Lancaster University

DATA CONDITIONED SIMULATION AND INFERENCE

Author
Chien Lin Terry Huang

Under the supervision of
Dr. Peter Neal

A thesis submitted for the degree of Doctor of Philosophy in the Department of Mathematics and Statistics.

21 October 2016

Declaration

I, CHIEN LIN TERRY HUANG, declare that this thesis titled, 'Data Conditioned Simulation and Inference' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

To mum, dad, and Hanna

Abstract

With the increasing power of personal computers, computational intensive statistical methods such as approximate Bayesian computation (ABC) are becoming an attractive and viable proposition to analyse complex statistical problems. There are three main aspects to ABC:

- Proposing parameters.
- Simulation of the process.
- Some acceptance or approximation criteria for assessing the simulation.

Majority of the publications on ABC explores the parameter proposition and the acceptance criteria aspects. Our work focuses on the simulation aspect of the algorithm. Our research has led us to the development of *Data Conditioned Simulation*. The data conditioned simulation utilises a mixture of the importance sampling algorithm and data augmentation to *steer* simulations towards the observed data with a corresponding weight to take account of the steering. The consequence of the steering is that even though each simulation takes more time than the unconditioned simulation, fewer simulations are required to make reasonable inference. Without the steering in the simulation, the acceptance rate can be extremely small. We demonstrate the data conditioned simulation through the *data conditioned approximate Bayesian Computation (dcABC)* and the *grouped independence Metropolis-Hastings (GIMH)* algorithm. The methodology is demonstrated through three examples, a homogeneous mixing SIR epidemic model, a time inhomogeneous Markov chain model and the stochastic Ricker model. The implementation of the methodology is problem-specific and we demonstrate the benefits of our approach in all three examples.

Acknowledgements

I would like to express my special appreciation to my supervisor Dr. Peter Neal, you have been a tremendous mentor for me. Your endless support and extreme patience have helped me greatly in completing this thesis. I couldn't have done it without your help.

Hanna Michnowicz, you have been a rock in my silly quest to self indulgence, albeit wobbly at times. However, you made it possible for an otherwise impossible job with the occasional tidying up and keeping everyone at Umezushi on their toes.

The Umezushi team, you have managed to hold it together while I was out there (in my room) typing away (procrastinating away). I have grown and so have you. Thank you for sticking with me. Let's make more yummy food!

I would also like to thank Gary Kendall for constantly indulge my thought experiments and being at the receiving end of my rebelliousness.

Charlie Reams, you are still fixing my English. Nothing has changed since 2002. I am always grateful and will always be!

Finally, my dear parents probably deserves this most. Without them, I wouldn't be here and the rest. Words can't express my appreciation. Your pension is on me, I guess?

Contents

List of Figures **9**

List of Tables **12**

List of Symbols and Acronyms **13**

List of Symbols and Acronyms **14**

1 Introduction **16**
1.1 Research objective 16
1.2 Thesis structure 17

2 A Review of Approximate Bayesian Computation **19**
2.1 Background knowledge 19
2.2 Exact Bayesian computation algorithm 21
2.2.1 The Algorithm 21
2.2.2 Why does exact Bayesian computation (EBC) work? . 22
2.2.3 A toy example 23
2.3 Rejection sampling ABC 24
2.3.1 Motivation 24
2.3.2 The use of sufficient statistics 26
2.3.3 The algorithm 27
2.3.4 Justification of the rejection sampling based approximate Bayesian Computation (rsABC) algorithm 28
2.3.5 Revisiting the Poisson toy example 30
2.4 A quick ABC summary 31
2.5 Sequential Monte Carlo ABC 32

2.5.1 The algorithm 32
2.6 Monte Carlo Markov Chain (MCMC)-ABC 34
2.7 Semi-automatic ABC 38
2.8 Piecewise ABC 39
2.9 ABC with regression correction 40
2.10 Monte Carlo within Metropolis algorithm 40
2.11 Grouped independence Metropolis Hastings algorithm 42

3 Data conditioned simulation methodology 45
3.1 Data conditioned simulation 46
3.2 Data conditioned ABC (dcABC) Algorithm 48
3.3 Example 1: Poisson distribution 52
3.3.1 Data condition simulation 52
3.3.2 dcABC algorithm 55
3.4 Example 2: exponential distribution 56
3.4.1 Data conditioned simulation 57
3.4.2 dcABC algoritm 60
3.5 A note on practical applications 61

4 SIR Model 62
4.1 Introduction 62
4.2 Model description 63
4.3 Algorithm implementation 67
4.3.1 The rsABC algorithm 67
4.3.2 The dcABC algorithm 68
4.3.3 The GIMH algorithm 71
4.3.4 Results 74
4.4 Chapter Conclusion 82

5 Time Inhomogeneous Markov Chain 84
5.1 Introduction 84
5.2 Model description 85
5.2.1 Data 85
5.2.2 The model 85
5.2.3 Time to the last event 86

5.2.4 Data simulation 97
5.3 Algorithm implementation 100
5.3.1 The rsABC algorithm 101
5.3.2 The dcABC algorithm 101
5.3.3 The GIMH algorithm 102
5.3.4 Results 103
5.4 Chapter Conclusion 108

6 Ricker Model **109**
6.1 Introduction 109
6.2 Model description 110
6.3 Exploratory research 111
6.3.1 The Ricker model without Poissonised observation . . . 117
6.3.2 The Ricker model with Poissonised observation on the log scale 129
6.3.3 Higher order dynamics 134
6.3.4 Posterior distribution of ϕ 138
6.3.5 Exploratory research conclusion 138
6.4 Summary statistics selection 139
6.4.1 Choices of summary statistics 139
6.4.2 For r 144
6.4.3 For σ 146
6.4.4 For ϕ 150
6.4.5 Summary statistics conclusion 152
6.5 Data simulation 153
6.5.1 Unconditioned Data Simulation 153
6.5.2 Data conditioned simulation 153
6.6 Algorithm implementation 159
6.6.1 The dcABC algorithm 159
6.6.2 The GIMH algorithm 163
6.6.3 Results 165
6.7 Chapter Conclusion 173

7 Conclusions **175**

7.1 Discussion 175
7.2 Further research 177

Bibliography **179**

List of Figures

3.1 Comparison between the real likelihood and the data conditioned likelihood estimates. 55
3.2 Comparison between the real likelihood and the data conditioned likelihood estimates. 60

4.1 Effects of groups sizes on different aspects of the performance of the GIMH algorithm. 76
4.2 Effects of σ on different aspects of the performance of the GIMH algorithm. 77
4.3 Effects of groups sizes on different aspects of the performance of the data augmented GIMH algorithm. 78
4.4 Effects of σ on different aspects of the performance of the data augmentated GIMH algorithm. 79
4.5 Effects of groups sizes on different aspects of the performance of the dcABC algorithm. 80
4.6 Boxplot of 100 estimates from all four algorithms 82

5.1 dcABC estimations v.s. True values 104
5.2 GIMH estimations v.s. True values 105
5.3 dcABC estimations v.s. GIMH estimations 107

6.1 Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25, \phi = 10$, $N_0 = 1$, T = 100 and L = 50 113
6.2 Three sets of observations simulated with $\sigma = 0.25, \phi = 10$, $N_0 = 1$, T = 100, L = 50, and varying r. 114
6.3 Three sets of observations simulated with $r = e^{3.8}, \phi = 10$, $N_0 = 1$, T = 100, L = 50, and varying σ. 115

6.4 Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25$, $N_0 = 1$, T = 100, L = 50 and varying ϕ. 116
6.5 Bifurcation diagram of the deterministic Ricker model, $r \in [1, 60], N_0 = 1$ 118
6.6 Bifurcation diagram of the Ricker model with a Gaussian error term, $r \in [1, 60], \sigma = 0.25$ 119
6.7 Phase diagram of the deterministic Ricker model with various values of r and $N_0 = 1$ 121
6.8 Phase diagram of the stochastic Ricker model with various value of r with $\sigma = 0.5$ and $N_0 = 1$ 122
6.9 Phase diagram of the stochastic Ricker model with various value of σ with $r = 45$ and $N_0 = 1$ 123
6.10 Phase diagram of the stochastic Ricker model on the log scale for fixed $\sigma = 0.5$ and $N_0 = 1$, and various value of r 125
6.11 Phase diagram of the stochastic Ricker model on the log scale with various value of r with $\sigma = 0.5$ and $N_0 = 1$ zoomed in at the turning point. 126
6.12 Phase diagram of the stochastic Ricker model on the log scale for various value of σ and fixed $r = e^4$, $N_0 = 1$ 127
6.13 Phase diagram of the stochastic Ricker model on the log scale for various value of σ and fixed $r = e^4$, $N_0 = 1$ zoomed in at the turning point. 128
6.14 Z_{t+1} plotted against Z_t for various values of r, σ with $\phi = 10$. 130
6.15 Corresponding $log(N_{t+1})$ plotted against $log(N_t)$ for Figure 6.14 131
6.16 Z_{t+1} against Z_t for various values of r, σ with $\phi = 30$ 133
6.17 Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25, \phi =$ 10, $N_0 = 1$, T = 100 and L = 50 134
6.18 Phase diagram of the deterministic Ricker model with different orders and fixed r. 135
6.19 Phase diagram of the stochastic Ricker model with different orders and fixed r and σ. 136
6.20 Fitting the mixture of two straight lines model. 142
6.21 All summary statistics plotted against $log(r)$ 145
6.22 All summary statistics plotted against $log(\sigma)$ 147

6.23 All summary statistics plotted against σ with fixed r 149
6.24 All summary statistics plotted against ϕ 151
6.25 dcABC estimations of the Ricker Model 166
6.26 GIMH estimations of the Ricker Model 167
6.27 GIMH estimations of the Ricker Model excluding the outliers. 168
6.28 synthetic likelihood MCMC (SMCMC) estimates 169
6.29 SMCMC estimates 170
6.30 Box plot of GIMH estimations and dcABC estimations 172

List of Tables

4.1 The mean estimates, effective sample sizes, and computation times over 100 runs for all algorithms. 81

5.1 Reporting the fitted linear models between the estimated values and the true values. 106
5.2 Summary of output from the dcABC algorithm and the GIMH algorithm. 107

6.1 Correlations between all summary statistics based on 5000 data sets each with 500 observations. 143
6.2 Linear correlations between r, $\log(r)$ and the summary statistics. 146
6.3 Correlations between $\log(\sigma)$ and the summary statistics. . . . 148
6.4 Correlations between σ, $log(\sigma)$ and the summary statistics with fixed $r = e^{3.8}$. 150
6.5 Correlations between ϕ, $\log(\phi)$ and the summary statistics. . . 152
6.6 Simple linear models fitted for the estimations of the dcABC algorithm again the true values. β_0 denotes the intercept term and β_1 denotes the coefficient term. 171
6.7 Linear models fitted for the estimations of the GIMH algorithm again the true values after removing the outliers. β_0 denotes the intercept term and β_1 denotes the coefficient term. 171
6.8 Linear models fitted for the estimations of the SMCMC algorithm again the true values after removing the outliers. β_0 denotes the intercept term and β_1 denotes the coefficient term. 171
6.9 Mean estimates of GIMH, dcABC, and SMCMC 172

List of Symbols and Acronyms

ABC approximate Bayesian computation.

CDF cummulative density function.

dcABC data conditioned approximate Bayesian Computation.

$\pi(\cdot)$ Throughout this thesis, $\pi(\cdot)$ is used to represent some probability density function or probability mass function.

EBC exact Bayesian computation.

$d(\cdot, \cdot)$ $d(\cdot, \cdot)$ denotes the Euclidean metric.

$\Gamma(\alpha, \beta)$ In this thesis, we adopt the shape and rate parameterisation of the gamma distribution. The first parameter α denotes the shape parameter, and the second parameter β denotes the rate parameter. The corresponding density function is $\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} x^{-\beta x}$.

$\Gamma(k)$ $\Gamma(k)$ is the standard gamma function with parameter k.

GIMH grouped independence Metropolis-Hastings.

I.I.D. independent and identically distributed.

$\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$ Throughout this thesis, $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$ is exclusively used to represent likelihood function of the model parameter.

MCMC Monte Carlo Markov Chain.

MCMC-ABC Markov Chain Monte Carlo ABC.

MCWM Monte Carlo within Metropolis.

$\pi(\boldsymbol{x}^*)$ $\pi(\boldsymbol{x}^*)$ is used to denote the marginal density of the observed data.

Poisson $Poisson(\theta)$ denotes the Poisson distribution with parameter θ.

$\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$ Throughout this thesis, $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$ is exclusively used to represent posterior distribution of the model parameter..

$\pi(\boldsymbol{\theta})$ Throughout this thesis, $\pi(\boldsymbol{\theta})$ is exclusively used to represent prior distribution of the model parameter.

$\mathbb{P}(\mathcal{E})$ $\mathbb{P}(\cdot)$ is used to denote the probability of event $\mathcal{E}$ occurring.

$\mathbb{R}^d$ $\mathbb{R}$ is used to denote the d dimensional real space.

rsABC rejection sampling based approximate Bayesian Computation.

SIR Susceptible - Infectious - Recovered.

SIS Susceptible - Infectious - Susceptible.

SMC-ABC sequential Monte Carlo ABC.

$\boldsymbol{T(\cdot)}$ $\boldsymbol{T(\cdot)}$ is exclusively used to represent sufficient statistics function. It can be either a scalar or a vector..

$\boldsymbol{S(\cdot)}$ $\boldsymbol{S(\cdot)}$ is exclusively used to represent sufficient statistics function. It can be either a scalar or a vector..

$\boldsymbol{\theta}$ Throughout this thesis, $\boldsymbol{\theta}$ is exclusively used to represent model parameter. The bold font is used to indicate that there may be more than one parameters.

$\boldsymbol{x}^*$ Throughout this thesis, $\boldsymbol{x}^*$ is used to represent data simulated from the model of interest. The lack of superscripted $*$ is to itself from the observed data.

$\boldsymbol{x}^*$ Throughout this thesis, $\boldsymbol{x}^*$ is exclusively used to represent observed data.

Chapter 1

Introduction

1.1 Research objective

The standard rsABC algorithm can be very inefficient in complex models. Most rsABC algorithms can be broken down into three parts: sampling of the parameters, sampling of the model, and rejection criterion. Much research effort has gone into improving the efficiency of the rsABC algorithm. We identified that most of the research has focused on an MCMC based approach for sampling of the parameters or improving the acceptance criterion, and there is little discussion on the simulation of the model itself, which we believe is an important element to all ABC related algorithms. Our aim in this work is to find a stable, robust, and efficient model simulation algorithm. The main innovation of our research is the *data conditioned simulation*, which uses a mixture of data augmentation algorithm and importance sampling algorithm to steer the simulation so that the simulated data matches with the observed data either in the full data form or some summary statistics. The advantage of this method is that every simulation contributes to the final estimation of the likelihood, and no information is thrown away which is the case for the rejection sampling based algorithm. We demonstrate the

idea of the data conditioned simulation in the data condition ABC algorithm (dcABC algorithm), and we also show how the dcABC algorithm can be applied to three very different types of problems: Susceptible - Infectious - Recovered (SIR) model, time inhomogeneous Markov chain mode and the Ricker model (a stochastic non-linear dynamic model).

1.2 Thesis structure

This thesis comprises 6 further chapters (Chapters 2 - 7); one literature review chapter, one theory chapter, three example chapters, and finally the conclusion.

In chapter 2, we being by laying the ground work for the literature review with the introduction of the *EBC* and *rsABC* and a toy example. It follows with a discussion of classification of ABC-related algorithms depending on which stage the innovations are made compared to the EBC algorithm. Finally, we provide an extensive review of some of the major ABC-related algorithms.

In chapter 3, we introduce the core idea of this thesis, the *data conditioned simulation*, through the dcABC with two toy examples: a discrete data model (Poisson distribution) and a continuous data model (exponential distribution).

In chapter 4, we look at the first of the three problems: the SIR model. We begin by introducing the SIR model and its properties. We then apply the rsABC algorithm, the dcABC algorithm, the GIMH, the data augmented GIMH to the problem. Finally, the outcomes of the analysis are presented and discussed.

In chapter 5, we look at a Susceptible - Infectious - Susceptible (SIS) model, which is in essence a time inhomogeneous Markov chain model. We begin with an involved discussion of time inhomogeneous Markov chain model,

which paves the way to the implementation of both the unconditioned simulation and the data conditioned simulation. We apply the rsABC algorithm, dcABC algorithm, and the GIMH algorithm on some simulated data as well as the true data. Finally, outcomes of the analysis are presented and discussed.

In chapter 6, we look at the Ricker model, which is a stochastic non-linear dynamic system devised for predicting fish stock in a fishery. We start with the model description and then an extensive exploratory research in order to establish what summary statistics to use. It follows by the description of the unconditioned simulation and the data conditioned simulation. We then apply the dcABC algorithm and the GIMH algorithm to some simulated data. Finally, outcomes of the analysis are presented and discussed.

In chapter 7, we discuss our findings from the three examples and identify possible areas for future research.

Chapter 2

A Review of Approximate Bayesian Computation

In this chapter, we give an overview of current developments in the ABC algorithm. We start by defining some notation and providing background knowledge in order to assist the examination of the EBC (Section 2.2), and rsABC (Section 2.3) in detail. We will then look at some variants of the ABC algorithm including the sequential Monte Carlo ABC (SMC-ABC) algorithm (Section 2.5), the Markov Chain Monte Carlo ABC (MCMC-ABC) algorithm (Section 2.6), the semi-automatic ABC algorithm (Section 2.7), the piecewise ABC algorithm (Section 2.8), the ABC algorithm with regression correction (Section 2.9), the Monte Carlo within Metropolis (MCWM) (Section 2.10), and the GIMH algorithm (Section 2.11).

2.1 Background knowledge

We assume that there exists a parametric model which gives rise to data $\boldsymbol{x}^*$, which are observed from a parametric model parameterised by $\boldsymbol{\theta}$. Let

$\pi(\boldsymbol{\theta})$ denote the prior distribution of the parameters $\boldsymbol{\theta}$, $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$ denote the likelihood of the observed data given $\boldsymbol{\theta}$, and $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$ denote the posterior distribution of the parameters $\boldsymbol{\theta}$ given $\boldsymbol{x}^*$.

Then by Bayes' theorem:

$$\begin{aligned}\pi(\boldsymbol{\theta}|\boldsymbol{x}^*) &= \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})}{\pi(\boldsymbol{x}^*)} \\ &= \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})}{\displaystyle\int_{\forall\boldsymbol{\theta}} \pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &\propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta}).\end{aligned}$$

Our primary interest is in obtaining samples from the posterior distribution, $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$, to estimate posterior quantities of interest such as $\mathbb{E}[\boldsymbol{\theta}|\boldsymbol{x}^*]$.

The likelihood, $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$, can be difficult to calculate or intractable in many complex statistical problems. In situations where the likelihood can be calculated either directly or through data augmentation, MCMC is usually the answer to obtain such samples (see Algorithm 6 below for an outline of the MCMC algorithm). The MCMC algorithm will, if started from $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$, produce an exact dependent sample from the posterior distribution. However, MCMC is computationally intensive and can be prohibitively slow or complicated for complex models with intractable likelihoods. Thus alternatives are required to estimate $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$.

Approximate Bayesian computation is a methodology to circumvent evaluation of likelihoods directly through simulation of the likelihood. The idea is that the likelihood can be approximated by simulation and hence obtaining samples from the posterior distribution.

The ABC algorithm can be applied to a broad range of models, and the details of the algorithm depend heavily on the nature of the models and data. This flexibility of ABC has contributed to the growing interest and

development of this methodology.

2.2 Exact Bayesian computation algorithm

2.2.1 The Algorithm

The exact Bayesian computation algorithm is the simplest form of the Bayesian computation algorithm with no approximation [White et al., 2013] and is a rejection sampling algorithm. It is generally a restrictive and inefficient algorithm, but nonetheless it is a good starting point for understanding the ABC algorithm.

Algorithm 1 (EBC Algorithm)**.**

1. Set $i = 1$.
2. Sample $\boldsymbol{\theta}$ from $\pi(\boldsymbol{\theta})$.
3. Generate data $\boldsymbol{x}$ from $\pi(\boldsymbol{x}|\boldsymbol{\theta})$, the model with parameters $\boldsymbol{\theta}$.
4. If $\boldsymbol{x} = \boldsymbol{x}^*$, then set $\boldsymbol{\theta_i} = \boldsymbol{\theta}$ and increment i by 1.
5. Let m be the desired sample size. If $i < m$, go to 2.
6. $\{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m}\}$ is an independent and identically distributed sample from $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$.

This version of the algorithm requires the simulated data to be precisely matched to the observed data before an acceptance. Thus, it requires the model to be a discrete data model. Otherwise $\mathbb{P}(\boldsymbol{x}_i = \boldsymbol{x}^*) = 0$, and no acceptance is possible. Even for a discrete data model, $\mathbb{P}(\boldsymbol{x}_i = \boldsymbol{x}^*)$ is often very small and hence the acceptance rate is prohibitively low, see Section 2.2.3. One upside is that this algorithm will produce independent and independent and identically distributed (I.I.D.) samples from the exact posterior distribution rather than the approximated posterior distribution.

2.2.2 Why does EBC work?

To show why the EBC algorithm works, we utilise the uniqueness property of cumulative density functions. That is, we have to show that

$$\int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|Accepted)d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|\boldsymbol{x}^*)d\boldsymbol{\theta} \tag{2.1}$$

where $\mathcal{A} \subseteq \mathbb{R}^d$, and d is the dimension of the parameters, $\boldsymbol{\theta}$, then we have the proof.

Given $\mathcal{A} \subseteq \mathbb{R}^d$, to compute the cumulative density function on the left hand side of (2.1) is equivalent to computing $\mathbb{P}\big((\boldsymbol{\theta} \in \mathcal{A})|Accepted\big)$. Thus, starting from the left hand side of (2.1):

$$\begin{aligned}
\int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|Accepted)d\boldsymbol{\theta} &= \mathbb{P}\big((\boldsymbol{\theta} \in \mathcal{A})|Accepted\big) \\
&= \frac{\mathbb{P}\big((\boldsymbol{\theta} \in \mathcal{A}) \cap Accepted\big)}{\mathbb{P}(Accepted)} \\
&= \frac{\displaystyle\int_{\boldsymbol{\theta}\in\mathcal{A}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\displaystyle\int_{\forall\boldsymbol{\theta}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \\
&= \frac{\displaystyle\int_{\boldsymbol{\theta}\in\mathcal{A}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^*)} \\
&= \frac{\displaystyle\int_{\boldsymbol{\theta}\in\mathcal{A}} \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\pi(\boldsymbol{x}^*)} \\
&= \int_{\boldsymbol{\theta}\in\mathcal{A}} \frac{\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{x}^*)}d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|\boldsymbol{x}^*)d\boldsymbol{\theta},
\end{aligned}$$

as required.

