediction intervals. In the fourth panel, the r parameter is much lower than the others. In practice, this has the effect of lowering the influence of nearby training points when making predictions. It can be seen that between training points, the emulator attempts to revert to the prior mean function (zero in this case).

5.1.4 Training data design

It is necessary to choose carefully the $\boldsymbol{\theta}_i$ at which to run the simulator. A poor choice can lead to parts of the design space not being well explored which in turn can lead to an inaccurate emulator in certain parts of the input space. In general, it is hoped that

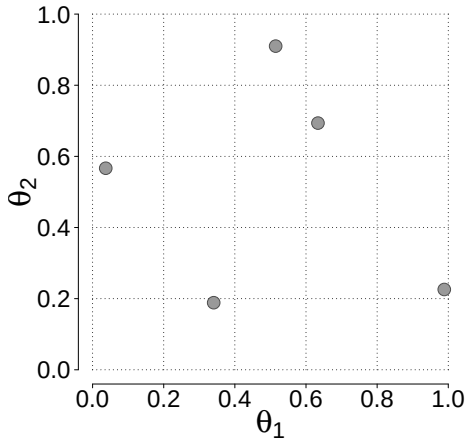

Figure 5.2: Latin hypercube design in 2–D with $n_d = 5$.

points are well spread out over the design space; a design is sought that is *space filling*.

The use of Latin hypercube sampling (LHS) was proposed by McKay et al. (1979) as a *space filling* design. The popularity of this design can in part be attributed to its ease of use. It was shown by Stein (1987) that when LHS is compared to simple random sampling, it produces a lower asymptotic variance for the mean of simulator output.

In the two dimensional case, the design is a Latin square constructed by splitting the space into n_p rows and columns and placing exactly one point in each row and column as shown in Figure 5.2. When the Latin hypercube is projected onto either axis, the points provide good coverage in that dimension. This generalises to higher dimensions, where projecting the points onto any subset of the inputs produces a well covered design. Although this design promises a well covered design when projected onto any subset of the inputs, this does not necessarily ensure the whole space is well covered. In the most extreme case, for two dimensions, the points could lie precisely on the diagonal and still produce a valid Latin square.

Morris and Mitchell (1995) proposed the maximin design. Conceptually, this works by maximising the distance between points. This is done by producing a list of the minimum distance between points for each design using the Euclidean distance $d(\boldsymbol{\theta}, \boldsymbol{\theta}')$. The maximin design maximises the minimum distance $d(\boldsymbol{\theta}, \boldsymbol{\theta}')$. Figure 5.3 compares the ordinary Latin hypercube design with the maximin design in 2D with $n_d = 20$. It

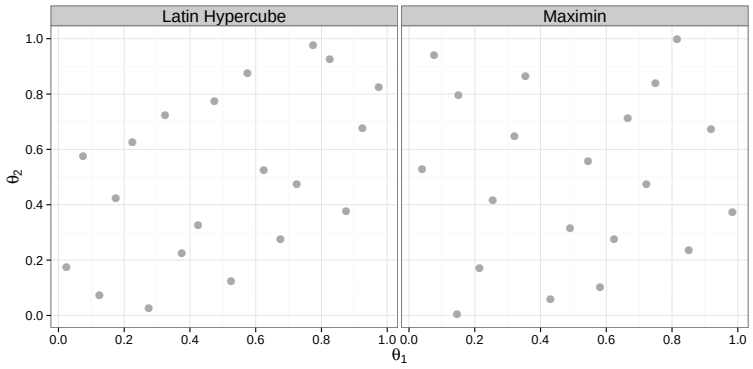

Figure 5.3: Left: Latin hypercube design. Right: Maximin design. Both with $n_d = 20$.

can be seen that maximin design appears to given slightly better coverage of the design space. In the following chapters, the maximin design will be used throughout.

The choice of n_d represents a trade off between accuracy and speed since the construction of an emulator involves multiple inversions of an $n_d \times n_d$ covariance matrix. This operation scales with $\mathcal{O}(n_d^3)$; thus keeping n_d as low as possible is vital. Chapman et al. (1994) suggest that n_d should be at least $10n_p$.

5.2 Estimating hyperparameters

The fitted emulator is conditional on hyperparameters $(a, \boldsymbol{r})$ which are unknown and can be estimated from the standardised training data $\boldsymbol{z}(\boldsymbol{\Theta}) = \boldsymbol{y}(\boldsymbol{\Theta}) - m(\boldsymbol{\Theta})$. The likelihood of these training data is

$$\begin{aligned}\pi(\boldsymbol{z}|a, \boldsymbol{r}) &= (2\pi)^{-n_d/2}|K(a, \boldsymbol{r})|^{-1/2} \exp\left(-\frac{1}{2}\boldsymbol{z}^T K(a, \boldsymbol{r})^{-1}\boldsymbol{z}\right) \\ &= (2\pi)^{-n_d/2} a^{-n_d/2}|R(\boldsymbol{r})|^{-1/2} \exp\left(-\frac{1}{2a}\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}\right),\end{aligned}$$

where the dependence of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ on $K(\cdot, \cdot)$ and $R(\cdot)$ has been dropped from the notation. A fully Bayesian analysis would specify a prior distribution for $(a, \boldsymbol{r})$ and proceed by obtaining the posterior distribution $\pi(a, \boldsymbol{r}|\boldsymbol{z})$, which is typically intractable. An MCMC scheme can be constructed to obtain realisations of the posterior and these can be used to obtain estimates of the hyperparameters.

Under the assumption that the prior distribution has independent components and $a \sim \text{InvGa}(c_0, d_0)$, a closed form for the marginal likelihood can be determined

$$\begin{aligned}\pi(\boldsymbol{z}|\boldsymbol{r}) &= \int_0^\infty \pi(\boldsymbol{z}|a,\boldsymbol{r})\pi(a)\,da \\ &= \int_0^\infty (2\pi)^{-n_d/2} a^{-n_d/2} |R(\boldsymbol{r})|^{-1/2} \exp\left\{-\frac{1}{2a}\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}\right\} \times \frac{d_0^{c_0} a^{-c_0-1} e^{-d_0/a}}{\Gamma(c_0)}\,da \\ &= \frac{(2\pi)^{-n_d/2} d_0^{c_0}}{\Gamma(c_0)} |R(\boldsymbol{r})|^{-1/2} \int_0^\infty a^{-c_0-n_d/2-1} \exp\left\{-\frac{1}{a}\left[d_0 + \frac{\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}}{2}\right]\right\} da.\end{aligned}$$

On noting that

$$\int_0^\infty a^{-c-1} e^{-d/c}\,da = \Gamma(c) d^{-c}$$

it follows that

$$\begin{aligned}\pi(\boldsymbol{z}|\boldsymbol{r}) &= \frac{(2\pi)^{-n_d/2} d_0^{c_0}}{\Gamma(c_0)} |R(\boldsymbol{r})|^{-1/2} \times \Gamma(c_0 + n_d/2) \left[d_0 + \boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}/2\right]^{-(c_0+n_d/2)} \\ &\propto |R(\boldsymbol{r})|^{-1/2} \left[1 + \frac{\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}}{2d_0}\right]^{-(c_0+n_d/2)}\end{aligned}$$

where the proportionality constant does not depend on $\boldsymbol{r}$. Further

$$\begin{aligned}\pi(a|\boldsymbol{r},\boldsymbol{z}) &\propto \pi(\boldsymbol{z}|a,\boldsymbol{r})\pi(a) \\ &\propto |K(a,\boldsymbol{r})|^{-1/2} \exp\left\{-\frac{1}{2}\boldsymbol{z}^T K(a,\boldsymbol{r})^{-1}\boldsymbol{z}\right\} a^{-c_0-1} e^{-d_0/a} \\ &\propto a^{-n_d/2} \exp\left\{-\frac{1}{2a}\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}\right\} a^{-c_0-1} e^{-d_0/a} \\ &\propto a^{-(c_0+n_d/2)-1} \exp\left\{-\frac{1}{a}\left[d_0 + \frac{\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}}{2}\right]\right\}\end{aligned}$$

and so

$$a|\boldsymbol{r},\boldsymbol{z} \sim \text{InvGa}\left(c_0 + n_d/2, d_0 + \boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}/2\right).$$

Therefore, assuming *a priori* $a \sim \text{InvGa}(c_0, d_0)$ independently of $\boldsymbol{r}$ allows an MCMC scheme to be constructed with state space $\boldsymbol{r}$ (rather than a and $\boldsymbol{r}$) which should

Algorithm 13 Estimating hyperparameters 1

For each iteration of the scheme:

1. Propose $\boldsymbol{r}^* \sim q(\boldsymbol{r}^*|\boldsymbol{r})$, where q is a symmetric random walk on the log scale.
2. Compute the acceptance probability $\alpha = \min\{1, A\}$ where

$$A = \frac{\pi(\boldsymbol{r}^*)}{\pi(\boldsymbol{r})}\frac{\pi(\boldsymbol{z}|\boldsymbol{r}^*)}{\pi(\boldsymbol{z}|\boldsymbol{r})}\frac{q(\boldsymbol{r}|\boldsymbol{r}^*)}{q(\boldsymbol{r}^*|\boldsymbol{r})} = \frac{\pi(\boldsymbol{r}^*)}{\pi(\boldsymbol{r})}\frac{\pi(\boldsymbol{z}|\boldsymbol{r}^*)}{\pi(\boldsymbol{z}|\boldsymbol{r})}\prod_{i=1}^{n_p}\frac{r_i^*}{r_i}.$$

   If it is assumed the r_i are independent *a priori* then the expression for A simplifies to

$$A = \frac{\pi(\boldsymbol{z}|\boldsymbol{r}^*)}{\pi(\boldsymbol{z}|\boldsymbol{r})}\prod_{i=1}^{n_p}\frac{r_i^*\pi(r_i^*)}{r_i\pi(r_i)}.$$

3. Accept and move the current state of the chain to $\boldsymbol{r}^*$ with probability α.
4. Simulate a from

$$a|\boldsymbol{r}^*, \boldsymbol{z} \sim \text{InvGa}\left(c_0 + n_d/2, d_0 + \boldsymbol{z}^T R(\boldsymbol{r}^*)^{-1}\boldsymbol{z}/2\right).$$

   .

have better convergence properties and be more efficient. This algorithm is given in Algorithm 13.

If it is assumed *a priori* $r_i \sim \text{LN}(c_i, 1/d_i)$, the numerator (or denominator) in the acceptance probability A is given by

$$\log\left\{\pi(\boldsymbol{z}|\boldsymbol{r})\prod_{i=1}^{n_p} r_i\pi(r_i)\right\} = k - \frac{1}{2}\log|R(\boldsymbol{r})| - \left(c_0 + \frac{n_d}{2}\right)\log\left(1 + \frac{\boldsymbol{z}^T R(\boldsymbol{r})^{-1}\boldsymbol{z}}{2d_0}\right) - \frac{1}{2}\sum_{i=1}^{n_p} d_i\left(\log r_i - c_i\right)^2.$$

In Chapter 6, emulators are built for logit proportions of extinction for populations governed by a birth–death model. When it is not possible to obtain exact proportions of extinction from the model, approximate proportions can be obtained using many runs of the simulator. This scenario was considered in detail in Section 4.2.6 of Chapter 4.

The uncertainty induced by this approximation can be accounted for in the emulator

by introducing a *nugget* term to the covariance function

$$\tilde{K}_t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) = K_t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) + f(\boldsymbol{\theta_i})\delta_{ij}$$

where δ_{ij} is the Kronecker delta function and further details on the choice of $f(\cdot)$ are given in Chapter 6. This amounts to adding an extra term to the leading diagonal of the covariance matrix.

This means that the covariance function cannot be written in the form $a \times R$. In this case, the scheme presented in Algorithm 13 can no longer be used, since a cannot be marginalised over. In this scenario, it is necessary to construct an MCMC scheme with state space $(a, \boldsymbol{r})$, that is, use a joint update as shown in Algorithm 14.

If it assumed *a prior* $r_i \sim \mathrm{LN}(c_i, 1/d_i)$ and $a \sim \mathrm{LN}(c_0, 1/d_0)$, the numerator (or denominator) in the acceptance probability A is given by

$$\log\left\{\pi(\boldsymbol{z}|a, \boldsymbol{r})a\pi(a)\prod_{i=1}^{n_p} r_i\pi(r_i)\right\} = k - \frac{1}{2}\log|\tilde{K}(a, \boldsymbol{r})| - \frac{1}{2}\boldsymbol{z}^T\tilde{K}(a, \boldsymbol{r})^{-1}\boldsymbol{z} \\ - \frac{d_0}{2}(\log a - c_0)^2 - \frac{1}{2}\sum_{i=1}^{n_p} d_i(\log r_i - c_i)^2 .$$

Once samples from the posterior have been obtained using either Algorithm 13 or 14, emulator construction could proceed using a plug-in approach. For example, $(a, \boldsymbol{r})$ could be estimated using their marginal posterior mean/mode. Alternatively, the posterior uncertainty could be taken into account and averaged over in the fitted emulators. Both of these approaches are compared in the numerical examples in Chapter 6. For the model considered, it was found that averaging over the posterior uncertainty in the hyperparameters makes negligible difference to the emulator predictions.

5.3 Other approaches to hyperparameter estimation

One of the main drawbacks of Algorithms 13 and 14 is that the evaluation of the likelihood, at each iteration, involves calculating both the determinant and the inverse

Algorithm 14 Estimating hyperparameters 2

For each iteration of the scheme:

1. Propose $(a^*, \boldsymbol{r}^*) \sim q(a^*, \boldsymbol{r}^*|a, \boldsymbol{r})$, where q is a symmetric random walk with independent components on the log scale.

2. Compute the acceptance probability $\alpha = \min\{1, A\}$ where

$$A = \frac{\pi(a^*)}{\pi(a)} \frac{\pi(\boldsymbol{r}^*)}{\pi(\boldsymbol{r})} \frac{\pi(\boldsymbol{z}|a^*, \boldsymbol{r}^*)}{\pi(\boldsymbol{z}|a, \boldsymbol{r})} \frac{q(a|a^*)}{q(a^*|a)} \frac{q(\boldsymbol{r}|\boldsymbol{r}^*)}{q(\boldsymbol{r}^*|\boldsymbol{r})}$$
$$= \frac{\pi(a^*)}{\pi(a)} \frac{\pi(\boldsymbol{r}^*)}{\pi(\boldsymbol{r})} \frac{\pi(\boldsymbol{z}|a^*, \boldsymbol{r}^*)}{\pi(\boldsymbol{z}|a, \boldsymbol{r})} \frac{a^*}{a} \prod_{i=1}^{n_p} \frac{r_i^*}{r_i},$$

   If it is assumed the r_i are independent *a priori* then the expression for A simplifies to

$$A = \frac{\pi(a^*)}{\pi(a)} \frac{\pi(\boldsymbol{z}|a^*, \boldsymbol{r}^*)}{\pi(\boldsymbol{z}|a, \boldsymbol{r})} \frac{a^*}{a} \prod_{i=1}^{n_p} \frac{r_i^* \pi(r_i^*)}{r_i \pi(r_i)}.$$

3. Accept and move the current state of the chain to $(a^*, \boldsymbol{r}^*)$ with probability α.

of the covariance matrix K. Inverting a matrix of size $n_d \times n_d$ is an $\mathcal{O}(n_d^3)$ operation, hence the algorithm will scale with $\mathcal{O}(n_d^3)$. Since the PolyQ model is high–dimensional, using a large n_d will be crucial to cover the parameter space. Consequently, a more computational efficient approach to estimating hyperparameters must be sought.

5.3.1 Sparse covariance approach

To achieve the required computational speed up, one approach is to take advantage of the near sparsity of the covariance matrix and use sparse matrix algorithms (Pissanetzky, 1984; Barry and Kelley Pace, 1997). This approach has received attention in the literature in the context of covariance tapering (Furrer et al., 2006; Kaufman et al., 2008), where the covariance matrix is multiplied by another sparse matrix to impose sparsity. These methods all assume the correlation function is isotropic i.e. $r_1 = r_2 = \ldots = r_{n_p}$. This assumption is too restrictive since it is expected that for the PolyQ model, inputs will impact the response differently.

The approach of Kaufman et al. (2011) allows anisotropic compactly supported

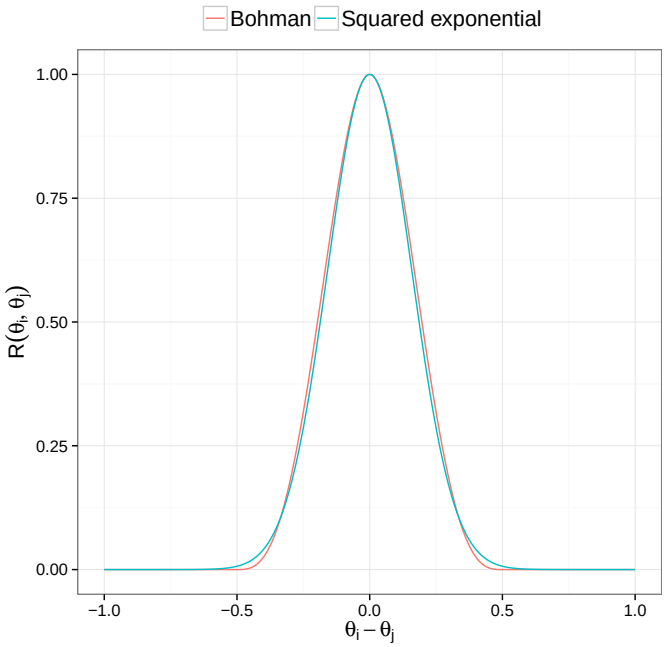

Figure 5.4: Comparison of the Bohman function and squared exponential kernel.

covariance functions. The inputs $\boldsymbol{\theta}$ are first scaled so that they are in $[0, 1]$ and the covariance function is approximated by a function such as the Bohman function

$$R_k(t; \tau_k)_{ij} = \begin{cases} (1 - t/\tau_k)\cos(\pi t/\tau_k) + \sin(\pi t/\tau_k)/\pi, & t < \tau_k \\ 0, & t \geq \tau_k \end{cases}$$

where $t = |\theta_{ik} - \theta_{jk}|$. The parameter τ_k governs the correlation in the output for input k. The Bohman function is shown in Figure 5.4 (red line) for $k = 1$ and $\tau_1 = 0.5$. It can be seen that the function is zero when $t \geq 0.5$. The blue line illustrates the squared exponential kernel where the hyperparameters are chosen such that the two functions exhibit very similar behaviour. This function asymptotes zero when t gets large, but unlike the Bohman function has no absolute cutoff for t after which R is taken to be zero. The advantage of using a covariance function such as the Bohman is that is it still creates a valid covariance matrix while allowing sparsity to be imposed.

The level of sparsity to be imposed on the covariance matrix is specified by the user and represents a trade off between computational efficiency and accuracy. For example,

$s = 0.99$ implies 99% of the off-diagonal elements of

$$R(t;\boldsymbol{\tau})_{ij} = \prod_{k=1}^{n_P} R_k(t;\tau_k)_{ij}$$

will be zero and this feeds into the prior for $\boldsymbol{\tau}$. It is assumed *a priori* that $\boldsymbol{\tau}$ is uniformly distributed over the space, such that each of the k inputs impacts the response equally

$$\frac{1}{n_p}\sum_{k=1}^{n_p}\tau_k \leq c$$

where c is chosen to satisfy $c(2-c) = (1-s)^{1/n_p}$.

To construct the covariance matrix, pairs of input values $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ such that $|\theta_{ik} - \theta_{jk}| < \tau_k$ for all dimensions k are identified. These pairs only are used to compute

$$R(\boldsymbol{\tau})_{ij} = \prod_{k=1}^{n_p} R_k(|\theta_{ik} - \theta_{jk}|;\tau_k).$$

The `spam` package (Furrer and Sain, 2010) in `R` uses computationally efficient sparse matrix methods to store and manipulate sparse covariance matrices. The the Cholesky decomposition of $R(\boldsymbol{\tau})$ is computed and used to circumnavigate the need to compute directly the inverse of the covariance matrix. An outline of the algorithm is given in Algorithm 15.

5.4 Diagnostics

Following the construction of an emulator, it is crucial to assess how well it performs as a surrogate for the simulator. Bastos and O'Hagan (2009) provide a detailed overview of diagnostics for emulators.

One reason for an inaccurate emulator could be that the hyperparameters of the covariance function (a and $\boldsymbol{r}$) have been incorrectly estimated. Misspecification of a affects the credible intervals for predictions from the emulator. An overestimation leads to credible intervals which are too wide and an underestimation leads to credible intervals

Algorithm 15 Estimating hyperparameters using sparse covariance approach

For each iteration of the scheme:

1. Propose $\boldsymbol{\tau}^* \sim q(\boldsymbol{\tau}^*|\boldsymbol{\tau})$, where q is a multivariate normal random walk.
2. Identify pairs of input values $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ such that $|\theta_{ik} - \theta_{jk}| < \tau_k^*$ for all dimensions k.
3. Use these pairs to compute

$$R(\boldsymbol{\tau^*})_{ij} = \prod_{k=1}^{n_p} R_k(|\theta_{ik} - \theta_{jk}|; \tau_k^*).$$

4. Use the `spam` package in `R` to compute the Cholesky decomposition of $R(\boldsymbol{\tau})$ and "backsolve" to obtain the quantities needed to construct the likelihood.
5. Accept/reject $\boldsymbol{\tau^*}$ based on the acceptance probability.

which are too narrow. When the $\boldsymbol{r}$ parameters are incorrectly estimated, the correlation between the output for inputs that are a certain distance apart is misrepresented. This affects the size of the credible intervals close to the training inputs.

Another reason for an inaccurate emulator, is that the assumptions made about the covariance function are not appropriate. It is assumed that the covariance function is stationary, that is, it depends only on $\theta_i - \theta_j$. In practice, this has the effect of assuming that the covariance of the simulator output at inputs a particular distance apart is the same for all areas of parameter space.

In certain scenarios, it may not be possible to use the simulator to generate any new training points, thus diagnostics must be constructed which only use the existing training runs. Rougier et al. (2009) suggest a *leave-one-out* approach which removes one data point at a time from the training data to fit the emulator, then uses the emulator to try to predict the omitted data point.

For the models featured in this thesis, it will be possible to obtain a set of training data which can be used for validation, thus the diagnostics used will all rely on the use of validation data. A new Latin hypercube design over the same input space as the

original is constructed which has $n_d^\dagger$ points

$$\boldsymbol{\Theta}^\dagger = \left[\boldsymbol{\theta}_1^\dagger, \ldots, \boldsymbol{\theta}_i^\dagger, \ldots, \boldsymbol{\theta}_{n_d^\dagger}^\dagger\right]$$

giving $n_d^\dagger$ outputs from the simulator

$$\boldsymbol{y}^\dagger = \left[y(\boldsymbol{\theta}_1^\dagger), \ldots, y(\boldsymbol{\theta}_i^\dagger), \ldots y(\boldsymbol{\theta}_{n_d^\dagger}^\dagger)\right].$$

This new data $(\boldsymbol{\Theta}^\dagger, \boldsymbol{y}^\dagger)$ will be known as the validation training dataset.

5.4.1 Individual prediction errors

The individual prediction errors (IPE) are defined as

$$D(\boldsymbol{\theta}_i^\dagger) = \frac{y(\boldsymbol{\theta}_i^\dagger) - m^*(\boldsymbol{\theta_i}^\dagger)}{\sqrt{K^*(\boldsymbol{\theta_i^\dagger}, \boldsymbol{\theta_i^\dagger})}} \quad \text{for} \quad i = 1, \ldots, n_d^\dagger.$$

It is useful to use graphical summaries of the $D(\boldsymbol{\theta}_i^\dagger)$ to assess emulator performance. If the emulator is fitting correctly, the distribution of the $D(\boldsymbol{\theta}_i^\dagger)$ should be standard normal. It would be expected that approximately 95% of the IPE are within the interval $(-2, 2)$. Figure 5.5 (left-hand panel) gives an example of how the IPE would look if the emulator was behaving as expected.

If the magnitude of the IPE is too large, this indicates that the emulator variance has been underestimated. Conversely, too many very small values indicate that the emulator variance is inflated.

Providing the emulator is fitting correctly, plots of the IPE would be expected to have random scatter around zero with no patterns appearing. Plotting $D(\boldsymbol{\theta}_i^\dagger)$ against $\boldsymbol{\theta}_i^\dagger$ will indicate particular areas of parameter space in which the emulator is badly fitting. Patterns in these plots could be a suggestion that the stationarity assumption of the covariance function is not appropriate.

5.4.2 Mahalanobis distance

The Mahalanobis distance is an extension to individual prediction errors which accounts for the correlation in the outputs. It is defined as

$$MD^2(\boldsymbol{\Theta}^\dagger) = \{\boldsymbol{y}(\boldsymbol{\Theta}^\dagger) - m^*(\boldsymbol{\Theta}^\dagger)\}^T K^*(\boldsymbol{\Theta}^\dagger, \boldsymbol{\Theta}^\dagger)^{-1}\{\boldsymbol{y}(\boldsymbol{\Theta}^\dagger) - m^*(\boldsymbol{\Theta}^\dagger)\}$$

and summarises the individual prediction errors in one single diagnostic. Conditional on the training data and hyperparameters, it has distribution

$$MD^2(\boldsymbol{\Theta}^\dagger) \sim \chi^2_{n_d^\dagger}$$

since

$$y(\boldsymbol{\theta}^\dagger) \sim N_{n_d^*}\left(m^*(\boldsymbol{\theta}^\dagger), K^*(\boldsymbol{\theta}^\dagger, \boldsymbol{\theta}^\dagger)\right).$$

5.4.3 Probability integral transform

Gneiting et al. (2007) suggest using the Probability integral transform (PIT) to check that the distributional assumptions of the emulator are reasonable. The PIT is defined as

$$P(\boldsymbol{\theta}_i^\dagger) = \Phi\{D(\boldsymbol{\theta_i}^\dagger)\} \quad \text{for} \quad i = 1, \ldots, n_d^\dagger.$$

If the distributional assumptions about the emulator are correct then $D(\boldsymbol{\theta}_i^\dagger)$ should have a standard normal distribution and $P(\boldsymbol{\theta}_i^\dagger)$ should have a standard uniform distribution. Plotting a histogram of $P(\boldsymbol{\theta}_i^\dagger)$ should look flat if the Gaussian assumption of the emulator is correct. Figure 5.5 (right-hand panel) illustrates how the PIT histogram should look for a well behaved emulator. The PIT can be regarded as an alternative graphical display of the IPE, allowing departures from the distributional assumptions of the emulator to be viewed more easily.

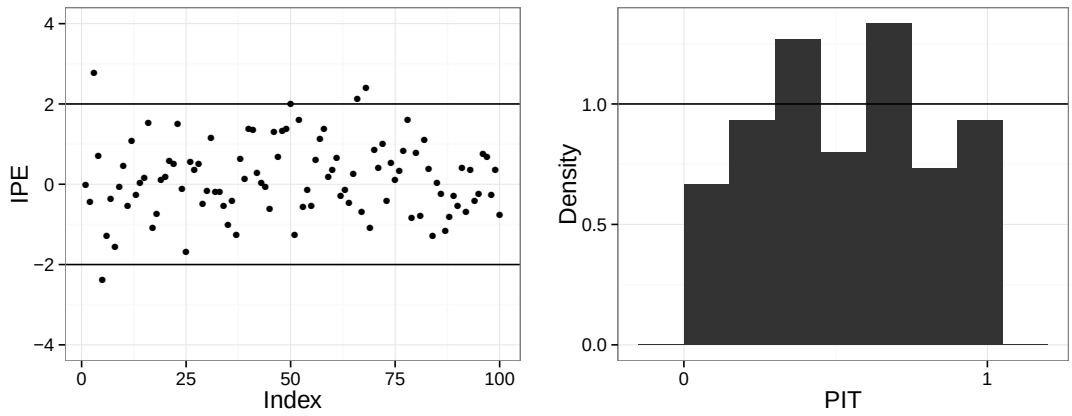

Figure 5.5: Example diagnostics with $n_d^\dagger = 50$. Plot of the IPE (left) and PIT histogram (right).

Chapter 6

Parameter inference using Gaussian process emulators

This chapter constructs Gaussian process emulators using the methodology outlined in Chapter 5 for proportions of extinction from the birth–death model. The fit of the emulators will be assessed using the diagnostics introduced in Chapter 5. Once the fitted emulators have been constructed, they will be embedded into an inference scheme and used in place of a simulator to make inferences about model parameters.

In Chapter 4, simulation–based inference schemes were successfully used to infer rate parameters. These inference schemes all relied on the ability to quickly obtain forward simulations from the model at each iteration. For the birth–death model, repeated simulation from the model is quick, hence these simulation–based algorithms are feasible. However, for larger models such as the PolyQ system, repeated simulation from the model at each iteration of an inference scheme is too slow for practical purposes. This motivates the need to find a fast approximation to the simulator and a Gaussian process emulator provides this speed–up.

Since the birth–death model is quick to simulate from, the results of parameter inference using the emulator can be compared to the results obtained in Chapter 4 using the simulator.

6.1 Emulator construction

As in Chapter 4, noisy observations x_t of proportions of extinction are observed discretely in time; a plot of the observed data was given in Figure 4.6 of Chapter 4. The data are assumed to have the following data model

$$y_t(\boldsymbol{\theta}) \equiv \text{logit}\, x_t(\boldsymbol{\theta}) = \text{logit}\, p_t(\boldsymbol{\theta}) + \sigma\epsilon_t, \quad t = 1, \ldots, T$$

where the ϵ_t are independent and $\epsilon_t \sim N(0,1)$. For the birth–death model, the proportion of extinction $p_t(\boldsymbol{\theta})$ for different choices of $\boldsymbol{\theta} = (\lambda, \mu)$ is given by

$$p_t(\boldsymbol{\theta}) = \begin{cases} \left(\frac{\mu - \mu e^{(\mu-\lambda)t}}{\lambda - \mu e^{(\mu-\lambda)t)}}\right)^{x_0}, & \text{for } \lambda \neq \mu \\ \left(\frac{\lambda t}{1+\lambda t}\right)^{x_0}, & \text{for } \lambda = \mu \end{cases} \tag{6.1}$$

where x_0 is the initial population level.

A Gaussian process emulator is required for output values $y_t(\boldsymbol{\theta}) = \text{logit}\, p_t(\boldsymbol{\theta})$. At each time point $t = 1, \ldots, T$, a Gaussian process emulator is fitted to output values

$$y_t(\boldsymbol{\theta}_i) = \text{logit}\, p_t(\boldsymbol{\theta}_i) \quad \text{for } \; i = 1, \ldots, n_d$$

by evaluating Equation 6.1 at input values $\boldsymbol{\theta}_i$, $i = 1, \ldots, n_d$. These fitted emulators have mean function

$$m_t^*(\boldsymbol{\Theta}^*) = m_t(\boldsymbol{\Theta}^*) + K_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta})^T K_t(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1}(\boldsymbol{y}_t - m_t(\boldsymbol{\Theta})) \tag{6.2}$$

and covariance function

$$K_t^*(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*) = K_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*) - K_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta})^T K_t(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} K_t(\boldsymbol{\Theta}, \boldsymbol{\Theta}^*), \tag{6.3}$$

these expression were derived in Section 5.1 of Chapter 5. Note that the explicit dependence of the fitted covariance function on hyperparameters is omitted here for

notational simplicity. The hyperparameters a and $\boldsymbol{r}$ are estimated using the scheme outlined in Algorithm 13 of Section 5.2. Further discussion regarding the estimation and treatment of hyperparameters is given in Sections 6.1.5 and 6.2.2.

6.1.1 Emulating approximate proportions

For models that are more complex than the birth–death model, such analytic results as Equation 6.1 are not available; consequently, this scenario must be explored. For the birth–death model, the proportion of extinction can be approximated using n runs of the simulator. This has been previously discussed in detail in Section 4.2.6 of Chapter 4, where the approximate proportions are denoted $\widehat{p}_{t,n}$. The empirical logit of the approximate proportion is

$$\operatorname{elogit} \widehat{p}_{t,n} = \log\left(\frac{n\widehat{p}_{t,n} + 0.5}{n - n\widehat{p}_{t,n} + 0.5}\right).$$

Gaussian process emulators can be constructed for output values

$$y(\boldsymbol{\theta}) = \operatorname{elogit} \widehat{p}_{t,n}(\boldsymbol{\theta}) \quad \text{for } i = 1, \ldots, n_d$$

by running the simulator at input values $\boldsymbol{\theta}_i$, $i = 1, \ldots, n_d$. The distribution of $\operatorname{elogit} \widehat{p}_{t,n}$ is known for large n and this sampling error can be incorporated by including a nugget term in the covariance function

$$\tilde{K}_t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) = K_t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) + \frac{\delta_{ij}}{[n \operatorname{eexpit} m_t(\boldsymbol{\theta}_i)\{1 - \operatorname{eexpit} m_t(\boldsymbol{\theta}_i)\}]}$$

where δ_{ij} is the Kronecker delta function and eexpit is the inverse of the empirical logit, i.e.

$$\operatorname{eexpit} m_t(\boldsymbol{\theta}_i) = \frac{e^{m_t(\boldsymbol{\theta}_i)}(n + 0.5) - 0.5}{n[1 + e^{m_t(\boldsymbol{\theta}_i)}]}.$$

In practice, the addition of the nugget term to the covariance function contributes only to the leading diagonal elements of the covariance matrix. This causes the mean of the

fitted emulator to no longer go through all of the training points, and the emulator variance is no longer zero at training points.

This effect can be seen for the one–dimensional birth–death example (with fixed μ) given in Figure 6.1. The panels show emulators fitted to the empirical logit of approximate proportions, simulated using $n = 10, 10^2, 10^3, 10^4$. The fitted mean function no longer goes precisely through all of the points and the fitted variance is no longer zero at training points. It can be seen that the nugget term has a much larger effect when n is small. When $n > 100$, the effect of the nugget can barely be noticed. This behaviour is desirable; when n is small, the approximation is much cruder than when n is large and the emulator uncertainty should reflect this. Unless otherwise stated, approximate proportions are constructed using $n = 1000$ runs of the simulator throughout the rest of the thesis.

For emulation with approximate proportions, the construction proceeds as in the previous section where K is replaced with $\tilde{K}$. The fitted mean and covariance are

$$m_t^*(\boldsymbol{\Theta}^*) = m_t(\boldsymbol{\Theta}^*) + \tilde{K}_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta})^T \tilde{K}_t(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1}(\boldsymbol{y}_t - m_t(\boldsymbol{\Theta})) \tag{6.4}$$

and

$$\tilde{K}_t^*(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*) = \tilde{K}_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*) - \tilde{K}_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta})^T \tilde{K}_t(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} \tilde{K}_t(\boldsymbol{\Theta}, \boldsymbol{\Theta}^*). \tag{6.5}$$

Since the covariance function $\tilde{K}(\cdot, \cdot)$ can no longer be written as $a \times R(\cdot, \cdot)$, the scheme for estimating hyperparameters introduced in Algorithm 13 of Section 5.2 may no longer be used. However, the scheme presented in Algorithm 14 where a is not marginalised over can be used.

Comment on emulator construction

The above approach fits Gaussian process emulators to output values $y_t(\boldsymbol{\theta}_i)$ at each time point $t = 1, \ldots, T$. This results in the construction of T emulators, each of which

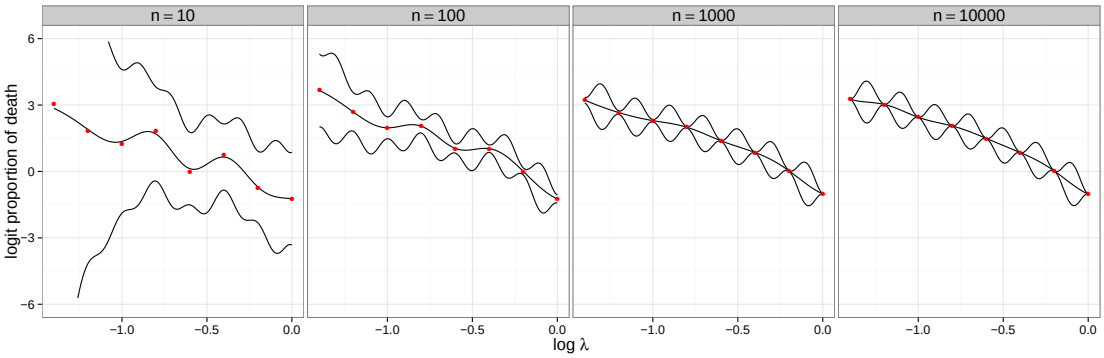

Figure 6.1: 1-D example of fitted emulator. Training data used to build emulator shown in red points. The mean and two standard deviations from the fitted emulator are shown in black. The different panels represent emulators built on approximate proportions where $n = 10, 10^2, 10^3, 10^4$ (where n is the number of forward simulations from the model used to construct proportions).

has input space with the same dimension as $\boldsymbol{\theta}$.

An alternative approach would have been to build a single emulator which has an extra dimension for time t. The advantage of this approach is that it only requires the construction of a single emulator; in this respect the implementation is simpler. This was tried in practice; however, it was found that the number of design points n_d required was much larger since they had to cover the space of the extra input, t.

It has been decided that the first approach will be most suitable. Firstly, the times at which experimental data are observed are known, there will be no need to make predictions at new time points. Secondly, the construction of the emulator scales with order $\mathcal{O}(n_d^3)$ since evaluating the likelihood for the hyperparameters involves inverting the $n_d \times n_d$ covariance matrix. Having T smaller emulators will be computationally beneficial and the construction of the emulators can also be parallelised.

6.1.2 Training data design

An appropriate area of parameter space must be selected over which to obtain the training data. For the birth–death model, the aim is to make inferences on rate parameters λ and μ. A sensible choice would be to use prior beliefs about λ and μ to guide this choice. In Chapter 4, it was assumed *a priori* that λ and μ were independent

and

$$\lambda \sim \text{Log-Normal}(\log 0.6, 0.5)$$

$$\mu \sim \text{Log-Normal}(\log 1, 0.5).$$

The Latin hypercube for the emulator will be constructed to cover the central 95% of the prior distributions for $\log \lambda$ and $\log \mu$. If there is found to be much posterior support in an area of parameter space not covered by the design space, then re-fitting the emulators over a larger design space may be sensible. It must also be noted that provided an appropriate prior mean function is specified, predictions for inputs outside of the design space should still be sensible.

The Latin hypercube will be constructed for the log parameters since the inference scheme uses a random walk over the log parameters. The range of the Latin hypercube is

$$-1.9 < \log \lambda < 0.88$$

$$-1.4 < \log \mu < 1.4.$$

The training data can be seen in Figure 6.2 for $n_d = 50$, where the Latin hypercube was constructed using the *maximin* design discussed in Chapter 5.

6.1.3 Choice of mean function

Before fitting the emulators, an appropriate prior mean function $m(\boldsymbol{\theta})$ must be chosen. When the fitted emulator attempts to make predictions for inputs which are far away from training inputs, the predicted mean reverts to the prior mean function. For this reason, a careful choice of prior mean function is necessary. Further discussion on the choice of prior mean function was given in Section 5.1.2 of Chapter 5.

Before deciding on a mean function, it is helpful to consider the shape of the function to be emulated. Figure 6.3 illustrates the birth–death process. The left hand panel

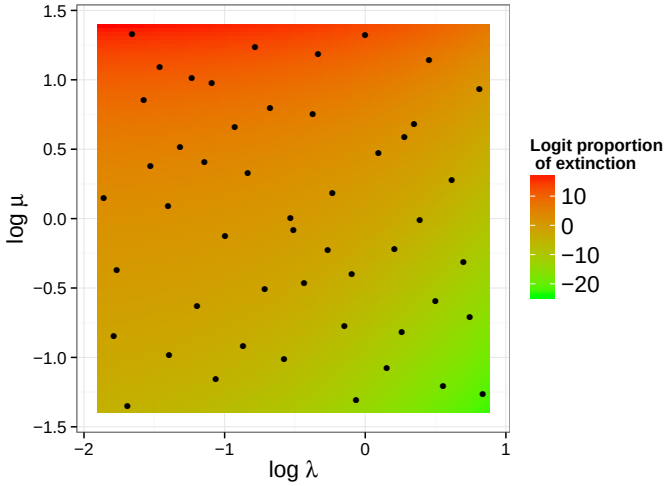

Figure 6.2: Birth–death model: training data used to build emulators shown in black dots; the background shows how the logit proportion of extinction varies with log λ and log μ .

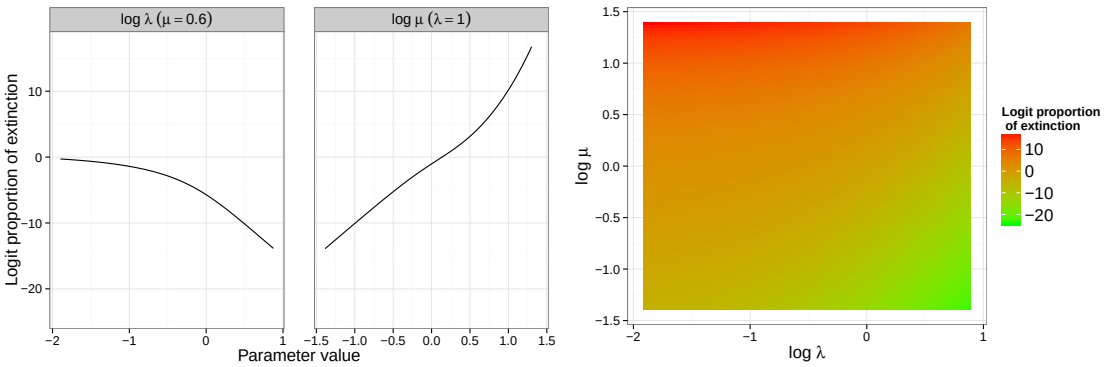

Figure 6.3: Birth–death model: logit proportions of extinction plotted against log λ for fixed μ (left), against log μ for fixed λ (middle), against log λ and log μ (right).

Number	Mean Function
a	β_0
b	$\beta_0 + \beta_1 \log\lambda + \beta_2 \log\mu$
c	$\beta_0 + \beta_1 \log\lambda + \beta_2 \log\mu + \beta_3(\log\lambda)^2 + \beta_4(\log\mu)^2 + \beta_5 \log\lambda \log\mu$

Table 6.1: Mean functions.

shows how the logit proportion of extinction changes with $\log\lambda$ (for fixed $\mu = 1$) and for $\log\mu$ (for fixed $\lambda = 0.6$). The right hand panel shows how the logit proportion of extinction changes for both $\log\lambda$ and $\log\mu$. Clearly it can be seen that the logit proportion of extinction decreases with λ and increases with μ. This is expected, since increasing the birth rate λ will cause the population level to increase, hence reducing the likelihood of the population becoming extinct. The converse is true for an increase in the death rate μ.

Table 6.1 presents three different choices of mean function. Mean function (a) represents a constant mean function which expresses no prior beliefs about the simulator output at different inputs. Mean function (b) includes linear terms in $\log\lambda$ and $\log\mu$ and mean function (c) includes higher order terms to reflect the non-linear behaviour. When fitting the Gaussian process emulator the β parameters in the mean function are fixed at the maximum likelihood (least squares) estimates.

6.1.4 Choice of covariance function

Since the birth–death process has two parameters, the squared exponential covariance function takes the form

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}) = a \exp\left\{ -\left(\frac{(\theta_{i1} - \theta_{j1})^2}{r_1^2} + \frac{(\theta_{i2} - \theta_{j2})^2}{r_2^2} \right) \right\}$$

and

$$K(\lambda_i, \mu_i, \lambda_j, \mu_j | a, \boldsymbol{r}) = a \exp\left\{ -\left(\frac{(\log\lambda_i - \log\lambda_j)^2}{r_1^2} + \frac{(\log\mu_i - \log\mu_j)^2}{r_2^2} \right) \right\}.$$

There are three hyperparameters to be estimated a, r_1 and r_2. The parameter r_1 relates to the effect the λ parameter has on the output and r_2 relates to the effect the μ parameter has on the output.

The use of this covariance function has the property that the mean of the fitted emulator will go though all of the design points and the variance will be zero at these points. This is appropriate behaviour for a deterministic simulator.

It was found in practice that this choice of covariance function presents computational difficulties when attempting to compute the inverse of the covariance matrix. This is due to the fact that the condition number (the ratio of the largest and smallest eigenvalues) becomes too large. It is well known that covariance matrices of this form suffer from this problem (Ababou et al., 1994; Neal, 1997). A common way to overcome this problem is to add a jitter (or nugget) parameter to the covariance function, i.e.

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | a, \boldsymbol{r}, g) = a \left(\exp \left\{ - \left(\frac{(\theta_{i1} - \theta_{j1})^2}{r_1^2} + \frac{(\theta_{i2} - \theta_{j2})^2}{r_2^2} \right) \right\} + g\delta_{ij} \right)$$

where δ_{ij} is the Kronecker delta function and g is the jitter parameter. In practice, this has the effect of adding a small constant down the leading diagonal of the matrix. This is very effective in reducing numerical instabilities. It has been suggested that the jitter term should be as close to zero as is possible while still overcoming the numerical instabilities (Ranjan et al., 2011).

In the context of the birth–death model, the addition of the jitter term was initially sought purely for the purposes of preventing the covariance function becoming ill-conditioned. Since g was thought to serve no real purpose, a small value of g was chosen (approximately 10^{-6}) and used for each emulator.

However, on further investigation it became clear that there are other advantageous reasons for including the jitter term. Gramacy and Lee (2010) strongly claim that even for deterministic simulators, omitting a jitter term is foolish. They present several reasons why including a jitter term is a good idea. These arguments focus on the better *statistical properties* which are achieved when a jitter term is included. In particular, Gramacy

Parameter	Exact proportions	Approximate proportions
a	Inv-Gamma$(0.001, 0.001)$	Log-Normal$(0, 10)$
r_1	Log-Normal$(0, 10)$	Log-Normal$(0, 10)$
r_2	Log-Normal$(0, 10)$	Log-Normal$(0, 10)$
g		Log-Normal$(-19, 10)$

Table 6.2: Prior distributions for hyperparameters.

and Lee (2010) note that the stationarity assumption of the covariance function is often too strong unless a particularly well fitting prior mean function is used. They also note the underlying assumptions about the correlation structure may not be a true reflection of reality. The addition of the jitter term provides protection to small violations of these assumptions.

For these reasons, it has been decided that the g parameter should be estimated in the MCMC scheme along with the other hyperparameters. Note that if the assumptions of the Gaussian process held precisely, there would be no information about the g parameter and it could not be estimated.

6.1.5 Hyperparameter estimation

The prior distributions for hyperparameters are given in Table 6.2, they represent vague prior knowledge about the hyperparameters. For exact proportions, the scheme presented in Algorithm 13 of Chapter 5 is used; the conjugate choice of prior distribution for a is Inverse-Gamma. For the approximate proportions, Algorithm 14 of Chapter 5 is used and there is no conjugate prior, a natural choice for the prior distribution of a is Log-Normal.

Figures 6.4 and 6.5 show marginal posterior distributions for the hyperparameters of the covariance function for emulators fitted to exact and approximate proportions of extinction.

It can be seen that the marginal posterior distributions for all hyperparameters are very informative compared to their prior distributions. One observation is that for each emulator, the posterior mean is lower as the mean function increases in complexity, most

notably for the a parameter. The a parameter can be thought of a general variance term and represents the variability in the output. As the mean function becomes more complex, more variability is removed from the residuals and it would be expected that the a parameter would be lower.

Additionally, r_1 and r_2 are generally lower as the mean function becomes more complex. This is also expected since larger values of r_i imply the output is more dependent on input i.

For the emulators fitted to exact proportions, there was the extra jitter parameter g to estimate. This parameter describes the lack of fit of the model, for example, due to violations of the assumptions of the Gaussian process emulators. It was seen that for the most simple mean function, the posterior is most peaked, suggesting that there is most information about g when a less complicated mean function is used. This is as expected; it is more likely that the assumptions of the Gaussian process have been violated if an inadequate mean function is used.