2.2.3 A toy example

Consider a data set, $\boldsymbol{x}^*$, consisting of 10 observations from a *Poisson* distribution, $Poisson(\theta^*)$, with $\theta^* = 5$, i.e.

$$\boldsymbol{x}^* = \{2, 4, 5, 5, 5, 6, 7, 7, 8, 10\}.$$

Since the Gamma distribution is the conjugate prior of the *Poisson* distribution, it is only natural to use $\Gamma(\alpha, \beta)$ as the prior distribution of θ, where $\alpha > 0$ is the shape parameter and $\beta > 0$ is the rate parameter. We choose $\alpha = 1$ and $\beta = 0.1$ to represent a fairly diffused prior to simulate a lack of prior knowledge.

Algorithm 2 (The EBC algorithm for the *Poisson* example)**.**

1. Set $i = 1$
2. Sample θ from $\Gamma(1, 0.1)$.
3. Sample 10 values, $\boldsymbol{x} = \{x_1, \ldots, x_{10}\}$ from $Poisson(\theta)$.
4. If $\boldsymbol{x} = \boldsymbol{x}^*$, we accept θ and increment i by 1. Otherwise we discard θ.
5. Let m be the desired sample size. If $i < m$, go to 2.

In this example, we set the desired sample size to be 100. It took 8.74×10^6 iterations to achieve that using Algorithm 2. That means an acceptance rate in the region of 1.14×10^{-5}, which is low for such a simple model. However, in practice this magnitude of acceptance rate is good compared with using ABC for real life problems.

2.3 Rejection sampling ABC

2.3.1 Motivation

From the toy example in Section 2.2.3, we see that even for such a simple model, the EBC algorithm is inefficient. Suppose that we know sufficient statistics of the parameter, then we can improve on the efficiency of the algorithm while still obtaining a sample from the exact posterior distribution. Below we will demonstrate using the toy example from Section 2.2.3.

Throughout this chapter, we will let $\boldsymbol{T(\cdot)}$ denote sufficient statistics, which can be either a scalar or a vector. In the context of the toy example from Section 2.2.3, we can identify a sufficient statistic for θ. By the factorisation theorem, the sufficient statistic for the parameter of a $Poisson$ distribution with n identically independent distributed samples is $\sum_{i=1}^{n} X_i$. Furthermore, the sum of independent $Poisson$ random variables is itself a $Poisson$ distribution with the parameter being the sum of all the parameters of the random variables. Thus $T(\boldsymbol{X}) \sim Poisson(10\theta)$.

Given any θ, we want to compare $\mathbb{P}(\boldsymbol{x} = \boldsymbol{x^*})$ and $\mathbb{P}\big(T(\boldsymbol{x}) = T(\boldsymbol{x^*})\big)$. This is not always possible, but in this example we can compute the probability exactly. Note that these are the marginal likelihoods, and correspond to the probabilities of getting an exact match (matching all 10 terms or the sum of the terms).

For $\mathbb{P}(\boldsymbol{x} = \boldsymbol{x}^*)$:

$$\begin{aligned}
\mathbb{P}(\boldsymbol{x} = \boldsymbol{x}^*) &= \int_0^\infty \mathbb{P}(\boldsymbol{x} = \boldsymbol{x}^*|\theta)\pi(\theta)d\theta \\
&= \int_0^\infty \frac{10!}{3!2!} \prod_{i=1}^{10} \mathbb{P}(x_i = x_i^*|\theta)\pi(\theta)d\theta \\
&= \int_0^\infty \frac{10!}{3!2! \prod_{i=1}^{10}(x_i^*!)} \theta^{\sum_{i=1}^{10} x_i^*} e^{-10\theta} 0.1e^{-0.1\theta} d\theta \\
&= \frac{9!}{3!2! \prod_{i=1}^{10}(x_i^*!)} \int_0^\infty \theta^{\sum_{i=1}^{10} x_i^*} e^{-10.1\theta} d\theta \\
&= \frac{9!}{3!2! \prod_{i=1}^{10}(x_i^*!)} \frac{\Gamma(\sum_{i=1}^{10} x_i^* + 1)}{10.1^{\sum_{i=1}^{10} x_i^* + 1}} \int_0^\infty \frac{10.1^{\sum_{i=1}^{10} x_i^* + 1}}{\Gamma(\sum_{i=1}^{10} x_i^* + 1)} \theta^{\sum_{i=1}^{10} x_i^*} e^{-10.1\theta} d\theta \\
&= \frac{9!}{3!2! \prod_{i=1}^{10}(x_i^*!)} \frac{\Gamma(\sum_{i=1}^{10} x_i^* + 1)}{10.1^{\sum_{i=1}^{10} x_i^*}} \\
&= 1.04 \times 10^{-5}.
\end{aligned}$$

The combinatorial term, $\frac{10!}{3!2!}$, is to account for the ordering of the sample. We consider $\boldsymbol{x} = \boldsymbol{x}^*$ as long as all elements match up regardless of the order.

And for $\mathbb{P}\big(T(\boldsymbol{x}) = T(\boldsymbol{x}^*)\big)$:

$$\begin{aligned}
\mathbb{P}\big(T(\boldsymbol{x}) = T(\boldsymbol{x}^*)\big) &= \int_0^\infty \mathbb{P}\big(T(\boldsymbol{x}) = T(\boldsymbol{x}^*)|\theta\big)\pi(\theta)d\theta \\
&= \int_0^\infty \frac{(10\theta)^{T(\boldsymbol{x}^*)} e^{-10\theta}}{T(\boldsymbol{x}^*)!} 0.1e^{-0.1\theta} d\theta \\
&= \frac{10^{T(\boldsymbol{x}^*)}}{10(T(\boldsymbol{x}^*)!)} \frac{\Gamma(T(\boldsymbol{x}^*) + 1)}{10.1^{T(\boldsymbol{x}^*)+1}} \int_0^\infty \frac{10.1^{T(\boldsymbol{x}^*)+1}}{\Gamma(T(\boldsymbol{x}^*) + 1)} \theta^{T(\boldsymbol{x}^*)} e^{-10.1\theta} d\theta \\
&= \frac{10^{T(\boldsymbol{x}^*)}}{10(T(\boldsymbol{x}^*)!)} \frac{\Gamma(T(\boldsymbol{x}^*) + 1)}{10.1^{T(\boldsymbol{x}^*)+1}} \\
&= 5.50 \times 10^{-3}
\end{aligned}$$

From the computation above, we can see that the algorithm using the sufficient statistics, $T(\boldsymbol{X})$, is 530 times more likely to have a match than the algorithm using the raw data, $\boldsymbol{X}$. This is a dramatic improvement of 2 orders of magnitude. However, in practice, finding sufficient statistics is often difficult and therefore non-sufficient summary statistics are often used. This leads to the next algorithm: the *rsABC algorithm*.

The version of rsABC that will be discussed in this section first appeared in [Pritchard et al., 1999]'s work on population genetics. This version of rsABC makes two modifications to the EBC algorithm (Algorithm 1):

- It adopts summary statistics instead of the raw data.
- It relaxes the acceptance criterion: for an acceptance, we only require the generated data to be "*close*" to the observed data rather than an exact match.

These two modifications introduce "approximation" into the EBC algorithm, hence the name *Approximate Bayesian Computation*.

2.3.2 The use of sufficient statistics

Suppose that sufficient statistics, $\boldsymbol{T(x)}$, can be identified for a model.

Recall Bayes' theorem:

$$\begin{aligned}\pi(\boldsymbol{\theta}|\boldsymbol{x}^*) &= \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})}{\int_{\forall\boldsymbol{\theta}} \pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &= K\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta}),\end{aligned}$$

where $K = \left[\int_{\forall\boldsymbol{\theta}} \pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}\right]^{-1}$.

Let $\boldsymbol{t}^*$ denote the sufficient statistic of the observed data, where $\boldsymbol{t}^* = \boldsymbol{T}(\boldsymbol{x}^*)$, then:

$$\begin{aligned}\pi(\boldsymbol{\theta}|\boldsymbol{x}^*) &= \pi(\boldsymbol{\theta}|\boldsymbol{x}^*, \boldsymbol{t}^*) \\ &= \pi(\boldsymbol{\theta}|\boldsymbol{t}^*) \\ &= \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{t}^*|\boldsymbol{\theta})}{\pi(\boldsymbol{t}^*)},\end{aligned}$$

hence $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*) \propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{t}^*|\boldsymbol{\theta})$.

This shows that Bayes' theorem still holds after substituting the raw data with a sufficient statistic. If we further substitute the sufficient statistic with some approximating summary statistic, it will then allow us to work with the modifications made to the rsABC algorithm in comparison to the EBC algorithm.

2.3.3 The algorithm

We use $\boldsymbol{S}(\boldsymbol{x})$ denote summary statistics of $\boldsymbol{x}$, in order to distinguish summary statistics from sufficient statistics. Let $d(\cdot, \cdot)$ denote the Euclidean metric, and let h denote a non-negative real number. The rsABC algorithm is as follows:

Algorithm 3 (rsABC algorithm)**.**

1. Set $i = 1$
2. Sample $\boldsymbol{\theta}$ from $\pi(\boldsymbol{\theta})$.
3. Generate data $\boldsymbol{x}$ from $\pi(\boldsymbol{x}|\boldsymbol{\theta})$.
4. If $d(\boldsymbol{S}(\boldsymbol{x}), \boldsymbol{S}(\boldsymbol{x}^*)) \leq h$, then set $\boldsymbol{\theta}_i = \boldsymbol{\theta}$ and increment i by 1.
5. Let m be the desired sample size. If $i < m$, go to 2.
6. $\{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m}\}$ is an independent and identically distributed sample from an approximation of $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$

Algorithm 3 is very similar to Algorithm 1 with the only difference being step 4., which is the comparing and deciding stage of the algorithm. This

modification compares summary statistics between the generated data and the observed data instead of the raw data and it also makes allowance for not requiring an exact match. Furthermore, it allows Algorithm 3 to be used with continuous data.

Samples produced using Algorithm 3 are from an approximation of the posterior distribution, but if we were able to identify the sufficient statistics and let $h \to 0$, then we have samples from the exact posterior distribution. The "closer" $\boldsymbol{S(x)}$ is to sufficiency,s and the smaller h is, the closer the generated sample will be to being a sample from the posterior distribution. However, often it is not possible to identify useful low order sufficient statistics for complex models, and thus we have to use alternative summary statistics.

2.3.4 Justification of the rsABC algorithm

To justify the rsABC algorithm, a similar argument to that used for the EBC algorithm can be made. Due to the two modifications made to the EBC algorithm, we will be looking at the limiting behaviour of the cummulative density function (CDF) of the sampled $\boldsymbol{\theta_i}$ that is indeed the same or approximately the same as the CDF of the posterior distribution.

Suppose that for a given region $\mathcal{A} \subseteq \mathbb{R}^d$, where d is the dimension of the parameter. The goal for the next step is to establish that:

$$\mathbb{P}(\boldsymbol{\theta} \in \mathcal{A}|Accepted) = \int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|\boldsymbol{x^*})d\boldsymbol{\theta}. \tag{2.2}$$

We proceed in the similar manner as in Section 2.2. Starting from the left hand side of (2.2):

$$\mathbb{P}\big((\boldsymbol{\theta} \in \mathcal{A})|Accepted\big) = \frac{\int_{\boldsymbol{\theta}\in\mathcal{A}} \mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}), \boldsymbol{S}(\boldsymbol{x}^*)\big) \leq h \Big| \boldsymbol{\theta}\Big)\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}), \boldsymbol{S}(\boldsymbol{x}^*)\big) \leq h\Big)}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \frac{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}), \boldsymbol{S}(\boldsymbol{x}^*)\big) \leq h \Big| \boldsymbol{\theta}\Big)\pi(\boldsymbol{\theta})}{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}), \boldsymbol{S}(\boldsymbol{x}^*)\big) \leq h\Big)} d\boldsymbol{\theta}.$$

Suppose that the posterior is continuous in h and $\boldsymbol{S}(\cdot)$, but not necessarily continuous in $\boldsymbol{X}$, so that we know the limit,

$$\lim_{h\to 0, \boldsymbol{S}(\cdot)\to\boldsymbol{T}(\cdot)} \mathbb{P}\big((\boldsymbol{\theta} \in \mathcal{A})|Accepted\big),$$

exists. Further suppose that the likelihood is continuous in h and $\boldsymbol{S}(\cdot)$, but not necessarily continuous in $\boldsymbol{X}$, so that we can swap the order of integral and limit, then we have:

$$\lim_{h\to 0, \boldsymbol{S}(\cdot)\to\boldsymbol{T}(\cdot)} \mathbb{P}\big((\boldsymbol{\theta}\in\mathcal{A})|Accepted\big)$$

$$= \lim_{h\to 0, \boldsymbol{S}(\cdot)\to\boldsymbol{T}(\cdot)} \int_{\boldsymbol{\theta}\in\mathcal{A}} \frac{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}),\boldsymbol{S}(\boldsymbol{x}^*)\big)<h\Big|\boldsymbol{\theta}\Big)\pi(\theta)}{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}),\boldsymbol{S}(\boldsymbol{x}^*)\big)<h\Big)} d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \lim_{h\to 0, \boldsymbol{S}(\cdot)\to\boldsymbol{T}(\cdot)} \frac{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}),\boldsymbol{S}(\boldsymbol{x}^*)\big)<h\Big|\boldsymbol{\theta}\Big)\pi(\theta)}{\mathbb{P}\Big(d\big(\boldsymbol{S}(\boldsymbol{x}),\boldsymbol{S}(\boldsymbol{x}^*)\big)<h\Big)} d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \frac{\mathbb{P}\big(\boldsymbol{T}(\boldsymbol{x})=\boldsymbol{T}(\boldsymbol{x}^*)\big|\boldsymbol{\theta}\big)\pi(\theta)}{\mathbb{P}\big(\boldsymbol{T}(\boldsymbol{x})=\boldsymbol{T}(\boldsymbol{x}^*)\big)} d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \frac{\pi\big(\boldsymbol{T}(\boldsymbol{x}^*)\big|\boldsymbol{\theta}\big)\pi(\theta)}{\pi\big(\boldsymbol{T}(\boldsymbol{x}^*)\big)} d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|\boldsymbol{t}^*) d\boldsymbol{\theta}$$

$$= \int_{\boldsymbol{\theta}\in\mathcal{A}} \pi(\boldsymbol{\theta}|\boldsymbol{x}^*) d\boldsymbol{\theta},$$

as required.

2.3.5 Revisiting the Poisson toy example

To demonstrate the rsABC algorithm, we use the same data from Section 2.2.3. We know from Section 2.2.3 that $T(\boldsymbol{X}) \sim Poisson(10\theta)$.

Now suppose that $h = 1$, $\theta \sim \Gamma(1, 0.1)$, $d(a,b) = (a^2+b^2)^{1/2}$, and $s^* = \sum_{i=1}^{10} x_i = 66$, then the rsABC algorithm is as follows:

Algorithm 4 (rsABC algorithm for the Poisson example)**.**

1. Set $i = 1$
2. Sample θ from $\Gamma(1, 0.1)$.
3. Sample a value s from $Poisson(10\theta)$.
4. Accept θ if $d(s, s^*) \leq 1$ and increment i by 1.
5. Let m be the desired sample size. If $i < m$, go to 2.
6. $\{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m}\}$ is an independent and identically distributed sample from an approximation of $\pi(\boldsymbol{\theta}|\boldsymbol{x^*})$

The rsABC algorithm achieved the desired 100 sample size with 5609 iterations. That means an acceptance rate of 1.78×10^{-2}, which is about 1561 times more efficient than the EBC algorithm.

2.4 A quick ABC summary

Section 2.3 provides a good example for the general framework of ABC algorithms, and indeed many ABC algorithms fall under the rsABC umbrella. However, there are other algorithms which do not fall under this category. For the more general ABC algorithm, there are still three key stages to any ABC algorithm:

1. Sampling of the parameters, $\boldsymbol{\theta}$
2. Sampling data $\boldsymbol{x}$ from $\pi(\boldsymbol{x}|\boldsymbol{\theta})$
3. Comparing $\boldsymbol{x}$ and $\boldsymbol{x^*}$

The majority of the research on ABC algorithms is focused on Stages 1. (SMC-ABC, MCMC-ABC) and 3.(semi-automatic ABC, ABC with regression correction), whereas this thesis concentrate on Stage 2. We will discuss some of these algorithms in the following sections before going into our research.

2.5 Sequential Monte Carlo ABC

There had been many studies on applying sequential techniques with ABC framework. [Bortot et al., 2007, Sisson et al., 2007, Robert et al., 2008, Toni et al., 2009, Beskos et al., 2011, Peters et al., 2012, Del Moral et al., 2012, Silk et al., 2012, Bonassi et al., 2015, Filippi et al., 2013]. In particular, SMC-ABC algorithm is probably the most studied algorithm under the sequential approach to the ABC algorithm. It is an algorithm which make an innovation at the parameter sampling stage. Although in the toy example, the acceptance rate is reasonably high even with a small error tolerance, this is often not the case in practice. Therefore, as the name suggests, SMC-ABC approaches the approximation via a sequence of approximations. Instead of trying to hit the best result at once as rsABC does, it starts by allowing a large tolerance, h, and hence a higher acceptance rate, and then gradually reduce h and samples only from the region of the previously accepted values. Although this may seem to duplicate the work at each stage, it allows the algorithm to explore and exclude unlikely parameter regions quickly, and therefore more computational effort can be concentrated on the important regions. Thus, the SMC-ABC algorithm is useful when there is a lack of prior knowledge about the parameters. Algorithm 5 is an early development of SMC-ABC proposed in [Bortot et al., 2007], which only considers the final set of the sample for inference. In the later development, see [Sisson et al., 2007], SMC-ABC considers samples from each stage with adjustment weights. Although Algorithm 5 is less efficient, but it serves as a good example to understand the sequential approach to SMC-ABC.

2.5.1 The algorithm

Let m denote the desired sample size, $\epsilon \sim D(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$ denote some perturbation distribution with mean μ and variance σ^2, and $h_1 > \ldots > h_k \geq 0$ denote a sequence of decreasing error tolerance.

Algorithm 5 (SMC-ABC algorithm)**.**

1. Set $i = 1$
2. Perform the rsABC algorithm with $h = h_1$ to get m accepted values $\{\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{i})}, \ldots, \boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{i})}\}$.
3. Increment i by 1.
4. Let the proposal distribution $q(\boldsymbol{\theta}) = \frac{1}{m} D(\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{i-1})}, \boldsymbol{\sigma}^{\mathbf{2}}) + \ldots + \frac{1}{m} D(\boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{i-1})}, \boldsymbol{\sigma}^{\mathbf{2}})$, i.e. the mixture distribution based on the previously accepted values.
5. Perform the rsABC algorithm with $h = h_i$ and the new proposal distribution to get a new set of $\{\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{i})}, \ldots, \boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{i})}\}$
6. If $i < k$, then repeat 3. - 5., or else $\{\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{k})}, \ldots, \boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{k})}\}$ is a sample from the posterior distribution.

NOTE 1. *Let* $\boldsymbol{\Theta}^{(\boldsymbol{i})} = \{\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{i})}, \ldots, \boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{i})}\}$*, then* $\boldsymbol{\Theta}^{(\mathbf{1})}, \ldots, \boldsymbol{\Theta}^{(\boldsymbol{k})}$ *are sequentially generated with conditionally independent components* $\{\boldsymbol{\theta}_{\mathbf{1}}^{(\boldsymbol{i})}, \ldots, \boldsymbol{\theta}_{\boldsymbol{m}}^{(\boldsymbol{i})}\}$ *given* $\boldsymbol{\Theta}^{(\boldsymbol{i-1})}$.

There are three key parts to SMC-ABC:

1. the reducing error tolerance $(h_1, \ldots, h_k)$,
2. the proposal distribution (also called the perturbation) $q(\boldsymbol{\theta})$,
3. the weights (omitted in Algorithm 5).

Careful choices of the error tolerance, the proposal distribution and the weights will improve the efficiency of the algorithm. In [Silk et al., 2012], an automated and adaptive scheme for choosing $h_1, \ldots, h_k$ is proposed, which balances between the need for minimising the error tolerance and the computation efficiency. In [Filippi et al., 2013], a locally adapted kernel (proposal distribution) is discussed which improves the acceptance rate with little computation cost. An adaptive weights scheme is proposed in [Bonassi et al., 2015], which also improves on the acceptance rate, and hence the efficiency of the algorithm.

2.6 MCMC-ABC

In rsABC, drawing $\boldsymbol{\theta}$ from $\pi(\theta)$ can lead to a very inefficient algorithm, especially with an uninformative prior. By combining the idea of the MCMC algorithm with likelihood simulation, it gives an algorithm that often explores the parameter space more efficiently at the expense of having dependent samples. [Marjoram et al., 2003, Bortot et al., 2007, Ratmann et al., 2007]

We will start by looking at the MCMC algorithm and then extend it to include a simulation based approximation of the likelihood. The idea is that once we establish that the MCMC algorithm can give us samples from the desired posterior distribution, we approximate the likelihood by an unbiased approximation. We will show later that the algorithm produces samples from the posterior distribution.

MCMC algorithm may be used to mean an array of different algorithms. However, we consider only the Metropolis-Hastings (MH) algorithm here. The core of any MCMC algorithm is to construct a Markov chain so that the corresponding stationary distribution is the posterior distribution we wish to sample from. Algorithm 6 is a generic MH algorithm for simulating from the posterior distribution.

Algorithm 6 (MH Algorithm)**.**

1. Let $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ be the proposal distribution. $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ gives the density of moving from $\boldsymbol{\theta}$ to $\boldsymbol{\theta'}$
2. Initiate $\boldsymbol{\theta}_1$ with an arbitrary number, and ideally from the posterior distribution if possible.
3. Propose a move from $\boldsymbol{\theta_i}$ to $\boldsymbol{\theta'}$ according to $q(\boldsymbol{\theta_i}, \boldsymbol{\theta'})$
4. Set $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta'}$ with probability $\min\left\{1, \dfrac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i})\pi(\boldsymbol{x^*}|\boldsymbol{\theta'})\pi(\boldsymbol{\theta'})}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'})\pi(\boldsymbol{x^*}|\boldsymbol{\theta_i})\pi(\boldsymbol{\theta_i})}\right\}$, or else $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta_i}$.
5. Repeat 2. to 4. until desired sample size is reached.

A sketch proof of MH algorithm. In order to show that the MH algorithm

does indeed gives us samples from the desired distribution we start by stating two well known Markov chain theorems, detailed balance and ergodicity, and then we will show how these two theorems are utilised in the algorithm. Our argument is based on the discrete time Markov chains, and we will let $\theta_1, \theta_2, \ldots$ represent observations from a Markov chain.

Theorem 1 (Detailed balance - existence of stationary distribution [Norris, 1997]). *Let $f(a)$ be some probability density, and $P(a, b)$ be the transition density of a discrete time and continuous state Markov chain, i.e. the density of jumping from state a to state b, then the detailed balance condition states that: $f(a)$ is the stationary density of the Markov chain, if $f(a)P(a, b) = f(b)P(b, a)$ for all a, b.*

Theorem 2 (Ergodicity - uniqueness of stationary distribution[Norris, 1997]). *The stationary distribution is unique, if $P(a, b)$, the transition density is ergodic meaning: 1) aperiodic and 2) positive recurrent.*

We are going to show that the transition density of the Markov chain constructed using the MH algorithm satisfies both the detailed balance and ergodicity with the correct stationary density (the posterior distribution). We will prove the case for continuous state Markov chain since this is usually the case required with MCMC. To begin, we will firstly establish the transition density, $P(\boldsymbol{\theta}, \boldsymbol{\theta'})$ of the Markov chain constructed by the MH algorithm. Our distribution of interest is the posterior distribution, which proportional to the product of the likelihood and the prior, i.e. $\pi(\boldsymbol{x^*}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, so for convenience, we will let $\phi(\boldsymbol{\theta}) = \pi(\boldsymbol{x^*}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$. The posterior distribution can then be expressed as $\pi(\boldsymbol{\theta}|\boldsymbol{x^*}) = \mathcal{C}\phi(\boldsymbol{\theta})$, where $\mathcal{C} = \int_{\forall \boldsymbol{\theta}} \pi(\boldsymbol{x^*}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}$ is the normalising constant for the posterior distribution.