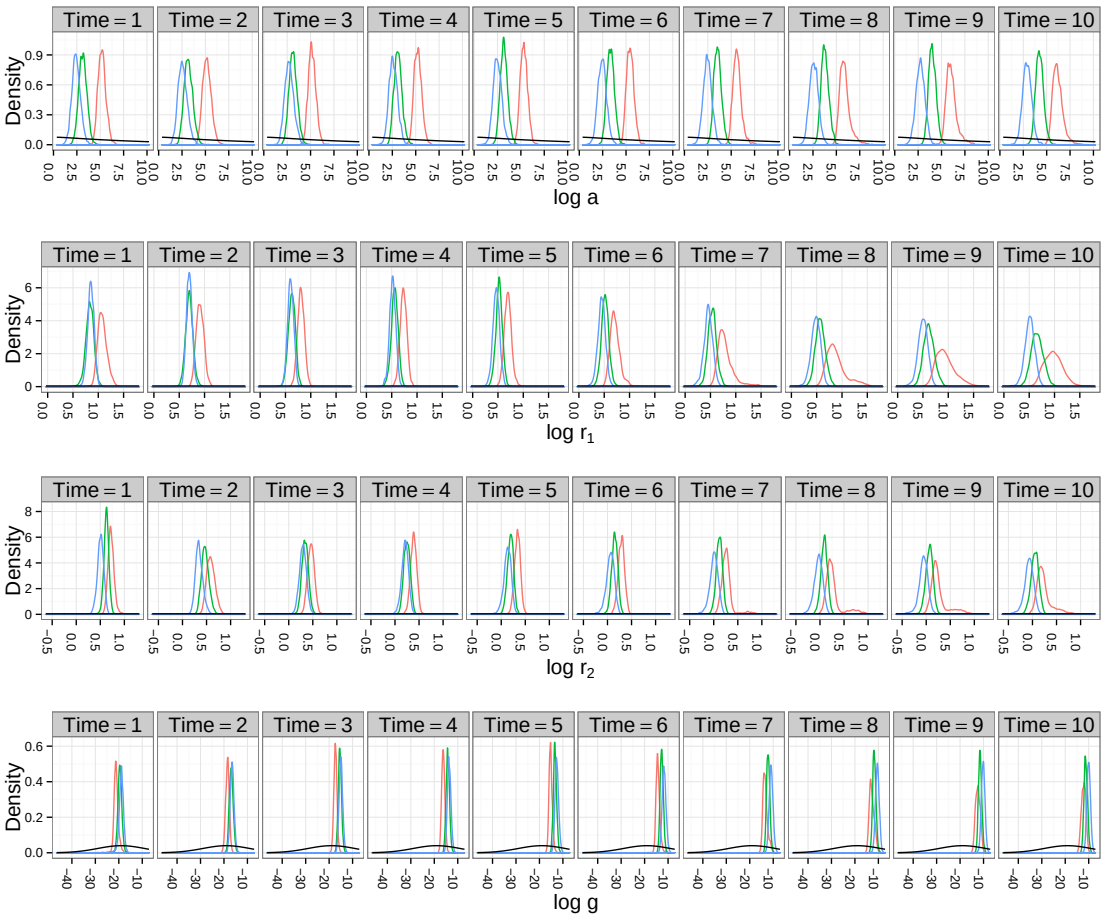

Figure 6.4: Marginal posterior distributions for hyperparameters for emulators fitted to exact proportions of extinction. Mean function (a) is given in red, mean function (b) green and mean function (c) in blue. Prior distributions are shown in black.

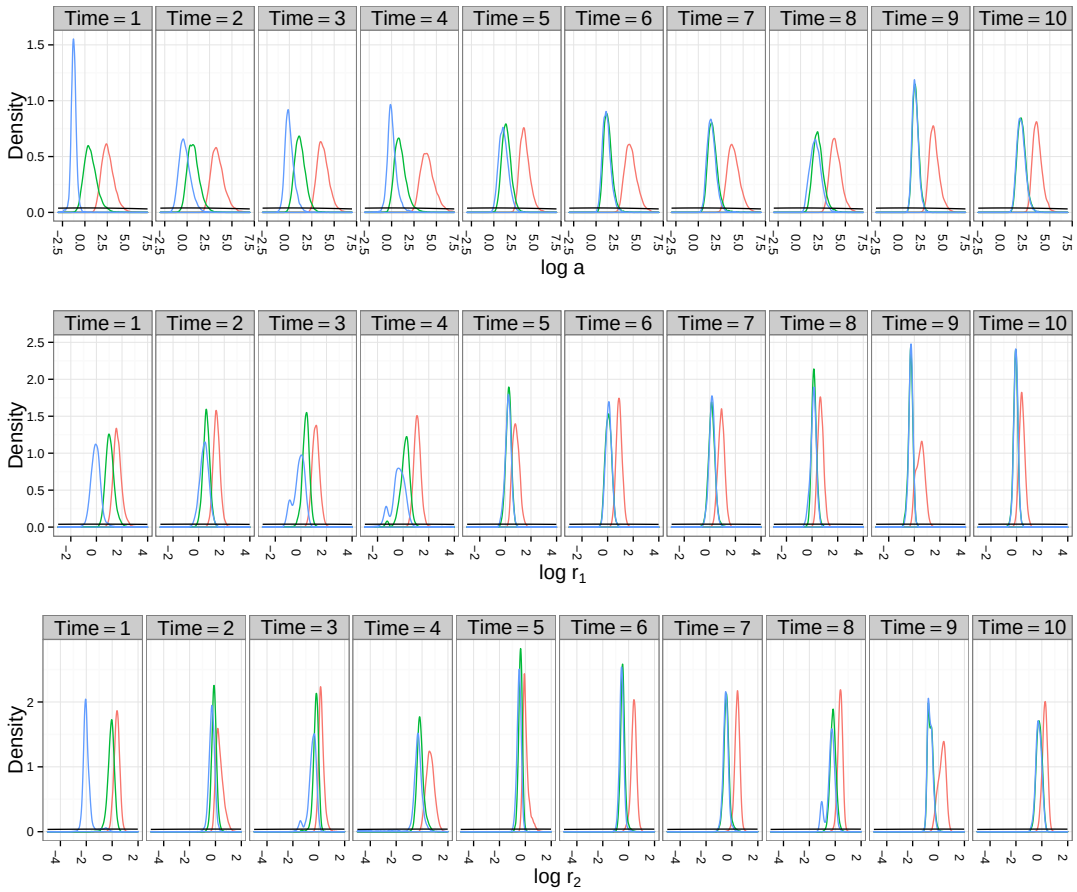

Figure 6.5: Marginal posterior distributions for hyperparameters for emulators fitted to approximate proportions of extinction. Mean function (a) is given in red, mean function (b) green and mean function (c) in blue. Prior distributions are shown in black.

6.1.6 Diagnostics

Diagnostics for the fitted emulators can be seen in Figures 6.6 (for exact proportions) and 6.7 (for approximate proportions) for a validation dataset simulated consisting of $n_d^\dagger = 25$ points. The Latin hypercube over which the validation dataset was constructed was over the same parameter range as the training data. The diagnostics used were explained in detail in Section 5.4 of Chapter 5.

In each figure, the top plot shows the IPE (individual prediction errors). If the emulators are fitting correctly, it would be hoped that most of the IPE values lie between ± 2. It would also be hoped that the points would be randomly scattered around zero with no obvious patterns.

The IPE (y-axis) are plotted against λ (x-axis) and coloured according to μ. This is to check that the emulators are fitting well in all areas of parameter space. It can be seen that there is no obvious pattern in the IPE for any of the choices of mean function.

The middle plot shows the PIT histograms for the fitted emulators. It would be hoped that the PIT values have a standard uniform distribution. It can be seen that there are deviations from the standard uniform distribution which suggests that the tails of the emulators are slightly too heavy.

The bottom plot shows the Mahalanobis distance, which takes into account the correlation in the residuals. For this diagnostic, there is only one value for each emulator and making comparisons between the fit of different emulators is more convenient. It can be seen that the Mahalanobis distances are all well below the upper 99% percentile of χ^2_{25} distribution suggesting well fitting emulators. It is concluded that the fit of the emulators is acceptable.

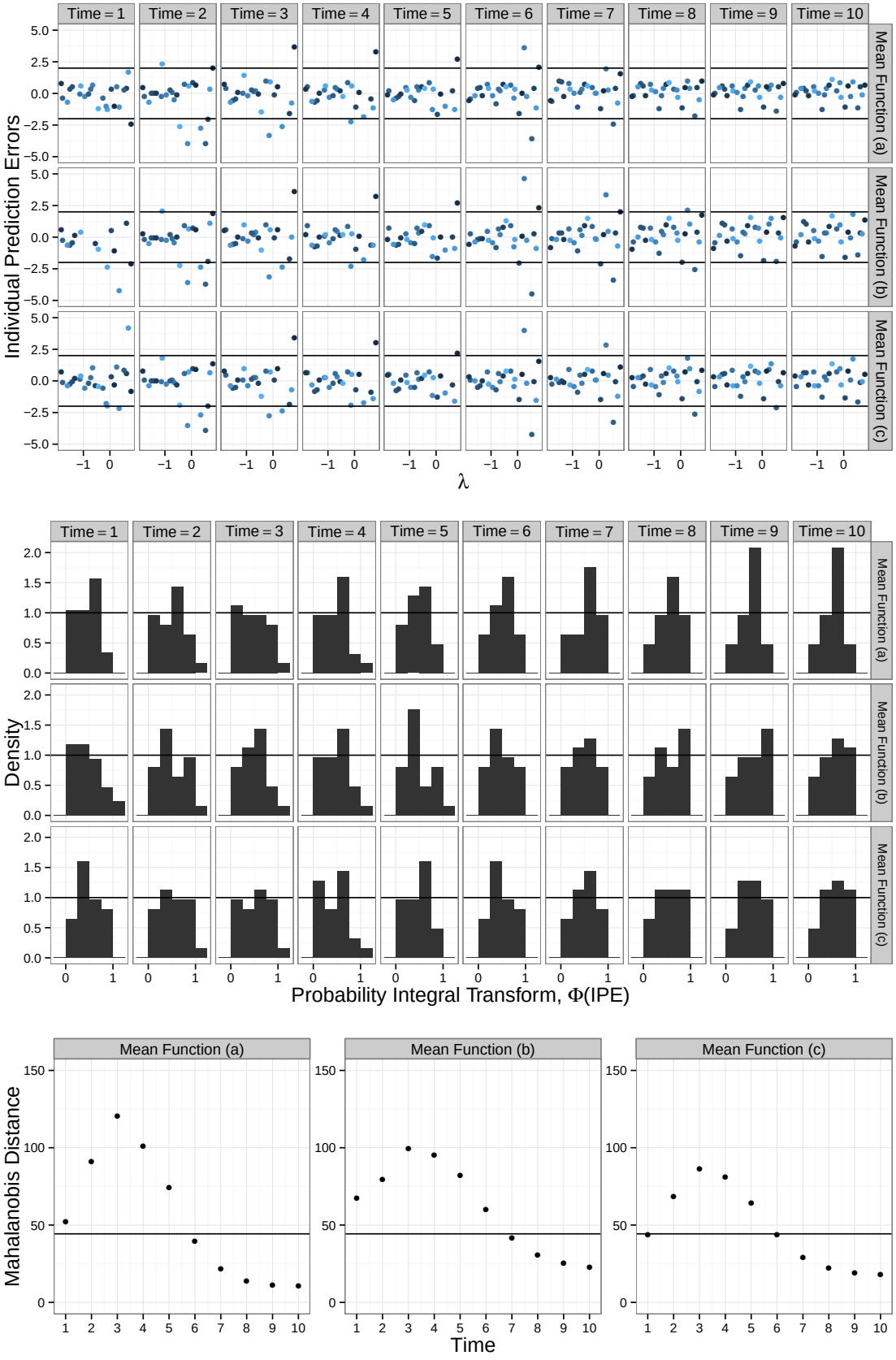

Figure 6.6: Diagnostic for the birth–death model for emulators fitted to exact proportions of extinction.

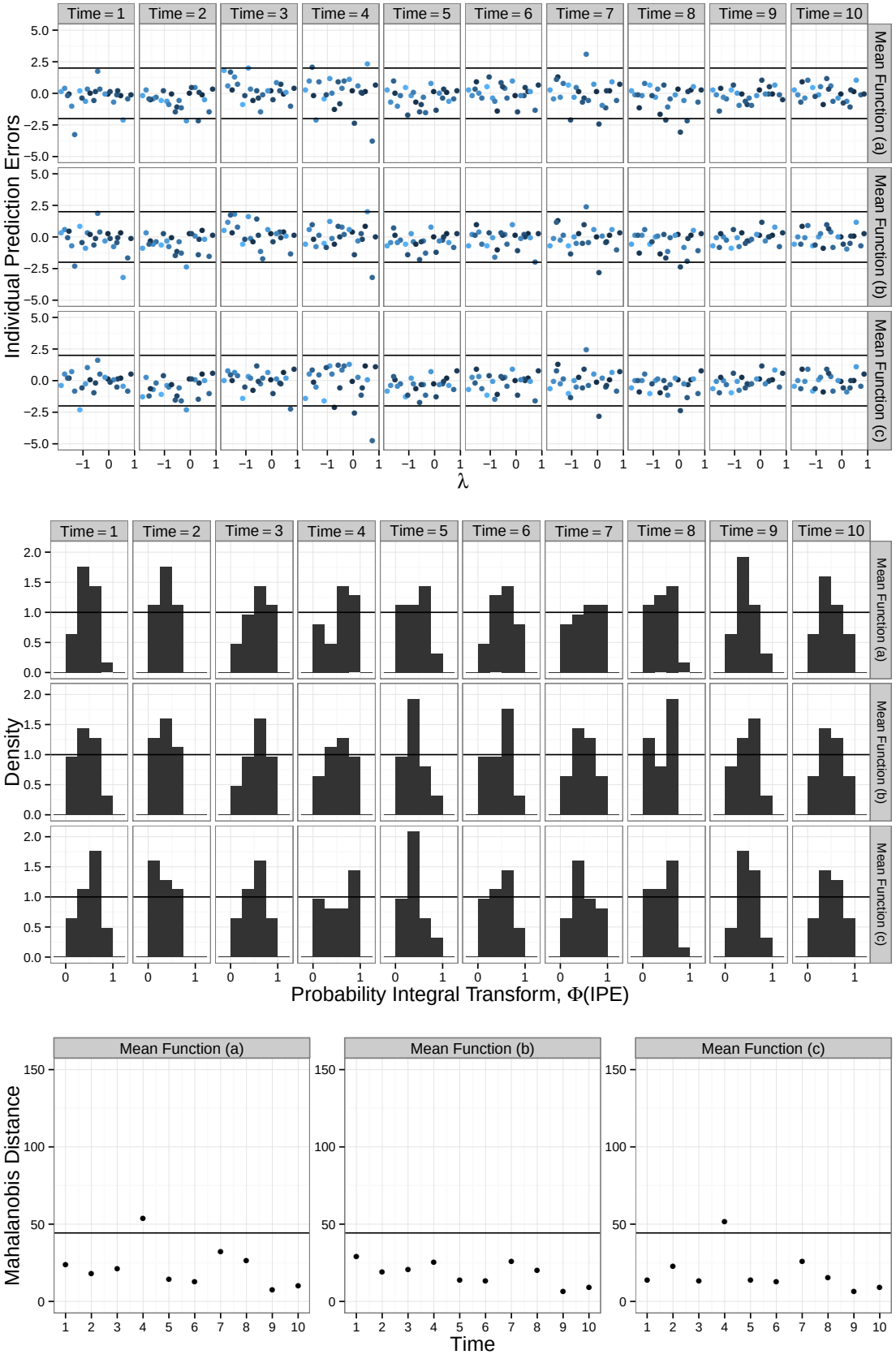

Figure 6.7: Diagnostic for the birth–death model for emulators fitted to approximate proportions of extinction (with $n = 1000$).

6.2 Inference for model parameters

Inference for the model parameters of the birth–death model, λ and μ using the emulators constructed previously is now explored. When emulators are fitted to exact proportions, the data model is, for $t = 1, \ldots, T$

$$y_t \equiv \operatorname{logit} x_t | \boldsymbol{\theta}, \sigma \sim N(m_t^*(\boldsymbol{\theta}), K_t^*(\boldsymbol{\theta}) + \sigma^2), \quad \text{independently}$$

and so

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}, \sigma) = \prod_{t=1}^{T} \phi(y_t | m_t^*(\boldsymbol{\theta}), K_t^*(\boldsymbol{\theta}) + \sigma^2).$$

and for approximate proportions

$$y_t \equiv \operatorname{logit} x_t | \boldsymbol{\theta}, \sigma \sim N(m_t^*(\boldsymbol{\theta}), \tilde{K}_t^*(\boldsymbol{\theta}) + \sigma^2), \quad \text{independently}$$

and so

$$\pi(\boldsymbol{y}|\boldsymbol{\theta}, \sigma) = \prod_{t=1}^{T} \phi(y_t | m_t^*(\boldsymbol{\theta}), \tilde{K}_t^*(\boldsymbol{\theta}) + \sigma^2).$$

If is assumed *a priori* that $\boldsymbol{\theta}$ and σ are independent, the posterior density is

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

If it is assumed *a priori* that $\theta_i \sim \mathrm{LN}(c_{\theta_i}, 1/d_{\theta_i})$ and $\sigma \sim \mathrm{LN}(c_\sigma, 1/d_\sigma)$ then

$$\log \left\{ \pi(\boldsymbol{y}|\boldsymbol{\theta}, \sigma) \sigma \pi(\sigma) \prod_{i=1}^{n_p} \theta_i \pi(\theta_i) \right\} = k - \frac{1}{2} \sum_{t=1}^{T} \log \left\{ K_t^*(\boldsymbol{\theta}) + \sigma^2 \right\} - \frac{1}{2} \sum_{t=1}^{T} \frac{\{y_t - m_t^*(\boldsymbol{\theta})\}^2}{K_t^*(\boldsymbol{\theta}) + \sigma^2} - \frac{d_\sigma}{2} (\log \sigma - c_\sigma)^2 - \frac{1}{2} \sum_{i=1}^{n_p} d_{\theta_i} (\log \theta_i - c_{\theta_i})^2.$$

The conditional distributions of $\boldsymbol{\theta}|\sigma, \boldsymbol{y}$ and $\sigma|\boldsymbol{\theta}, \boldsymbol{y}$ do not have recognisable forms; an MCMC scheme using a Metropolis-Hastings step is given in Algorithm 16.

Algorithm 16 MCMC scheme for model parameters using emulators

For each iteration of the scheme:

1. Sample $(\boldsymbol{\theta}^*, \sigma^*)^T$ from a symmetric proposal distribution q, on the log scale.
2. Compute the acceptance probability

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

3. Set $(\boldsymbol{\theta}, \sigma) = (\boldsymbol{\theta}^*, \sigma^*)$ with probability α, otherwise retain $(\boldsymbol{\theta}, \sigma)$. else keep $(\boldsymbol{\theta}, \sigma)^T$.

6.2.1 Results of inference using emulators

Results of inference on model parameters using emulators are given in Figure 6.8 (using the same experimental data from Chapter 4), where the top panel relates to inference using emulators with exact proportions of extinction, and the bottom panel relates to inference with emulators using approximate proportions (with $n = 1000$). In Chapter 4, these marginal posterior distributions were obtained using a simulator rather than an emulator, the posterior distributions resulting from these schemes will be referred to as *exact* or *vanilla*. It is advantageous that these *gold standard* posterior distributions exist, since they provide a reference point for comparing emulator performance.

Firstly, considering the results for the exact proportions (top panel), it can be seen that for all choices of mean function, comparing the marginal posterior distributions to the exact posterior (red line), the difference is indistinguishable. This suggests that the emulators are acting as an excellent surrogate to the simulator and are fairly insensitive to the choice of mean function.

Secondly, considering the results for the approximate proportions (bottom panel), it can be seen that, for all choices of mean function, the posterior distributions are very close to those obtained when using the simulator.

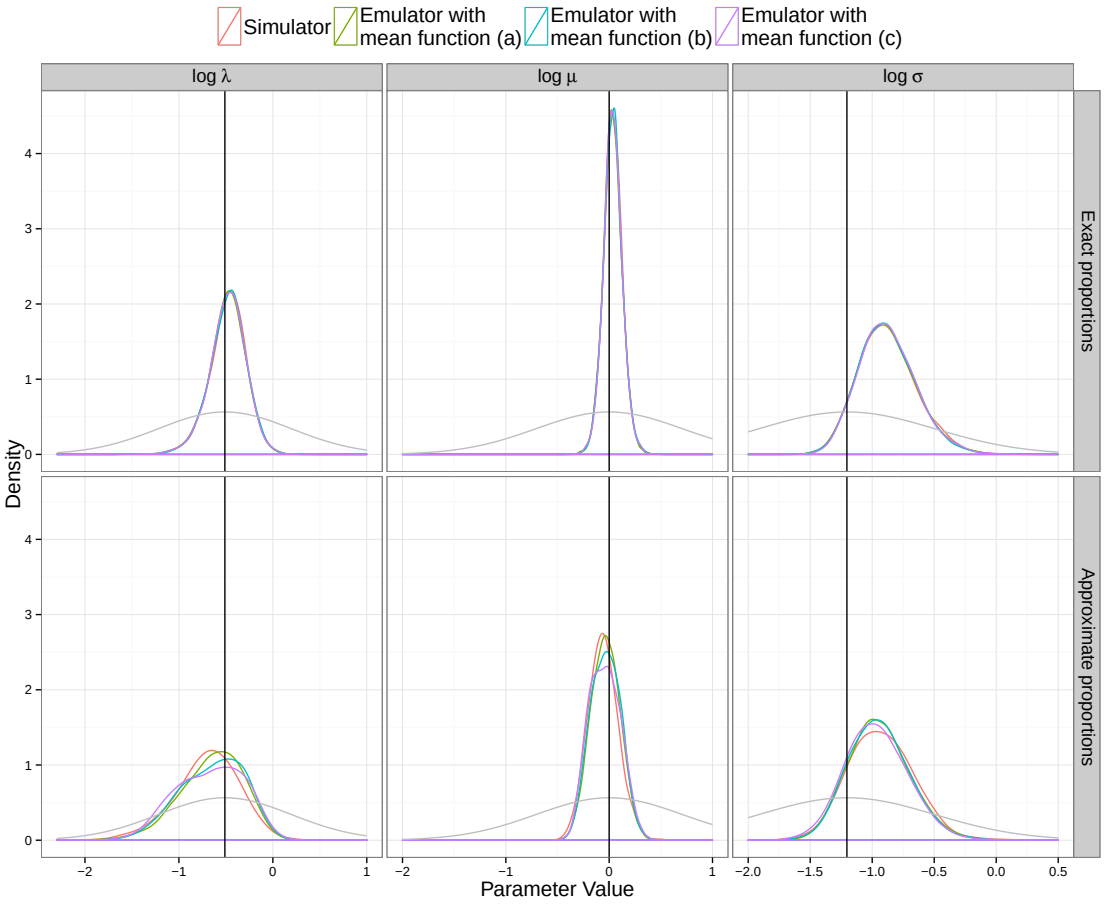

Figure 6.8: Results of inference on the birth–death model. Top panel: using emulators fitted to exact proportions of extinction. Bottom panel: using emulators fitted to approximate proportions of extinction.

6.2.2 Considering the uncertainty of hyperparameters

It must be noted that the fitted mean function and covariance function are dependent on hyperparameters $\boldsymbol{\psi}$ from the covariance function. Note $\boldsymbol{\psi} = (a, r_1, r_2, g)^T$ when exact proportions are used and $\boldsymbol{\psi} = (a, r_1, r_2)^T$ when approximate proportions are used. Explicitly, the distribution of the fitted emulator is

$$y(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\beta}, \boldsymbol{\psi}) \sim N\big(m_t^*(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi}), K_t^*(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi})\big).$$

However, an approximation to this distribution is found by noting that

$$y(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi}) = E_{a,\boldsymbol{r}|\boldsymbol{\Theta}}\big[y(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi})\big] \simeq \frac{1}{m}\sum_{i=1}^{m} y(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi}_i)$$

and

$$\frac{1}{m}\sum_{i=1}^{m} y(\boldsymbol{\Theta}^*|\boldsymbol{\psi}_i) \sim N\left(\frac{1}{m}\sum_{i=1}^{m} m_t^*(\boldsymbol{\Theta}^*|\boldsymbol{\psi}_i), \frac{1}{m}\sum_{i=1}^{m} K_t^*(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*|\boldsymbol{\psi}_i)\right).$$

A cruder approximation is found by ignoring the posterior uncertainty in $\boldsymbol{\psi}$, giving

$$\frac{1}{m}\sum_{i=1}^{m} m_t(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \boldsymbol{\psi}_i) \simeq m_t^*(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \mathbb{E}[\boldsymbol{\psi}|\boldsymbol{\Theta}])$$

and

$$\frac{1}{m}\sum_{i=1}^{m} K_t(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*|\boldsymbol{\psi}_i) \simeq K_t^*(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*|\boldsymbol{\Theta}, \mathbb{E}[\boldsymbol{\psi}|\boldsymbol{\Theta}]).$$

This is the approach that has been used previously in this chapter when fitting emulators. However, it is necessary to consider how sensitive the emulator predictions are to this choice. Figure 6.9 considers the sensitivity of the marginal posterior distributions for model parameters to fixing the emulators hyperparameters at their posterior means.

For emulation of exact proportions (top panel), it can be seen that averaging over the posterior uncertainty in hyperparameters produces marginal posterior distributions which are indistinguishable to those when the posterior uncertainty is ignored. It can

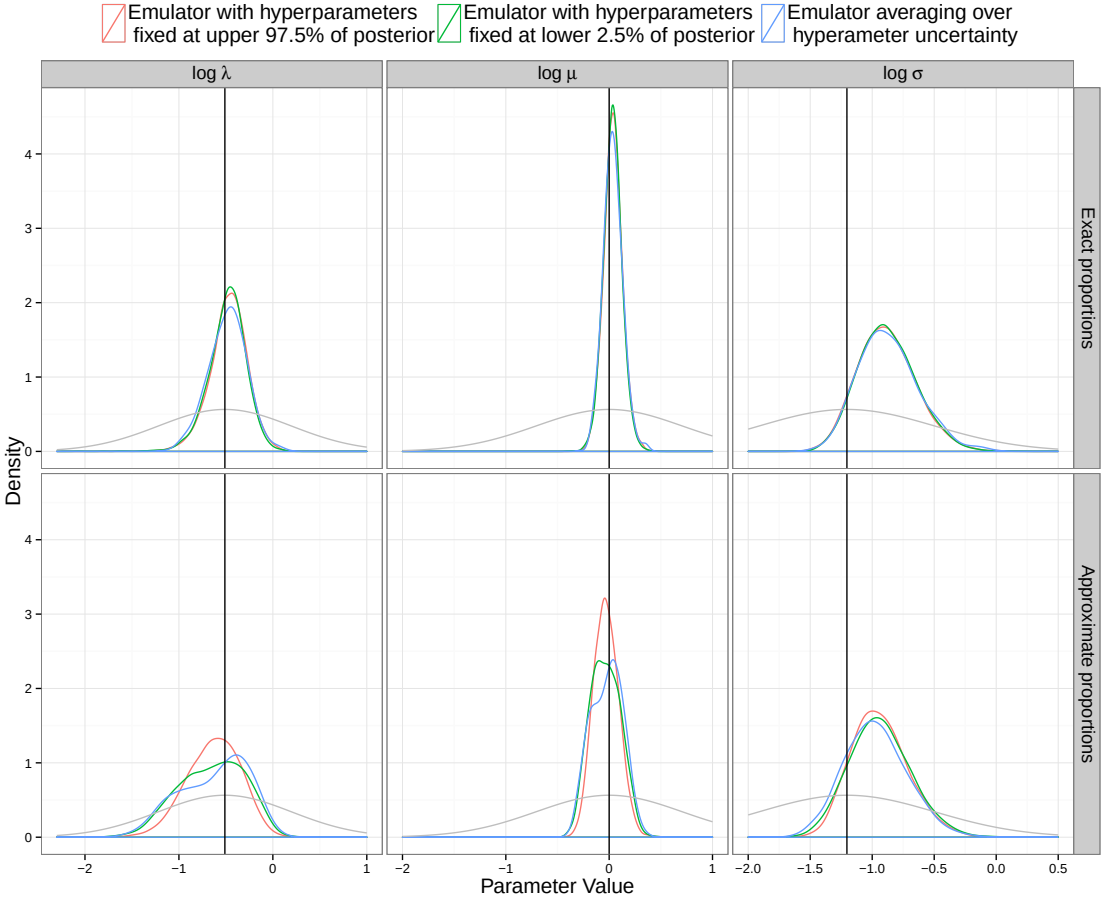

Figure 6.9: Results of inference on the birth–death model, considering the sensitivity to the posterior uncertainty in hyperparameters. Top panel: using emulators fitted to exact proportions of extinction. Bottom panel: using emulators fitted to approximate proportions of extinction

also be seen that fixing the hyperparameters at the lower 2.5% and upper 97.5% points of their posterior distributions has an imperceptible effect on the posterior distributions obtained for the model parameters.

When emulating approximate proportions (bottom panel), these corresponding results are not quite as similar, however, they are still very close. These observations suggest that the fitted emulators are insensitive to the posterior uncertainty in the hyperparameters and suggest that fixing them at their posterior mean is perfectly adequate.

6.3 Emulators with sparse covariance matrices

Emulators with sparse covariance matrices were also fitted to the same training data used in previous sections; this approach was outlined in detail in Section 5.3.1 of Chapter 5. The idea is to take advantage of the near sparsity of covariance matrices by constructing them in a way such that they can be stored as sparse matrices. Computationally efficient sparse matrix algorithms can then be used to speed up operations such as matrix inversions which would otherwise scale with $\mathcal{O}(n_d^3)$.

Kaufman et al. (2011) use a rich prior mean structure which takes into account large scale variation in the output. They suggest using a linear combination of basis functions, such as Legendre polynomials. The advantage of using a more complex mean structure is that it reduces the amount of covariance structure which the Gaussian process must model.

For the sparse covariance approach, an alternative covariance function is used which has hyperparameters τ_1 and τ_2. Here, τ_i represents the distance between two inputs in the *ith* direction before the output is assumed uncorrelated (see Section 5.3.1 of Chapter 5 for more details). Note, that all inputs are first scaled such that they lie between 0 and 1 meaning that τ_i can only take values in $[0, 1]$. *A priori* it is assumed

that $\boldsymbol{\tau}$ is uniformly distributed over the space

$$\frac{1}{n_p}\sum_{k=1}^{n_p}\tau_k \leq c$$

where c is chosen to satisfy $c(2-c) = (1-s)^{1/n_p}$.

The marginal posterior distributions for $\log \tau_1$ and $\log \tau_2$ can be seen in Figure 6.10 for different levels of sparsity. As the sparsity level increases, τ_1 and τ_2 get closer to zero. This is to be expected as the greater the level of sparsity imposed on the matrix, the smaller distance apart inputs can be before their outputs are considered uncorrelated.

Diagnostics for these emulators can be seen in Figures 6.11 (exact proportions) and 6.12 (approximate proportions). Using the criteria discussed previously, it can be seen that plots of the IPE show no unsually large values, however, the PIT histograms confirm that there appear to be departures from normality in the IPE. However, it must be noted that the sparse covariance approach represents an approximation with considerable speed up, and it would be expected that these emulators do not fit as well as those built under the original scheme.

The results of inference on model parameters using emulators can be seen in Figure 6.13. They are compared with the marginal posterior distributions obtained in Chapter 4 using the simulator.

It can be seen that for exact proportions (top panel), emulators with sparse covariance matrices provide posterior distributions for model parameters which are very similar to those obtained using the simulator. However, it must be noted that as the sparsity level increases, the posterior distributions start to look more different to results obtained using the simulator.

For approximate proportions (bottom panel), λ and μ are well recovered. However, the emulator has failed to capture the measurement error structure and the σ parameter is not well recovered. This suggests that the emulator variance is inaccurate.

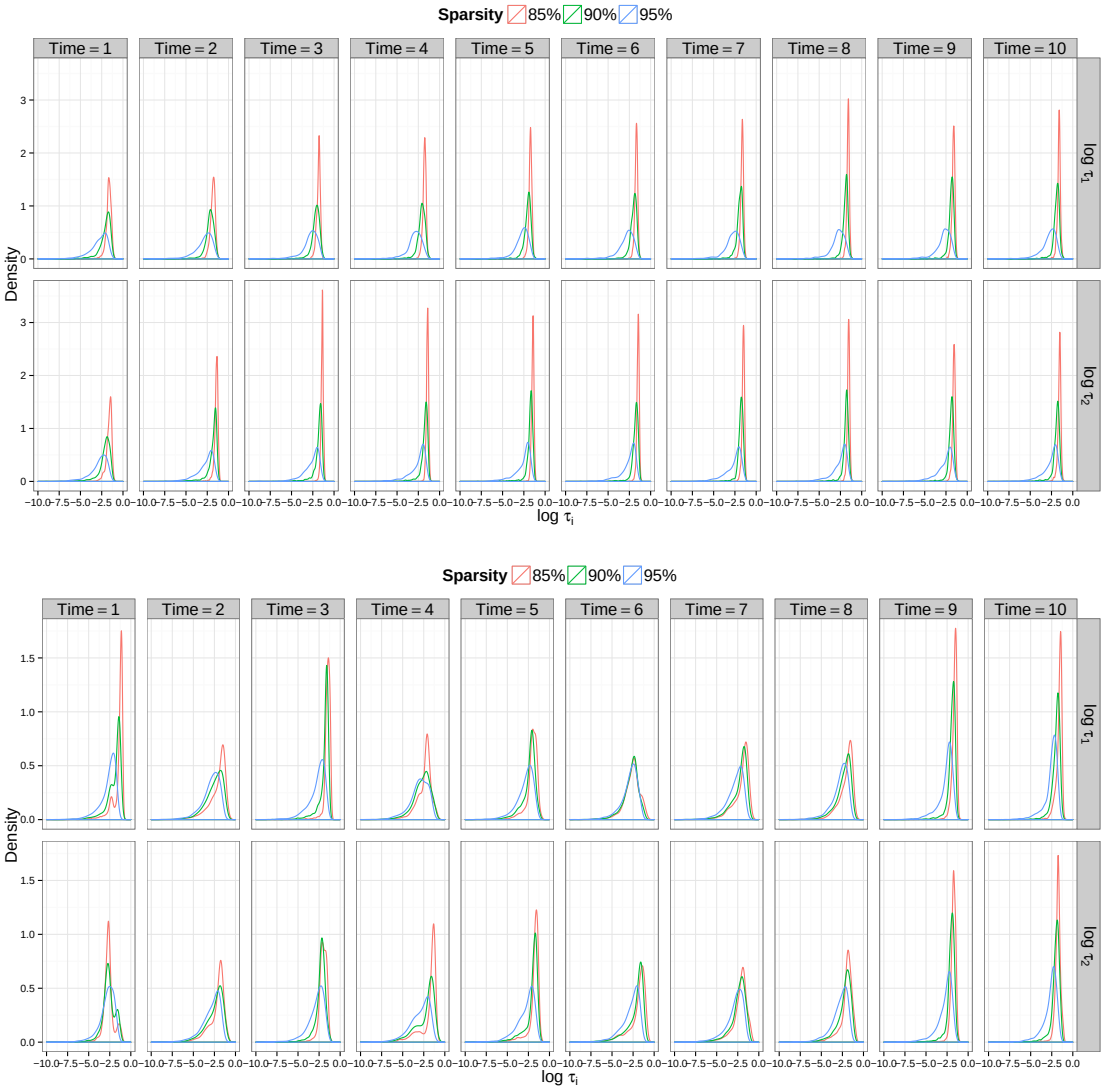

Figure 6.10: Marginal posterior distributions for hyperparameters using sparse covariance matrices

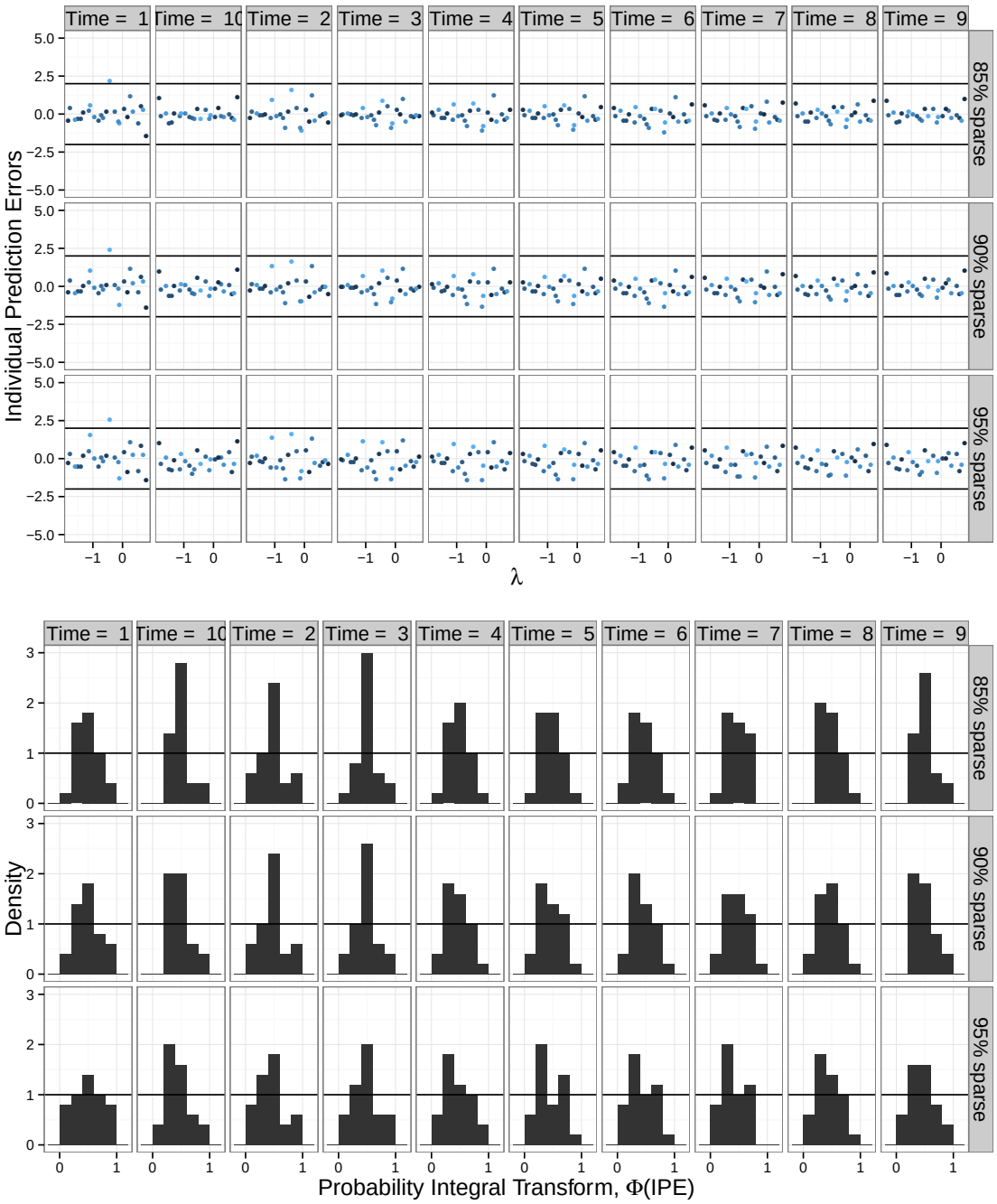

Figure 6.11: Diagnostics for emulators with sparse covariance matrices fitted to exact proportions.

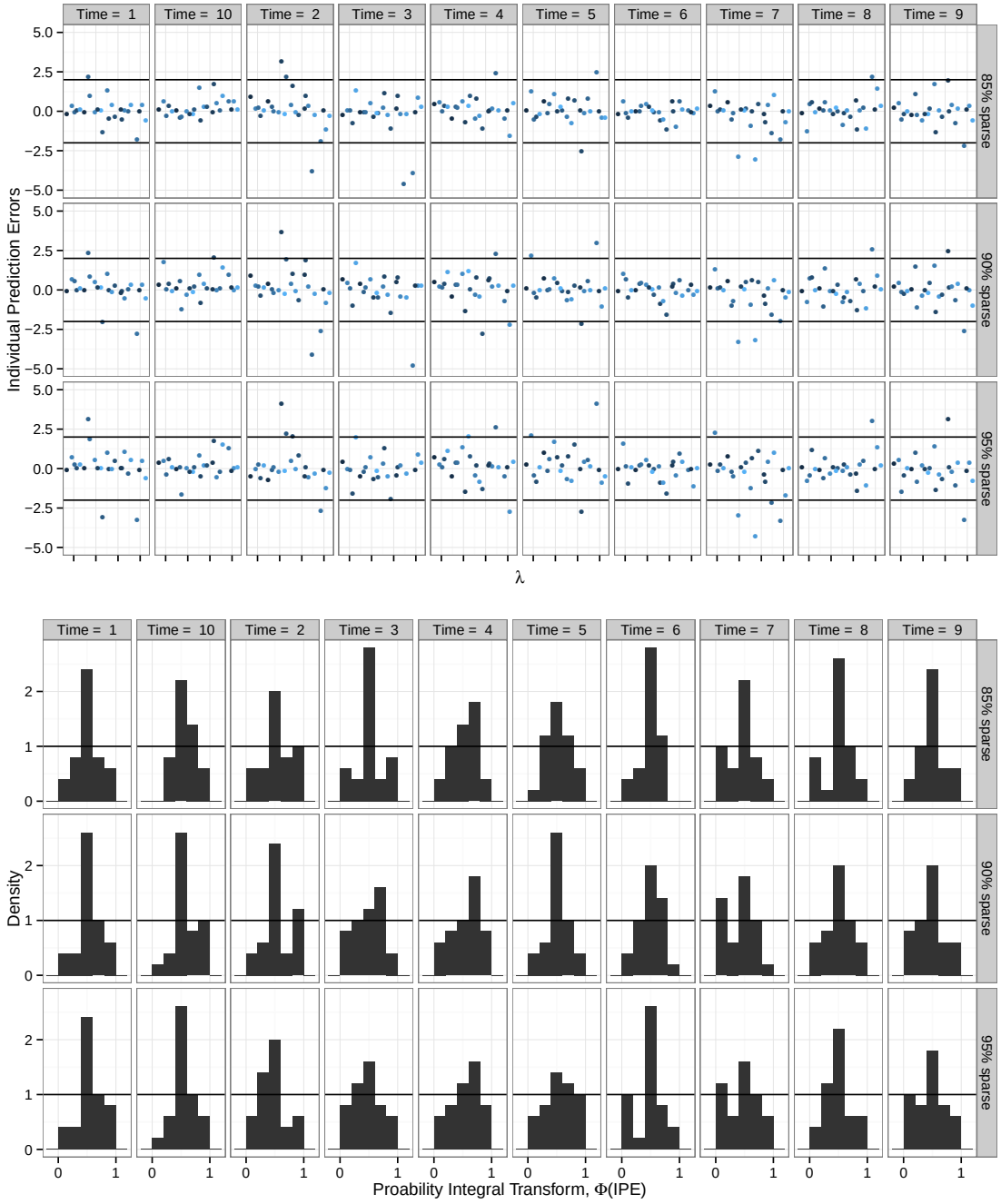

Figure 6.12: Diagnostics for emulators with sparse covariance matrices fitted to approximate proportions.

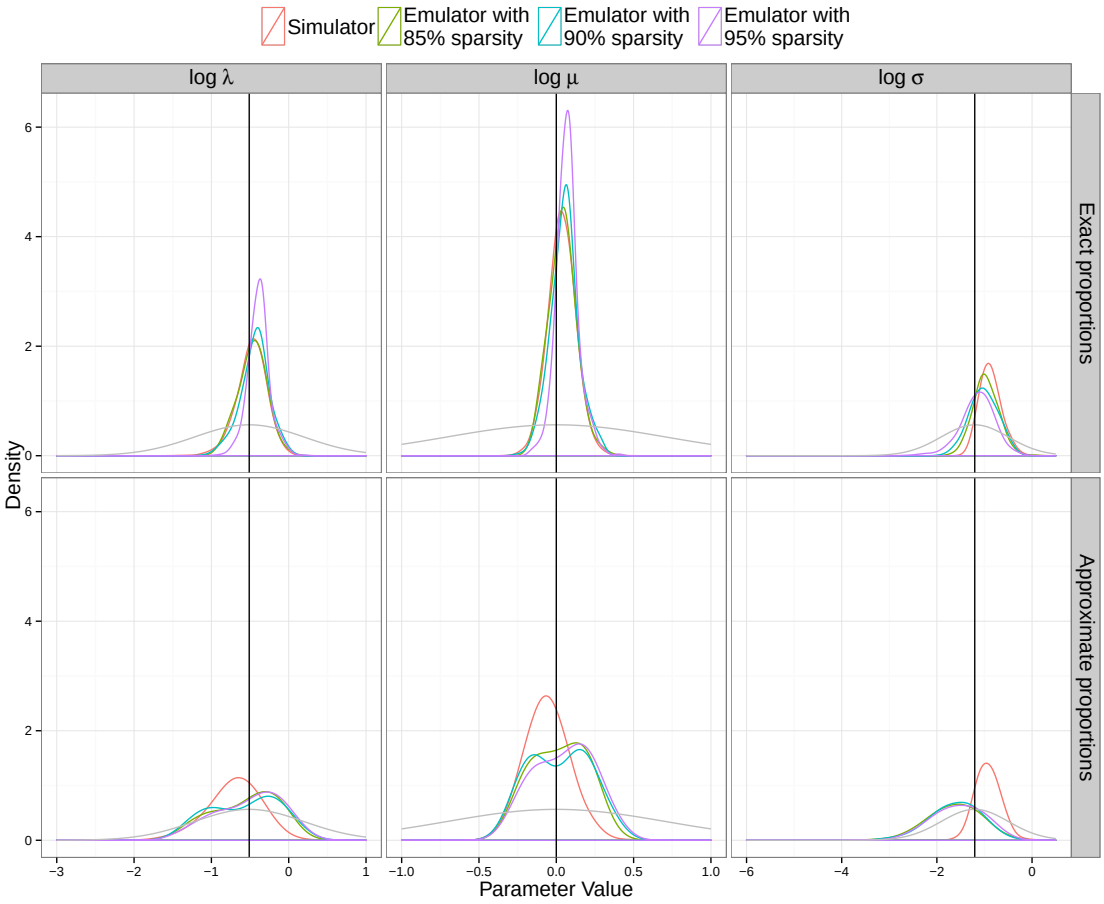

Figure 6.13: Marginal posterior distributions for model parameters using emulators with sparse covariance matrices. Top panel: using emulators fitted to exact proportions of extinction. Bottom: using emulators fitted to approximate proportions of extinction.

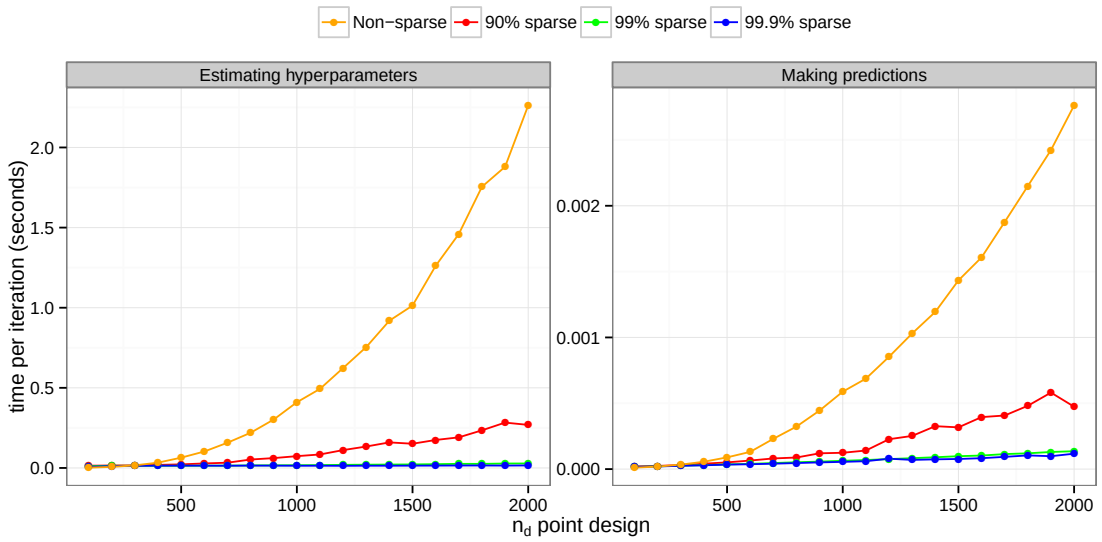

Figure 6.14: Comparison of emulators with sparse and non–sparse covariance matrices

6.4 Comparing emulators with sparse and non-sparse covariance matrices

It has been seen that using emulators with sparse covariance matrices, are not quite as accurate as standard emulators for parameter inference when the number of training points, n_d is 50. Since the birth–death model has only two parameters, a relatively small n_d could be used and computational expense was not an issue. However, the sparse covariance approach was introduced to overcome the problem that the computational cost scales with $\mathcal{O}(n_d^3)$ and a much larger n_d will be required for larger models.