The MH algorithm can be viewed as two Markov chains coupled together to form one new Markov chain. The two Markov chains in play here are associated with the new move proposal step and the acceptance-rejection step, and their transition density are $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ and $\min\left\{1, \dfrac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i})\pi(\boldsymbol{x^*}|\boldsymbol{\theta'})\pi(\boldsymbol{\theta'})}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'})\pi(\boldsymbol{x^*}|\boldsymbol{\theta_i})\pi(\boldsymbol{\theta_i})}\right\}$ for accepting the proposed move respectively.

Let $P(\boldsymbol{a}, \boldsymbol{b})$ denote the transition density of the MH algorithm, where $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{\Theta}$. To establish the transition density of the MH algorithm, we want to show that $P(\boldsymbol{a}, \boldsymbol{b})$ is indeed ergodic and detailed balanced with the stationary distribution being $\phi(\boldsymbol{\theta})$. We will consider $P(\boldsymbol{\theta}, \boldsymbol{\theta'})$ for $\boldsymbol{\theta'} \neq \boldsymbol{\theta}$ and $P(\boldsymbol{\theta}, \boldsymbol{\theta})$. We start with the case of $P(\boldsymbol{\theta}, \boldsymbol{\theta'})$. For $\boldsymbol{\theta} \neq \boldsymbol{\theta'}$:

$$\begin{aligned}
P(\boldsymbol{\theta}, \boldsymbol{\theta'}) =& f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta'}, \text{Accepted } \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
&+ f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta'}, \text{not Accepted } \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta'}, \text{Accepted } \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta'} | \text{Accepted } \boldsymbol{\theta'}, \boldsymbol{\theta}_t = \boldsymbol{\theta}) f(\text{Accepted} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& q(\boldsymbol{\theta}, \boldsymbol{\theta'}) \min \left(1, \frac{\phi(\boldsymbol{\theta'}) q(\boldsymbol{\theta'}, \boldsymbol{\theta})}{\phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'})} \right) \quad \text{by the MH algorithm} \\
=& q(\boldsymbol{\theta}, \boldsymbol{\theta'}) \min \left(1, \frac{\pi(\boldsymbol{\theta'} | \boldsymbol{x}^*) q(\boldsymbol{\theta'}, \boldsymbol{\theta})}{\pi(\boldsymbol{\theta} | \boldsymbol{x}^*) q(\boldsymbol{\theta}, \boldsymbol{\theta'})} \right). \qquad (2.3)
\end{aligned}$$

The detailed balance follows since:

$$\begin{aligned}
\phi(\boldsymbol{\theta}) P(\boldsymbol{\theta}, \boldsymbol{\theta'}) &= \phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'}) \min \left(1, \frac{\pi(\boldsymbol{\theta'} | \boldsymbol{x}^*) q(\boldsymbol{\theta'}, \boldsymbol{\theta})}{\pi(\boldsymbol{\theta} | \boldsymbol{x}^*) q(\boldsymbol{\theta}, \boldsymbol{\theta'})} \right) \\
&= \phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'}) \min \left(1, \frac{\phi(\boldsymbol{\theta'}) q(\boldsymbol{\theta'}, \boldsymbol{\theta})}{\phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'})} \right) \\
&= \min \left(\phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'}), \phi(\boldsymbol{\theta'}) q(\boldsymbol{\theta'}, \boldsymbol{\theta}) \right) \\
&= \phi(\boldsymbol{\theta'}) q(\boldsymbol{\theta'}, \boldsymbol{\theta}) \min \left(1, \frac{\phi(\boldsymbol{\theta}) q(\boldsymbol{\theta}, \boldsymbol{\theta'})}{\phi(\boldsymbol{\theta'}) q(\boldsymbol{\theta'}, \boldsymbol{\theta})} \right) \\
&= \phi(\boldsymbol{\theta'}) P(\boldsymbol{\theta'}, \boldsymbol{\theta})
\end{aligned}$$

Next, by choosing an appropriate proposal distribution $q(\cdot, \cdot)$ which allows us to get from anywhere to anywhere (irreducible), we can ensure ergodicity. We will now look at $P(\boldsymbol{\theta}, \boldsymbol{\theta})$.

$$
\begin{aligned}
P(\boldsymbol{\theta}, \boldsymbol{\theta}) =& f(\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& f(\boldsymbol{\theta'} = \boldsymbol{\theta}, \text{Accepted } \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
&+ f(\boldsymbol{\theta'} \neq \boldsymbol{\theta}, \text{Rejected } \boldsymbol{\theta'} | \boldsymbol{\theta}_t = \boldsymbol{\theta}) \\
=& q(\boldsymbol{\theta}, \boldsymbol{\theta}) \min \left\{ 1, \frac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i}) \pi(\boldsymbol{x^*} | \boldsymbol{\theta'}) \pi(\boldsymbol{\theta'})}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'}) \pi(\boldsymbol{x^*} | \boldsymbol{\theta_i}) \pi(\boldsymbol{\theta_i})} \right\} \\
&+ \int_{\boldsymbol{\theta'} \neq \boldsymbol{\theta}} q(\boldsymbol{\theta}, \boldsymbol{\theta'}) \left(1 - \min \left\{ 1, \frac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i}) \pi(\boldsymbol{x^*} | \boldsymbol{\theta'}) \pi(\boldsymbol{\theta'})}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'}) \pi(\boldsymbol{x^*} | \boldsymbol{\theta_i}) \pi(\boldsymbol{\theta_i})} \right\} \right) d\boldsymbol{\theta'}.
\end{aligned}
\tag{2.4}
$$

In this case, it is trivial to show that the detailed balance is satisfied:

$$\phi(\boldsymbol{\theta}) P(\boldsymbol{\theta}, \boldsymbol{\theta}) = P(\boldsymbol{\theta}, \boldsymbol{\theta}) \phi(\boldsymbol{\theta}),$$

and the ergodicity is also ensured by irreducible $q(\cdot, \cdot)$. Therefore, we have established that the transition density, $P(\cdot, \cdot)$, of the MH algorithm does indeed satisfies the detailed balance and ergodicity with the desired stationary distribution, $\phi(\boldsymbol{\theta})$. □

We can replace the likelihood $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ in the MH algorithm with an unbiased and non-negative estimator $\widehat{\pi(\boldsymbol{x}|\boldsymbol{\theta})}$ and still get a sample from the posterior distribution [Andrieu and Roberts, 2009], which gives us the MCMC-ABC algorithm in Algorithm 7.

Algorithm 7 (MCMC-ABC)**.**

1. Let $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ be the proposal distribution. $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ gives the density of moving from θ to θ'
2. Initiate $\boldsymbol{\theta}_1$ using the rsABC algorithm so that $\{d\big(\boldsymbol{S(x)}, \boldsymbol{S(x^*)}\big) \leq h\}$
3. Propose a move from $\boldsymbol{\theta_i}$ to $\boldsymbol{\theta'}$
4. Set $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta'}$ with probability $\min \left\{ 1, \dfrac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i}) \pi(\boldsymbol{\theta'}) \widehat{\pi(\boldsymbol{x^*}|\boldsymbol{\theta'})}}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'}) \pi(\boldsymbol{\theta_i}) \widehat{\pi(\boldsymbol{x^*}|\boldsymbol{\theta})}} \right\}$, or else $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i}$
5. Repeat 3. to 4. until desired sample size is reached

where, $S(\cdot)$ denote some summary statistics. Let $\boldsymbol{x'}$ denote some data generated form $\pi(\boldsymbol{x}|\boldsymbol{\theta'})$. In step 4., if we replace $\frac{\widehat{\pi(\boldsymbol{x^*}|\boldsymbol{\theta'})}}{\widehat{\pi(\boldsymbol{x^*}|\boldsymbol{\theta})}}$ with $\frac{\mathbf{1}_{\left\{d\left(\boldsymbol{S(x')},\boldsymbol{S(x^*)}\right)\leq h\right\}}}{\mathbf{1}_{\left\{d\left(\boldsymbol{S(x^*)},\boldsymbol{S(x')}\right)\leq h\right\}}}$, then we will get samples from an approximate posterior.

Justification. We have shown that Algorithm 7 will generate a sample with probability density function (PDF) proportional to

$$\pi(\boldsymbol{\theta})\mathbf{1}_{\left\{d\left(\boldsymbol{S(x^*)},\boldsymbol{S(x')}\right)\leq h\right\}}$$

in the earlier proof for the MH algorithm, and from Section 2.3, we know that such a sample is indeed a sample from the approximate posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x^*})$. □

2.7 Semi-automatic ABC

All the algorithms discussed so far require choosing summary statistics that are reasonable approximation of the sufficient statistics, which can be a challenging task when the model is complex and low order sufficient statistics cannot be identified. The Semi-automatic ABC algorithm discussed in [Fearnhead and Prangle, 2012] systemises the summary statistics selection somewhat by transforming the problem of selecting "appropriate" summary statistics into a more standard model selection problem.

[Fearnhead and Prangle, 2012] suggests that instead of using well approximated (to the sufficient statistics) summary statistics, they construct a vector valued regression model based on a larger set of (possibly) less well approximated summary statistics, for which the summary statistics are used as independent variables in the model and the dependent variable is used as the new summary statistics.

This is an algorithm that improves on the comparison stage of the algorithm.

2.8 Piecewise ABC

The piecewise ABC algorithm or pwABC algorithm suggested by [White et al., 2013], is a comparatively efficient algorithm for analysing discretely observed data with the Markov property, i.e. $\pi(x_i|x_1, \ldots, x_{i-1}) = \pi(x_i|x_{i-1})$.

The main innovation of the pwABC algorithm is that it utilises the Markov property to factorise the posterior distribution of the whole observed data into products of posterior distributions of connected data pairs. If there are n observations, then the full posterior distribution will be factorised into $n-1$ factors. After the factorisation stage, each of the posterior factors can be treated as independent problem given the observed data and any of the ABC sampling technique can be adopted. rsABC algorithm was used in [White et al., 2013]. The process is then repeated for each posterior factor and at the end of the process, a set of accepted parameters will be collected. To estimate the overall posterior, a kernel smoothing method is performed on the parameter estimates from each sample.

The major benefit of the factorisation is that it greatly increases the acceptance probability at each sub problem. The higher acceptance probability also means that the acceptance tolerance, h, can be reduced and in some cases circumvent the need to use lower order summary statistics.

The advantage of pwABC algorithm is that it only requires the likelihood to be factorisable with respect to the data, and [White et al., 2013] demonstrated in the paper that a broad class of models fit in this framework including: stochastic differential equation model, autoregressive time-series mode, and dynamical predator-prey model.

2.9 ABC with regression correction

A major drawback of the standard rsABC algorithm (Section 2.3) is that the efficiency of the algorithm decreases rapidly as the number of summary statistics increases. As the number of summary statistics increases, we are forced to increase the tolerance, h, otherwise the acceptance rate will be prohibitively low.

In [Beaumont et al., 2002], a regression correction step is added to the rsABC algorithm which allows it to run with a more relaxed choice of the tolerance h but with a correction afterwards. In order to implement the regression correction step, the rsABC algorithm is modified to record both the accepted values, $\boldsymbol{\theta_i}$ as well as its corresponding summary statistics, $\boldsymbol{s_i} = \boldsymbol{S}(\boldsymbol{x_i})$. The regression correction step assumes a local-linear relationship in the vinivity of $\boldsymbol{s^*}$. The correction uses the discrepancy between the sample summary statistics and the observed summary statistics, i.e. $|\boldsymbol{s_i} - \boldsymbol{s^*}|$, as weights to construct a local-linear regression model.

2.10 Monte Carlo within Metropolis algorithm

MCWM algorithm [O'Neill et al., 2000] is an algorithm closely related to the MCMC-ABC algorithm. The key difference between the two algorithm is that the MCWM algorithm uses data augmentation and importance sampling algorithm to approximate the likelihood $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)|\boldsymbol{\theta}\big)$, which is otherwise difficult to calculate. Let $\boldsymbol{Y}$ denote some latent random variable, such that $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y}, \boldsymbol{\theta}\big)$ is easily computable for some value of $\boldsymbol{Y}$. Probability theory tells us that:

$$\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{\theta}\big) = \int_{\forall \boldsymbol{y}} \pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y}, \boldsymbol{\theta}\big)\pi\big(\boldsymbol{y}\big|\boldsymbol{\theta}\big)d\boldsymbol{y}.$$

If $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y}, \boldsymbol{\theta}\big)$ is computable for any $\boldsymbol{Y}$, then the estimation of $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{\theta}\big)$ can be achieved by applying Monte Carlo integration. If $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y}, \boldsymbol{\theta}\big)$ is

only computable for a specific set of $\boldsymbol{Y}$, then we can use the importance sampling algorithm to ensure that we only sample from these computable values. Given $\boldsymbol{\theta}$, importance sampling algorithm allows to us to estimate the likelihood as below:

$$\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta}\big) = \frac{1}{N}\sum_{i=1}^{N}\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y_i},\boldsymbol{\theta}\big)\frac{\pi\big(\boldsymbol{y_i}\big|\boldsymbol{\theta}\big)}{r(\boldsymbol{y_i}|\boldsymbol{\theta})} \tag{2.5}$$

where $r(\boldsymbol{y})$, is the density function of the importance sampling proposal distribution with respect to $\boldsymbol{y}$. In some cases, by choosing the proposal distribution carefully, we can ensure that $\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)\big|\boldsymbol{y_i},\boldsymbol{\theta}\big) = 1$. The quantity $\frac{\pi(\boldsymbol{y_i}|\boldsymbol{\theta})}{r(\boldsymbol{y_i}|\boldsymbol{\theta})}$ is the importance sampling weight, which is also the measurement of the amount of "steering". The closer the importance sampling weight is to 1, the less the steering. This idea of data augmentation and importance sampling play a crucial role in this thesis, and we will explore this further in Chapter 3. Below is the MCWM algorithm.

Algorithm 8 (MCWM)**.**

1. Let $q(\boldsymbol{\theta},\boldsymbol{\theta'})$ be the proposal distribution. $q(\boldsymbol{\theta},\boldsymbol{\theta'})$ gives the density of moving from $\boldsymbol{\theta}$ to $\boldsymbol{\theta'}$
2. Initiate $\boldsymbol{\theta}_1$ with an arbitrary number, and ideally from the posterior distribution if possible.
3. Propose a move from $\boldsymbol{\theta_i}$ to $\boldsymbol{\theta'}$ according to $q(\boldsymbol{\theta_i},\boldsymbol{\theta'})$
4. Generate data $\{\boldsymbol{x'_1},\ldots,\boldsymbol{x'_N}\}$ from $r(\boldsymbol{y}|\boldsymbol{\theta'})$.
5. Calculate $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta_i}\big)$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta'}\big)$ according to (2.5).
6. Set $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta'}$ with probability $\min\left\{1, \dfrac{q(\boldsymbol{\theta'},\boldsymbol{\theta_i})\pi(\boldsymbol{\theta'})\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta'}\big)}{q(\boldsymbol{\theta_i},\boldsymbol{\theta'})\pi(\boldsymbol{\theta_i})\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta_i}\big)}\right\}$, or else $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i}$.
7. Repeat 3. to 6. until desired sample size is reached.

Often steps 4. and 5. are done as one single step because $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta_i}\big)$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta'}\big)$ need to be calculated iteratively.

Theoretical properties of the MCWM are well studied in [Andrieu and Roberts, 2009, Medina-Aguayo et al., 2015, Alquier et al., 2016]. The MCWM algo-

rithm is also referred to as the noisy MH algorithm in [Medina-Aguayo et al., 2015]. Because both $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta}_i\big)$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta}'\big)$ are re-estimated at each iteration, which means that they are independently samples between iterations, the resulting Markov chain is still $\{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \boldsymbol{\theta_3}, \ldots\}$. However, the target distribution of the MCWM, when exists, is

$$\pi(\boldsymbol{\theta})\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x}^*)}\big|\boldsymbol{\theta}\big),$$

i.e. the target invariant density is an approximation of the posterior up to a constant proportionality. An appropriate choice of the importance sampling distribution is required for MCWM to inherent the ergodicity and the detailed balance and therefore to have an invariant distribution, see [Medina-Aguayo et al., 2015] for details. Next, we will look at another closely related algorithm, Grouped Independence Metropolis Hastings algorithm.

2.11 Grouped independence Metropolis Hastings algorithm

The GIMH algorithm is firstly introduced in [Beaumont, 2003]. Below is the general GIMH algorithm:

Algorithm 9 (GIMH)**.**

1. Let $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ be the proposal distribution. $q(\boldsymbol{\theta}, \boldsymbol{\theta'})$ gives the density of moving from $\boldsymbol{\theta}$ to $\boldsymbol{\theta'}$
2. Initiate $\boldsymbol{\theta}_1$ with an arbitrary number, and ideally from the posterior distribution if possible. Calculate $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta}\big)$ according to (2.5).
3. Propose a move from $\boldsymbol{\theta_i}$ to $\boldsymbol{\theta'}$ according to $q(\boldsymbol{\theta_i}, \boldsymbol{\theta'})$
4. Generate data $\{\boldsymbol{x'_1}, \ldots, \boldsymbol{x'_N}\}$ from $r(\boldsymbol{x}|\boldsymbol{\theta'})$.
5. Calculate $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta'}\big)$ according to (2.5).
6. Set $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta'}$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big) = \pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta'}\big)$ with probability $\min\left\{1, \dfrac{q(\boldsymbol{\theta'}, \boldsymbol{\theta_i})\pi(\boldsymbol{\theta'})\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta'}\big)}{q(\boldsymbol{\theta_i}, \boldsymbol{\theta'})\pi(\boldsymbol{\theta_i})\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big)}\right\}$, or else set $\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i}$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_{i+1}}\big) = \pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big)$.
7. Repeat 3. to 6. until desired sample size is reached.

In MCWM, $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta}\big)$ and $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta'}\big)$ are calculated using new $\{\boldsymbol{x'_1}, \ldots, \boldsymbol{x'_N}\}$ at each iteration; in the GIMH algorithm, $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big)$ is inherited from the previous iteration. If $\boldsymbol{\theta'}$ is accepted, then we set $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_{i+1}}\big) = \pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta'}\big)$, otherwise $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_{i+1}}\big) = \pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big)$. This give a clear advantage of GIMH over MCWM is the computation time, because the likelihood approximation step is usually the most computationally intensive.

Both MCWM and GIMH are special cases of pseudo-marginal methods.GIMH have been studied in some depth usually under the umbrella of pseudo marginal methods, see [Andrieu and Vihola, 2014, Andrieu and Roberts, 2009, Andrieu et al., 2015, Beaumont, 2003]. For notation simplicity, we let $w_i = \pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta_i}\big)$. Because in GIMH we recycle w_i between iterations, the resulting Markov chain is in the pairs of $\{\boldsymbol{\theta_i}, w_i\}$ for $i = 1, 2, 3, \ldots$. The remarkable property of GIMH is that the target invariant distribution is in fact the exact posterior, provided that $\pi\big(\widehat{\boldsymbol{S}(\boldsymbol{x^*})}\big|\boldsymbol{\theta}\big)$ is an unbiased and non-negative estimator of the likelihood, see [Andrieu and Roberts, 2009]. The GIMH algorithm is computationally advantageous, but vulnerable to being

stuck, see [Medina-Aguayo et al., 2015]. In practice, often more computation power are given to sampling through $\boldsymbol{\theta}$'s than estimating w_i at a particular θ, which means that the estimation can have a large Monte Carlo error. If by chance that an w_i is large, then the algorithm is very likely to reject most most $\boldsymbol{\theta}$'s afterwards, and therefore the Markov chain gets stuck in one place. However, this is less of a problem in MCWM because at each iteration w_i and w' are calculated afresh.

Chapter 3

Data conditioned simulation methodology

In this chapter we introduce the core idea of the thesis, *data conditioned simulation*. Most of the ABC algorithms discussed so far have their focus on either the exploration of the parameter space or the acceptance criterion stage (Stages 1 and 3 in Section 2.4). Here we explore a selective approach to the likelihood simulation stage (Stage 2). In particular, we control the simulation of the likelihood dynamically depending on the data sampled so that we can steer the sampled data to closely match the observed data. We first introduce what we term, the data conditioned simulation. We then introduce the data conditioned ABC (dcABC) algorithm, which is in essence the rsABC algorithm in Section 2.3, but modified to incorporate the data conditioned simulation. Two toy examples are used to illustrate the data conditioned simulation and dcABC algorithm in practice. Three applications of the algorithm in a homogeneous mixing SIR epidemic model, a time-inhomogeneous Markov model, and a stochastic population model, are discussed later in Chapters 4, 5 and 6 respectively.

3.1 Data conditioned simulation

The data conditioned simulation is a method of reaching an estimate for the likelihood through steered simulation. We steer the simulation to ensure that the outcome of the simulation matches closely, if not exactly, with the observed data through the application of data augmentation and importance sampling. Let $\boldsymbol{x}^*$ denote some observed data from a model parameterised by $\boldsymbol{\theta}$ with an intractable likelihood $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$.

Suppose that there exists some latent random variables $\boldsymbol{Y}$ with density $\pi(\boldsymbol{y}|\boldsymbol{\theta})$, such that if we have a realisation $\boldsymbol{y}$ from $\boldsymbol{Y}$, then $\pi(\boldsymbol{x}^*|\boldsymbol{y},\boldsymbol{\theta})$ is tractable. This allows us to apply the data augmentation algorithm:

$$\pi\big(\boldsymbol{x}^*|\boldsymbol{\theta}\big) = \int \pi\big(\boldsymbol{x}^*|\boldsymbol{y},\boldsymbol{\theta}\big)\pi(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{y} \tag{3.1}$$

Applying Monte Carlo integration, (3.1) can be approximated by

$$\frac{1}{k}\sum_{i=1}^{k} \pi\big(\boldsymbol{x}^*|\boldsymbol{y_i},\boldsymbol{\theta}\big), \tag{3.2}$$

where k is some positive integer. As $k \to \infty$, (3.2) converges to $\pi\big(\boldsymbol{x}^*|\boldsymbol{\theta}\big)$.