Figure 6.14 compares computational efficiency of the sparse and non-sparse emulators. It can clearly be seen that as n_d gets large, the original emulators become increasingly slow when compared to the sparse emulator. Note, the fitting of emulators with sparse covariance matrices is done in `R` using the `SparseEm` package of Kaufman et al. (2011).

6.5 Conclusion

In this chapter, emulators were built for the birth–death model and used in place of the simulator in an inference scheme for the model parameters.

In the first scenario, the training data were built on the exact proportions of extinction since an analytic expression exists for this quantity. For emulators built under this framework, it was found that a jitter parameter must be added to the covariance function to ensure numerical stability. As discussed, the addition of this jitter term has other desirable properties. It is confirmed by the emulator diagnostics that the emulators are fitting acceptably.

When attempting to infer model parameters using emulators built on exact proportions, it was found that using training data with $n_d = 50$ gave inferences which were indistinguishable from the true posterior. These posterior distributions appear to be insensitive to the mean function. The posterior distributions also seem insensitive to ignoring the posterior uncertainty in the hyperparameters.

It was found that using emulators with sparse covariance matrices fitted to exact proportions provided inferences which were very similar to that of the inferences obtained using the simulator. The much improved computational scaling of the sparse emulators was clearly demonstrated.

In the second scenario, emulators were built for approximate proportions of extinction simulated using $n = 1000$ runs of the simulator. In this case, a nugget term was added to the covariance function to deal with the extra uncertainty induced. It was found that even in this scenario, emulators performed as a very good surrogate for the simulator, although not quite as good as when exact proportions were used.

The emulators with sparse covariance matrices did not perform as well when used on approximate proportions. The fitted emulator mean is very good when compared to the actual mean, although the emulator variance is inaccurate. Although initially, it may seem that this approach will not be appropriate for approximate proportions, the ease of use and speed of implementation make this approach ideal as a first attempt at a larger problem.

Chapter 7

Mitochondrial DNA model

This chapter considers parameter inference, using Gaussian process emulators, for a medium sized model of mitochondrial DNA (mtDNA). This model was first introduced in Chapter 2 and will be studied in further detail in this chapter.

Inference has been attempted previously for this model. Henderson et al. (2009) consider inference, when the experimental data are noisy measurements on one of the underlying chemical species in the model. In a further paper, Henderson et al. (2010) use an additional dataset containing counts of surviving neurons to make inferences about model parameters.

In this chapter, interest lies in performing inference using only the data on proportions of surviving neurons. For the purposes of this analysis, the binomial error on the proportions is ignored. Firstly, inference will be attempted on simulated data, where the true parameter values are known. This will give an idea of how well the method performs. Secondly, inference will be attempted on the experimental data. It is hoped that using only the data on proportions will be informative about model parameters, and that these inferences are consistent with Henderson et al. (2009, 2010).

7.1 A stochastic model

Sufferers of Parkinson's disease exhibit symptoms which are related to the area of the brain which controls motor function. It is also known that neuron loss in the *substantia nigra* region of the human brain, located just above the spinal cord, is associated with these symptoms.

The model aims to describe the process of neuron loss in the *substantia nigra* region. Neuron loss is thought to be related to mtDNA deletions. Deletion mutations in the mtDNA of the *substantia nigra* region are known to occur with ageing in healthy adults, although higher levels have been observed in patients with Parkinson's disease. It is of scientific interest to gain a better understanding of how mtDNA deletions affect neuron loss.

The focus of Henderson et al. (2009) was to use experimental data to perform inference for the parameters of the mtDNA model. The motivation behind the analysis in this chapter is to see what information is lost on model parameters by using only proportions. The benefit of having posterior estimates of parameters is that they can be used in a computer model which allows virtual experiments to be carried out. These experiments could be used to determine interventions which can stop or reverse neuron decline for suffers of Parkinson's disease.

Model details

A brief introduction was given to the mtDNA model in Chapter 2. The model, for a single neuron, involves two chemical species $\boldsymbol{X} = (X_1, X_2)'$, where X_1 represents mtDNA with no deletions (healthy mtDNA) and X_2 represents mtDNA with deletions (unhealthy mtDNA). At any time, the number of copies of X_1 and X_2 is given by x_1 and x_2.

There are five possible reactions in the system which are given in Table 7.1. The first reaction describes the process of mutation, whereby healthy mtDNA become unhealthy mtDNA. The second and fourth reactions describe the process of synthesis

Label	Reaction	Hazard	Description
R_1	$X_1 \rightarrow X_2$	$h_1(\boldsymbol{x}, \theta_1) = \theta_1 x_1$	Mutation
R_2	$X_1 \rightarrow 2X_1$	$h_2(\boldsymbol{x}, \theta_3) = \frac{1000\theta_3 x_1}{x_1+x_2}$	Synthesis
R_3	$X_1 \rightarrow \emptyset$	$h_3(\boldsymbol{x}, \theta_3) = \theta_3 x_1$	Degradation
R_4	$X_2 \rightarrow 2X_2$	$h_4(\boldsymbol{x}, \theta_3) = \frac{1000\theta_3 x_2}{x_1+x_2}$	Mutant Synthesis
R_5	$X_2 \rightarrow \emptyset$	$h_5(\boldsymbol{x}, \theta_3) = \theta_3 x_2$	Mutant Degradation

Table 7.1: Reactions and their hazards for the mtDNA model.

(reproduction); this can happen in both the healthy and unhealthy mtDNA. The third and fifth reactions describe degradation (death), which can also happen in both the healthy and unhealthy mtDNA.

The original model given in Table 2.3 of Chapter 2 contains parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)^T$. Henderson et al. (2009) make the assumption that healthy and unhealthy mtDNA synthesise and degrade at the same rate, reducing the parameter space to $\boldsymbol{\theta} = (\theta_1, \theta_3)^T$, and giving the reaction hazards in Table 7.1. The rate laws for reactions R_2 and R_4 are constructed to ensure that the total number of mtDNA in the cell $x_1 + x_2$ remains approximately constant (at 1000) throughout the lifetime of the cell.

The model contains a mechanism for cell death (neuron death). This is modelled by a deterministic process in which the proportion of unhealthy mtDNA reaches some critical threshold and the cell dies. The proportion of unhealthy mtDNA is $p = x_2/(x_1 + x_2)$ and the model imposes cell death when $p \geq \tau$, for some threshold parameter $\tau \in (0, 1]$.

Deletion accumulation data

The experimental data used for inference in Henderson et al. (2009) were taken from post–mortems of 15 individuals who ranged from $19 - 91$ in age, non of whom were suffering from Parkinson's disease. For each individual a slice of brain tissue from the *substantia nigra* region was used to obtain a sample of 25 neurons. The data consist of RT-PCR (real–time polymerase chain reaction) measurements $y_i = -\log_2(1 - p_i)$, where p_i is the proportions of mtDNA deletions in the sample of 25 neurons for individual i.

Henderson et al. (2009) used these data only to infer parameters in the model. They also used an emulator to approximate the stochastic kinetic model since simulation from the model is slow.

Neuron survival data

Henderson et al. (2009) also introduced another dataset, which they called the neuron survival data. This data is taken from Fearnley and Lees (1991) and is given in Table 7.2. The dataset considers 36 individuals without Parkinson's disease. For each individual, their age at death, along with a count of surviving neurons take from a sample of brain tissue (post–mortem) is recorded. These data are a corrected version of Fearnley and Lees (1991), where the correct number of neurons observed for the person aged 22 was 792 instead of 692. The data are shown graphically in the left hand panel of Figure 7.1. It can be seen that as individuals increase in age, the number of surviving neurons appears to decrease, as would be expected.

In Henderson et al. (2009), this dataset was only used for external validation for the parameter inferences obtained using the deletion accumulation data. They constructed 95% predictive probability intervals for neuron survival by sampling parameters from their posterior distributions and simulating from the model. These simulations were then compared to the validation data and they showed that the predictive intervals were consistent with the experimental data. Later, in Henderson et al. (2010), both the deletion accumulation data and the neuron survival data are used in the same inference scheme to reduce uncertainty on model parameters.

Age	Observed number of surviving neurons	Age	Observed number of surviving neurons	Age	Observed number of surviving neurons
21	692	60	642	78	503
22	792	61	587	79	520
29	695	61	585	80	556
31	657	65	403	81	543
44	633	65	518	81	448
47	583	69	702	84	648
53	613	70	406	85	616
54	692	70	615	86	471
55	653	71	558	87	540
56	658	75	493	89	578
58	588	75	504	91	426
58	544	77	390	91	394

Table 7.2: Neuron survival data

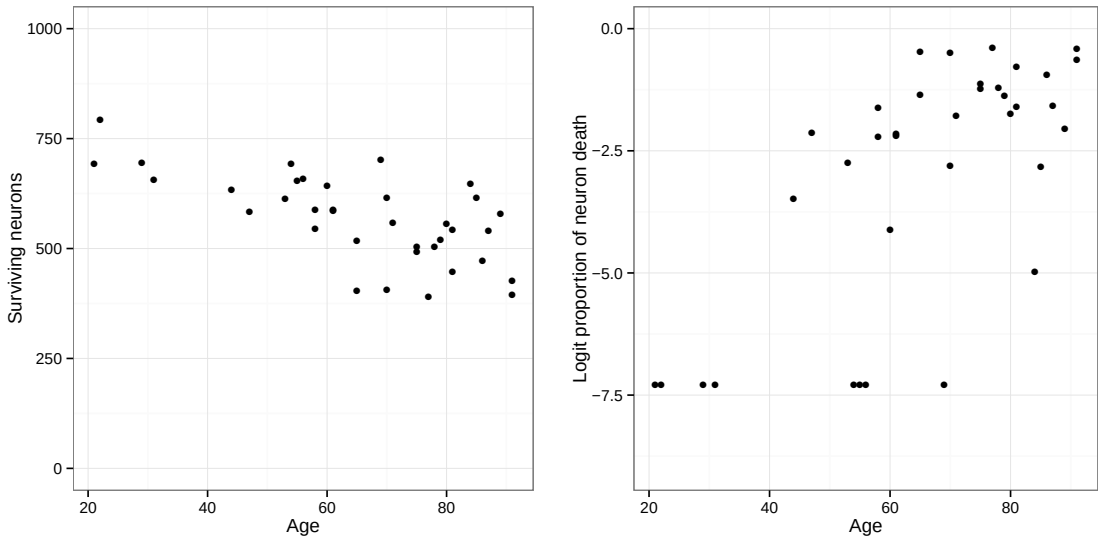

Figure 7.1: Neuron survival data

7.1.1 Modelling neuron survival

The model for neuron survival data, developed by Henderson et al. (2010), has the following hierarchical structure for 36 individuals:

$$\begin{aligned} z_i|y_i,\phi &\sim \text{Bin}(y_i,\phi) \quad \text{for } i=1,\ldots,36 \\ y_i|N,\boldsymbol{\theta},x_i &\sim \text{Bin}(N,\text{expit}[\rho(\boldsymbol{\theta},x_i)]) \quad \text{for } i=1,\ldots,36 \\ \phi &\sim \text{Beta}(a,b) \\ N &\sim \text{Poisson}(N^*). \end{aligned}$$

Here the z_i are the observed data (counts of surviving neurons). The model assumes that surviving neurons are observed with binomial error, where the true number of surviving neurons for person i is y_i. Each surviving neuron is observed independently, with probability ϕ, where ϕ has a beta prior distribution. Henderson et al. (2010) choose the parameters of this beta distribution to be $a = 90$ and $b = 10$, so that ϕ has a prior distribution fairly concentrated around its mean of 0.9. The thinking here is that roughly 90% of surviving neurons are observed.

The model assumes that people are born with N neurons, meaning at any time, the true number of surviving neurons y_i can be at most N. The logit transformed probability of each neuron surviving, $\rho(\boldsymbol{\theta}, x_i)$ depends on the individual's age x_i, along with the model parameters $\boldsymbol{\theta}$. The model includes prior beliefs for N, the number of neurons present at birth, and describes this with a Poisson distribution, with mean $N^* = 795$.

The model used in previous chapters for proportions of death (rather than proportions of survival) takes the form

$$y_i \equiv \text{logit}\, x_i = \text{logit}\, p_i(\boldsymbol{\theta}) + \sigma\epsilon_i,$$

where $\epsilon_i \sim N(0,1)$ and the y_i are the observed data, which are noisy logit proportions of death. This model has a much simpler error structure than that of Henderson et al.

(2010). However, they both capture roughly the same mean structure. This can be seen as follows. The simpler model has $p(\boldsymbol{\theta})$ as a probability of death and a corresponding probability of survival $q(\boldsymbol{\theta}) = 1 - p(\boldsymbol{\theta})$. Conditioning on parameters, the mean number of surviving neurons for a particular individual age x is

$$E[\boldsymbol{z}] = E_{\boldsymbol{y}}\left[E(\boldsymbol{z}|\boldsymbol{y})\right] = E_{\boldsymbol{y}}[\phi\boldsymbol{y}] = \phi E_{\boldsymbol{y}}[\boldsymbol{y}] = \phi N \text{expit}[\rho(\boldsymbol{\theta}, x)].$$

Therefore

$$\text{logit}\left\{\frac{E(\boldsymbol{z})}{\phi N}\right\} = \rho(\boldsymbol{\theta}, x).$$

In the simpler model

$$E(\text{logit}\, x) = \text{logit}\, q(\boldsymbol{\theta}, x)$$

and so, roughly speaking, the mean structure of the two models are the same if it is assumed that $x = z/(\phi N)$. Unfortunately, there is a slight problem with using this scaling of z, namely that it is possible to have observed values with $\frac{z}{\phi N} > 1$. One work around for this problem, and which is used in the analysis of the data in Table 7.2, is to specify N and ϕ and take $x = \min\left\{\frac{z}{\phi N}, 1\right\}$. This restriction affects 8 data points in Table 7.2. The right hand panel of Figure 7.1 shows the experimental data on proportions of neuron death plotted on the logit scale. Note that the 8 data points affected by the scaling described above have be set equal to $\log\{0.5/(725 + 0.5)\}$. The alternative to using the empirical logit would be simply to remove these data points from the analysis.

7.2 Emulation for neuron survival

Henderson et al. (2010) suggest that, on average, individuals are born with 725 neurons (this is their posterior mean for N^*). Simulating the lifetime of 725 neurons takes around two minutes. While this is not excessively slow, if an inference scheme was used where forward simulations from the model were required at each iteration, it would take around a fortnight to obtain 10K iterations (typically, many more iterations would be

required). Consequently, an emulator will be constructed for neuron survival which will be a fast approximation to the stochastic kinetic model.

A separate emulator for each of the 29 unique ages in the experimental data will be constructed. The input is three dimensional, since the model parameters for which inference is required are θ_1, θ_3 and τ.

7.2.1 Obtaining training data

A suitable range for the inputs must be chosen over which to construct the emulators. The prior distributions used by Henderson et al. (2010) were used to guide the region over which training data was constructed. These prior distributions were elicited from expert's beliefs which were themselves based on previous literature. The parameters are taken to be independent *a priori* and

$$\log \theta_1 \sim N(-10.4, 1.8^2)$$
$$\log \theta_3 \sim N(-3.8, 0.37^2)$$
$$\tau \sim U(0.5, 1).$$

Parameter sets used to construct the training data were chosen to cover the middle 95% of the prior distribution for $\log \theta_1$ and $\log \theta_3$ and, since τ was uniformly distributed, 100% prior coverage was used.

The number of training points, n_d, was chosen to be 200 and this seemed to work well in practice – the resulting diagnostics suggested that the emulators fitted well. For each training point, the lifetimes of $n = 725$ neurons were simulated (up to 91 years). For each age, the counts of surviving neurons, (out of 725) were recorded and converted to death counts. The empirical logit of these death counts form the output values on which emulators were built.

High–throughput computing

Simulating from the mtDNA model is slow, hence the motivation for building emulators. However, the construction of emulators still requires training data to be obtained. The simulator must be ran for n_d training inputs. As n_d gets large, obtaining this training data becomes increasingly time consuming. Since each training run of the simulator is independent, running the simulations in parallel affords large speed–ups.

The HTCondor system is a form of high–throughput computing (HTC) which was used to harness the power of computing resources within Newcastle University. The HTCondor system is a workload management system that can be used for computer intensive tasks. When obtaining the training data, n_d jobs are submitted to the HTCondor system and placed in a queue. Each job is then ran on the first suitable machine which becomes available.

In practice, the HTCondor system has worked very well for this task. It required very little in the way of time spent setting it up and allowed around 100 jobs to be run at the same time.

7.2.2 Mean and covariance function

Multiple linear regression with normal errors was performed to advise on the choice of mean function. Terms were added sequentially, starting with the linear terms, then adding squared terms and interactions. Any non–significant terms were discarded and the final mean function was taken as

$$m(\theta_1, \theta_3, \tau) = \widehat{\beta}_0 + \widehat{\beta}_1 \log\, \theta_1 + \widehat{\beta}_2\, \log\, \theta_3 + \widehat{\beta}_3 \tau + \widehat{\beta}_4 (\log\, \theta_1)^2 + \widehat{\beta}_5 \log\, \theta_1 \log\, \theta_3$$

where the $\widehat{\beta}_i$ are the least squares estimates from the regression. The squared exponential covariance function takes the form

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \tau_i, \tau_j | a, \boldsymbol{r}) = a \exp\left\{ - \left(\frac{(\log\, \theta_{i1} - \log\, \theta_{j1})^2}{r_1^2} + \frac{(\log\, \theta_{i3} - \log\, \theta_{j3})^2}{r_2^2} + \frac{(\tau_i - \tau_j)^2}{r_3^2} \right) \right\}$$

for this model.

7.2.3 Estimating hyperparameters and diagnostics

Hyperparameter estimation for the emulators was performed using Algorithm 14 of Chapter 5. The prior distributions used were assumed to be independent *a priori*, where

$$a \sim \text{Log-Normal}(0, 100)$$
$$r_1 \sim \text{Log-Normal}(0, 100)$$
$$r_2 \sim \text{Log-Normal}(0, 100)$$
$$r_3 \sim \text{Log-Normal}(0, 100).$$

These prior distributions represent vague prior knowledge about the hyperparameters and are shown in red in Figure 7.2. The marginal posterior distributions are shown in black. It can be seen that all marginal posterior distributions are different from the prior distributions and that the training data have been very informative.

Diagnostics for the emulators can be seen in Figure 7.3. A validation dataset was constructed over the same range as the original Latin hypercube with $n_{d^\dagger} = 100$ points. It can be seen that all diagnostics appear to behave reasonably, suggesting that the emulators are fitting well.

7.3 Analysis of simulated data

A synthetic dataset, containing proportions of neuron survival, was simulated using model parameters which were fixed at the posterior means obtained by Henderson et al. (2010). This dataset contains 36 individuals whose ages corresponded to the ages in the experimental dataset of Table 7.2.

In previous chapters, proportions of death have been considered rather than proportions of survival. Consequently, the synthetic data were converted to proportions of neuron death. In the simple model, the measurement error structure is normal on the

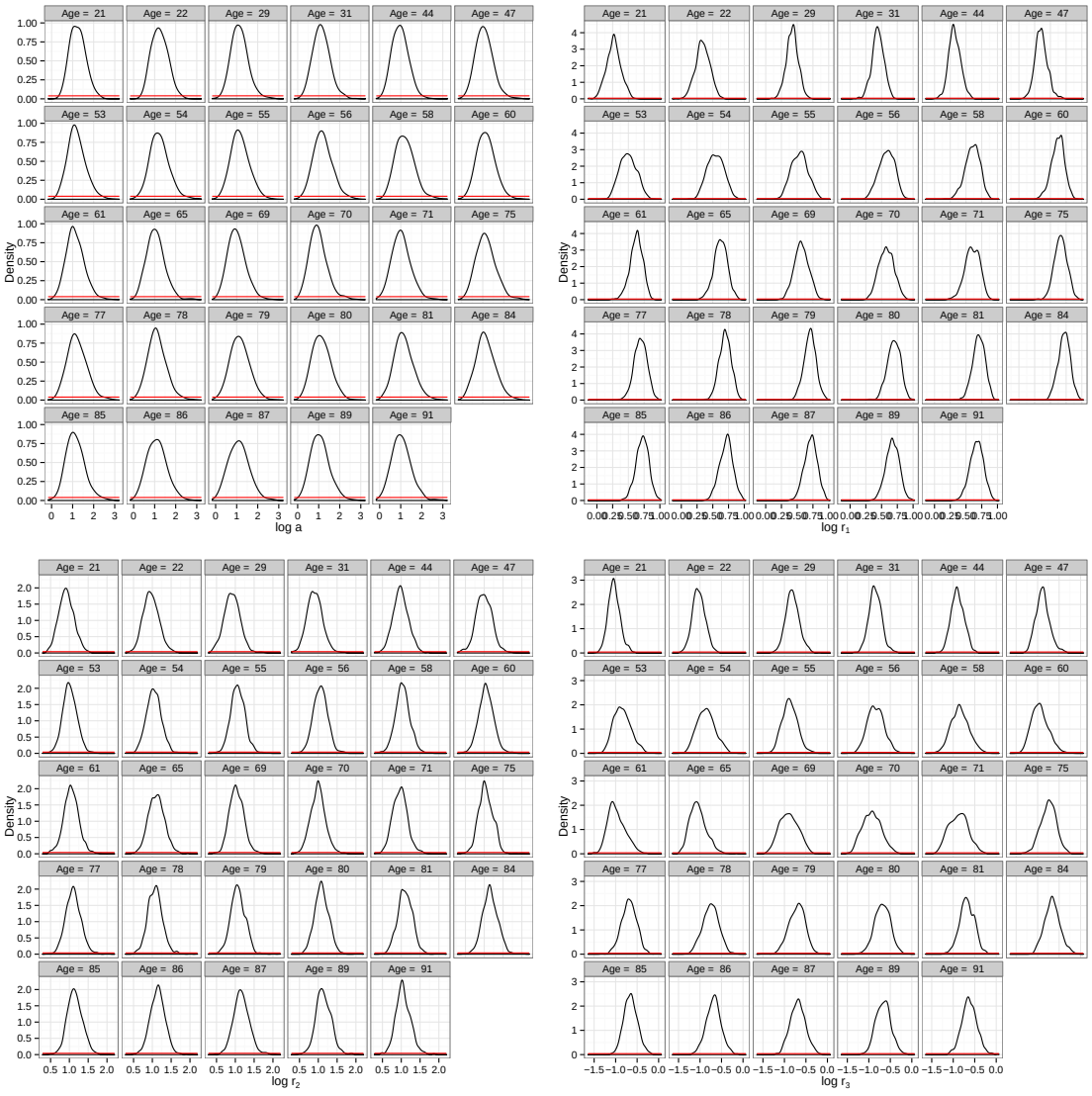

Figure 7.2: Marginal posterior distributions for hyperparameters for the mtDNA model. Prior distributions given in red.

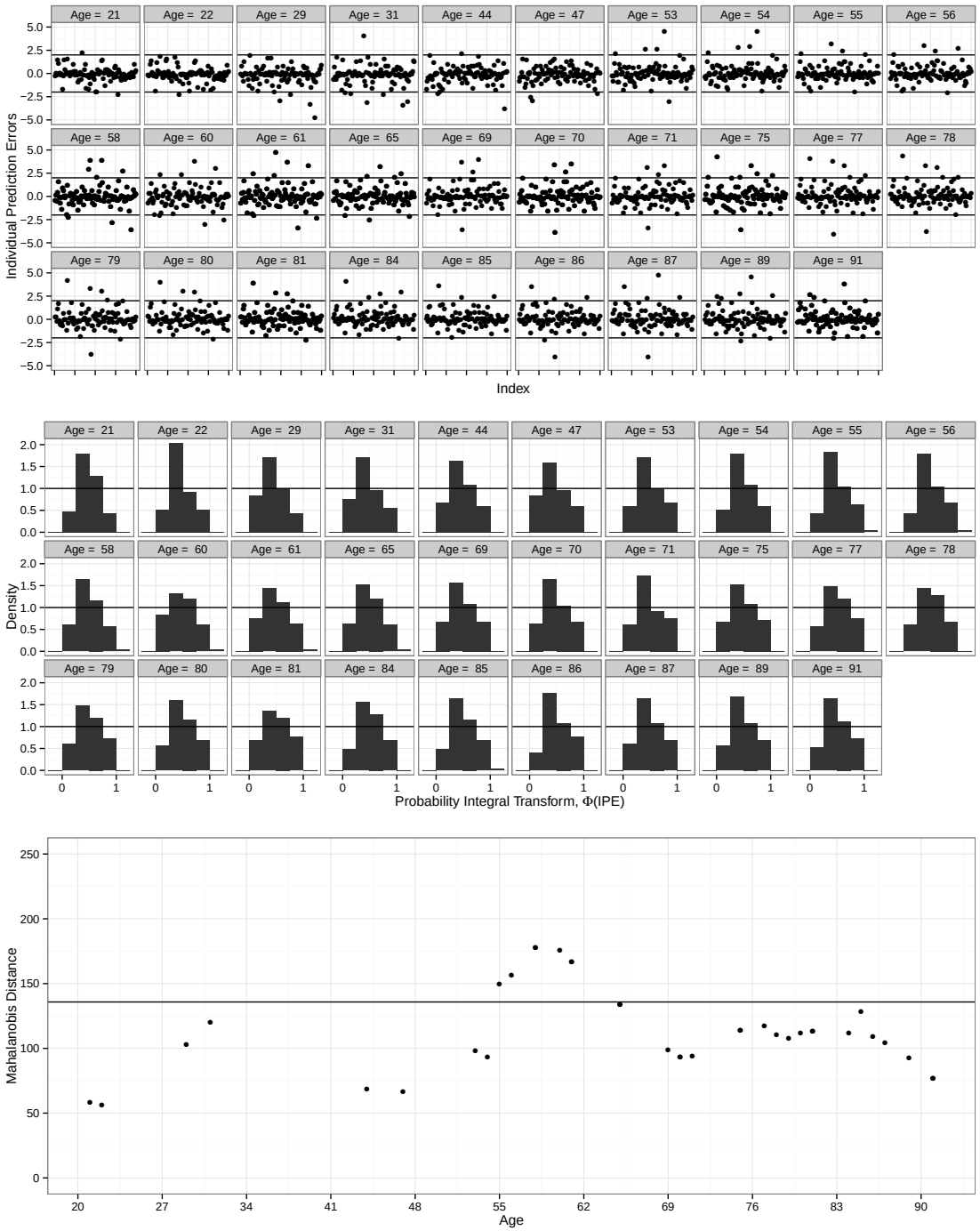

Figure 7.3: Diagnostics for emulators for the mtDNA model.

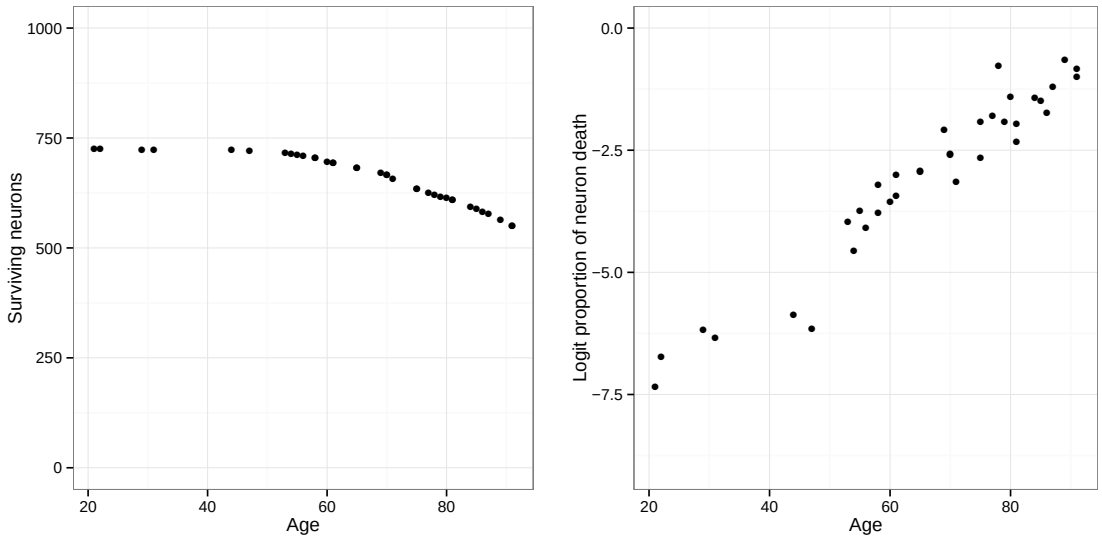

Figure 7.4: Simulated data. Left: number of surviving neurons. Right: noisy logit proportion of death neurons.

logit scale. The logit of the proportion of dead neurons was taken and normal noise with mean zero and standard deviation $\sigma = 0.3$ was added. The simulated number of surviving neurons can be seen in the left hand plot of Figure 7.4. The right hand plot shows the noisy logit proportions of neuron death.

Inference was attempted on the noisy logit proportions of neuron death using the scheme introduced in Section 6.2 of Chapter 6. The prior distribution for model parameters was taken from Henderson et al. (2010); this has independent components, with

$$\log \theta_1 \sim N(-10.4, 1.8^2)$$
$$\log \theta_3 \sim N(-3.8, 0.37^2)$$
$$\tau \sim U(0.5, 1)$$

and for the measurement error

$$\log \sigma \sim N(\log\ 0.3, 5).$$

The MCMC scheme was run for 100K iterations, then the sample was thinned, keeping every 10th iteration. The trace plots and autocorrelation plots shown in the middle and right hand panels of Figure 7.5 suggest convergence has been achieved.

Histograms of the marginal posterior distributions for the model parameters are given in the left hand panels of Figure 7.5. The prior distributions are given in red and the true parameter values are displayed in green. It can be seen that all model parameter seem to be well recovered.

7.4 Analysis of experimental data

The results of inference using the experimental data on neuron survival given in Table 7.2 can be seen in Figure 7.6. The vertical green lines represent the posterior means obtained in Henderson et al. (2010), where both the neuron survival data and the deletion accumulation data were considered.

The details of the inference scheme, including the prior distributions (shown in red) are the same as for the simulated data. The trace plots and autocorrelation plots in the middle and right hand panels of Figure 7.6 suggest convergence.

It can be seen that, in general, the analysis returns similar posterior means to that of Henderson et al. (2010). This is very encouraging since it would be expected that there is a lot of information in the deletion accumulation data.

A more formal comparison is given in Table 7.3. The first column shows the posterior means and 95% equal-tailed posterior probability intervals obtained from Henderson et al. (2009), when only the deletion accumulation data was used ($\mathcal{D}_1$). The second column shows the equivalent information taken from Henderson et al. (2010), when the deletion accumulation and the neuron survival data were used ($\mathcal{D}_2$). The third column shows the results of the analysis from this chapter, when only the neuron survival data was used ($\mathcal{D}_3$).

It can be seen that for all parameters, the analysis using $\mathcal{D}_3$ gives wider intervals than the analysis using $\mathcal{D}_2$. This is perhaps not surprising, however, it is encouraging to note

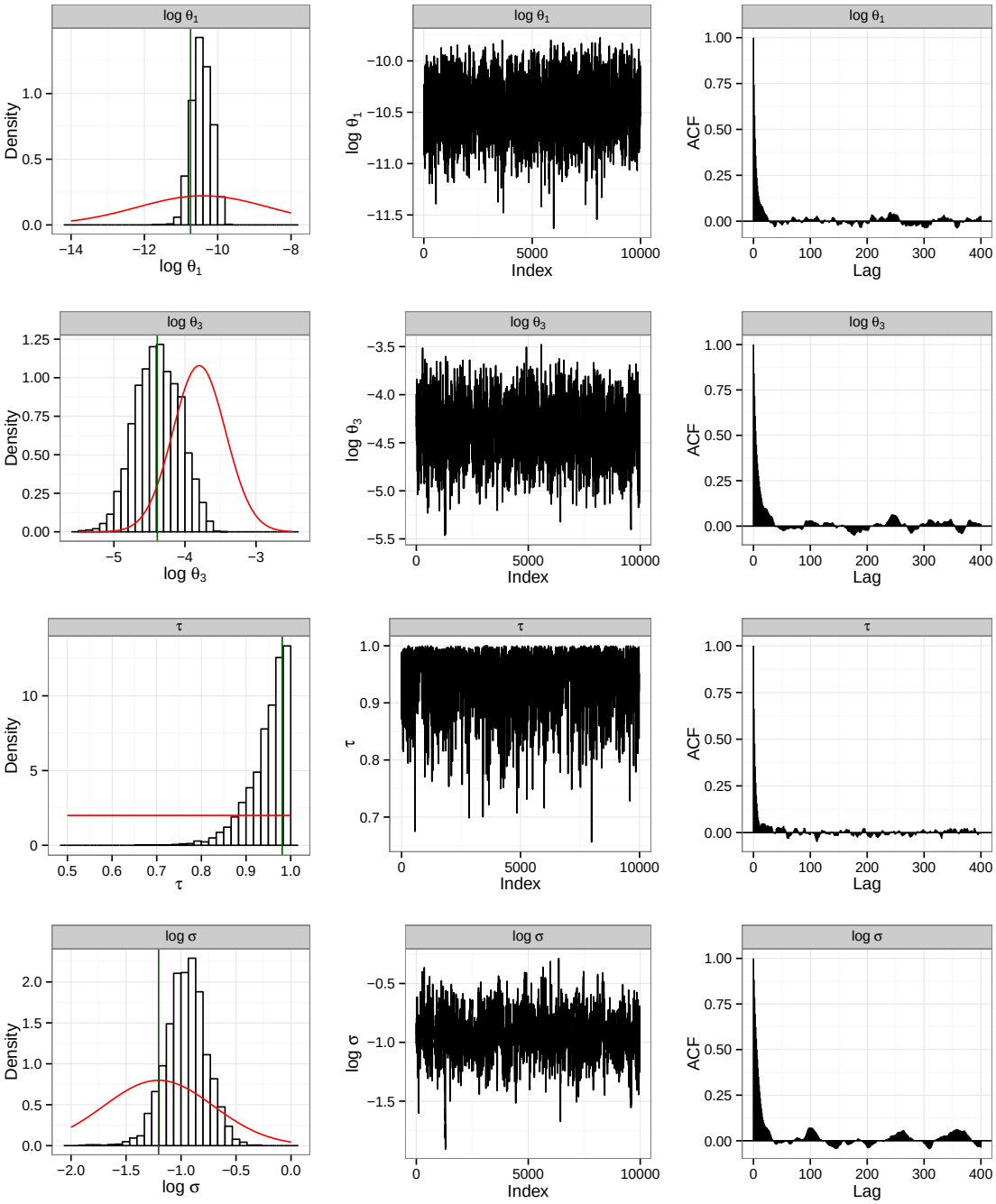

Figure 7.5: Results of parameter inference on model parameters for the mtDNA model using simulated data.

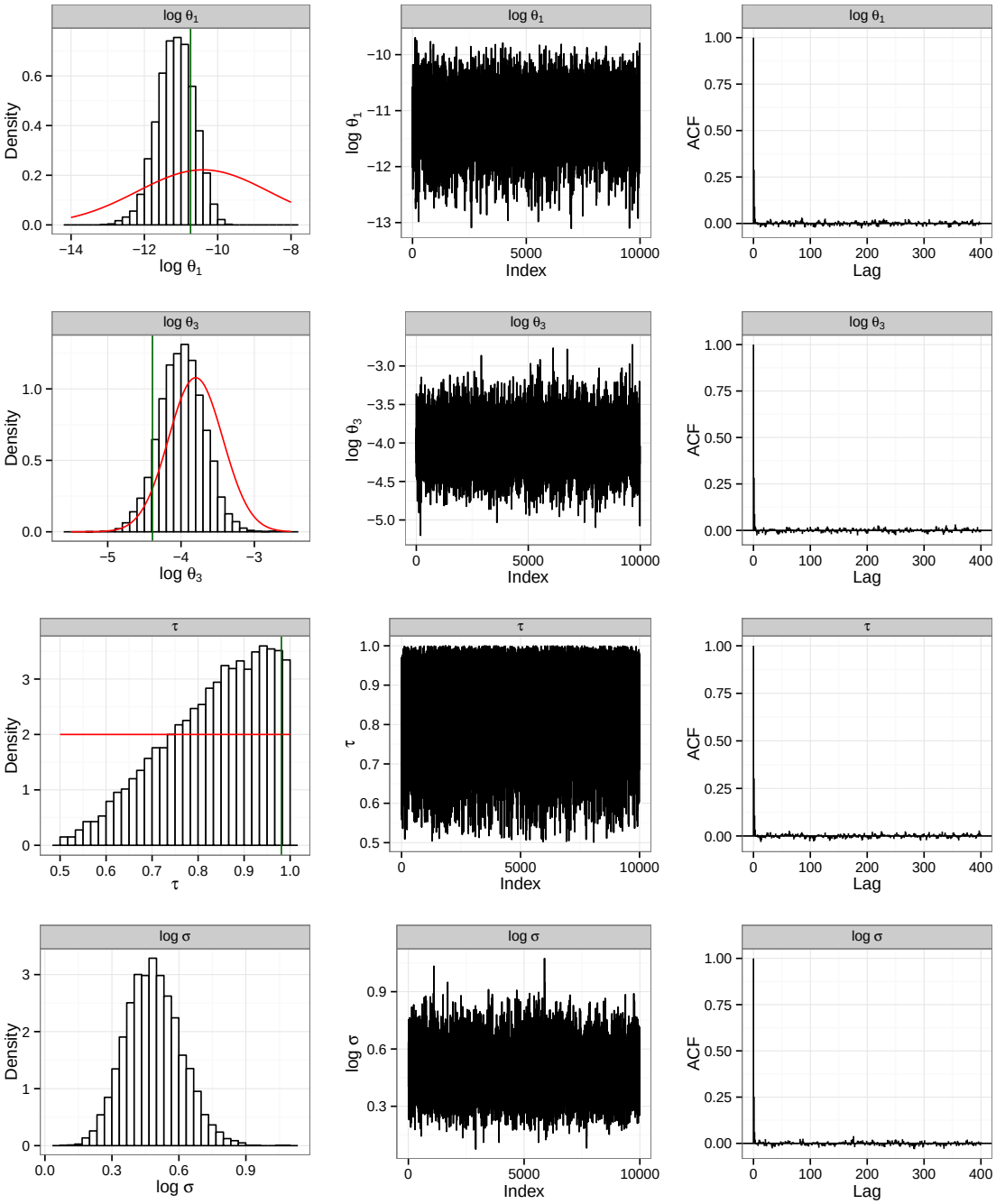

Figure 7.6: Results of parameter inference on model parameters for the mtDNA model using experimental data.

	$\mathcal{D}_1$ Deletion accumulation data only	$\mathcal{D}_2$ Deletion accumulation data and neuron survival data	$\mathcal{D}_3$ Neuron survival data only
$\log \theta_1$	-10.18 $(-10.57, -9.79)$	-10.74 $(-10.94, -10.55)$	-11.15 $(-12.18, -10.22)$
$\log \theta_3$	-4.51 $(-5.09, -3.92)$	-4.39 $(-4.95, -3.91)$	-4.00 $(-4.60, -3.40)$
τ	0.962 $(0.868, 0.999)$	0.981 $(0.930, 0.999)$	0.83 $(0.59, 0.993)$

Table 7.3: Posterior means and 95% equal-tailed posterior probability intervals (in parenthesis), for inference using different datasets.

that the posterior means from the $\mathcal{D}_2$ analysis all lie well within the 95% equal-tailed posterior probability intervals for the $\mathcal{D}_3$ analysis.

This is also the case when comparing the $\mathcal{D}_1$ analysis with the $\mathcal{D}_3$ analysis, apart from the $\log \theta_1$ parameter where the posterior mean obtained in the $\mathcal{D}_1$ analysis of -10.18 lies slightly outside of the posterior interval of $(-12.18, -10.22)$ obtained from the $\mathcal{D}_3$ analysis.

Figure 7.7 shows a plausible range of logit proportions determined via 99% predictive intervals for each age. The intervals were constructed by taking 300 samples from the posterior distribution then simulating a dataset from the model using each of these parameter sets. The central 99% of these simulations is plotted (light grey). The crosses on the plot represent the experimental data. It can be seen that in general, this interval gives good coverage of the experimental data suggesting that the parameter choices found are consistent with the data.

7.5 Conclusions

Previous attempts to calibrate the mtDNA model have been successful, however, no attempts have been made using only the neuron survival data presented in Table 7.2. In this chapter, inference has successfully been performed for the model parameters of the mtDNA model using the neuron survival data, along with a comparable synthetic dataset.

Firstly, analysis of a synthetic dataset where the true parameters were known was

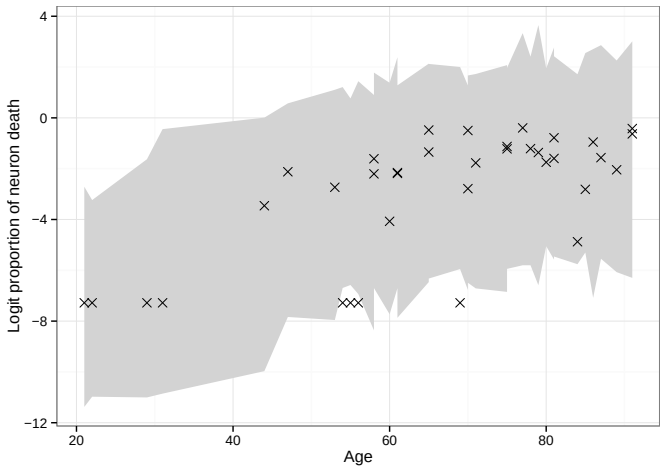

Figure 7.7: Plausible range of logit proportions determined via 99% predictive intervals for each age.

attempted. This resulted in marginal posterior distributions which were centred around their true parameter values and significant information was learned from the prior distribution.

Next, analysis of the neuron survival data was attempted. The resulting marginal posterior distributions were consistent with that of Henderson et al. (2010). This was encouraging since, unlike Henderson et al. (2010), no observations on the underlying chemical species were included. The predictive posterior distribution demonstrates that the posterior distributions for model parameters are consistent with the experimental data. However, two notable improvement which could be made to the model are to include a mechanism which recognises the binomial sampling error in the data and to include prior information on the initial number of neurons at birth, N. This could be implemented by adding an extra input to the design space of the emulator.

Chapter 8

PolyQ model

8.1 Introduction

The PolyQ model, developed by Tang et al. (2010), is a large stochastic kinetic model capturing biological processes at the molecular level within human cells as they undergo ageing. Modelling and understanding these processes are important for the treatment of neurodegenerative diseases such as Huntington's disease.

The model was introduced in Chapter 2, where a brief background was given along with some example simulations. The model contains 25 chemical species which are listed in Table 2.4. These chemical species typically represent numbers of molecules of a particular protein which can react through a series of reactions. There are 69 reactions each of which will typically involve an increase or decrease in the number of molecules of a certain protein. A list of reactions was given in Table 2.5 along with the associated rates. s The model is represented graphically in Figure 8.1. Oval shapes (nodes) represent chemical species and an arrow between two nodes represents a reaction which can take place involving the two chemical species. It is not intended that this network diagram should be studied in detail, rather that it gives the reader an impression of the size and complexity of the model.

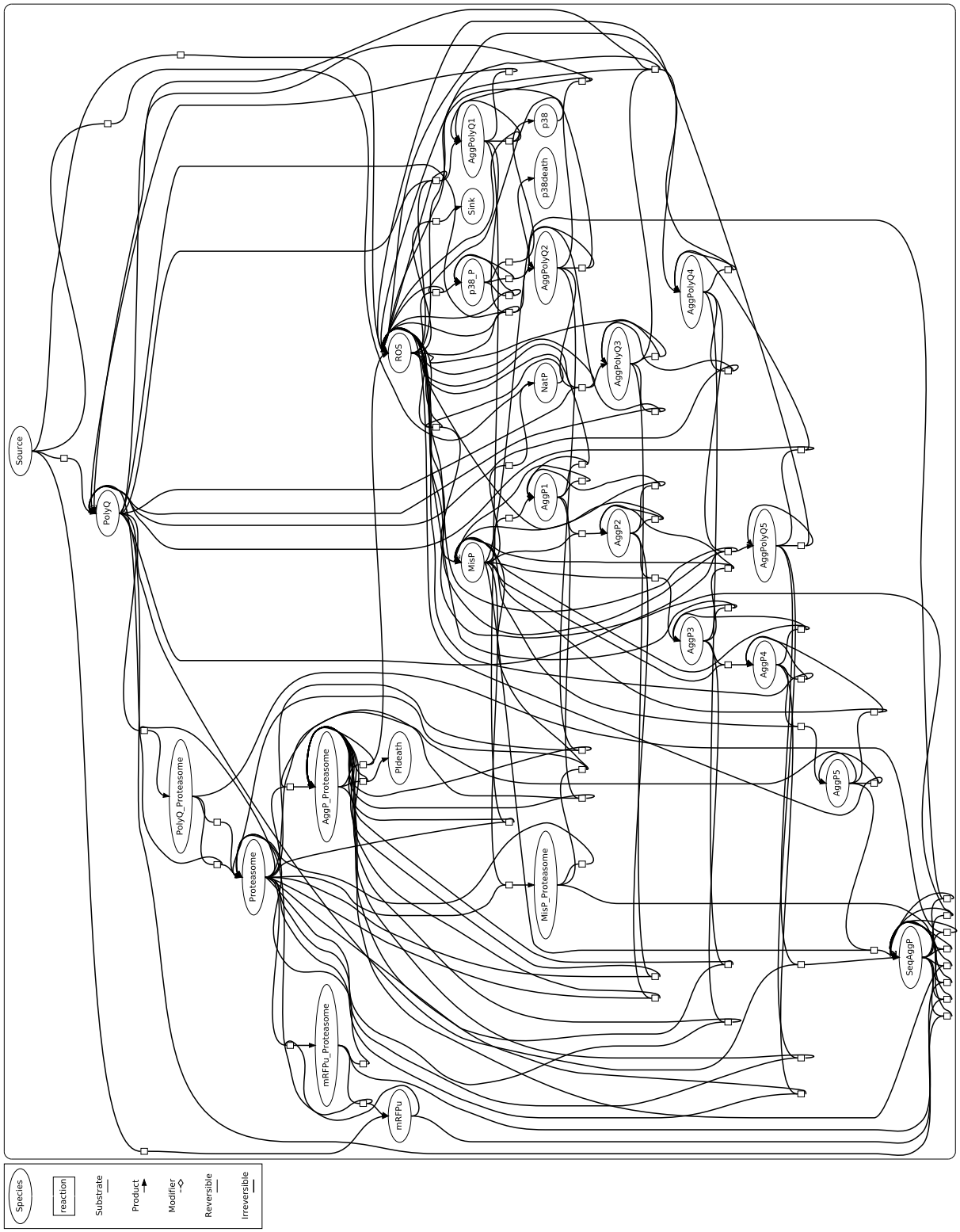

Figure 8.1: Network diagram of reactions in the PolyQ model

Current research is based on fixing the parameters in the model at the expert's best guesses and then manually adjusting them to match experimental data. This chapter considers parameter inference for parameters in the PolyQ model using two different synthetic datasets on proportions of cell death.

The number of parameters in the PolyQ model is large and attempting to make inference on all 40 parameters would be a very ambitious task. Since in previous chapters, inference on a maximum of three model parameters (plus measurement error) has been considered, it was decided that this should be extended to four parameters for the PolyQ model. In conjunction with the mathematical modellers, the four parameters were chosen which are most important for cell death and inference was considered for these parameters; all other parameters values were fixed at the expert's best guess. Since the model is slow to simulate from, Gaussian process emulators are built for proportions of cell death. These emulators are then embedded into inference schemes.

8.2 The stochastic model

A full list of the 25 chemical species in the model is given in Table 2.4 of Chapter 2. The initial number of each chemical species assumed by Tang et al. (2010) is also given. Table 8.1 is a condensed version of Table 2.4 containing only the chemical species which will be of direct interest for the analysis in this chapter. Table 2.5 of Chapter 2 contains a full list of all reactions which can take place in the model. A condensed version of this table is given in Table 8.2, which includes only the reactions whose rates involve the four parameters included in this analysis.

Some further biological details are given below on the parts of the model which are directly related to the parameters of interest in this chapter. However, it must be noted that there are many more processes happening in the model which are not discussed below.