The simulation step in EBC and rsABC algorithm can be seen as a special case of this framework, where we treat the outcome of the model under consideration as the "latent" variable. Given $\boldsymbol{\theta}$, we simulate some data, $\boldsymbol{y}$, from the model, then $\pi\big(\boldsymbol{x}^*|\boldsymbol{y},\boldsymbol{\theta}\big) = 1_{\{\boldsymbol{y}=\boldsymbol{x}^*\}}$. $1_{\{\boldsymbol{y}=\boldsymbol{x}^*\}}$ is an indicator function such that if $\boldsymbol{y} = \boldsymbol{x}^*$ then it returns 1, otherwise 0. We can do better than waiting for a match between $\boldsymbol{y}$ and $\boldsymbol{x}^*$ by adding an importance sampling step which ensures a match and therefore that $1_{\{\boldsymbol{y}=\boldsymbol{x}^*\}}$ is always 1.

The importance sampling step added here is to ensure that when we simulate from the model, we only simulate from the region where the simulated data matches the observed data. Suppose that we can find such a proposal distribution for the importance sampling. We denote this proposal distribution

as $q(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta})$, then:

$$\begin{aligned}\pi\big(\boldsymbol{x}^*|\boldsymbol{\theta}\big) &= \int \pi\big(\boldsymbol{x}^*|\boldsymbol{y}, \boldsymbol{\theta}\big)\pi(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{y} \\ &= \int \pi\big(\boldsymbol{x}^*|\boldsymbol{y}, \boldsymbol{\theta}\big)\frac{\pi(\boldsymbol{y}|\boldsymbol{\theta})}{q\big(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)}q\big(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)d\boldsymbol{y} \\ &= \int 1 \times \frac{\pi(\boldsymbol{y}|\boldsymbol{\theta})}{q\big(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)}q\big(\boldsymbol{y}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)d\boldsymbol{y}. \end{aligned} \tag{3.3}$$

Again, by Monte Carlo integration, we can approximate (3.3) by

$$\frac{1}{k}\sum_{i=1}^{k}\frac{\pi(\boldsymbol{y_i}|\boldsymbol{\theta})}{q\big(\boldsymbol{y_i}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)}, \tag{3.4}$$

where k is some positive integer.

$q\big(\boldsymbol{y_i}|\boldsymbol{x}^*, \boldsymbol{\theta}\big)$ is essentially the truncated density of $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ over a smaller region. It is the flexibility in the choice of $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ and the definition of the truncated region that allows us to "steer" the simulation to match the observed data and perform data conditioned simulation. The problem of choosing $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ and the truncated region is problem specific and it will be discussed in detail for each case. The likelihood estimation can be calculated using (3.5). The same argument can be applied when we substitute the raw observed data, $\boldsymbol{x}^*$, with some summary statistics $\boldsymbol{S}(\boldsymbol{x}^*)$, and therefore the likelihood of the summary statistics can be estimated using the following:

$$\pi\big(\boldsymbol{S}(\boldsymbol{x}^*)|\boldsymbol{\theta}\big) = \frac{1}{k}\sum_{i=1}^{k}\frac{\pi(\boldsymbol{y_i}|\boldsymbol{\theta})}{q\big(\boldsymbol{y_i}|\boldsymbol{S}(\boldsymbol{x}^*), \boldsymbol{\theta}\big)}. \tag{3.5}$$

Even though for large k, we can achieve a better likelihood estimate, in practice, often k is taken to be 1.

3.2 Data conditioned ABC (dcABC) Algorithm

Let $\boldsymbol{x}^*$ denote some observed data from a model parameterised by $\boldsymbol{\theta}$ with an intractable likelihood $\pi(\boldsymbol{x}^*|\boldsymbol{\theta})$. The augmented random variable $\boldsymbol{Y}$ is defined as in section 3.1. We wish to estimate the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)$. The dcABC algorithm produces a weighted sample from the posterior distribution, which we can use to estimate posterior statistics using a weighted mean. We now use the data conditioned likelihood estimation to help us define a generic dcABC algorithm below.

Algorithm 10 (dcABC algorithm)**.**

1. Set $i = 1$.
2. Sample $\boldsymbol{\theta_i}$ from the prior $\pi(\boldsymbol{\theta})$.
3. Sample k $\boldsymbol{y}$'s from $\boldsymbol{Y}|\boldsymbol{\theta_i}, \boldsymbol{S(x^*)}$, the model sample space which satisfies the summary statistics.
4. Let $p_i = \dfrac{1}{k}\displaystyle\sum_{j=1}^{k} \dfrac{\pi(\boldsymbol{y_j}|\boldsymbol{\theta_i})}{q\big(\boldsymbol{y_j}|\boldsymbol{S(x^*)}, \boldsymbol{\theta_i}\big)}$.
5. Record $(\boldsymbol{\theta_i}, p_i)$.
6. Increment i and repeat 2. to 5. until desired number of samples.

The weighted sample is used to estimate statistics of interest using a weighted mean. For example, let $\lambda_\Phi = \mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}^*}\big[\Phi(\boldsymbol{\theta})\big]$ and $var(\lambda_\Phi)$ is finite, then we estimate λ_Φ by

$$\widehat{\lambda}_\Phi = \frac{\sum_{i=1}^{m} \Phi(\boldsymbol{\theta}_i) p_i}{\sum_{i=1}^{m} p_i}. \tag{3.6}$$

We will show that the weighted mean yields a consistent estimator for the statistic of interest with respect to the posterior distribution. Consider

$\mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}^*}\big[\Phi(\boldsymbol{\theta})\big]$:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}^*}[\Phi(\boldsymbol{\theta})] &= \int \Phi(\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{x}^*)d\boldsymbol{\theta} \\ &= \int \Phi(\boldsymbol{\theta})\frac{\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{x}^*)}d\boldsymbol{\theta} \\ &= \frac{\int \Phi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\int \pi(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &= \frac{\mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big]}{\mathbb{E}_{\boldsymbol{\theta}}\big[\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big]}. \end{aligned} \tag{3.7}$$

Returning to (3.6), we multiply both numerator and denominator of (3.6) by $\frac{1}{m}$, where m is the sample size, we have

$$\widehat{\lambda}_{\Phi} = \frac{\frac{1}{m}\sum_{i=1}^{m}\Phi(\boldsymbol{\theta}_i)p_i}{\frac{1}{m}\sum_{i=1}^{m}p_i}. \tag{3.8}$$

We want to establish that the numerator and the denominator of (3.8) are unbiased and consistent estimators of the numerator and the denominator of (3.7) respectively.

For the unbiasedness of the numberator, we look at the expectation of the numerator of the weighted mean. Since $\boldsymbol{\theta_i}$'s are identically independently distributed, we have $\mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta}_1)p_1\big] = \mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta}_2)p_2\big] = \ldots = \mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta}_m)p_m\big]$. From section 3.1, we know that $p_i \approx \pi(\boldsymbol{x}^*|\boldsymbol{\theta_i})$, and therefore:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}}\left[\frac{1}{m}\sum_{i=1}^{m}\Phi(\boldsymbol{\theta}_i)p_i\right] &= \mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta}_1)\pi(\boldsymbol{x}^*|\boldsymbol{\theta}_1)\big] \\ &= \mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big].\end{aligned}$$

The above demonstrates that the numerator of the weighted mean is an unbiased estimator.

For consistency, we will look at the variance,

$$\begin{aligned} var\left(\frac{1}{m}\sum_{i=1}^{m}\Phi(\boldsymbol{\theta}_i)p_i\right) &= \frac{1}{m^2}\sum_{i=1}^{m} var\big(\Phi(\boldsymbol{\theta}_i)p_i\big) \\ &= \frac{1}{m} var\big(\Phi(\boldsymbol{\theta}_1)p_1\big) \\ &= \frac{1}{m}\Big(\mathbb{E}\big[\Phi(\boldsymbol{\theta}_1)^2 p_1^2\big] - \mathbb{E}\big[\Phi(\boldsymbol{\theta}_1)p_1\big]^2\Big). \end{aligned}$$

Then since $var(\lambda_\Phi)$ is assumed finite, we have

$$\frac{1}{m}\Big(\mathbb{E}\big[\Phi(\boldsymbol{\theta}_1)^2 p_1^2\big] - \mathbb{E}\big[\Phi(\boldsymbol{\theta}_1)p_1\big]^2\Big) \to 0 \text{ as } m \to \infty.$$

$$\Rightarrow var\left(\frac{1}{m}\sum_{i=1}^{m}\Phi(\boldsymbol{\theta}_i)p_i\right) \to 0 \text{ as } m \to \infty.$$

That means $\frac{1}{m}\sum_{i=1}^{m}\Phi(\boldsymbol{\theta}_i)p_i$ is a consistent (and unbiased) estimator of the numerator in (3.7). We will treat $\frac{1}{m}\sum_{1}^{m} p_i$ similarly. Again, we first look at the expected value of the denominator:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}}\left[\frac{1}{m}\sum_{i=1}^{m} p_i\right] &= \mathbb{E}_{\boldsymbol{\theta}}[p_1] \\ &= \mathbb{E}_{\boldsymbol{\theta}}\big[\pi(\boldsymbol{x}^*)|\boldsymbol{\theta}\big]. \end{aligned}$$

i.e. the denominator is an unbiased estimator of $\pi(\boldsymbol{x}^*)|\boldsymbol{\theta}$. Next, we will examine the variance of the denominator:

$$\begin{aligned} var\left(\frac{1}{m}\sum_{1}^{m} p_i\right) &= \frac{1}{m^2}\sum_{i=1}^{m} var(p_i) \\ &= \frac{1}{m} var(p_1) \end{aligned}$$

Since $p_1 \in [0,1]$, $var(p_1)$ is also bounded and therefore:

$$\frac{1}{m} var(p_1) \to 0 \text{ as } m \to \infty$$

$$\Rightarrow var\left(\frac{1}{m}\sum_{i=1}^{m} p_i\right) \to 0 \text{ as } m \to \infty.$$

$\frac{1}{m}\sum_1^m p_i$ is therefore a consistent (and unbiased) estimator of $\mathbb{E}_{\boldsymbol{\theta}}\big[\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big]$.

Hence:

$$\frac{\frac{1}{m}\sum_{1}^{m}\Phi(\boldsymbol{\theta}_i)p_i}{\frac{1}{m}\sum_{i}^{m}p_i} = \frac{\sum_{i}^{m}\Phi(\boldsymbol{\theta}_i)p_i}{\sum_{i}^{m}p_i}$$

$$\rightarrow \frac{\mathbb{E}_{\boldsymbol{\theta}}\big[\Phi(\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big]}{\mathbb{E}_{\boldsymbol{\theta}}\big[\pi(\boldsymbol{x}^*|\boldsymbol{\theta})\big]} \text{ as } m \rightarrow \infty$$

$$= \mathbb{E}_{\boldsymbol{\theta}|\boldsymbol{x}^*}\big[\Phi(\boldsymbol{\theta})\big] \text{ by (3.7)}$$

In order to sample from $\boldsymbol{Y}|\boldsymbol{\theta_i}, \boldsymbol{S(x^*)}$ so that it matches the observed data, we need to be able to find the appropriate proposal distribution $q(\cdot)$. We were able to find the appropriate $q(\cdot)$, so that $\pi\big(\boldsymbol{S(x^*)}\big|\boldsymbol{y}, \boldsymbol{\theta}\big) = 1$ for all the examples in this thesis. However, it is possible that the augmented data allows us to calculate $\pi\big(\boldsymbol{S(x^*)}\big|\boldsymbol{y}, \boldsymbol{\theta}\big)$ explicitly, see [Neal and Huang, 2015]. The link between the steering mechanism and the importance sampling proposal distribution is demonstrated in sections 3.4.2 and 3.3.2 with relatively simple proposal distributions. The idea is easily generalised to accommodate for a more complex proposal distribution. We also see that the total amount of steering is quantified by the importance sampling weight. The exact mechanism of steering the simulation is problem dependent, and varies greatly from problem to problem. However, the general approach is to use the conditional distribution of the underlying model, for which, sometimes it involves introducing further latent variables to the model. In practice, because of the nature of the conditioning we placed, the conditional distributions used are the truncated versions of the unconditional distributions. This will become apparent in the toy examples. Steps 3. and 4. are often one single step as the sample of $\boldsymbol{y}$ from $\boldsymbol{Y}|\boldsymbol{\theta_i}, \boldsymbol{S(x^*)}$ are done iteratively. We can further generalise the algorithm by sampling $\boldsymbol{\theta_i}$ from a probability density$r(\boldsymbol{\theta})$ other than the prior with attached weight $\frac{\pi(\boldsymbol{\theta})}{r(\boldsymbol{\theta})}$ at Step 2.

3.3 Example 1: Poisson distribution

Here we revisit the toy example in Section 2.2.3 but modifying the algorithm to fit the dcABC framework. This is an example where the likelihood can be calculated easily without requiring a data augmentation step, but it is useful to demonstrate the idea of the weighted samples of the posterior distribution. Consider a data set, $\boldsymbol{x}^*$, consisting of 10 observations from a Poisson distribution, $Poisson(\theta^*)$, with $\theta^* = 5$.

Given that:

$$\boldsymbol{x}^* = \{2, 3, 5, 6, 7, 7, 7, 8, 9, 12\}$$

Suppose θ has the prior distribution $\Gamma(1, 0.1)$.

3.3.1 Data condition simulation

Although the likelihood can be calculated directly here, we can still implement the data conditioned simulation. Given any θ, the likelihood can be calculated as in section 2.3:

$$\begin{aligned}\mathbb{P}(\boldsymbol{x} = \boldsymbol{x}^*|\theta) &= \frac{10!}{3!}\prod_{i=1}^{10}\mathbb{P}(x_i = x_i^*|\theta) \\ &= \frac{10!}{3!}\frac{\theta^{\sum_{i=1}^{10} x_i^*}e^{-10\theta}}{\prod_{i=1}^{10} x_i^*!}.\end{aligned}$$

We can use this to compare the likelihoods estimate from the data conditioned simulation.

In order to implement the data conditioned simulation, we need to decide on two things: $\boldsymbol{Y}$ the data to be augmented, and $q(\boldsymbol{y}|\boldsymbol{x}^*, \theta)$ the importance proposal density. In this scenario, $\boldsymbol{Y}$ is set to be the output of the model itself, which is 10 Poisson realisations. We want to design $q(\boldsymbol{y}|\boldsymbol{x}^*, \theta)$ such

that we can ensure that $\boldsymbol{y} = \boldsymbol{x}^*$. To achieve this, we will utilise a series of truncated Poisson distributions.

We start by defining regions of importance, which we will restrict the sampling over these "important" region only. Let $\mathcal{A}_1 = \{2, 3, 5, 6, 7, 8, 9, 12\}$, a set contains only unique members of $\boldsymbol{x}^*$, then $Y_1|\mathcal{A}_1 \sim Poisson(\theta)|Y_1 \in \mathcal{A}_1$ and the corresponding density function is defined as:

$$f_{Y_1|\mathcal{A}_1}(y) = \begin{cases} \frac{\frac{\theta^y e^{-\theta}}{y!}}{\sum_{\forall z \in \mathcal{A}_1} \frac{\theta^z e^{-\theta}}{z!}} & \text{if } y \in \mathcal{A}_1 \\ 0 & \text{otherwise} \end{cases}$$

where z is a dummy variable for summation.

Let $\mathcal{A}_i$ denote subsequent importance region, then $\mathcal{A}_i = \boldsymbol{x}^* \setminus \{y_1, \ldots, y_{i-1}\}$ and $Y_i|\mathcal{A}_i \sim Poisson(\theta)|Y_i \in \mathcal{A}_i$. We can now write down the importance proposal density as below:

$$q(\boldsymbol{y}|\boldsymbol{x}^*, \theta) = \prod_{i=1}^{10} f_{Y_i|\mathcal{A}_i}(y_i).$$

Using (3.3) setting $k = 1$, we can write down the data conditioned likelihood estimate as:

$$\pi\big(\boldsymbol{x}^*|\theta\big) = \prod_{i=1}^{10} \sum_{\forall x \in \mathcal{A}_i} \frac{\theta^x e^{-\theta}}{x!}, \tag{3.9}$$

i.e. the product of the normalising factors of the truncated Poisson distributions.

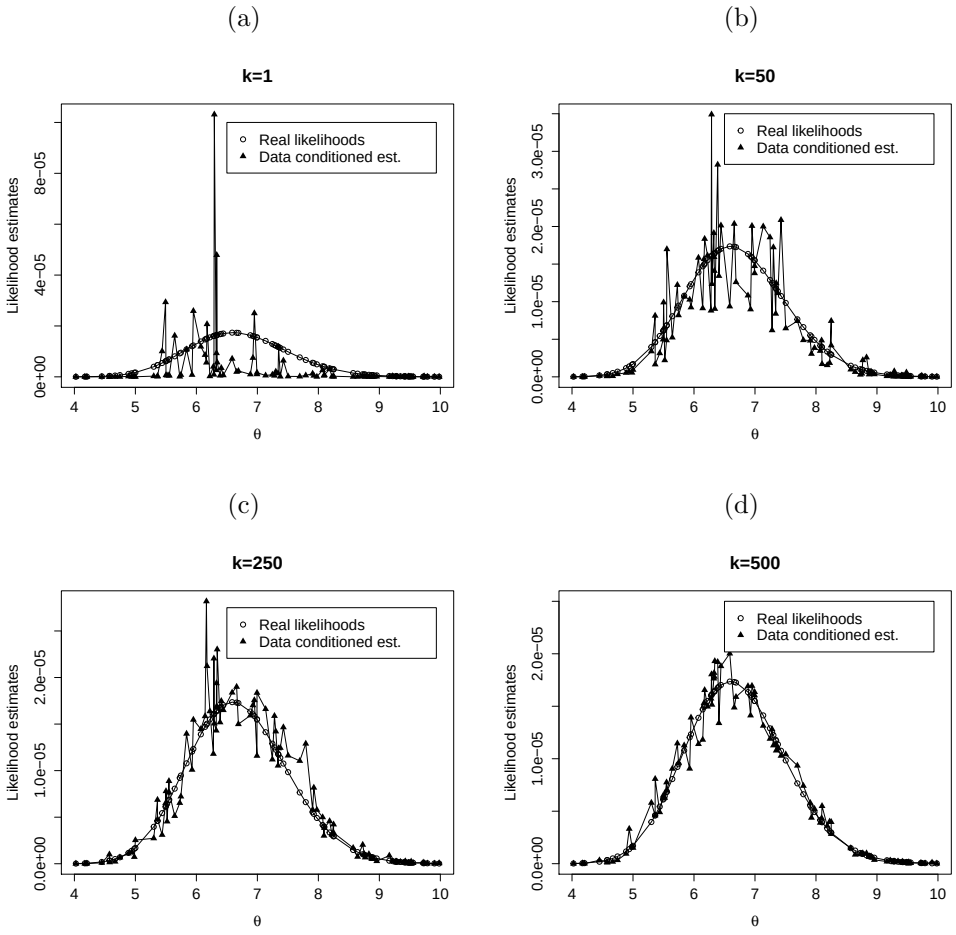
The data conditioned simulation is now straightforward. There are 10 observations in total, and therefore Y_i's are defined for $i = 1, \ldots, 10$. The data conditioned simulation is as follow:

Algorithm 11 (Data conditioned simulation for the Poisson example)**.**

1. For $i = 1, \ldots, 10$.
2. Sample a y_i from $Poisson(\theta)|Y_i \in \mathcal{A}_i$.
3. Calculate the likelihood estimate using (3.9).
4. Repeat steps 1. to 3. k times.
5. For the data conditioned simulation estimate, we simply take the mean of the k estimates.

Figure 3.1 show a comparison between the true likelihood and the data conditioned likelihood estimates. The figure is based on 100 θ's drawn from $U(4, 10)$, and $k = 1, 50, 250, 500$. The uniform distribution is used because it allows to compare the estimate evenly across the parameter space, whereas using the Gamma distribution will produce a more heavy tailed sample of θ's. We can observe that as k increases, the data conditioned estimate gets closer to the true likelihood with decreasing variability. Even though at $k = 1$ the data conditioned estimate does not resemble the true likelihood, the data conditioned estimates place most of the weights at the desired area (between 5 - 8). This allows us to still use the data conditioned estimates as weights when making inference later.

Figure 3.1: Comparison between the real likelihood and the data conditioned likelihood estimates.



3.3.2 dcABC algorithm

Armed with the data conditioned simulation in section 3.3.1, the dcABC algorithm is straightforward.

Algorithm 12 (dcABC algorithm with the full observed data)**.**

1. Set $i = 1$.
2. Sample a θ_i from $\Gamma(1, 0.1)$.
3. Calculate p_i, the likelihood estimate, using algorithm 11.
4. Record (θ_i, p_i).
5. If $i < m$, increment i by 1 and go to step 2, where m is the desired sample size.

Since we used the raw data in the data conditioned simulation with no tolerance for discrepency, the resulting algorithm gives a sample from the exact posterior distribution rather than the approximate posterior distribution. We can write down the posterior distribution as follow:

$$\pi(\theta|\boldsymbol{x}^*) = \Gamma\left(1 + \sum_{i=1}^{10} x^i, 0.1 + 10\right)$$

$$= \Gamma\left(67, 10.1\right).$$

3.4 Example 2: exponential distribution

The next example we will look at is a set of observations from an exponential distribution. We use exponential distribution to demonstrate how the data conditioned simulation is adapted to continuous random variables. We will also be using the sufficient statistic of the exponential distribution parameter in this example. Consider $\boldsymbol{x}^*$, a set of identically independently observed data from an exponentially distributed random variable $X \sim Exp(\theta^*)$, with $\theta^* = 1$. We wish to estimate θ given $\boldsymbol{x}^*$.

Suppose $\boldsymbol{x}^* = \{1.5334, 0.5060, 0.6447, 1.0254, 0.5944,$
$2.3562, 0.9453, 1.0464, 0.1113, 0.0440\}$, are 10 samples drawn from $Exp(1)$.

For the exponential distribution, we can identify the sufficient statistic to

be $T(\boldsymbol{x}^*) = \sum_{i=1}^{10} x_i^*$. Let t^* denote the sufficient statistic, and therefore $t^* = 8.8071$. We will follow a similar structure in this example to that in the section 3.3.2 and demonstrate how we can utilise the sufficient statistics in the data conditioned simulation. For which, we will first look at the data conditioned simulation to estimate the likelihood, and then utilise that to help us define the dcABC algorithm for the exponential model.