Name	Description	Initial amount
`Proteasome`	26S Proteasome	1000
`AggPolyQi`	PolyQ aggregate of size i ($i = 1, \ldots, 5$)	0
`AggPProteasome`	Aggregated protein bound to proteasome	0
`ROS`	Reactive oxygen species	10
`p38P`	Active P38MAPK	0
`p38`	Inactive p38MAPK	100
`AggPi`	Small aggregate of size i ($i = 1, \ldots, 5$)	0
`PIdeath`	Dummy species to record cell death due to proteasome inhibition	0
`p38death`	Dummy species to record cell death due to p38MAPK activation	0

Table 8.1: List of species

ID	Reaction name	Reaction	Rate law
17a	Proteasome inhibition	$\texttt{AggPolyQ}_1 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggPPolyQ}_1\texttt{]}\,\texttt{[Proteasome]}$
17b		$\texttt{AggPolyQ}_2 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggPPolyQ}_2\texttt{]}\,\texttt{[Proteasome]}$
17c		$\texttt{AggPolyQ}_3 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggPPolyQ}_3\texttt{]}\,\texttt{[Proteasome]}$
17d		$\texttt{AggPolyQ}_4 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggPPolyQ}_4\texttt{]}\,\texttt{[Proteasome]}$
17e		$\texttt{AggPolyQ}_5 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggPPolyQ}_5\texttt{]}\,\texttt{[Proteasome]}$
37a		$\texttt{AggP}_1 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggP}_1\texttt{]}\,\texttt{[Proteasome]}$
37b		$\texttt{AggP}_2 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggP}_2\texttt{]}\,\texttt{[Proteasome]}$
37c		$\texttt{AggP}_3 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggP}_3\texttt{]}\,\texttt{[Proteasome]}$
37d		$\texttt{AggP}_4 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggP}_4\texttt{]}\,\texttt{[Proteasome]}$
37e		$\texttt{AggP}_5 + \texttt{Proteasome} \rightarrow \texttt{AggPProteasome}$	$k_{inhprot}\,\texttt{[AggP}_5\texttt{]}\,\texttt{[Proteasome]}$
20a	ROS generation	$\texttt{AggPolyQ}_1 \rightarrow \texttt{AggPolyQ}_1 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggPolyQ}_1\texttt{]}$
20b		$\texttt{AggPolyQ}_2 \rightarrow \texttt{AggPolyQ}_2 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggPolyQ}_2\texttt{]}$
20c		$\texttt{AggPolyQ}_3 \rightarrow \texttt{AggPolyQ}_3 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggPolyQ}_3\texttt{]}$
20d		$\texttt{AggPolyQ}_4 \rightarrow \texttt{AggPolyQ}_4 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggPolyQ}_4\texttt{]}$
20e		$\texttt{AggPolyQ}_5 \rightarrow \texttt{AggPolyQ}_5 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggPolyQ}_5\texttt{]}$
21		$\texttt{AggPProteasome} \rightarrow \texttt{AggPProteasome} + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AppPProteasome]}$
38a		$\texttt{AggP}_1 \rightarrow \texttt{AggP}_1 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggP}_1\texttt{]}$
38b		$\texttt{AggP}_2 \rightarrow \texttt{AggP}_2 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggP}_2\texttt{]}$
38c		$\texttt{AggP}_3 \rightarrow \texttt{AggP}_3 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggP}_3\texttt{]}$
38d		$\texttt{AggP}_4 \rightarrow \texttt{AggP}_4 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggP}_4\texttt{]}$
38e		$\texttt{AggP}_5 \rightarrow \texttt{AggP}_5 + \texttt{ROS}$	$k_{genROSAggP}\,\texttt{[AggP}_5\texttt{]}$
39		$\texttt{p38P} \rightarrow \texttt{p38P} + \texttt{ROS}$	$k_{genROSp38} * k_{p38act}\,\texttt{[p38P]}$
22	p38MAPK activation	$\texttt{ROS} + \texttt{p38} \rightarrow \texttt{ROS} + \texttt{p38P}$	$k_{actp38}\,\texttt{[ROS]}\,\texttt{[p38]}$
41	p38 cell death	$\texttt{p38P} \rightarrow \texttt{p38P} + \texttt{p38death}$	$k_{p38death} k_{p38act}\,\texttt{[p38P]}$
42	PI cell death	$\texttt{AggPProteasome} \rightarrow \texttt{AggPProteasome} + \texttt{pIdeath}$	$k_{pIdeath}\,\texttt{[AggPProteasome]}$

Table 8.2: List of reactions and hazards for the PolyQ model

AggP and AggPolyQ

The $\texttt{AggP}_i$ represent small aggregates of proteins of size i where $i = 1, \ldots, 5$. For example, $\texttt{AggP}_1$ represents aggregates of proteins of size one. The formation of these aggregates is due to presence of misfolded proteins in the cell. In the model, the production of these aggregates is encoded using a set of reactions whereby an aggregate of $\texttt{AggP}_i$ binds to a misfolded protein producing an aggregate of $\texttt{AggP}_{i+1}$.

Similarly, the $\texttt{AggPolyQ}_i$ represent aggregates of PolyQ proteins of size i ($i = 1, \ldots, 5$) where $i = 1, \ldots, 5$. PolyQ aggregates are formed when PolyQ proteins clump together.

ROS

Reactive oxygen species (ROS) are molecules of oxygen with an unpaired electron, making them very reactive. The generation of ROS as a by product of reactions involving oxygen is normal in all cells, however, increased levels of ROS can be dangerous, putting the cell under oxidative stress. Increased levels of ROS in neurons is a typical feature of age-related neurodegenerative disease such as Huntington's disease.

It can be seen from Table 8.2, that `ROS` is produced as a result of protein aggregation (reactions 20a–20e, 21, 38a–38e) and due to the production of `p38P` (reaction 39).

Proteasome

The proteasome are protein complexes found in the nucleus of all eukaryotic cells. The purpose of the proteasome is to break down and remove damaged and misfolded proteins in the cell. If the proteasome are not able to do their job, the cell is put under stress. Inhibition of the proteasome can ultimately cause a chain of reactions which leads to cell death.

Reactions 17a–e and 37a–e of Table 8.2 describe the process of protein aggregates (either $\texttt{AggP}_i$ or $\texttt{AggPolyQ}_i$) inhibiting the proteasome (represented by `AggPProteasome`). Reaction 42 is the mechanism for cell death due to proteasome inhibition. This reaction is more likely to happen when there are high levels of `AggPProteasome`.

p38 and p38P

p38MAPK (mitogen-activated protein kinases) are part of a class of proteins within the cell which are responsible for regulating processes such as gene expression and mitosis. In the PolyQ model, `p38` represents inactive p38MAPK and `p38P` and represent active p38MAPK.

When PolyQ proteins are being produced in a cell, p38MAPK is known to play a role in cell death. Tang et al. (2010) conclude that the inhibition of p38MAPK, reduces cell death due to this pathway.

In the PolyQ model, cells are assumed to have an initial number of 100 molecules of inactive p38MAPK (`p38`). It can be seen in reaction 22 of Table 8.2 that high levels of ROS increase the likelihood of p38MAPK activation. Cell death is more likely to occur in the model for high levels of p38MAPK activation. The dummy species `p38death` is included in the model as a binary variable. While the cell is alive, `p38death` $= 0$ and if cell death occurs via this pathway, `p38death` $= 1$.

Model parameters

A full list of parameters in the PolyQ model is given in Table 8.3 along with their current values, which are the modellers best guesses. There are four model parameters that will be considered in the analysis in this chapter which are highlighted in colour in the table. These are

- $k_{inhprot}$: controls the rate of proteasome inhibition which leads to cell death. Features in 10 of reactions in Table 2.5 and is highlighted in red.
- $k_{genROSAggP}$: controls the amount of ROS produced after stress which leads to the activation of p38 and cell death. Features in 11 reactions in Table 2.5 and is highlighted in blue.
- $k_{genROSp38}$: Similar function to $k_{genROSAggP}$. This parameter features in reaction 39 and is highlighted in green.

Parameter	Value	Parameter	Value
$k_{aggPolyQ}$	5.0×10^{-8}	k_{genROS}	1.7×10^{-3}
$k_{disaggPolyQ1}$	5.0×10^{-7}	k_{remROS}	2.0×10^{-4}
$k_{disaggPolyQ2}$	4.0×10^{-7}	$k_{genROSAggP}$	5.0×10^{-6}
$k_{disaggPolyQ3}$	3.0×10^{-7}	$k_{genROSSeqAggP}$	1.0×10^{-7}
$k_{disaggPolyQ4}$	2.0×10^{-7}	k_{actp38}	5.0×10^{-6}
$k_{disaggPolyQ5}$	1.0×10^{-7}	$k_{inactp38}$	2.0×10^{-3}
$k_{seqPolyQ}$	8.0×10^{-7}	$k_{seqMisP}$	1.0×10^{-9}
$k_{inhprot}$	5.0×10^{-9}	$k_{seqAggPProt}$	5.0×10^{-7}
$k_{aggMisP}$	1.0×10^{-11}	$k_{seqPolyQProt}$	5.0×10^{-7}
$k_{agg2MisP}$	1.0×10^{-10}	$k_{seqMisPProt}$	5.0×10^{-7}
$k_{disaggMisP1}$	5.0×10^{-7}	$k_{seqmRFPuProt}$	5.0×10^{-7}
$k_{disaggMisP2}$	4.0×10^{-7}	$k_{seqmRFPu}$	1.0×10^{-10}
$k_{disaggMisP3}$	3.0×10^{-7}	$k_{synNatP}$	2.4
$k_{disaggMisP4}$	2.0×10^{-7}	$k_{misfold}$	2.0×10^{-6}
$k_{disaggMisP5}$	1.0×10^{-7}	k_{refold}	8.0×10^{-5}
$k_{synmRFPu}$	1.38×10^{-1}	$k_{binMisPProt}$	5.0×10^{-8}
$k_{binmRFPu}$	5.0×10^{-7}	$k_{relMisPProt}$	1.0×10^{-8}
$k_{relmRFPu}$	1.0×10^{-8}	$k_{degMisP}$	1.0×10^{-2}
$k_{degmRFPu}$	5.0×10^{-3}	$k_{genROSp38}$	7.0×10^{-4}
$k_{synPolyQ}$	7.0×10^{-3}	k_{p38act}	1
$k_{binPolyQ}$	5.0×10^{-8}	$k_{PIdeath}$	2.5×10^{-8}
$k_{relPolyQ}$	1.0×10^{-9}	$k_{proteff}$	1.0
$k_{degPolyQ}$	2.5×10^{-3}	k_{alive}	1.0

Table 8.3: PolyQ parameters and the values used for simulating data.

- $k_{kactp38}$: controls the rate at which p38 is activated leading to cell death. Features in reaction 22 and is highlighted in purple.

For notation simplicity the following reparametrisation will be used

$$\theta_1 = \log k_{inhprot}, \quad \theta_2 = \log k_{genROSAggP}, \quad \theta_3 = \log k_{genROSp38}, \quad \theta_4 = \log k_{actp38}.$$

8.3 Experimental data

The experimentalists start off with a large number of cells and use a technique called *propidium iodide exclusion* to identify the viability (death status) of the cells over time. The cells are treated and then stained with propidium iodide, a fluorescent dye. Propidium iodide has the property of only binding to non-viable cells. The fluorescent

Experimental conditions	24hrs	36hrs	48hrs
GFP	0.179	0.182	0.285
H25	0.164	0.197	0.300
H103	0.244	0.210	0.387

Table 8.4: Proportions of cell death observed under different experimental conditions.

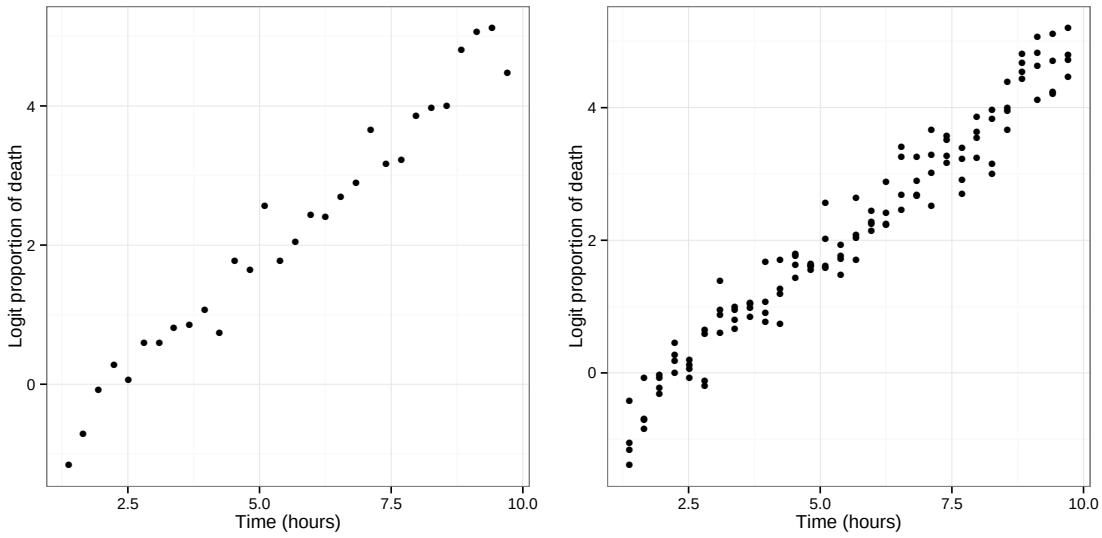

Figure 8.2: PolyQ simulated data

dye can be viewed under a microscope and an estimates of the proportion of cell death can be observed over time.

An example of some experimental data is given in Table 8.4 where each row corresponds to experiments carried out under slightly different experimental conditions. The analysis in this chapter only considers the first experimental condition.

Simulated data from the PolyQ model will be used in place of experimental data to test out the performance of methods of inference. Two synthetic datasets will be simulated using the parameter values given in Table 8.3. The first datasets has data points at 30 unique time points, the second dataset has data at the same 30 unique time points but has four repeats at each time point (giving a total of 120 points). Plots of the data are given in Figure 8.2.

8.4 Emulating proportions of death from the PolyQ model

Emulators will be built for the PolyQ model using the framework developed in previous chapters. For both synthetic datasets, there are observations at 30 unique time points. Consequently, a emulator will be built for the logit proportions of cell death at each of the 30 time points.

8.4.1 Mean function and covariance function

As with the mtDNA model, multiple linear regression was performed to advise the choice of mean function. Terms were added sequentially, starting with the linear terms, then adding squared terms and interactions. Any non–significant terms were discarded and the final mean function was

$$m(\theta_1,\theta_2,\theta_3,\theta_4) = \widehat{\beta}_0 + \widehat{\beta}_1\theta_1 + \widehat{\beta}_2\theta_2 + \widehat{\beta}_3\theta_3 + \widehat{\beta}_4\theta_4 + \widehat{\beta}_5\theta_1^2 + \widehat{\beta}_6\theta_2^2 + \widehat{\beta}_7\theta_3^2 + \widehat{\beta}_8\theta_4^2 + \\ \widehat{\beta}_9\theta_1\theta_2 + \widehat{\beta}_{10}\theta_1\theta_3 + \widehat{\beta}_{11}\theta_1\theta_2\theta_3 + \widehat{\beta}_{12}\theta_1\theta_2\theta_4 + \widehat{\beta}_{13}\theta_1\theta_2\theta_3\theta_4,$$

where the $\widehat{\beta}_i$ are the least square estimates from the regression. The squared exponential covariance function will take the form

$$K(\boldsymbol{\theta_i},\boldsymbol{\theta_j}|a,\boldsymbol{r}) = a\exp\left\{-\left(\frac{(\theta_{i1}-\theta_{j1})^2}{r_1^2} + \frac{(\theta_{i2}-\theta_{j2})^2}{r_2^2} + \frac{(\theta_{i3}-\theta_{j3})^2}{r_3^2} + \frac{(\theta_{i4}-\theta_{j4})^2}{r_4^2}\right)\right\}.$$

The five hyperparameters (a, r_1, r_2, r_3, r_4) will be estimated from the training data.

8.4.2 Training data

Model parameters were thought to be independent *a priori* and chosen to represent fairly vague prior knowledge, with

$$\theta_1 \sim N(\log 5 \times 10^{-9}, 5)$$
$$\theta_2 \sim N(\log 5 \times 10^{-6}, 5)$$
$$\theta_3 \sim N(\log 7 \times 10^{-4}, 5)$$
$$\theta_4 \sim N(\log 5 \times 10^{-6}, 5).$$

The emulators were fitted using a Latin hypercube design with $n_d = 1000$ points distributed over the middle 95% of these prior distributions using the *maximin* design introduced in Chapter 5. The HTCondor system which was described in Chapter 7 was used to obtain the training runs. In total, this took less than 24 hours.

8.4.3 Estimating hyperparameters and diagnostics

The prior distributions used for hyperparameters were assumed to be independent *a priori*, where

$$a \sim \text{Log-Normal}(0, 100) \qquad \text{and} \qquad r_i \sim \text{Log-Normal}(0, 100) \quad \text{for } i = 1, 2, 3, 4,$$

and represent vague prior knowledge about hyperparameters. The marginal posterior distributions for hyperparameters can be seen in Figure 8.3 where the five panels represent the five hyperparameters estimated using the scheme in Algorithm 13 of Chapter 5. The prior distributions can be seen in red, clearly the training data has been very informative. There appears to be a gradual shift in the posterior distributions for hyperparameters across time. In particular, the posterior mean for the log a parameter increases from approximately -0.75 at time 0 to approximately 0.25 at time 20. This suggests that the variance on the proportions of cell death increases with time – as would be expected.

Due to the relatively large size of Latin hypercube required for this model, it takes around one day to run each scheme for 1000 iterations, of which there are 30 emulators. Since fitting each emulator is independent of all other, the HTCondor system can be used to parallelise this task.

Diagnostics for the emulators can be seen in Figure 8.4. These diagnostics were constructed using a validation dataset generated using another *maximin* Latin hypercube covering the same range as the original containing $n_{d^\dagger} = 100$ points. The diagnostics seem to be reasonable, suggesting that in general the emulators are fitting acceptably. Some of the Mahalanobis distances are slightly higher than would be expected, especially for emulators at times 20 – 30, however, the IPE and PIT histograms look satisfactory for these emulators.

8.5 Analysis of simulated data

The results of analysis of simulated data can be seen for the dataset with 30 points in Figure 8.5 and for the dataset with 120 points in Figure 8.6. For each analysis, the MCMC scheme was run for 10K iterations and samples were thinned such that every 10th iteration was kept. The marginal posterior distributions are given in black and the prior distributions in red. The trace plots and autocorrelation plots are given in the middle and right hand panels of each figure. These suggest convergence has been achieved for both schemes.

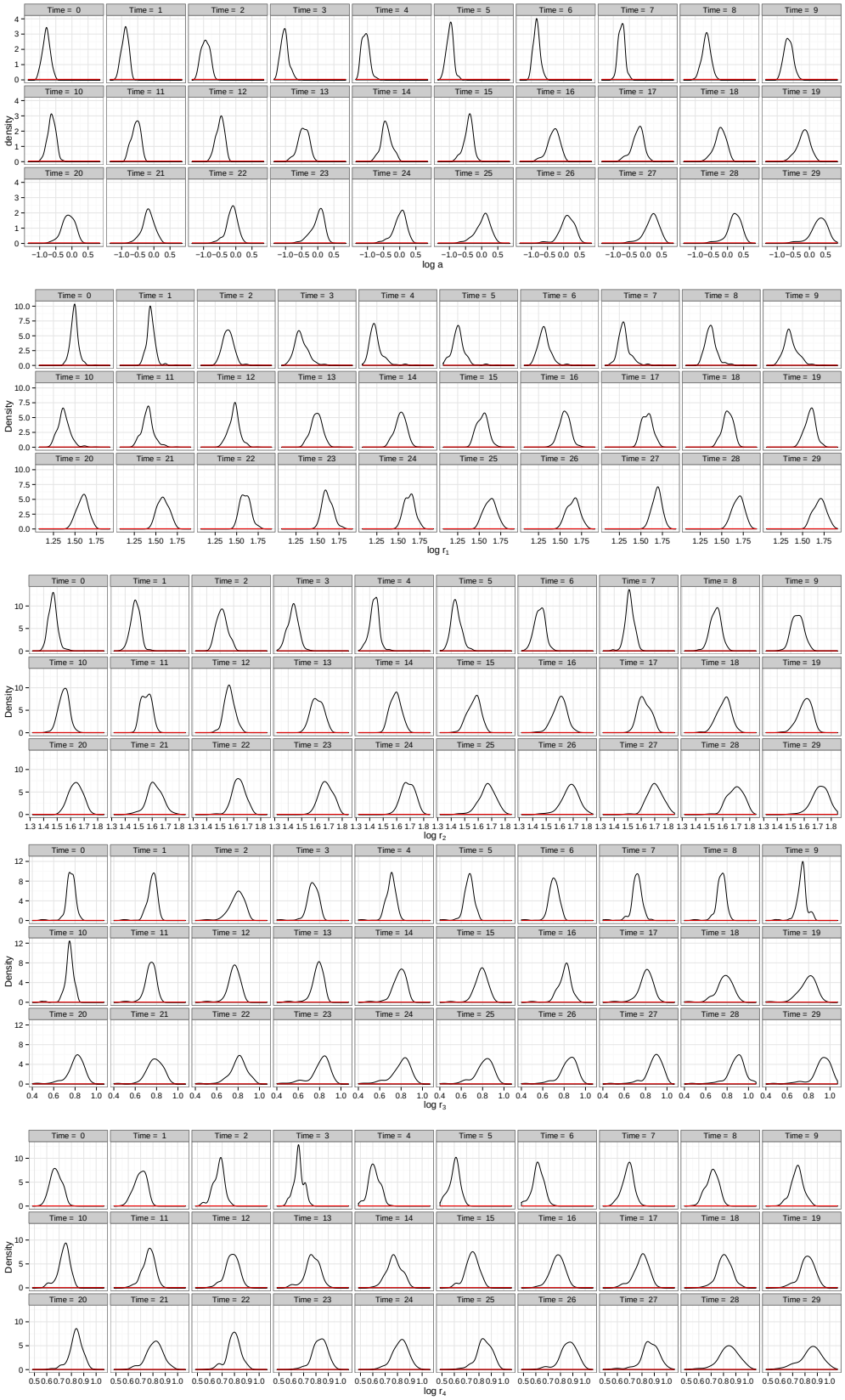

Figure 8.3: Marginal posterior distributions for hyperparameters.

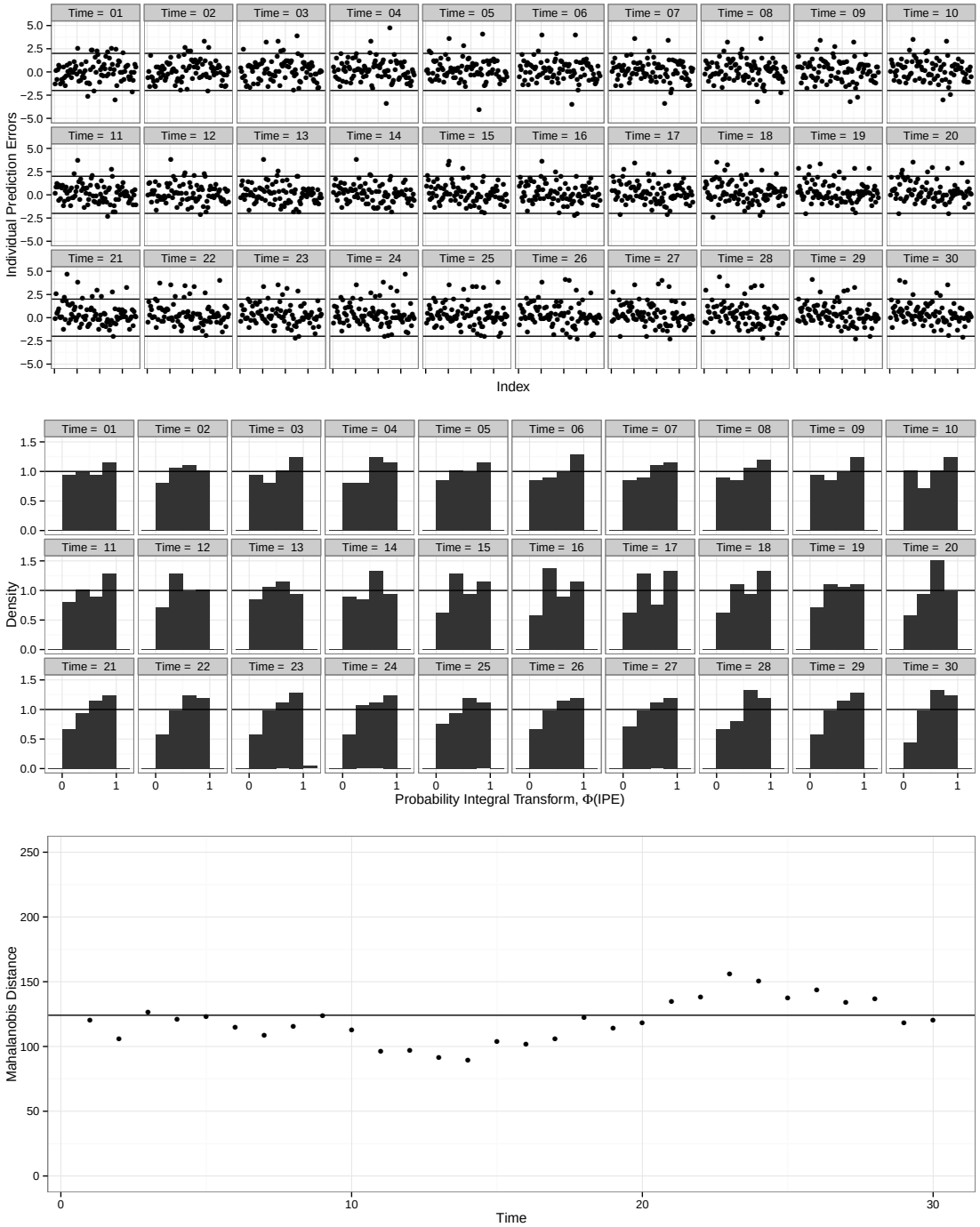

Figure 8.4: Diagnostics for emulators for the PolyQ model.