3.4.1 Data conditioned simulation

The aim for the data conditioned simulation for this example is to produce weighted samples such that they have the same sufficient statistic as $\boldsymbol{x}^*$. We know that the sufficient statistic of an exponential distribution is the sum of all observations. A similar process to that used in section 3.3.1 is used here. For the data augmentation, we set the augmented data, $\boldsymbol{Y}$ to be output from the model itself as before. For the importance regions, we start by letting $\mathcal{A}_1 = \sum_{i=1}^{10} x_i^* = 8.8071$ and given any θ, $Y_1|\mathcal{A}_1 \sim Exp(\theta)|Y_1 \leq \mathcal{A}_1$. $Exp(\theta)|Y_1 \leq \mathcal{A}_1$ is the truncated exponential distribution. Let $Z \sim Exp(\theta)$, we define $q_i = \mathbb{P}(z \leq \mathcal{A}_i) = 1 - e^{-\theta \mathcal{A}_i}$. Then the density function of the conditioned Y_i's is as follow:

$$f_{Y_i|\mathcal{A}_i}(y) = \begin{cases} \dfrac{\theta e^{-\theta y}}{q_i} & \text{if } y \leq \mathcal{A}_1 \\ 0 & \text{otherwise} \end{cases}.$$

For $i = 2, \ldots, 9$, we let $\mathcal{A}_i = 8.8071 - \sum_{j=1}^{i-1} Y_j$, and hence $Y_i|\mathcal{A}_i \sim Exp(\theta)|Y_i \leq \mathcal{A}_i$. For the last term $i = 10$, a different treatment is required because once $Y_1, \ldots, Y_9$ are sampled, then Y_{10} is fully determined. For $i = 10$, $\mathcal{A}_{10}$ is defined the same way as before, but in order to have the desired sufficient statistic we must have $Y_{10} = \mathcal{A}_{10}$. Therefore, $Y_{10}|\mathcal{A}_{10}, \theta$ is in fact a determin-

istic distribution, which has the density function:

$$f_{Y_{10}|\mathcal{A}_{10}}(y) = \begin{cases} 1 & \text{if } y_{10} = \mathcal{A}_{10} \\ 0 & \text{otherwise} \end{cases}.$$

And therefore (3.5) for the exponential example is as follow:

$$\begin{aligned}
\frac{\pi(\boldsymbol{Y}|\theta)}{\pi(\boldsymbol{Y}|T(\boldsymbol{x}^*),\theta)} &= \frac{\pi(Y_1|\theta)}{\pi(Y_1|T(\boldsymbol{x}^*),\theta)} \cdots \frac{\pi(Y_9|Y_1,\ldots,Y_8,\theta)}{\pi(Y_9|Y_1,\ldots,Y_8,T(\boldsymbol{x}^*),\theta)} \frac{\pi(Y_{10}|Y_1,\ldots,Y_9,\theta)}{\pi(Y_{10}|Y_1,\ldots,Y_9,T(\boldsymbol{x}^*),\theta)} \\
&= \frac{\pi(Y_1|\theta)}{\pi(Y_1|\mathcal{A}_1,\theta)} \cdots \frac{\pi(Y_9|\theta)}{\pi(Y_9|\mathcal{A}_9,\theta)} \frac{\pi(Y_{10}|\theta)}{\pi(Y_{10}|\mathcal{A}_{10},\theta)} \\
&= \prod_{i=1}^{9} \frac{\pi(Y_i|\theta)}{\pi(Y_i|\mathcal{A}_i,\theta)} \times \frac{\pi(Y_{10}|\theta)}{\pi(Y_{10}|\mathcal{A}_{10},\theta)} \\
&= \prod_{i=1}^{9} \frac{\theta e^{-\theta Y_i}}{\frac{\theta e^{-\theta Y_i}}{q_i}} \times \frac{\pi(Y_{10}|\theta)}{\pi(Y_{10}|\mathcal{A}_{10},\theta)} \\
&= \prod_{i=1}^{9} q_i \times \theta e^{-\theta(T(\boldsymbol{x}^*)-\sum_{j=1}^{9} Y_j)}. \qquad (3.10)
\end{aligned}$$

Following from the above, the data conditioned algorithm is simple.

Algorithm 13 (Data conditioned simulation for the exponential example)**.**

1. For $i = 1, \ldots, 9$, sample a y_i from $Exp(\theta)|Y_i \leq \mathcal{A}_i$, and calculate q_i.
2. For $i = 10$, set $y_{10} = \mathcal{A}_{10}$, and let $q_{10} = \pi(Y_{10} = \mathcal{A}_{10}|\theta)$.
3. Calculate the likelihood estimate using $\prod_{j=1}^{10} q_j$
4. Repeat steps 1. to 3. k times.
5. For the data conditioned simulation estimate, we simply take the mean of the k estimates.

We will now see how the data conditioned simulation estimates compare with the true likelihood. Utilise the fact that sum of identically independently

distributed exponential random variables has a gamma distribution [Feller, 1957, Chapter 1], we can write down the likelihood for sufficient statistic given θ:

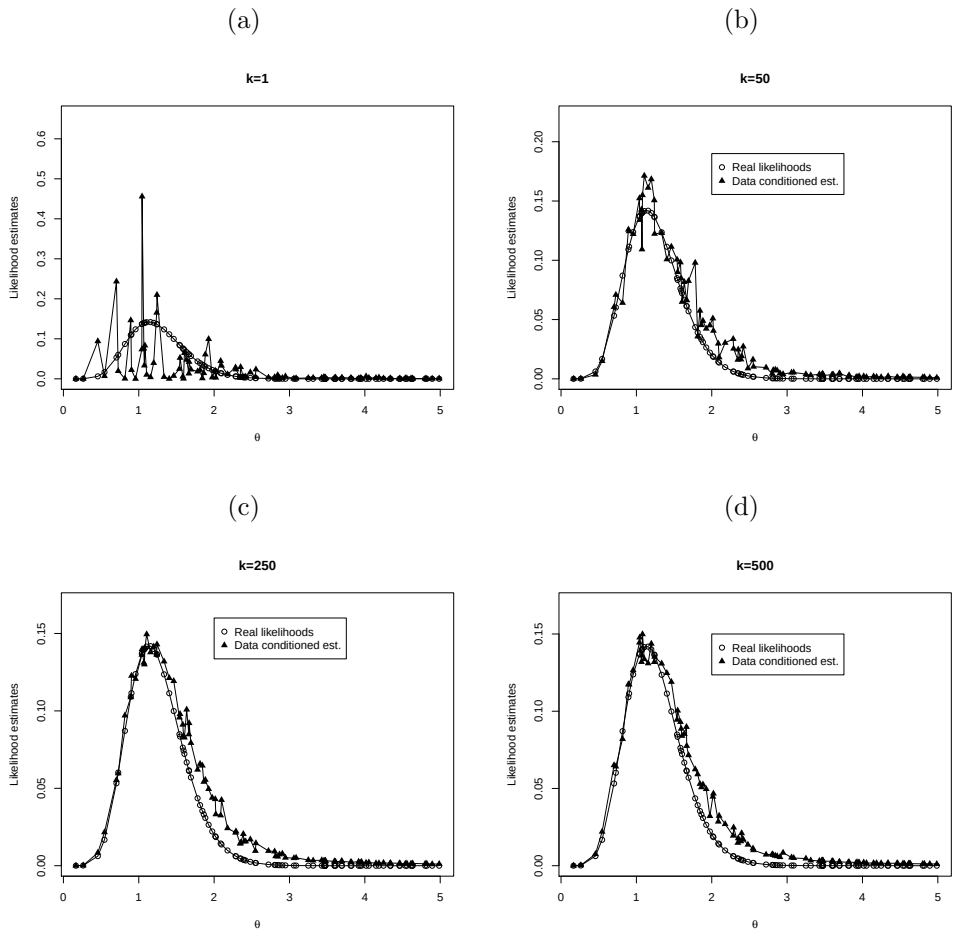
$$\sum_{i=1}^{10} X_i \sim \Gamma(10, \theta^{-1}),$$

and therefore the likelihood is:

$$\pi\big(T(\boldsymbol{x}^*), \theta\big) = \frac{\theta^{10}}{\Gamma(10)} T(\boldsymbol{x}^*)^9 e^{-\theta T(\boldsymbol{x}^*)}.$$

Figure 3.2 show the likelihood estimates of the data conditioned simulation and the true likelihood for $k = 1, 50, 250, 500$. It is apparent that as k increases, the two estimates match more closely. Although, it appears that the data conditioned estimates has a slightly heavier tail than the true likelihood.

Figure 3.2: Comparison between the real likelihood and the data conditioned likelihood estimates.



3.4.2 dcABC algoritm

The dcABC algorithm for the exponential distribution example is very similar to the one in section 3.3.2. The main difference is in step 3., where the likelihood estimate is calculated using algorithm 13. We choose to use the same prior distribution, $\Gamma(1, 0.1)$, because the conjugate prior of an exponen-

tial distribution is also the Gamma distribution. It is therefore the natural choice for the prior. For purpose of demonstrating the effectiveness of the algorithm, we want to choose a prior so that the corresponding posterior has a mean that is fairly far from the true value. We let $\theta \sim \Gamma(1, 0.1)$, where the parameters are the shape parameter and rate parameter respectively. The corresponding posterior distribution, $\pi(\boldsymbol{x}^*|\theta^*)$, is $\Gamma\left(11, 0.1 + \sum_{i=1}^{10} x_i^*\right)$. Note that the posterior distribution for the exponential distribution has a different format to the one for the Poisson distribution.

Algorithm 14 (dcABC algorithm for the exponential distribution example)**.**

1. Set $i = 1$.
2. Sample a θ_i from $\Gamma(1, 0.1)$.
3. Calculate p_i, the likelihood estimate, using algorithm 11.
4. Record (θ_i, p_i).
5. If $i < m$, increment i by 1 and go to step 2, where m is the desired sample size.

3.5 A note on practical applications

Both examples in this chapter are chosen for the illustrative purpose. In both cases, the data conditioned estimates of the likelihood are in the magnitude of 10^{-5} and 10^{-1}. As the size of data increases or the complexity of the model increases, the data conditioned estimates quickly decreases to the lower accuracy limit of the system. This could be a problem because in which case, all estimates are 0. Since we are considering the data conditioned estimates as weights, one solution is to apply a constant factor to each estimate, which we have done in chapters 4, 5, and 6. Applying a constant factor does not alter the outcome of the weighted mean calculation. However, if one were overly generous with the factor, then we are soon faced with reaching the upper accuracy limit of the system. We will discuss this further in chapter 7.

Chapter 4

SIR Model

4.1 Introduction

In this chapter we will apply the dcABC algorithm to a homogeneously mixing SIR epidemic model. The data of interest are the final size from a smallpox outbreak in Abakiliki, Nigeria in 1967. It is a well studied outbreak report [Thompson et al., 1968] from the epidemic literatures. For example see [Bailey, 1975, page 125] and [Neal, 2012]. It is worth noting that the final size of the epidemic is a sufficient statistic and therefore the algorithm will produce a parameter estimates from the exact posterior distribution rather than an approximation. The study of the SIR model in this chapter forms part of [Neal and Huang, 2015] Section 3.

We start with an introduction to the homogeneous mixing SIR epidemic model (Section 4.2). This is followed by the dcABC algorithm devised for analysing the SIR model (Section 4.3). Finally we will compare the results of a number of different ABC algorithms to demonstrate the benefits of the dcABC algorithm on the smallpox data (Section 4.3.4).

4.2 Model description

The model we will be investigating here is one of the simplest epidemic models, the homogeneously mixing SIR model. It has the following properties:

- The population is assumed to be closed. This means that no individual can enter (e.g. births or immigration) or leave (e.g. death or emigration).
- Each individual can only be in one of the three states: susceptible, infectious, or recovered.
- The status of each individual can only move in one direction from susceptible to infectious and to recovered. Once an individual is recovered, the individual stays completely immune throughout the remainder of the epidemic.
- We assume that a susceptible individual becomes immediately infectious when they come into contact with an infectious individual. Such an assumption does not affect the validity of our analysis since we are focusing on the final size data. As the final size data does not provide any information about the latent period (exposed period), we are unable to distinguish between an SIR and SEIR (susceptible, exposed, infectious, recovered) model, see [Ludwig, 1975].
- Every individual is equally likely to come into contact with every other (homogeneous mixing).
- Each individual stays infectious for a pre-determined period of time, which is called the infectious period. Infectious periods may differ between individuals but are assumed to be independently and identically distributed.
- Throughout this chapter we assume that at the beginning of the epidemic, there is only one infectious individual. This work can easily be generalised to multiple initial infectious individuals.

- We assume that individuals whilst infectious make contact at the points of a homogeneous Poisson point process. Any contact made during the infectious period is an infectious contact. A consequence of that is the number of infectious contacts made for a given length of the infectious period is Poisson distributed. If the recipient is in the susceptible state, then they will become infectious, and if the recipient is either infectious or recovered, then the contact has no effect on the recipient.

- The *final size* of the epidemic is the total number of individuals infected during the course of the epidemic. This is the total number of recovered individuals at the end of the epidemic when there are no more infectives remaining in the population.

For the epidemic model described above, the final size of the data can be simulated as follow. We label the initial infectious individual 1, and the rest of the population $2, \ldots, N$. Then the kth individual is characterised by (I_k, η_k), where I_k is the infectious period should k become infected, and η_k is the history of infectious contacts made by k during its infectious period respectively. We are free to arbitrarily choose any $I_k \geq 0$. η_k represents a Poisson point process with rate θ. We will illustrate the simulation method using the General Stochastic epidemic model from [Bailey, 1975], which assumes an Exponentially distributed infectious period. Note that, we cannot distinguish between multiplying the infectious rate by a constant factor $c > 0$ and dividing the infectious period by the same constant factor, and therefore we assume without loss of generality the infectious period is distributed as $Exp(1)$. Given the infectious rate θ, let X_k denote the total number of infectious contacts made by the kth individual during its infectious period, then $X_k \sim Poisson(\theta I_k)$ and $\mathbb{E}[X_k] = \theta\mathbb{E}[I_k] = \theta$. Thus θ represents the basic reproduction number. The basic reproduction number of an epidemic is defined to be the average number of secondary infections made by one infectious individual in a completely susceptible population, see [Dietz, 1993].

Suppose that there are currently $N - l$ susceptible individuals in the population, then the probability that an infectious contact is made with a sus-

ceptible individual is $\frac{N-l}{N}$. Considering the population as a whole, the total number of infectious contacts from the lth individual being infected to the $l+1$th individual being infected follows a geometric distribution with success probability $\frac{N-l}{N}$. Let $G_l \sim Geometric(\frac{N-l}{N})$ for $l \in (1, \ldots, N-1)$ and $G_N = \infty$. The final size, M, of the epidemic is given by:

$$M = \min \left\{ m \geq 1; \sum_{k=1}^{m} X_k < \sum_{l=1}^{m} G_l \right\} \tag{4.1}$$

An alternative construction for the above epidemic model is described in [Sellke, 1983]. The Selke construction gives us a more convenient framework to implement the dcABC algorithm. This alternative construction is also used in [Neal, 2012] for the purpose of Bayesian analysis, and we will adopt the same notation here. Under this construction, the ith individual is characterised by (I_i, T_i), an infectious period and an infectious threshold (or resistance to infection) respectively. I_i is defined the same way as before. The T_i for all $i \in \{1, \ldots, N\}$ are independently and identically distributed and follow an $Exp(1/N)$ distribution. Individual i becomes infected once the infectious pressure in the population during the course of the epidemic exceeds T_i. For an infectious individual j, its contribution to the infectious pressure is θI_j, and therefore the total population infectious pressure is $\sum_{j \in G} \theta I_j$, where G denotes the set of all infectious individuals. Hence, individual i becomes infectious if $T_i < \sum_{j \in G} \theta I_j$, and the probability that individual i avoid infection is $\mathbb{P}(T_i > \sum_{j \in G} \theta I_j) = e^{-\theta \sum_{j \in G} I_j / N}$.

In the original SIR model construction, let Y_{ji} denote the number of infectious contacts made by an infectious individual j with a particular individual, i, in the population, then $Y_{ji} \sim Poisson(\frac{\theta I_j}{N})$. Hence the probability that an infectious individual j with infectious period I_j fails to make an infectious contact with a given individual i is $\mathbb{P}(Y_{ji} = 0) = e^{(-\theta I_j / N)}$. Then the probability that a particular individual i avoids all infectious contacts from the set of infectives G is $\mathbb{P}(Y_{Gi} = 0) = e^{-\theta \sum_{j \in G} I_j / N}$. Therefore we can conclude that the two constructions are indeed equivalent for the final size of the data of the SIR model.

Under the Sellke construction, suppose that there is only one infectious individual at the beginning of the process, we can order individuals in increasing order of their infectious threshold, and relabel the population as $\{(\tilde{T}_1 = 0, \tilde{I}_1), \ldots, (\tilde{T}_N, \tilde{I}_N)\}$. Then the final size, M, the number of removed individuals at the end of the infection process satisfies:

$$M = \min\left\{ m : \tilde{T}_{m+1} > \theta \sum_{j=1}^{m} \tilde{I}_j \right\} \tag{4.2}$$

Note that $\tilde{T}_j$ is dependent on $\tilde{T}_{j-1}$ in (4.2). We can write, exploiting the memoryless property of exponential distributions, that $\tilde{T}_i = \sum_{j=1}^{i-1} L_j$, where $L_j \sim Exp\left(\frac{N-j}{N}\right)$. For the jth threshold, we are looking for the minimum time of $N - j$ exponentials with mean N and hence L_j. Then:

$$M = \min\left\{ m : \frac{\sum_{j=1}^{m} L_j}{\sum_{j=1}^{m} \tilde{I}_j} > \theta \right\} \tag{4.3}$$

The formulation in (4.3) allows us to break up the dependency between $\tilde{T}_j$'s into independently distributed L_j's. This is important to our implementation of the dcABC algorithm, as it allows us to simulate the epidemic process iteratively.

For homogeneous mixing SIR, it is possible to write down the exact likelihood as a recursive relation, see [Britton, 2010, Ball, 1986]. Let N denote the population size as before, M denote the final size as before, ι_0 denote the number of infectious individual presented in the population at time 0, $\delta = M - \iota_0$, and $p_\delta^{(N-\iota_0)}$ denote the likelihood of δ individuals infected during the epidemic excluding the initially infected individuals (ι_0). Therefore ι_0 can be any integer between $[1, M]$, but in this chapter we only consider $\iota_0 = 1$; δ can be any integer between $[0, M - \iota_0]$. Then:

$$\begin{aligned} p_\delta^{(N-\iota_0)} =& \binom{N-\iota_0}{\delta} \mathcal{L}\left(\frac{(N-\iota_0-\delta)\theta}{N}\right)^{\iota_0+\delta} \\ & - \sum_{i=0}^{\delta-1} \binom{N-\iota_0-i}{\delta-i} \mathcal{L}\left(\frac{(N-\iota_0-\delta)\theta}{N}\right)^{\delta-i} p_i^{(N-\iota_0)}, \end{aligned} \tag{4.4}$$

where $\mathcal{L}(\eta) = \mathbb{E}_I(e^{-\eta I})$ for $\eta \geq 0$, is the moment generating function of the infectious period. Let $p(M)$ denote the likelihood of the epidemic having final size M, then $p(M) = p_{M-\iota_0}^{(N-\iota_0)}$. The challenge in evaluating (4.4) is that the binomial factor reaches the system accuracy limit ($\approx 10^{308}$ for R on OSX 10 operating system) for $N > 1050$ and $\delta > 500$. Also, in order to calculate $p_{\delta}^{(N-\iota_0)}$, we need to calculate $p_0^{(N-\iota_0)}, p_1^{(N-\iota_0)}, \ldots, p_{\delta-1}^{(N-\iota_0)}$. We will look data conditioned methods of estimating the likelihood in the next section.

4.3 Algorithm implementation

Four different algorithms will be discussed in this section, namely: rsABC, dcABC, GIMH, and a data augmentation GIMH.

4.3.1 The rsABC algorithm

We begin by outlining the rsABC algorithm.

Algorithm 15 (rsABC algorithm for the SIR model)**.**

1. Sample a θ from its prior: $Exp(1)$.
2. Simulate $I_j \sim Exp(1)$ unconditionally for $j = 1, \ldots, N$, where N is the size of the population.
3. Simulate $L_j \sim Exp(\frac{N-j}{N})$ for $j = 1, \ldots, N$.
4. Determine the final size of the simulated output using equation (4.3), $\{I_1, \ldots, I_N\}$, and $\{L_1, \ldots, L_N\}$.
5. If the final size of the simulated output matches the observed data, then accept θ.
6. Repeat 1. to 5. until a desired number, m, of iterations are completed.

Note that, we have defined step 6. of the algorithm to be the number of iterations completed instead of the number of accepted values. Although it

is a less conventional implementation to the rsABC algorithm, it is useful in comparing the rsABC algorithm with other algorithms in this section. We can ensure that all algorithms have the same number of iterations.

The final size of the data is a discrete value random variable and we have constructed an rsABC algorithm with zero error tolerance and therefore the above algorithm will allow us to generate a sample from the exact posterior distribution.

4.3.2 The dcABC algorithm

We start with the dcABC algorithm for the SIR model, and then we will use a numerical example to demonstrate the algorithm.

Algorithm 16 (dcABC algorithm for the SIR model)**.**

1. Set $i = 1$
2. Sample θ_i from its prior: $Exp(1)$.
3. Simulate $I_j \sim Exp(1)$ unconditionally for $j = 1, \ldots, M$, where M is the size of the infected population.
4. Importance sampling is used to simulate $L_j \sim Exp\left(\frac{N-j}{N}\right)$ subject to constraints imposed by (4.3) for $j = 1, \ldots, M$. We denote the final importance sampling weight as P_i (See section below for a numerical walk-through).
5. Record (θ_i, P_i).
6. Increment i and repeat 2. to 5. until desired number of sample is reached, say m.

Then, $\mathbb{E}\big[\theta\big]$ or any other statistics of interest, $\mathbb{E}\big[\Phi(\theta)\big]$, can be estimated using $\widehat{\Phi(\theta)} = \frac{\sum_{i=1}^{m} P_i \Phi(\theta_i)}{\sum_{i=1}^{m} P_i}$.

A numerical walk-through of the dcABC algorithm

Suppose that the population size $N = 10$, and the final size of the epidemic $m = 4$. Our aim is to simulate an epidemic that will result in a final size of 4 in a population of size 10 for a given θ. There are many ways to achieve this. We choose to simulate the infectious periods freely from $Exp(1)$ distribution and then simulate the infectious threshold conditionally given θ and infectious periods $\{I_1, \ldots, I_{10}\}$.

Utilising (4.3), we want to generate L_j such that $\frac{\sum_{j=1}^{m} L_j}{\sum_{j=1}^{m} I_j} < \theta$ for $m = 1, \ldots, 3$ and $\frac{\sum_{j=1}^{m} L_j}{\sum_{j=1}^{m} I_j} > \theta$ for $m = 4$. For the purpose of this algorithm, simulation of L_4 is not necessary, and we will demonstrate why later. For $m = 5, \ldots, 10$, L_i's are unconstrained and their values are unimportant to the purpose of the dcABC algorithm. The dcABC algorithm is demonstrated below.