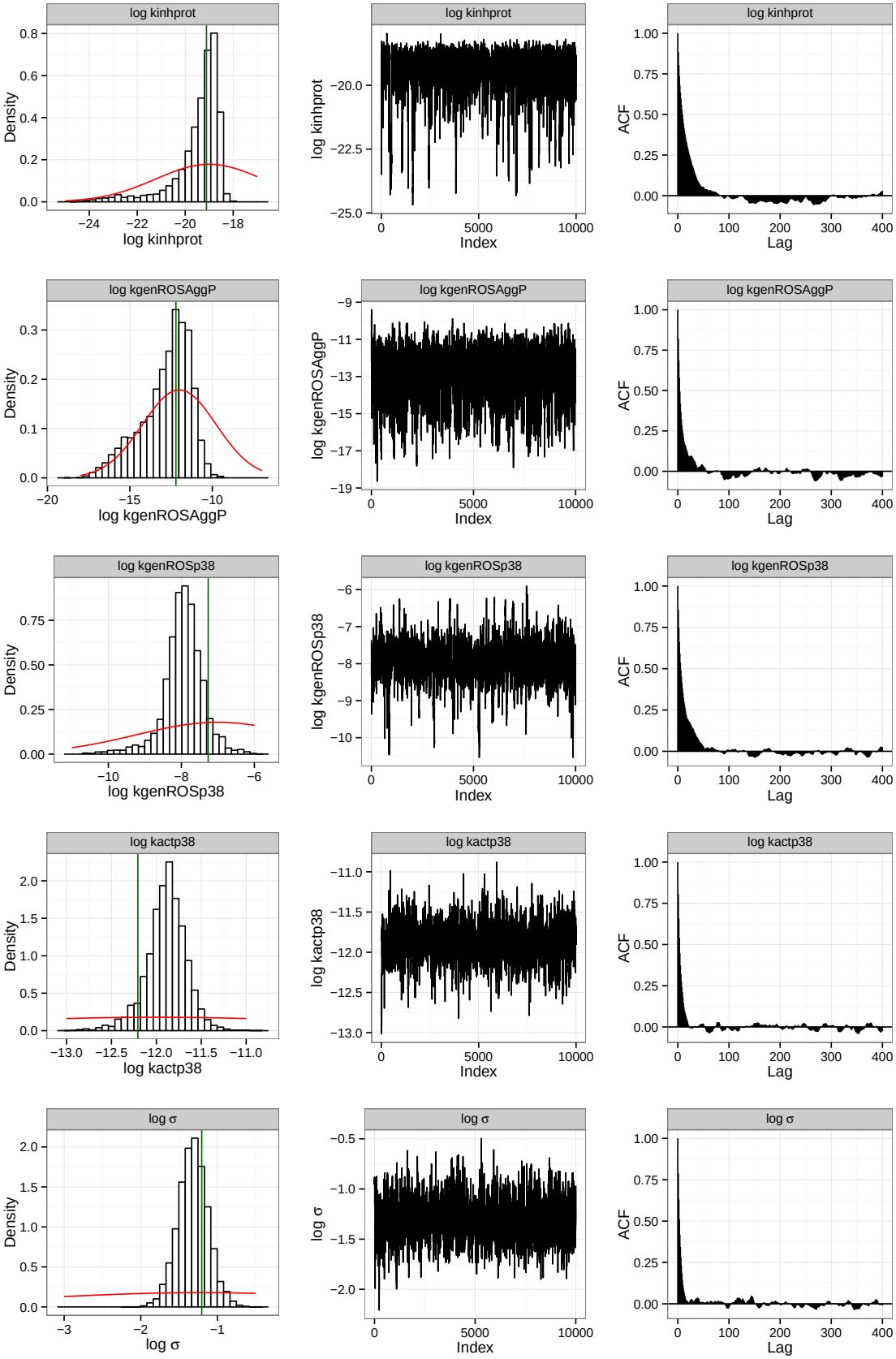

Figure 8.5: Results of parameter inference on model parameters for the PolyQ model using simulated data with 30 points. Marginal posterior distributions given in black. Prior distributions given in red.

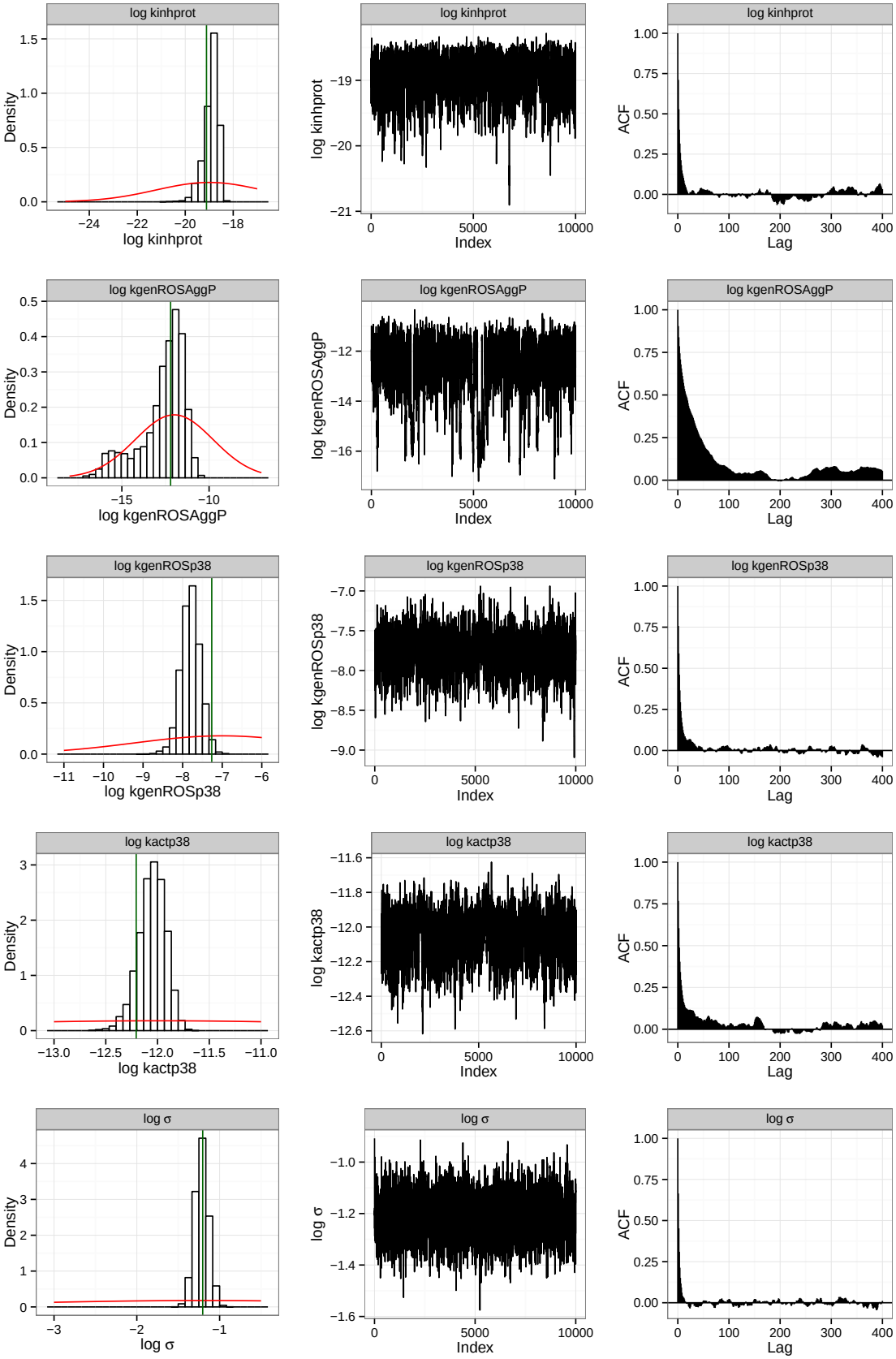

Figure 8.6: Results of parameter inference on model parameters for the PolyQ model using simulated data with 120 points. Marginal posterior distributions given in black. Prior distributions given in red.

8.6 Further considerations

The focus of this thesis was to look at parameter inference for the PolyQ model. This has in part been achieved for simulated data, albeit for only four kinetic rate constants and one scaling measurement error parameter.

The number of parameters in the PolyQ model is large and attempting to make inference on all 40 parameters would be a very ambitious task. However, some valid biological assumptions can be made which reduce the number of parameters. This can be done by constraining parameters of similar processes, as suggested by the biologists involved. A few of these suggestions are outlined below.

In the modelling assumptions, it was assumed that as a protein aggregate got larger, the rate of disaggregation became lower. Currently there are five unique rate constants which control the rate of disaggregation for aggregates of different sizes, $kdisaggPolyQ_i$ for $i = 1, \ldots, 5$. However, the model could allow the rate of disaggregation to decrease as the aggregate gets larger by taking, for example

$$\begin{aligned} kdisaggPolyQ_2 &= 0.8\, kdisaggPolyQ_1 \\ kdisaggPolyQ_3 &= 0.6\, kdisaggPolyQ_1 \\ kdisaggPolyQ_4 &= 0.4\, kdisaggPolyQ_1 \\ kdisaggPolyQ_5 &= 0.2\, kdisaggPolyQ_1. \end{aligned}$$

This would have the effect of removing four parameters from the model. Similar assumptions could be made for parameters which control the rate of disaggregation of misfolded proteins $kdisaggMisP_i$ for $i = 1, \ldots, 5$.

The biologists also suggest that it would be reasonable to assume that any protein bound to the proteasome would be sequestered into an inclusion at the same rate, that is, take

$$kseqAggPProt = kseqPolyQProt = kseqMisPProt = kseqmRFPuProt.$$

There is also reasonably strong prior knowledge which exists about another 16 parameters which could be incorporated in the analysis. This, along with the reduction of dimensionality of the parameter space, could mean that making inference for all parameters in the PolyQ model is a much more realistic aim with moderate sized data. Further discussions on the scope for extra analysis on the PolyQ model is given in Chapter 9.

Chapter 9

Conclusions and future work

The purpose of this thesis was to build a framework for inference on the parameters of stochastic kinetic models when the experimental data are proportions. This complicates the analysis as no observations are available on the underlying chemical species levels. This work was motivated by the large PolyQ model, developed by Tang et al. (2010). Experimental data on the proportions of cell death are available, although they are noisy measurements, at only very few time points. There is a need to formally calibrate the model, with the ultimate aim being to have a model which can be used for *in silico* experiments. The results of these fast and cheap, computer based (*in silico*) experiments could then be used to inform future lab based experiments.

The birth-death model was an ideal toy model on which to develop and test methods. In this model, a population becoming extinct was a convenient surrogate for cell death. The availability of an analytic expression for the probability of extinction facilitated inference on model parameters, where the target posterior distribution was the exact posterior distribution. This was especially useful when faster approximate methods were considered, since the true posterior distribution was available as a *gold standard* with which to compare and assess performance.

Several simulation-based inference schemes were introduced. When ignoring the availability of an analytic expression for the proportion of cell death, approximate proportions can be constructed using n forward simulations from the simulator. When

tested on the birth-death model, using $n = 1000$ forward simulations appeared to provide an adequate approximation to the exact proportion. It was seen that the posterior distribution using the approximate proportions was very close to that using the analytic expression for the probability of cell death.

For models of reasonable size and complexity, the reliance of these simulation-based algorithms on multiple simulations from the model, at each iteration, rendered them computationally infeasible. Gaussian process emulators were successfully used as fast surrogates for the simulator. For the birth-death model, a thorough exploration into fitting Gaussian process emulators was undertaken. The emulators seemed insensitive to the choice of mean function and it was found that fixing hyperparameters at their posterior means was appropriate.

Next, the medium sized mtDNA model was considered. Inference was undertaken using a set of publicly available experimental data. Previous attempts in the literature at parameter inference have all assumed that there was some data available on the underlying chemical species levels. However, the analysis in Chapter 7 has shown that parameter inference is possible using only the proportions of survival. Encouragingly, the inferences were found to be consistent with those of previous studies.

Finally, the PolyQ model was considered. The lack of experimental data meant that attempting to make inference using the very limited data which was available was futile. However, inference was considered using simulated datasets with 30 and 4×30 data points. For both datasets, inference was considered on up to four model parameters along with the measurement error, and the parameters were well recovered.

9.1 Future work

There are many natural extensions to the work in this thesis. Most notably, a more thorough exploration of inference for the PolyQ model is required. So far, only tentative inferences have been made on the PolyQ model using simulated data. Thus far, inference for a maximum of four model parameters (plus measurement error) has been considered;

the rest of the parameters were considered fixed. These attempts appear successful as model parameters can be recovered with their marginal posterior distributions having much reduced variability compared to the prior. It would be interesting to observe how the current framework performs when trying to infer more parameters than has been considered thus far.

Ideally, inference would be made on all parameters of the model. However, this is an ambitious task, especially in the current data–poor scenario. It is expected that if inference was attempted using just the experimental data given in Table 8.4 of Chapter 8, very little would be learned about model parameters and the posterior distribution would be very similar to the prior distribution.

The analysis in Chapter 8 considers inference for the PolyQ model where there are repeated observations at the same time point. However, it would be interesting to explore the scenario where there are repeats of experiments under different conditions: Tang et al. (2010) have data of this form, given in Table 8.4. It may also be of interest to consider including the level on one chemical species for the PolyQ model to see how this changes the inference.

When considering emulation for the PolyQ model, another foreseeable challenge is the increasing size of the Latin hypercube required as the number of parameters gets larger. For example, consider an emulator built with 15 inputs, placing a design point at the maximum and minimum of each input, leads to $2^{15} = 32768$ design points. It has been seen that using the approach of Kaufman et al. (2011), which takes advantage of the near sparsity of the covariance matrix, provides much better scaling than the original approach. However, clearly a different approach is required when the number of inputs is of this order.

One possible way forward is to use history matching. This technique rules out large areas of parameter space which are implausible before emulation begins. The method proceeds iteratively, where the parameter space is reduced further at each iteration. This leaves a much smaller area of parameter space on which to build the final emulator. This approach has successfully been implemented in the modelling of galaxy

formation (Bower et al., 2010; Vernon et al., 2010) and oil formation (Craig et al., 1996, 1997). To be able to implement history matching, it is necessary to have a good idea of the size of the observational error (measurement error). For the PolyQ model, the measurement error is not known, and has to be estimated. However, in conjunction with the experimentalists, it may be possible to elicit an upper bound on this error and use history matching using this value.

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*, 26:99–133.

Aitchison, J. (1986). *The statistical analysis of compositional data*. Monographs on statistics and applied probability. Chapman and Hall.

Alfonsi, A., Cances, E., Turinici, G., Ventura, B., and Huisinga, W. (2005). Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proceedings*, 14:1–13.

Andrieu, C., Doucet, A., and Holenstein, R. (2009). *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72:269–342.

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of Statistics*, 37(2):697–725.

Bailey, N. (1964). *The Elements of Stochastic Processes with Applications to the Natural Sciences*. Wiley, New York.

Barry, R. P. and Kelley Pace, R. (1997). Kriging with large data sets using sparse matrix techniques. *Communications in Statistics - Simulation and Computation*, 26:619–629.

Bastos, L. S. and O'Hagan, A. (2009). Diagnostics for Gaussian process emulators. *Technometrics*, 51:425–438.

Bates, R. A. and Kenett, R. S. (2006). Achieving robust design from computer simulations. *Quality Technology & Quantitative Management*, 3:161–177.

Beaumont, M. A. (2003). Estimation of Population Growth or Decline in Genetically Monitored Populations. *Genetics*, 164:1139–1160.

Bender, A., Krishnan, K. J., Morris, C. M., Taylor, G. A., Reeve, A. K., Perry, R. H., Jaros, E., Hersheson, J. S., Betts, J., Klopstock, T., Taylor, R. W., and Turnbull, D. M. (2006). High levels of mitochondrial DNA deletions in substantia nigra neurons in aging and Parkinson disease. *Nature Genetics*, 38:515–7.

Bower, R. G., Vernon, I., Goldstein, M., Benson, A. J., Lacey, C. G., Baugh, C. M., Cole, S., and Frenk, C. S. (2010). The parameter space of galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 407:2017–2045.

Cao, Y., Gillespie, D. T., and Petzold, L. R. (2006). Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124:044109.

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*, 99:919.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140:640–651.

Craig, P. S. and Goldstein, M. (2001). Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, 96:717 – 729.

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1996). Bayes linear strategies for matching hydrocarbon reservoir history. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistics 5*, pages 69–95.

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1997). Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments. In Gatsonis, C., Hodges, J. S., Kass, R. E., McCulloch, R., Rossi, P., and Singpurwalla, N. D., editors, *Case Studies in Bayesian Statistics*, pages 36–93. Springer, New York.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86:953–963.

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer.

Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46:193–227.

Doucet, A., Pitt, M., and Kohn, R. (2012). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *arXiv:1210.1871v2*.

Elson, J. L., Samuels, D. C., Turnbull, D. M., and Chinnery, P. F. (2001). Random intracellular drift explains the clonal expansion of mitochondrial DNA mutations with age. *American Journal of Human Genetics*, 68:802–6.

Fearnley, J. M. and Lees, A. J. (1991). Ageing and Parkinson's disease: substantia nigra regional selectivity. *Brain*, 114:2283–301.

Feller W. (1939). Die Grundlagen der Volterraschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung. *Acta Biotheoretica*, 5:11–40.

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15:502–523.

Furrer, R. and Sain, S. R. (2010). spam: A sparse matrix R package with emphasis on MCMC methods for Gaussian Markov random fields. *Journal of Statistical Software*, 36:1–25.

Gelman, A., Roberts, G., and Gilks, W. (1996). Efficient Metropolis jumping rules. *Bayesian Statistics 5*, pages 599–608.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457–472.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In Bernado, J., Berger, J., Dawid, A., and Smith, A., editors, *Bayesian statistics 4*. Clarendon Press, Oxford, 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*, 104:1876–1889.

Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81:2340–2361.

Gillespie, D. T. (1992). A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188:404–425.

Gillespie, D. T. (2000). The chemical Langevin equation. *Journal of Chemical Physics*, 113:297–306.

Gillespie, D. T. (2001). Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115:1716.

Gillespie, D. T. and Petzold, L. R. (2003). Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 119:8229.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69:243–268.

Goldstein, M. and Rougier, J. (2006). Bayes linear calibrated prediction for complex systems. *Journal of the American Statistical Association*, 101:1132–1143.

Golightly, A. and Gillespie, C. S. (2013). Simulation of stochastic kinetic models. In Schneider, M. V., editor, *Methods in Molecular Biology*, volume 1021, pages 169–87. 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*, 1:807–820.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140:107–113.

Gramacy, R. B. and Lee, H. K. H. (2010). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22:713–722.

Hassler, R. (1938). Zur Pathologie der Paralysis Agitans und des Postenzephalitischen Parkinsonismus. *Journal of Psychiatry and Neurology*, 48:387–476.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Heidelberger, P. and Welch, P. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, 31:1109 – 1145.

Hemmings, J., Barciela, R., and Bell, M. (2008). Ocean color data assimilation with material conservation for improving model estimates of air-sea CO2 flux. *Journal of Marine Research*, 66:87–126.

Henderson, D. A., Boys, R. J., Krishnan, K. J., Lawless, C., and Wilkinson, D. J. (2009). Bayesian emulation and calibration of a stochastic computer model of mitochondrial DNA deletions in substantia nigra neurons. *Journal of the American Statistical Association*, 104(485):76–87.

Henderson, D. A., Boys, R. J., and Wilkinson, D. J. (2010). Bayesian calibration of a stochastic kinetic computer model using multiple data sources. *Biometrics*, 66(1):249–56.

Higham, D., Intep, S., Mao, X., and Szpruch, L. (2011). Hybrid simulation of autoregulation within transcription and translation. *BIT Numerical Mathematics*, (51):177–196.

Imarisio, S., Carmichael, J., Korolchuk, V., Chen, C.-W., Saiki, S., Rose, C., Krishna, G., Davies, J. E., Ttofi, E., Underwood, B. R., and Rubinsztein, D. C. (2008). Huntington's disease: from pathology and genetics to potential therapies. *The Biochemical journal*, 412(2):191–209.

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.

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555.

Kendall, D. G. (1948). On the generalized 'birth-and-death' process. *Annals of Mathematical Statistics*, 19:1–15.

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):425–464.

Kiehl, T. R., Matteyses, R. M., and Simmons, M. K. (2004). Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322.

Kitano, H. (2001). *Foundations of Systems Biology*. MIT Press.

Kleijnen, J. P. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707–716.

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 uutput from a computer code. *Technometrics*, 21(2):239.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Morris, M. D. and Mitchell, T. J. (1995). Explanatory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402.

Neal, R. M. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical report.

Novozhilov, A. S. (2006). Biological applications of the theory of birth-and-death processes. *Briefings in Bioinformatics*, 7(1):70–85.

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.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300.

Palmer, J. and Totterdell, I. (2001). Production and export in a global ocean ecosystem model. *Deep Sea Research Part I: Oceanographic Research Papers*, 48(5):1169–1198.

Pissanetzky, S. (1984). *Sparse Matrix Technology*. Academic Press.

Pitt, M. K., dos Santos Silva, R., Giordani, P., and Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., and Feldman, M. W. (1999). Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798.

Puchaka, 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):1357–72.

Raferty, A. E. and Lewis, S. (1992). How many iterations in the Gibbs sampler? *Bayesian Statistics*, 4:763–773.

Ranjan, P., Haynes, R., and Karsten, R. (2011). A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data. *Technometrics*, 53(4):366–378.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Renshaw, E. (1993). *Modelling Biological Populations in Space and Time*. Cambridge University Press.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *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.

Rubinsztein, D. C. and Carmichael, J. (2004). Huntington's disease: molecular basis of neurodegeneration. *Expert Reviews in Molecular Medicine*, 5(20):1–21.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4:409–435.

Salis, H. and Kaznessis, Y. (2005). Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *Journal of Chemical Physics*, 122:54103.

Sandmann, W. (2009). Streamlined formulation of adaptive explicit-implicit tau-leaping with automatic tau selection. In *Winter Simulation Conference (WSC), Proceedings of the 2009*, pages 1104–1112. IEEE.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. Springer, New York.

Sherlock, C. (2013). Optimal scaling of the random walk Metropolis: general criteria for the 0.234 acceptance rule. *Journal of Applied Probability*, 50(1):1–15.

Sherlock, C. and Roberts, G. (2009). Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets. *Bernoulli*, 15(3):774–798.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151.

Tang, M. Y., Proctor, C. J., Woulfe, J., and Gray, D. A. (2010). Experimental and computational analysis of polyglutamine-mediated cytotoxicity. *PLoS Computational Biology*, 6(9).

Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. (1997). Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–18.

Vernon, I., Goldstein, M., and Bower, R. G. (2010). Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, 5:619–669.

Wilkinson, D. J. (2012). *Stochastic Modelling for Systems Biology*. Chapman & Hall/CRC Press, Boca Raton, Florida, 2nd edition.

Wood, S. N. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–4.

Yule, G. U. (1925). A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F.R.S. *Philosophical Transactions of the Royal Society of London, Series B, Containing Papers of a Biological Character*, 213(402-410):21–87.