Algorithm 17 (A numerical walk-through of the dcABC algorithm)**.**

1. Sample a θ from its prior $Exp(1)$, say $\theta = 0.1245$.
2. Generate the infectious periods from the infectious period distribution $Exp(1)$, say:

$$\{I_1, \ldots, I_{10}\} = \{3.2826, 0.6872, 1.5217, 0.6123, 0.1554,$$
$$1.0467, 1.1264, 2.1385, 2.6168, 0.4713\}$$

   Note that we are free to choose the infectious period distribution.
3. Sample L_j's conditioning on θ and $\{I_1, \ldots, I_{10}\}$. By rearranging (4.3), we can deduce that $L_j \sim Exp(\frac{N-j}{j}) \Big| \theta \sum_{k=1}^{j} I_i - \sum_{i=1}^{j-1} L_i < \theta$ for $j = 1, 2, 3$ and $L_j \sim Exp(\frac{N-j}{j}) \Big| \theta \sum_{k=1}^{j} I_i - \sum_{i=1}^{j-1} L_i > \theta$ for $j = 4$. We don't need to consider L_j for $j = 5, \ldots, 10$ at this step, because they are unconstrained and therefore the contribution to the final estimate of the likelihood is of factor 1, i.e. no effect under multiplication. An example of the simulation is demonstrated below:
   1. For $j = 1$: Sample $L_1 \sim Exp(\frac{N-1}{N}) \Big| L_1 < 0.4087$. Say $l_1 = 0.2614$ then $p_1 = \mathbb{P}[L_1 < 0.4087] = 0.3078$.
   2. For $j = 2$: Sample $L_2 \sim Exp(\frac{N-2}{N}) \Big| L_2 < 0.2328$. Say $l_2 = 0.0199$ then $p_2 = \mathbb{P}[L_2 < 0.2328] = 0.1699$.
   3. For $j = 3$: Sample $L_3 \sim Exp(\frac{N-3}{N}) \Big| L_3 < 0.4023$. Say $l_3 = 0.3921$ then $p_3 = \mathbb{P}[L_3 < 0.4023] = 0.2454$.
   4. For $j = 4$: $L_4 \sim Exp(\frac{N-4}{N}) \Big| L_4 > 0.0864$. We don't need to generate an actual realisation for l_4 as the data simulation ends at this step, and the contribution to the importance sampling weight is $p_4 = \mathbb{P}[L_4 > 0.0864] = 0.9494$.
   5. The importance sampling weight for the given θ is then $P = \Pi_{j=1}^{4} p_j = 0.01219$
4. Record the data pair (θ, P). For a sample of size m, repeat steps 1. to 3. m times.

We will next look at the implementation of the GIMH algorithm for the SIR model.

4.3.3 The GIMH algorithm

Algorithm 18 (GIMH algorithm for the SIR model)**.**

1. An ideal initial value for θ_1 is to be drawn from the posterior distribution, and therefore to ensure that we can use the rsABC algorithm to initialise the first value. This can be achieved by running the rsABC algorithm until the first acceptance.
2. Estimate $\widehat{\pi(\boldsymbol{x}^*|\theta_1)}$ using steps 3. and 4. of algorithm 16 with a given importance sampling size, g say.
3. Let $\theta' \sim |N(\theta, \sigma^2)|$, then we propose a move from θ to θ' by taking a sample from $|N(\theta, \sigma^2)|$. $|N(\theta, \sigma^2)|$ is the folded Normal distribution with $mean = \theta$ and $variance = \sigma^2$.
4. Estimate $\widehat{\pi(\boldsymbol{x}^*|\theta')}$ using steps 2. and 3. of the dcABC algorithm in Section 4.3.2 with a given importance sampling size, g say.
5. Set $\theta_{i+1} = \theta'$ and $\widehat{\pi(\boldsymbol{x}^*|\theta_{i+1})} = \widehat{\pi(\boldsymbol{x}^*|\theta')}$ with probability $\min\left\{1, \dfrac{q(\theta', \theta_i)\widehat{\pi(\boldsymbol{x}^*|\theta')}\pi(\theta')}{q(\theta_i, \theta')\widehat{\pi(\boldsymbol{x}^*|\theta_i)}\pi(\theta_i)}\right\}$, or else set $\theta_{i+1} = \theta_i$ and $\widehat{\pi(\boldsymbol{x}^*|\theta_{i+1})} = \widehat{\pi(\boldsymbol{x}^*|\theta_i)}$. Note that we do not re-estimate $\widehat{\pi(\boldsymbol{x}^*|\theta_i)}$ for each iteration, and we simply record it from the previous iteration. Here, any unbiased estimator of $\pi(\boldsymbol{x}^*|\theta)$ can be used for $\pi(\boldsymbol{x}^*|\theta')$ and $\pi(\boldsymbol{x}^*|\theta_i)$ ([Andrieu and Roberts, 2009]). We use step 3. and 4. of the dcABC algorithm for their estimations and denoted using the $\widehat{\cdot}$ sign. $\pi(\cdot)$ is the density function of $Exp(1)$.
6. Repeat 3. to 5. $m-1$ times for a sample of size m.

This algorithm is in its core a Metropolis-Hasting random walk algorithm, with the only difference being that the exact likelihood is replaced with an importance sampling approximation. Since we are able to identify the sufficient statistics, as the group size, g, increases, the estimation of the likelihood will better reflect the true likelihood with decreasing variance that is proportional to $1/g$.

Data augmented GIMH algorithm

In the context of earlier GIMH implementation 4.3.3, the infection periods I_js are treated as latent variables which are independently and identically generated for each θ_i's. One variation of the GIMH algorithm is adding an extra layer of data augmentation alternating between updating I and updating θ. The resulting algorithm is as follow.

Algorithm 19 (data augmented GIMH algorithm for the SIR model)**.**

1. An ideal initial value for θ_1 is to be drawn from the posterior distribution, and therefore to ensure that we can use the rsABC algorithm to initialise the first value. This can be achieved by running the rsABC algorithm until the first acceptance.
2. Initialise $\boldsymbol{I_1}$ by taking N independent samples from their period distribution $Exp(1)$.
3. Estimate $\pi(\widehat{\boldsymbol{x}^*|\theta_1, \boldsymbol{I_1}})$ using steps 3. and 4. of algorithm 16 with a given importance sampling size, g say.
4. Again, we use the Folded Normal Distribution as the proposal distribution. Let $q(\theta, \theta^{'}) = f_{N(\theta,\sigma^2)}(\theta^{'})$ be the proposal distribution. $q(\theta, \theta^{'})$ gives the density of moving from θ to $\theta^{'}$.
5. Propose a move from θ to $\theta^{'}$ according to $q(\theta, \theta^{'})$. Again, we estimate $\pi(\widehat{\boldsymbol{x}^*|\theta^{'}, \boldsymbol{I_i}})$ using steps 2. and 3. of the sABC algorithm with a given importance sampling size, g say.

   Set $\theta_{i+1} = \theta^{'}$ and $\pi(\widehat{\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I_i}}) = \pi(\widehat{\boldsymbol{x}^*|\theta^{'}, \boldsymbol{I_i}})$ with probability $\min\left\{1, \dfrac{q(\theta^{'}, \theta_i)\pi(\widehat{\boldsymbol{x}^*|\theta^{'}, \boldsymbol{I_i}})\pi(\theta^{'})}{q(\theta_i, \theta^{'})\pi(\widehat{\boldsymbol{x}^*|\theta_i, \boldsymbol{I_i}})\pi(\theta_i)}\right\}$, or else set $\theta_{i+1} = \theta_i$ and $\pi(\widehat{\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I_i}}) = \pi(\widehat{\boldsymbol{x}^*|\theta_i, \boldsymbol{I_i}})$.

   Proposal $\boldsymbol{I'}$ by taking N independent samples from $Exp(1)$. We chose the proposal distribution to be the same as the prior distribution so that they would cancel each other out in the calculation of the acceptance probability and hence saving on computational.
6. Estimate $\pi(\widehat{\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I'}})$ as before.
7. Set $\boldsymbol{I_{i+1}} = \boldsymbol{I'}$ and and $\pi(\boldsymbol{x}^*|\widehat{\theta_{i+1}, \boldsymbol{I_{i+1}}}) = \pi(\widehat{\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I'}})$ with probability $\min\left\{1, \dfrac{\pi(\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I'})}{\pi(\boldsymbol{x}^*|\theta_{i+1}, \boldsymbol{I_i})}\right\}$, or else set $\boldsymbol{I_{i+1}} = \boldsymbol{I_i}$ and $\pi(\boldsymbol{x}^*|\widehat{\theta_{i+1}, \boldsymbol{I_{i+1}}}) = \pi(\boldsymbol{x}^*|\widehat{\theta_{i+1}, \boldsymbol{I_i}})$.
8. Repeat 4. to 7. $m-1$ times for a sample of size m.

The added variability from the importance sampling makes most of the studies in choosing the optimal σ^2 for the random walk Metropolis inapplicable. We will be using the effective sample size and the expected squared jumping distance to help us determine the choice of σ^2. In the results section, we will use σ instead of σ^2.

The effective sample size (ESS) of an MCMC output estimates the equivalent number of independent samples from the chain. Therefore, given the same number of iterations [Gamerman and Lopes, 2006, p.126], the higher the ESS the better the efficiency of the algorithm. The effective sample size for an MCMC output is defined as below:

$$ESS = \frac{n}{1 + 2\sum_1^\infty \rho_k}$$

where n is the size of X and ρ_k is the autocorrelation at lag k.

The expected square jumping distance (ESJD) is examining the distance between consecutive jumps. Large ESJD means good mixing of the chain and large ESJD also indicates small first order autocorrelation and therefore large ESS. In random walk Metropolis algorithms, ρ_k is often approximately ρ_1^k, see [Neal and Roberts, 2008]. The ESJD is defined as below:

$$ESJD = \frac{1}{n-1}\sum_1^{n-1}(X_{i+1} - X_i)^2$$

4.3.4 Results

In this section, we start with diagnostic analysis of the GIMH algorithm and the data augmentation GIMH algorithm to support our choice of σ (standard deviation of the normally distributed proposal distribution) and group size. Then, we will make a comparative analysis between the four algorithms discussed in this chapter. Throughout, we also use $Exp(1)$ as the prior distribution for θ.

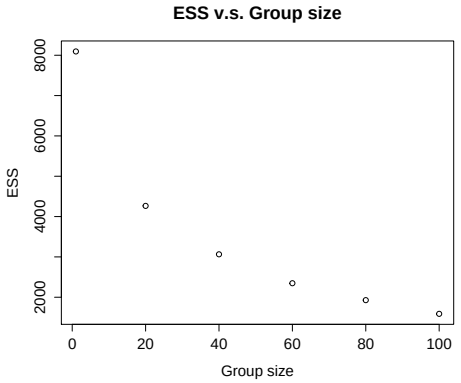
The GIMH algorithm

We start by looking at the effect of groups size and then follow by the effect of σ. The group sizes tested were $1, 20, 40, 60, 80, 100$ with $\sigma = 1$, and each group size was repeated 100 times with burnin set at 10%. The σ values tested were $0.10, 0.25, 0.40, \ldots, 3.70, 3.85, 4.00$, and as before, each value is repeated 100 times with 10% burnin.

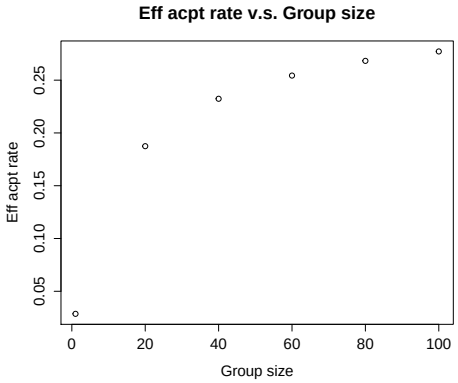
Figure 4.1c shows a significant difference between the time taken for group size of 1 and time taken for the larger group sizes. This is most likely to be caused by the overhead of function calls. Overall, Figure 4.1 indicates that group size at around 20 gives the highest ESS per second, which implies that it is the most time efficient choice, and overall Figure 4.2 show that σ at around 1 is the most efficient choice as it has the highest ESS per second.

Figure 4.1: Effects of groups sizes on different aspects of the performance of the GIMH algorithm.

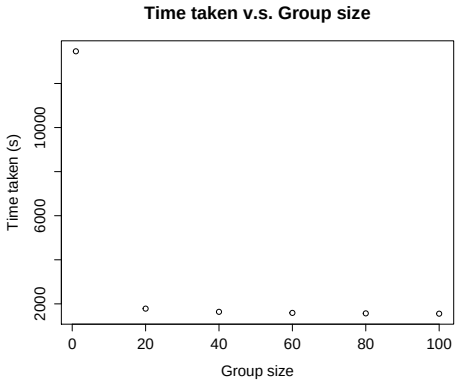
(a) ESS v.s. group size



(b) Effc. acpt. rate v.s. group size



(c) Time taken v.s. group size



(d) ESS per second v.s. group size

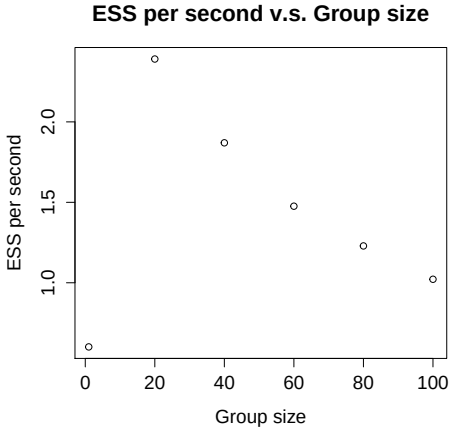
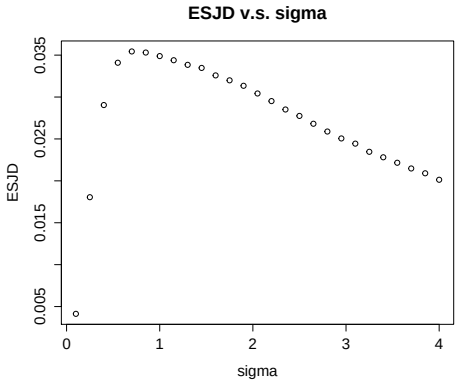
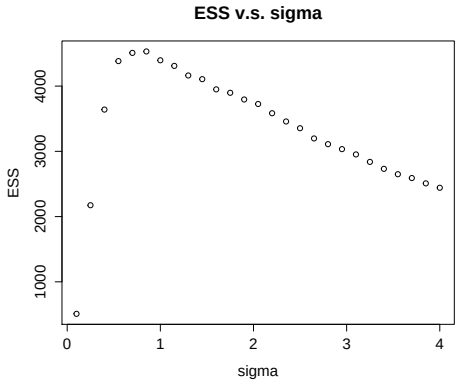


Figure 4.2: Effects of σ on different aspects of the performance of the GIMH algorithm.

(a) ESJD v.s. σ



(b) ESS v.s. σ



(c) ESS per second v.s. σ

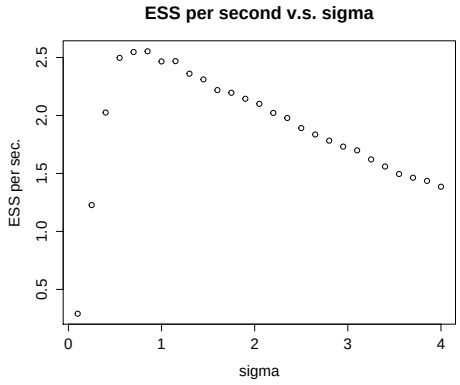
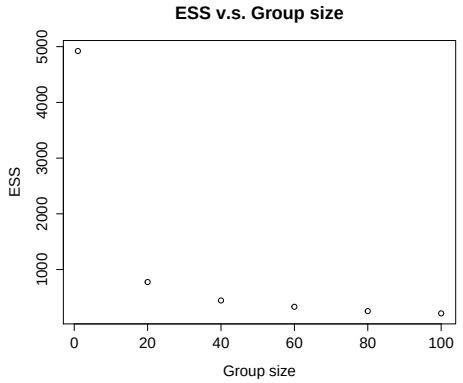
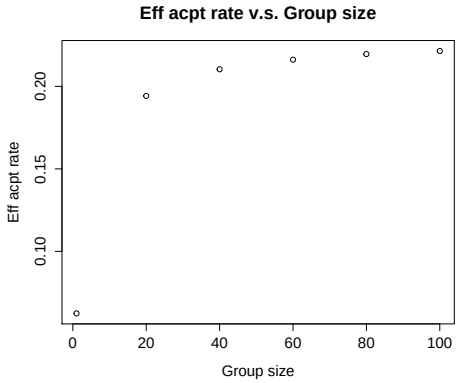


Figure 4.3: Effects of groups sizes on different aspects of the performance of the data augmented GIMH algorithm.

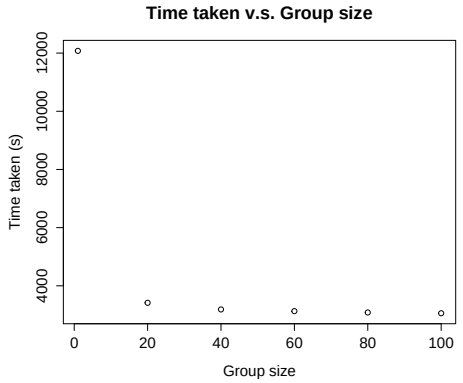
(a) ESS v.s. group size



(b) Effc. acpt. rate v.s. group size



(c) Time taken v.s. group size



(d) ESS per second v.s. group size

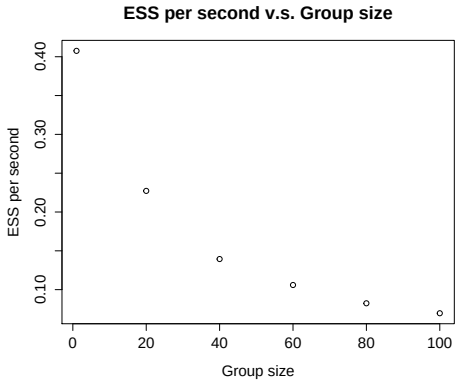
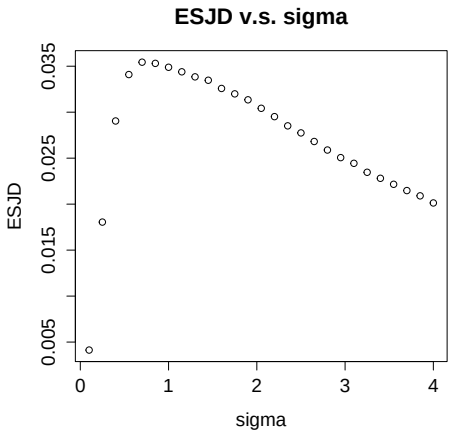
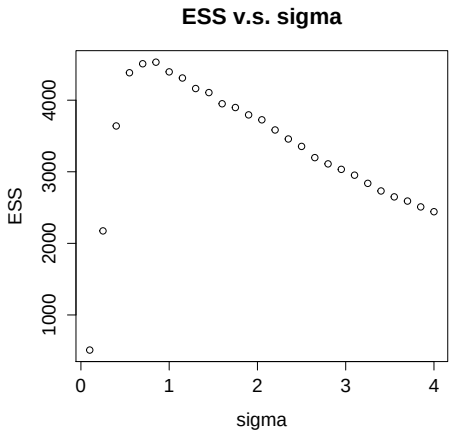


Figure 4.4: Effects of σ on different aspects of the performance of the data augmentated GIMH algorithm.

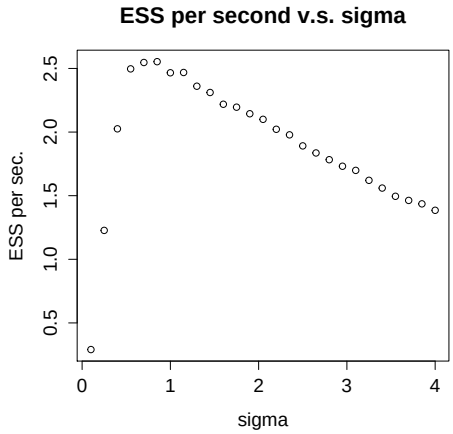
(a) ESJD v.s. σ



(b) ESS v.s. σ



(c) ESS per second v.s. σ



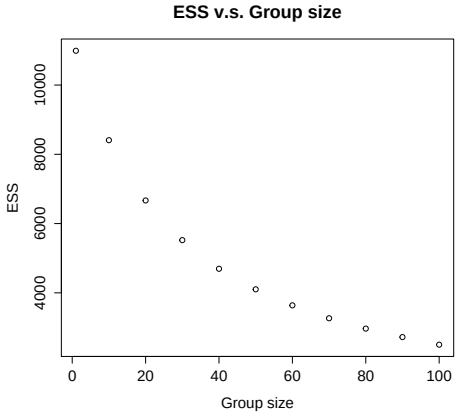
It can be observed that the data augmented GIMH algorithm favours a slight different set of group size and σ. Firgure 4.3 shows that group size 1 is the more efficient, and Figure 4.4 indicates that σ is most efficient at around 0.7. It is worth noting that even at its best, the data augmented GIMH is almost

5 times less efficient comparing to the GIMH algorithm measuring by ESS per second.

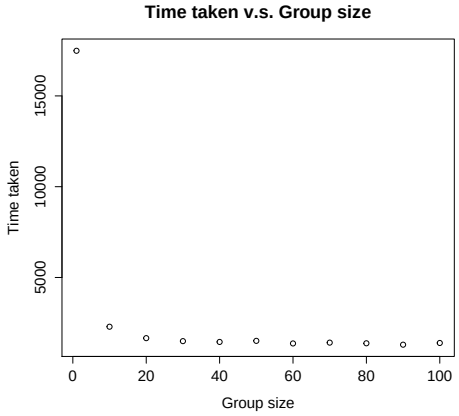
The dcABC algorithm

Figure 4.5: Effects of groups sizes on different aspects of the performance of the dcABC algorithm.

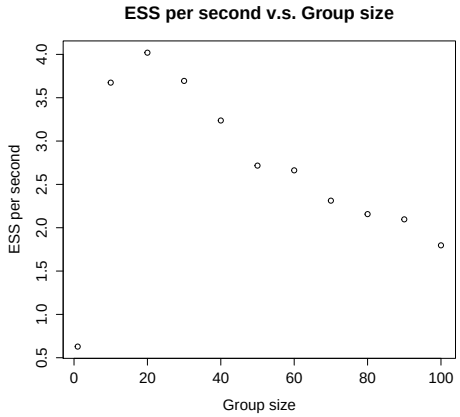
(a) ESS v.s. group size



(b) Time taken v.s. group size



(c) ESS per second v.s. group size



It is clear from Figure 4.5, that the most efficient group size for the dcABC algorithm is also at around 20.

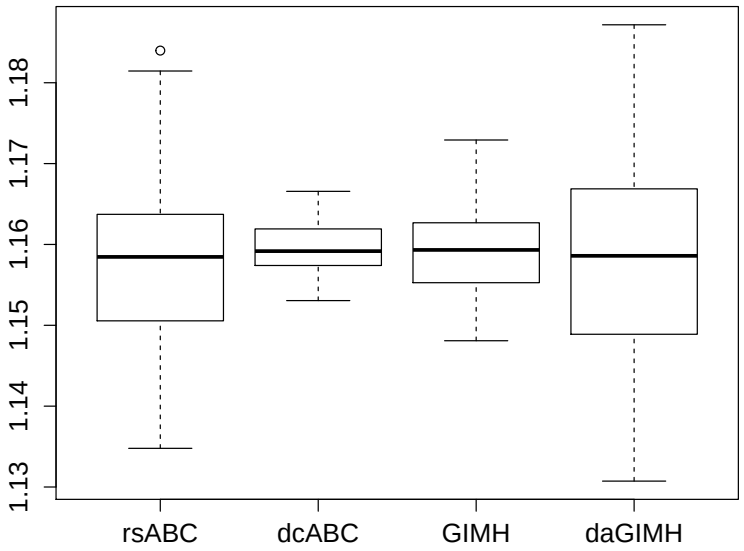
Comparison across all algorithms

To achieve a computationally fair comparison, the rsABC algorithm was ran with 1,000,000 samples, and the dcABC algorithm, the GIMH algorithm and data-augmentation MCMC are all ran with the condition that sample size $\times$ importance sampling sample size (group size) $= 10^6$. For the dcABC algorithm, we chose the group size to be 20. For the GIMH algorithm, we used group size 20 and $\sigma = 1$. For the data augmented GIMH algorithm, we used group size 1 and $\sigma = 0.75$.

Table 4.1: The mean estimates, effective sample sizes, and computation times over 100 runs for all algorithms.

	rsABC	dcABC	GIMH	daGIMH
$\mathbb{E}[\theta\|\boldsymbol{x}^*]$	1.1582	1.1626	1.1588	1.1579
ESS	15539	6675.4	4011.0	7477.2
Duration (s)	996.06	1957.4	2037.3	3927.0

Figure 4.6: Boxplot of 100 estimates from all four algorithms



From Table 4.1 and Figure 4.6, we can see that all four algorithms give reasonable mean estimate. Although the rsABC is the fastest, the dcABC gives the most consistent estimates whilst maintaining a reasonable computation time when comparing to the GIMH algorithm. A somewhat surprising observation is that the daGIMH algorithm is least consistent and gives estimations with large variation.

4.4 Chapter Conclusion

The SIR model has served as a good demonstration for the application of the dcABC algorithm, and provide a useful comparison between the rsABC algorithm, the dcABC algorithm, and the GIMH algorithm. The dcABC al-

gorithm has a clear advantage when compared to the other algorithms in the example. Although the dcABC algorithm is not the fastest, but we can see that the extra computation time is well spent on giving consistent estimates with small variation. In order to have a more mathematically objective comparison, all algorithms are ran in series, even though the dcABC algorithm falls under the embarrassingly parallel type of algorithms. Embarrassingly parallel type of algorithms means that certain part of the algorithm can be executed independently, see [Foster, 1995]. In the case of dcABC algorithm, θ's are sampled independently and therefore the likelihood of each sampled θ can be estimated independently and simultaneously. Whereas in MCMC type of algorithms, the dependence of between θ's means that one would have to wait for the acceptance/rejection of θ before moving onto the next iteration. That means, in theory, the efficiency of dcABC is linear dependent on the number of computer available, with the limit of the time taken to estimate the likelihood of one θ, and it happens when there are as many computers as the number of θ's.

Chapter 5

Time Inhomogeneous Markov Chain

5.1 Introduction

In this chapter will look at a different class of problem to the one in Chapter 4. In the SIR model, we view the population collectively and model the entire population at once. Whereas in this chapter, we are using a time inhomogeneous Markov chain to model individual's disease progression. It is in essence a Susceptible - Infectious - Susceptible (SIS) model with time dependent infectious rate and constant recovery rate.

We start by providing some background knowledge for continuous time Markov chain and defining the model in Section 5.2, which includes the implementation of both unconditioned and data conditioned simulation in Section 5.2.4, and then the implementation of the dcABC algorithm and the GIMH algorithm in Section 5.3. Finally, we conclude our findings in Section 5.4.

5.2 Model description

5.2.1 Data

The data of interest here was from a longitudinal study of five related diseases on animals in the Tanga region of Tanzania. The study tested 381 animals from 81 farms and each was tested between 1 to 11 times for the presence of five diseases. Our main interest is in one of the five diseases because this is illustrative of the effect for the other diseases. In our data, each record has the farm identifier, the animal identifier, age of the animal in days when the test was carried out, and presence of the disease. There are other covariates in the study, which can be used for further analysis.

5.2.2 The model

The disease progression can be modelled as a SIS epidemic model. It is believed that the chance of an animal catching the disease is dependent on its age, and the chance of an animal recovering from the disease is not dependent on age, and thus constant over time. Each animal can be in either of the two states, susceptible or infectious and the status of each animal can switch between susceptible and infectious multiple times. Each entry in the data set consists of 4 values: (X_0, X_t, t, a_0). X_0 represents the state of the individual at the start. X_t represents the state of the individual time t. The state of each individual can be either 0 or 1. 0 denotes that the individual is susceptible and 1 denotes that the individual is infectious. t denotes the time interval between the first observation of the individual and the second observation. a_0 is the age of the animal at the start.

To model the disease progression on an animal, a time-inhomogeneous Markov

model is proposed with the following infinitesimal generator matrix:

$$Q(u;\beta_0,\beta_1,\rho) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} -e^{\beta_0+\beta_1 a_u} & e^{\beta_0+\beta_1 a_u} \\ \rho & -\rho \end{pmatrix} \end{array}$$

where 0 indicates susceptible and 1 indicates infectious, and $0 \to 1$ is called infection, $1 \to 0$ is called recovery. a_u is the age of the subject at time u, and $a_u = a_0 + u$. We assume that $\beta_0 < 0, \beta_1 > 0$, and $\rho > 0$ are common parameters shared by all individuals. We are interested in estimating the values of β_0, β_1, and ρ from the observed data.

For each animal, we only observe at time 0 and at time t. However, a series of unobserved infections and recoveries may occur between time 0 and time t. We are only interested in time when the last of these unobserved event occurred before t, and we denote this time as s, where $0 < s < t$. The reason for that is because the final status of the individual is fully determined by what happened at time s.

In order to estimate the parameters using ABC algorithms, data augmented simulation of the model is required because the last event time, s, is unobserved. In the next section, we will derive an expression for the time to the first event in a time inhomogeneous Markov chain which will help us perform the data augmentation required and therefore enable us to simulate the model.

5.2.3 Time to the last event

In this section, we start with a short introduction to continuous time Markov chain. This is followed by a derivation of the distribution for the time between the last transition and the final observation. We will show how the distribution of the holding time in continuous time, time-homogeneous Markov chain (CTTHMC) can help us to derive the distribution for the time between the last transition and the final observation, and we follow the work done by

[Whitt, Amir] for the derivation of the CTTHMC. We then extend this to binary state continuous time, time-inhomogeneous Markov chain (CTTIMC), and the general theory of the CTTIMC we follow the work on [Mereacre].

Continuous time Markov chain

The two building blocks of a continuous time Markov chain are the distribution of the holding times in each state and the transition probability of the embedded Markov chain. The holding time of a continuous time Markov chain (CTMC) is defined as the amount of time that the Markov chain will remain in a given state. Suppose that $X(0) = x$, and let T_x denote the time that the Markov chain moves away from state x. For the distribution of T_x, we let $a, b \in \mathbb{R}^+$ and consider:

$$\begin{aligned}
&\mathbb{P}\Big(T_x > a + b \Big| T_x > a\Big) \\
&\quad = \mathbb{P}\Big(X(t) = x; t \in [0, a+b] \Big| X(t) = x; t \in [0, a]\Big) \\
&\quad = \mathbb{P}\Big(X(t) = x; t \in [a, a+b] \Big| X(a) = x\Big) (\text{by Markov property}) \\
&\quad = \mathbb{P}\Big(X(t) = x; t \in [0, b] \Big| X(0) = x\Big) (\text{by time homogeneity}) \\
&\quad = \mathbb{P}\Big(T_x > b\Big)
\end{aligned}$$

We observe that the holding time satisfies the memoryless property and therefore that the holding time must be exponentially distributed. We let $\lambda(x)$ denote the parameter for the exponential distribution of the holding time of state x, i.e. $T_x \sim Exp(\lambda(x))$. T_x tells us when the Markov chain is moving away from state x, but it does not tell us where it is going to. We will look at this next, which leads us to defining the transition probability of the embedded Markov chain and behaviour of the continuous time Markov chain in an infinitesimal time interval.

For $y \neq x$, we let $p_{xy} = \mathbb{P}\big(X(T_x) = y \big| X(0) = x\big)$ be the probability of the chain entering state y after leaving x. We also let $\lambda(x, y) = \lambda(x)p_{xy}$ be the rate for going from state x to state y.

We will examine the behaviour of the chain in an infinitesimal time interval. For small $h > 0$, we have:

$$\begin{aligned}\mathbb{P}(T_x < h) &= 1 - e^{-\lambda(x)h} \\ &= \lambda(x)h + o(h)\end{aligned}$$

where the $o(h)$ above represents the higher order terms of the Taylor expansion.

An important remark to make here is that from the Markov property of the continuous time Markov chain, we can deduce that the holding time and the transition probability of the embedded Markov chain are two independent random variables, which will become useful in the following. Considering a small h, for $y \neq x$:

$$\begin{aligned}&\mathbb{P}\big(X(h) = y \big| X(0) = x\big) \\ &\qquad = \mathbb{P}\big(T_x < h, X(T_x) = y \big| X(0) = x\big) + o(h) \\ &\qquad = \lambda(x)hp_{xy} + o(h) \\ &\qquad = \lambda(x, y)h + o(h)\end{aligned}$$

where the $o(h)$ in the above equation represents the probability of two or more transitions occurred between the time $[0, h]$. Therefore $\lambda(x, y)$ represents the local rate of transitioning from state x to y. It is easy to show that

$\sum_{y\neq x}\lambda(x,y) = \sum_{y\neq x}\lambda(x)p_{xy} = \lambda(x)$. And therefore:

$$\begin{aligned}
&\mathbb{P}\big(X(h)=x\big|X(0)=x\big)\\
&\qquad = 1-\sum_{y\neq x}\mathbb{P}\big(X(h)=y\big|X(0)=x\big)\\
&\qquad = 1-\sum_{y\neq x}\lambda(x,y)h+o(h)\\
&\qquad = 1-\lambda(x)h+o(h)\text{as}
\end{aligned}$$

Equipped with the above, a CTTHMC with transition rate $\lambda(x,y)$ can be defined to be a stochastic process $X(t)$ taking values in a finite or countably infinite state space S satisfying

$$\mathbb{P}\big(X(t+h)=x\big|X(t)=x\big) = 1-\lambda(x)h+o(h).$$

and

$$\mathbb{P}\big(X(t+h)=y\big|X(t)=x\big) = \lambda(x,y)h+o(h).$$

The transition probabilities can be calculated using the equation below:

$$p_{xy} = \frac{\lambda(x,y)}{\lambda(x)} = \frac{\lambda(x,y)}{\sum_{y\neq x}\lambda(x,y)}.$$

We now look at how the local behaviour construction of the CTTHMC relates to the infinitesimal generator matrix, and this is achieved through Kolmogorov equations, and in particular the forward equations. Kolmogorov equations are a set of differential equations developed by Andrei Kolmogorov to characterise continuous time Markov processes [Norris, 1997] . We begin by defining the matrix $\boldsymbol{P(t)}$, which has components satisfying:

$$P_{ij}(t) = \mathbb{P}\big(X(t)=j\big|X(0)=i\big)$$

for $i,j\in S$, the state space and $t>0$.

We further assume that the Markov chain is non-explosive with finite state space, so that there are no problems with interchanging the order of summation and limit. For *non-explosive*, we meant that there is only a finite number

of transitions in a finite amount of time. The Kolmogorov forward equations state that $\boldsymbol{P}'(t) = \boldsymbol{P}(t)\boldsymbol{Q}$, where $\boldsymbol{Q}$ is the infinitesimal generating matrix, $\boldsymbol{P}'(t) = \frac{d}{dt}\boldsymbol{P}(t)$, and $\boldsymbol{P}$ is defined below. We investigate how $P_{ij}(t)$ changes over time by studying its derivative from first principals conditioning on the last jump before our time of interest:

$$
\begin{aligned}
P_{ij}^{'}(t) &= \lim_{h\to 0} \frac{P_{ij}(t+h) - P_{ij}(t)}{h} \\
&= \lim_{h\to 0} \frac{1}{h}\Big(\mathbb{P}\big(X(t+h) = j \big| X(0) = i\big) - \mathbb{P}\big(X(t) = j \big| X(0) = i\big)\Big) \\
&= \lim_{h\to 0} \frac{1}{h}\Big(\sum_{y\in S} \mathbb{P}\big(X(t+h) = j \big| X(t) = y, X(0) = i\big)\mathbb{P}\big(X(t) = y \big| X(0) = i\big) \\
&\quad - \mathbb{P}\big(X(t) = j \big| X(0) = i\big)\Big) \\
&= \lim_{h\to 0} \frac{1}{h}\Big(\mathbb{P}\big(X(t+h) = j \big| X(t) = j, X(0) = i\big)\mathbb{P}\big(X(t) = j \big| X(0) = i\big) \\
&\quad + \sum_{y\neq j} \mathbb{P}\big(X(t+h) = j \big| X(t) = y, X(0) = i\big)\mathbb{P}\big(X(t) = y \big| X(0) = i\big) \\
&\quad - \mathbb{P}\big(X(t) = j \big| X(0) = i\big)\Big) \\
&= \lim_{h\to 0} \frac{1}{h}\Big(\big(1 - \lambda(j)h + o(h)\big)P_{ij}(t) + \sum_{y\neq j} \big(\lambda(y,j)h + o(h)\big)P_{iy}(t) - P_{ij}\Big) \\
&= \lim_{h\to 0} \frac{1}{h}\Big(-\lambda(j)hP_{ij}(t) + \sum_{y\neq j} \lambda(y,j)hP_{ij}(t) + \sum_{y\in S} P_{iy}(t)o(h)\Big) \\
&= -\lambda(j)P_{ij}(t) + \sum_{y\neq j} P_{iy}(t)\lambda(y,j)
\end{aligned}
$$

Note that the non-explosive assumption ensures that

$$\frac{\sum_{y\in S} P_{iy}(t)o(h)}{h} \to 0 \text{ as } h \to 0.$$

We can now define the infinitesimal generator matrix, $\boldsymbol{Q}$, as follow:

$$Q_{ij} = \begin{cases} -\lambda(i), \text{ for } i = j \\ \lambda(i,j), \text{ for } i \neq j \end{cases}$$

This allows us to write down the matrix differential equation for $\boldsymbol{P}(t)$:

$$\boldsymbol{P}'(t) = \boldsymbol{P}(t)\boldsymbol{Q}$$

and we can easily validate the matrix differential equation above by looking at its components. Using the initial condition that $\boldsymbol{P}(0) = \boldsymbol{I}$, the identity matrix, the system of differential equation has the following solution:

$$\boldsymbol{P}(t) = \boldsymbol{P}(0)e^{t\boldsymbol{Q}} = e^{t\boldsymbol{Q}}$$

where $e^{t\boldsymbol{Q}}$ is the matrix exponential defined as:

$$e^{t\boldsymbol{Q}} = \sum_{k=0}^{\infty} \frac{t^k \boldsymbol{Q}^k}{k!}.$$

This concludes the derivation for the Kolmogorov forward equations, and we will now turn to the Kolmogorov backward equations. The Kolmogorov backward equations are a system of equations that satisfy the identity: $\boldsymbol{P}'(\boldsymbol{t}) = \boldsymbol{Q}\boldsymbol{P}(\boldsymbol{t})$, and valid on general Markov processes. There are only subtle differences between the forward and backward equations. Both sets of equations start by considering the same problem: finding an expression for $P_{ij}(t) = \mathbb{P}\big(X(t) = j \big| X(0) = i\big)$. The forward equations achieves its solution by considering the last jump as having occurred between the time $[0, t]$, and the backward equations achieves its solution by considering the first jump as having occurred between the time $[0, t]$. To recover the backward equations,

suppose J_1 is the time of the first jump, and consider:

$$\begin{aligned}
P_{ij}^{'}(t) =& \lim_{h\to 0} \frac{P_{ij}(t+h) - P_{ij}(t)}{h} \\
=& \lim_{h\to 0} \frac{1}{h}\Big(\sum_{k\in S} \big(P_{ik}(h)P_{kj}(t)\big) - P_{ij}(t)\Big) \\
=& \lim_{h\to 0} \frac{1}{h}\Big(\sum_{k\in S} \big(P_{ik}(h) - \delta(i,k)\big)P_{kj}(t)\Big) \\
=& \lim_{h\to 0} \frac{1}{h}\Big(\sum_{k\neq i} \big(\lambda(i,k)h + o(h)\big)P_{kj}(t) - \big(\lambda(i)h + o(h)\big)P_{ij}(t)\Big) \\
=& \sum_{k\neq i} \lambda(i,k)P_{kj}(t) - \lambda(i)P_{ij}(t) \\
=& Q_{ik}P_{kj}(t)
\end{aligned}$$

We have the Kolmogorov backward equation:

$$\boldsymbol{P}^{'}(t) = \boldsymbol{Q}\boldsymbol{P}(t)$$

where $\boldsymbol{Q}$ is the infinitesimal generating matrix defined as before.

Using the same initial condition as the forward equations case, the backward equation has the same solution as the forward equations:

$$\boldsymbol{P}(t) = e^{t\boldsymbol{Q}}.$$

Time to the last event - CTTHMC

We now consider the time to the last event in a continuous time time homogeneous binary state Markov chain with non-zero $\boldsymbol{Q}$ matrix, which echoes the model proposed in section 5.2.2. Binary state Markov chain makes our approach of representing the Markov chain with a marked Poisson process easy to define. The guaranteed reversibility of a binary state Markov chain also

allows us to translate the first jump time to last event time (to be defined later). A similar approach can be taken for some special cases of Markov chains with more than 2 states, but they are beyond the scope of this thesis. Let $X(t)$ denote the states of a CTTHMC at time t with states $\{0, 1\}$ and infinitesimal generator matrix:

$$Q_X = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} -\alpha_X & \alpha_X \\ \beta_X & -\beta_X \end{pmatrix} \end{array}$$

An alternative representation to the two state CTTHMC above is a marked Poisson point process with events occurring at rate $\alpha_X + \beta_X$. Each point (event) on the Poisson process has a mark either 1 or 0 with probability $\frac{\alpha_X}{\alpha_X+\beta_X}$ or $\frac{\beta_X}{\alpha_X+\beta_X}$ respectively. Then $X(t)$ denotes the state of the last event (point) prior to time t. We use the word "event" because in this alternative representation, an event does not always changes the state of the process. For example, it is possible that the process was in state 0, and an event occurred which sets the process to being in state 0 again and therefore no change takes place.

To show that the Poisson process representation is indeed equivalent to the CTTHMC, we start by looking at the first jump time. Let T denote the random variable of the first jump time, then T can be described as below:

$$T = \min\{s > 0 : X(s) \neq X(0)\}.$$

Further, let T_0 and T_1 denote the holding time at state 0 and 1 respectively. For the two states CTTHMC above, we can rewrite T in terms of T_0 and T_1:

$$T = min\{T_0, T_1\}.$$

Given the fact that $T_0 \sim Exp(\alpha_X)$ and $T_1 \sim Exp(\beta_X)$, using the property of exponential distributions, we can write down the distribution of T:

$$T \sim Exp(\alpha_X + \beta_X).$$

T tells us when the first jump occurs, and to know where it jumps to, we

turn to the following probabilities $\mathbb{P}(T = T_0)$ and $\mathbb{P}(T = T_1)$.

$$\begin{aligned}
\mathbb{P}\big(T = T_0\big) =& \mathbb{P}\big(T_1 > T_0\big) \\
=& \int_0^\infty \mathbb{P}\big(T_1 > T_0 \big| T_0 = t\big) f_{T_0}(t) dt \\
=& \int_0^\infty \mathbb{P}\big(T_1 > t\big) f_{T_0}(t) dt \\
=& 1 - \int_0^\infty \mathbb{P}\big(T_1 < t\big) f_{T_0}(t) dt \\
=& 1 - \int_0^\infty \big(1 - e^{-\beta_X t}\big) \alpha_X e^{-\alpha_X t} dt \\
=& 1 - \int_0^\infty \alpha_X e^{-\alpha_X t} dt - \int_0^\infty \alpha_X e^{-(\alpha_X + \beta_X)t} dt \\
=& 1 - 1 + \Big[\frac{\alpha_X}{\alpha_X + \beta_X} e^{-}(\alpha_X + \beta_X)t\Big]_0^\infty \\
=& \frac{\alpha_X}{\alpha_X + \beta_X}.
\end{aligned}$$

Similarly we can show that $\mathbb{P}(T = T_1) = \frac{\beta_X}{\alpha_X + \beta_X}$. Combining the first jump time and probabilities derived here, we can reconstruct the two-state CT-THMC without conditioning on the current state. By using the first jump time, we can imagine the process starts afresh after each jump, and the two probabilities help us to determine the nature of each jump. To conclude, this is a stochastic process with exponentially distributed waiting time. The above process is reversible (see below), so that the distribution of the time from the start to the first event is the same as the time from the end to the last event. We will explore another angle of the Poisson process which relaxes the homogeneous rate assumption, and will allow us to simulate the time point of the last even between $[0, t]$.

We check for reversibility of $X(t)$ by solving the following for $\boldsymbol{\pi}$:

$$\pi_1 Q_{1,2} = \pi_2 Q_{2,1}$$

$$\Rightarrow \pi_1 \alpha_X = \pi_2 \beta_X.$$

Since there are two unknowns (π_1, π_2), but we can only form 1 equations. There are infinite many solutions. However, one simple solution would be that $\pi_1 = \beta_X$ and $\pi_2 = \alpha_X$. Hence we have the reversibility. This allows us to reverse the Markov chain, and what we derived for the first jump time can also be used for the last event time. Let S_X denote the time point of the last event on the interval $[0, t_X]$ of the process $X(t)$, where t_X denotes the end time of the process. Let $N_X(a, b)$ denote the number of events that occurred between time interval $[a, b]$. Then

$$N_X(a, b) \sim Poisson\Big(\int_a^b \alpha_X + \beta_X du \Big).$$

We observe that $\mathbb{P}(S_X < s) = \mathbb{P}(N_X(s, t_X) = 0)$. Therefore,

$$\begin{aligned} \mathbb{P}\big(S_X < s\big) =& \mathbb{P}\big(N_X(s, t_X) = 0\big) \\ =& e^{-\int_s^{t_X} (\alpha_X + \beta_X) du}. \end{aligned}$$

To simulate S_X, we use the inversion of the CDF method. Let $U \sim U(0, 1)$, and we can obtain a sample of S_X by rearranging:

$$e^{-\int_s^{t_X} (\alpha_X + \beta_X) du} = U.$$

Further, if we let $V \sim Exp(1)$. We can again apply inversion of the CDF method, using $e^{-V} \overset{D}{=} U$, i.e. $e^{-V} \sim U(0, 1)$. Then:

$$e^{-\int_s^{t_X} (\alpha_X + \beta_X) du} = e^{-V}$$

$$\Rightarrow \int_s^{t_X} \alpha_X + \beta_X du = V$$

$$\Rightarrow s = t_X - \frac{V}{\alpha_X + \beta_X}$$

We will consider the time inhomogeneous case below.

Time to the last event - CTTIMC

For the CTTIMC, the idea is similar. We will construct an equivalent Poisson process and then analyse it based on properties of the Poisson process. However, the constructive argument given in the CTTHMC case does not work here. The time dependence of the infinitesimal generating matrix complicates the process. However, when considering a two-state Markov chain, we can still find a Poisson process that has the same distribution. Firstly, we let $Y(t)$ denote the state of a CTTIMC at time t with transition matrix:

$$Q_Y(t) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} -\alpha_Y(t) & \alpha_Y(t) \\ \beta_Y(t) & -\beta_Y(t) \end{pmatrix} \end{array}$$

We propose an equivalent marked Poisson process with points (events) occurring at rate $\alpha_Y(t)+\beta_Y(t)$. There are two types of events 1 and 0 and each happens with probability $\frac{\alpha_Y(t)}{\alpha_Y(t)+\beta_Y(t)}$ and $\frac{\beta_Y(t)}{\alpha_Y(t)+\beta_Y(t)}$ respectively. To show that the type of the last event (mark) in the Poisson process prior to time t is equivalent to $Y(t)$, we want to show that the rate of the event $0 \to 1$ and $1 \to 0$ are $\alpha_Y(t)$ and $\beta_Y(t)$ respectively. We can achieve this by the idea of thinning. We will look at the case of $0 \to 1$ transition in particular, and the other way round can be achieved similarly. The process $Z(t)$ has two states and the idea of thinning is to look at the process of a particular type of event, $Z(t) = 1$ say. Let $Z_1(t)$ denote such a process. To construct $Z_1(t)$, we simply let the $Z(t)$ run its course and only accept events such that $Z(t) = 1$. Then the rate for $Z_1(t)$ is simply $\big(\alpha_Y(t) + \beta_Y(t)\big)\frac{\alpha_Y(t)}{\alpha_Y(t)+\beta_Y(t)} = \alpha_Y(t)$, i.e. rate of events occurring $\times$ Probability an event is of type 1. Similarly $Z_0(t)$ has rate $\beta_Y(t)$. Given $Z(t) = 0$, process $Z_0(t)$ would promote no change and therefore $Z_1(t)$ dictate when the next transition will happen, which means $\alpha_Y(t)$ is the transition rate for $0 \to 1$. We can do the same for the $1 \to 0$ transition. We have shown that process $Z(t)$ and $Y(t)$ have the same distribution.

Let S_Y denote the time point where the last event occurred for process $Y(t)$, t_Y denote the time of the end of the process, and $N_Y(a, b)$ denote the number

of events that occurred between time interval $[a, b]$ for $b > a > 0$. Then

$$N_Y(a, b) \sim Poisson\Big(\int_a^b \alpha_Y(u) + \beta_Y(u)du\Big).$$

We proceed as before, and we have

$$\begin{aligned}\mathbb{P}\big(S_Y < s\big) =& \mathbb{P}\big(N_Y(s, t_Y) = 0\big) \\ =& e^{-\int_s^{t_Y} \alpha_Y(u)+\beta_Y(u)du}.\end{aligned}$$

Again, to sample for s, we apply the inversion of the CDF method to simulate S_Y. Let $V \sim Exp(1)$, and we know that $e^{-V} \sim U(0, 1)$, and therefore to obtain S_Y, we rearrange

$$e^{-\int_s^{t_Y} \alpha_Y(u)+\beta_Y(u)du} = e^{-V}$$

$$\Rightarrow \int_s^{t_Y} \alpha_Y(u) + \beta_Y(u)du = V.$$

To obtain s, we rearrange and solve the above for s. We will see how this can help us to simulate the desired data in the next section.

5.2.4 Data simulation

We start by looking at the straightforward data simulation based on what we developed in the previous section, and then we will look at data conditioned simulation.

Unconditioned data simulation

We recall that each data point consists of 4 values (X_0, X_t, t, a_0), status at time 0, status at time t, time elapsed between X_0 and X_t, and age at time 0 respectively. Although in practice, each animal may have their first

observation taken at different times, we can assume that they all started at the same time, because we know the age when the first observation is made. We also recall that the proposed Markov chain has the following infinitesimal generating matrix:

$$Q(u;\beta_0,\beta_1,\rho) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} -e^{\beta_0+\beta_1 a_u} & e^{\beta_0+\beta_1 a_u} \\ \rho & -\rho \end{pmatrix} \end{array}.$$

Given $X_0, t, a_0, \alpha, \beta, \rho$, we want to simulate X_t, which is simply the event occurred last before time t. Suppose $V \sim Exp(1)$, consider

$$\int_s^t \Big(e^{\beta_0+\beta_1(a_0+u)} + \rho\Big)du = V$$

$$\Rightarrow \frac{e^{\beta_0+\beta_1 a_0}}{\beta_1}\big(e^{\beta_1 t} - e^{\beta_1 s}\big) + \rho(t-s) = V$$

$$\Rightarrow \frac{e^{\beta_0+\beta_1 a_0+\beta_1 s}}{\beta_1} + \rho s = \frac{e^{\beta_0+\beta_1 a_0+\beta_1 t}}{\beta_1} + \rho t - V. \tag{5.1}$$

The above equation cannot be solved directly, and therefore some form of numerical method should be employed. We observe that $\frac{e^{\beta_0+\beta_1 a_0+\beta_1 s}}{\beta_1} + \rho s$ is a strictly decreasing as s decreases, and therefore it can only have at most one solution. We are only interested if the solution is in $(0, t)$, and therefore a simple binary search will be able to locate the solution to the desired accuracy quickly. Once we have the solution for s, we can determine whether it was an infection or recovery with probability $\frac{e^{\beta_0+\beta_1(a_0+s)}}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$ and $\frac{\rho}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$ respectively using a Bernoulli trial. We can now fully determine X_t. We repeat the above for each individual for the unconditioned data simulation. To implement the EBCrsABC algorithm is simple from here. For EBC, we only accept β_0, β_1, ρ, if $(X_{(t}|X_0^*, t^*, a_0^*) = (X_t^*|X_0^*, t^*, a_0^*)$ for all animals, which is very rare. For rsABC, we allow an acceptance for a certain number of matches instead of matching everything.

Data conditioned simulation (steered simulation)

In the unconditioned case above, we sample V and the Bernoulli distribution freely to determine the final value of X_t. Whereas here in the data conditioned simulation we use the observed X_t to steer the simulation so that the simulation is always in agreement with the observation. To achieve this, we consider all 4 possible cases of (X_0, X_t) for every individual: $(X_0, X_t) = (0, 0)$, $(X_0, X_t) = (0, 1)$, $(X_0, X_t) = (1, 0)$, and $(X_0, X_t) = (1, 1)$. We will be referring to the amount of steering required for each individual as weight, which is the importance sampling weight in section 3.1. The total importance sampling weight is the product of weights contributed by every individual. The total importance sampling weight is also an estimate of the likelihood. Recall section 3.1, the two elements of the data conditioned simulation are data augmentation and importance sampling. In simpler models, we can sample from the model directly and then by applying importance sampling to the model, we can ensure a close match between the sampled outcome and the observed data. When the model is more complicated, like we have now, extra steps may be required in order to sample from the model. In the two states CTTIMC case, we require the additional information of the last event time, s, in order to determine the state of X_t. We will see how this is implemented in the data conditioned simulation below.

For $(X_0, X_t) = (0, 0)$ We cannot deduce if an event occurred during $(0, t)$, and therefore place no restriction on the last event time s, and hence no restriction on the sampling of V. The weight contribution from the sampling of V is therefore 1 (integrating over the whole sample space). Given V we solve for s, if $s < 0$, then there no event occurred between interval $[0, t]$, and therefore the weight is 1. If $0 \leq s \leq t$, then we want the last event to be a recovery, and therefore the we choose the last event to be a recovery which has probability $\frac{\rho}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$. The total weight is then $1 \times \frac{\rho}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$. Similarly, it can be deduced that the sampling weight for $(X_0, X_t) = (1, 1)$ is either 1 or $\frac{e^{\beta_0+\beta_1(a_0+s)}}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$ depending on whether $s < 0$ or $0 < s < t$.

For $(X_0, X_t) = (0, 1)$ We can deduce that at least one event occurred during $(0, t)$, and therefore s must be $0 \leq s \leq t$, and therefore $0 \leq V \leq U$, where U is an upper bound. We can find U by substituting s with 0 in (5.1) and solving for U. We know V must lay between the two bounds because of the strictly decreasing property of V with respect to s. Therefore, the weight contributed from the steered sampling of V is $\mathbb{P}(0 < V < U) = \int_0^U e^{-v}dv$. We can calculate this probability explicitly: $\mathbb{P}(0 < V < U) = 1 - exp\left\{\frac{exp\Big(\beta_0+\beta_1(a_0+t)\Big)-exp\Big(\beta_0+\beta_1 a_0\Big)}{\beta_1} + \rho t\right\}$. Given V, we choose the last event to be an infection which has probability $\frac{e^{\beta_0+\beta_1(a_0+s)}}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$. The final weight is then $\int_0^U e^{-v}dv \times \frac{e^{\beta_0+\beta_1(a_0+s)}}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$. Similarly, the sampling weight for $(X_0, X_t) = (1, 0)$ is $\int_0^U e^{-v}dv \times \frac{\rho}{e^{\beta_0+\beta_1(a_0+s)}+\rho}$.

To find the total sampling weight for the whole data set, we perform the above steered sampling for each observation. Let p_i denote the sampling weight for the i^{th} observation. The total sampling weight, denoted by P, of the data given parameters $\beta_0, \beta_1,$ and ρ is $P = \prod_1^N p_i$, where N is the total number of data points in the observed data. N may be greater than the number of animals. Because of the Markov property, if an animal had more than two observations, we can take any pair combination of the observations and consider them as an independent observation from another animal. Note that by design, we have ensured that $\pi(\boldsymbol{X}^*|\boldsymbol{X}, \beta_0, \beta_1, \rho) = 1$. We will see how the steered simulation is applied in the next section.

5.3 Algorithm implementation

In this section, we will look at the implementation of three algorithms: rsABC, dcABC, and GIMH. Let $(X_0^{(i)}, X_t^{(i)}, t^{(i)}, a_0^{(i)})$ denote the ith data point, and N denote the data size. We also let $\boldsymbol{X_0} = \{X_0^{(1)}, \ldots, X_0^{(N)}\}$, and $\boldsymbol{X_t}, \boldsymbol{t}, \boldsymbol{a_0}$ are defined similarly. As before, superscript * is used to indicate the observed data. We assume that $\beta_0 \sim U(-8, 0), \beta_1 \sim U(0, 8), \rho \sim U(0, 6)$. Uniform priors are used to indicate the lack of prior knowledge and the

boundaries are chosen based on the stability of the algorithm.

5.3.1 The rsABC algorithm

Algorithm 20 (rsABC algorithm for the TIMC model)**.**

1. Sample β_0, β_1, ρ from their corresponding prior.
2. Take $\boldsymbol{X_0^*}, \boldsymbol{t^*}, \boldsymbol{a_0^*}$ and sampled β_0, β_1, ρ in the unconditioned data simulation described in Section 5.2.4 to generate $\boldsymbol{X_t}$.
3. If $\boldsymbol{X_t} = \boldsymbol{X_t^*}$, then we accept β_0, β_1, ρ. Otherwise, we reject.
4. Repeat 1. to 3. until a desired number, m, of iterations are completed.

During testing, we ran the algorithm for up to 3^6 iterations without any acceptance, and therefore the rsABC algorithm is excluded in the final comparison.

5.3.2 The dcABC algorithm

Algorithm 21 (dcABC algorithm for the TIMC model)**.**

1. Sample β_0, β_1, ρ from their corresponding prior.
2. Take $\boldsymbol{X_0^*}, \boldsymbol{X_t^*}, \boldsymbol{t^*}, \boldsymbol{a_0^*}$ and sampled β_0, β_1, ρ in the data conditioned simulation described in Section 5.2.4 to estimate the importance sampling weight P.
3. Record $\{\beta_0, \beta_1, \rho, P\}$. We are safe to use P as the estimate for likelihood because by design $\pi(\boldsymbol{X^*}|\boldsymbol{X}, \beta_0, \beta_1, \rho) = 1$.
4. Repeat 1. to 3. until a desired number, m, of iterations are completed.

5.3.3 The GIMH algorithm

We have chosen to use three independent folded normal distributions as the proposal distribution for the three parameters. Let $|N(\mu, \sigma^2)|$ denote the folded normal distribution, which means that the corresponding random variable only lies in $\mathbb{R}^+ \cup \{0\}$. For β_0, we use $-\left|N(\mu, 0.25)\right|$ as the proposal, because $\beta_0 < 0$. For β_1, we use $\left|N(\mu, 0.25)\right|$ as the proposal. Finally, for ρ we use $\left|N(\mu, 0.64)\right|$ as the proposal. We let $q_{\beta_0}(a, b), q_{\beta_1}(a, b)$ and $q_\rho(a, b)$ denote the density functions of the folded normal proposal distributions. Then $q_{\beta_0}(a, b) = f_{\mu=b,\sigma^2=0.25}(a) + f_{\mu=b,\sigma^2=0.25}(-a)$, where $f_{\mu=b,\sigma^2=0.25}(x)$ is the density function of $N(b, 0.25)$. $q_{\beta_1}(a, b)$ and $q_\rho(a, b)$ can be defined similarly. There are two main reasons for choosing the folded normal distribution as the proposal distribution. The folded normal distribution still preserves the symmetrical property. Take $q_{\beta_0}(a, b)$ as an example:

$$\begin{aligned} q_{\beta_0}(a, b) =& f_{\mu=b,\sigma^2=0.25}(a) + f_{\mu=b,\sigma^2=0.25}(-a) \\ =& f_{\mu=a,\sigma^2=0.25}(b) + f_{\mu=a,\sigma^2=0.25}(-b) \\ =& q_{\beta_0}(b, a). \end{aligned}$$

As the result, when it comes to calculating the acceptance probability, the proposal distribution cancels each other out and therefore we save on computation time. The second advantage is that the folded normal distribution allows faster sampling when we only wish to sample from $\mathbb{R}^+$ or $\mathbb{R}^-$ than rejecting samples that fall in the undesired region, which also helps to improve on computation time. As discussed in section 5.2.4, the missing data for implementing the data conditioned simulation (or pseudo-marginal method) are the last event time, s, and the state of the animal at time t, X_t.

Algorithm 22 (The GIMH algorithm for the TIMC model)**.**

1. Initialise $\{\beta_0, \beta_1, \rho\}$ with the mean of their corresponding prior distributions. Although, one could achieve a better initial point, by conducting a short run of the dcABC algorithm. We then calculate the likelihood estimate (importance sampling weight), P, using the initial parameters and the observed data.
2. Propose a new set of parameters, $\{\beta_0', \beta_1', \rho'\}$ from $\{\beta_{0,i}, \beta_{1,i}, \rho_i\}$.
3. Calculate the corresponding likelihood estimate P' using Section 5.2.4.
4. Set $\{\beta_{0,i+1}, \beta_{1,i+1}, \rho_{i+1}\} = \{\beta_0', \beta_1', \rho'\}$ and $P_{i+1} = P'$ with probability $\min\left\{1, \frac{q_{\beta_0}(\beta_0', \beta_{0,i}) q_{\beta_1}(\beta_1', \beta_{1,i}) q_\rho(\rho', \rho_i) \pi(\beta_0') \pi(\beta_1') \pi(\rho') P'}{q_{\beta_0}(\beta_{0,i}, \beta_0') q_{\beta_1}(\beta_{1,i}, \beta_1') q_\rho(\rho_i, \rho') \pi(\beta_{0,i}) \pi(\beta_{1,i}) \pi(\rho_i) P_i}\right\}$, otherwise set $\{\beta_{0,i+1}, \beta_{1,i+1}, \rho_{i+1}\} = \{\beta_{0,i}, \beta_{1,i}, \rho_i\}$ and $P_{i+1} = P_i$.
5. Repeat 2. to 4. until a desired sample size, m say, is reached.

5.3.4 Results

Our analysis is delivered in two sections. The first one we test the dcABC algorithms and the GIMH algorithm on simulated data, so that we can see how the two algorithms compare both in terms of efficiency and accuracy. We then use both algorithms to analyse the actual data described in section 5.2.

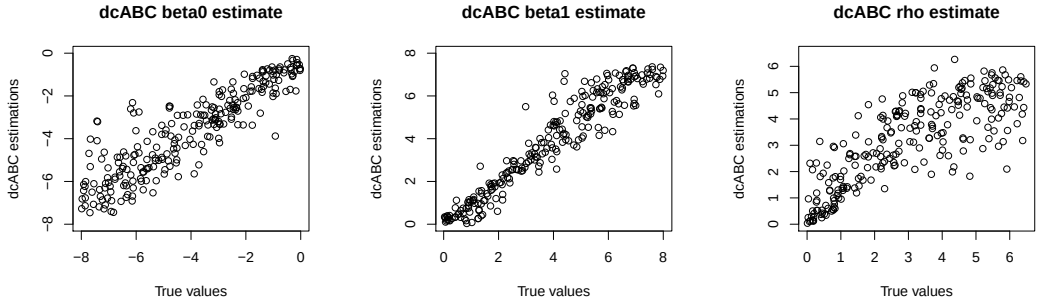
On simulated data

We draw 250 distinct sets of parameters from their corresponding prior distributions and for each set of parameters, a data of size 500 is generated.

We analyse these data using both the dcABC algorithm and the GIMH algorithm with group size 20 and sample size 50000. Out preliminary test showed that the rsABC algorithm showed 0 acceptance for the same sample size, and therefore we leave out the rsABC algorithm.

Figure 5.1: dcABC estimations v.s. True values

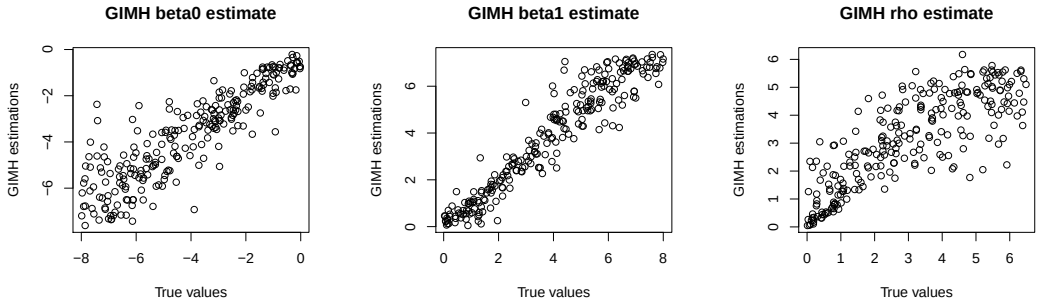
(a) β_0 (b) β_1 (c) ρ



In Figure 5.1, we can observe clear linear correlations in all three parameters. β_0 has higher accuracy when the true $\beta_0 > -4$. β_1 has the most consistent estimations with one obvious outlier, but the results seem best when the true $\beta_1 < 4$. ρ looks like the least correlated parameter between the estimation and the true value. It can be observed that the algorithm does not perform well for $\rho > 4$.

Figure 5.2: GIMH estimations v.s. True values

(a) β_0 (b) β_1 (c) ρ



In Figure 5.2, we can observe very similar trend to the output of the dcABC algorithm. β_0 estimations perform well for true $\beta_0 > -4$, β_1 estimations perform well overall but worsen slightly when true $\beta_1 > 4$, and estimations of ρ are lease correlated of the three with better correlation when true $\rho < 4$.

In order to assess the performance of the algorithms, we fitted a simple linear model (an intercept term, and one coefficient) between the estimations and true values for each parameters from both algorithms. For a good parameter estimating algorithm, we should observe that the intercept term is close to 0, the coefficient is close to 1, and R^2 value is close to 1. The models fitted is presented in Table 5.2.

Table 5.1: Reporting the fitted linear models between the estimated values and the true values.

	intercept	regression coefficient	adj. R^2
β_0 from dcABC	-0.5880	0.7584	0.7975
β_0 from GIMH	-0.6333	0.7555	0.7828
β_1 from dcABC	0.0473	0.9701	0.9141
β_1 from GIMH	0.0670	0.9653	0.9079
ρ from dcABC	1.1199	0.6822	0.6460
ρ from GIMH	1.0518	0.6893	0.6628

It can be observed in Table 5.2 that there is very little difference in terms of the models fitted for both the dcABC algorithm and the GIMH algorithm. Both algorithms have very similar model coefficient estimation as well as the adjusted R^2 value.

In terms of the computation time, the two algorithms are comparable with mean time elapsed of dcABC being 18,582s and mean time elapsed GIMH being 17,529s.

On the original data

Again, we repeat both algorithms 250 times with the group size 20 and sample size 50000.

Figure 5.3: dcABC estimations v.s. GIMH estimations

(a) β_0 (b) β_1 (c) ρ

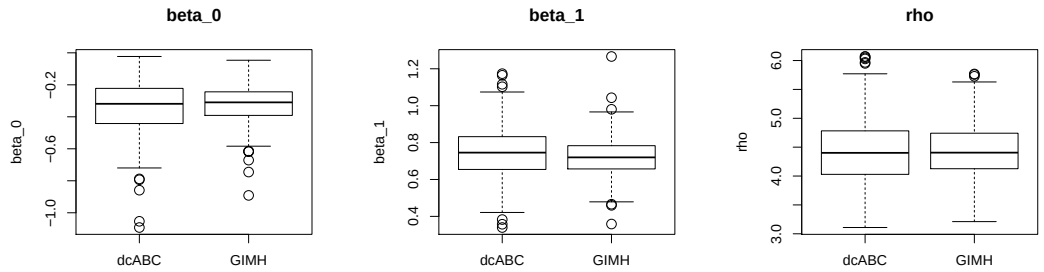


Table 5.2: Summary of output from the dcABC algorithm and the GIMH algorithm.

	$\mathbb{E}(\widehat{\beta_0})$	$Var(\widehat{\beta_0})$	$\mathbb{E}[\widehat{\beta_1}]$	$Var(\widehat{\beta_1})$	$\mathbb{E}[\widehat{\rho}]$	$Var(\widehat{\rho})$	Time elapsed (s)
dcABC	-0.3442	0.0311	0.7476	0.0210	4.4479	0.3595	30,912
GIMH	-0.3224	0.0160	0.7214	0.0129	4.4396	0.2067	29,075

In Figure 5.3, the dcABC algorithm has consistently wider inter quartile range than the GIMH algorithm. However, both algorithm agree well enough on their mean estimate. We can further confirm this in Table 5.2. The means of the parameter estimates for both algorithms are in close agreement. However, the Monte Carlo error is clearly higher for the dcABC algorithm. Again, the GIMH algorithm has slightly lower computation time. The big discrepancy between the computation time for the real data and the simulated data is due to the data size. The real data has just over 1000 data points, and we only used 500 data points in the simulated data case, which explains the increase in the computation time.

5.4 Chapter Conclusion

From the results above we can see that the dcABC algorithm and the GIMH algorithm perform almost equally well especially in the simulated data case. However, the GIMH algorithm achieved a more consistent parameter estimates for the real data case as well as the being more time efficient.

Chapter 6

Ricker Model

6.1 Introduction

In this chapter, we will look at a completely different class of problem, stochastic dynamic model, and in particular the Ricker model. The Ricker model has been well studied under the synthetic likelihood context in [Wood, 2010] and under the semi-automatic ABC context in [Fearnhead and Prangle, 2012]. We will see how the data conditioned simulation can be adapted in this situation and eventually make estimations on its parameters. This is probably the most challenging problem of the three examples.

This chapter follows a similar structure as before. We begin by introducing the Ricker model in Section 6.2, follow by an exploratory research of the Ricker model in Section 6.3. We dedicate Section 6.4 for finding the suitable summary statistics. Unconditioned simulation and data conditioned simulation are discussed in Section 6.5, and follow by the implementation of the dcABC algorithm and the GIMH algorithm in Section 6.6. Finally, we have the chapter conclusion on Section 6.7.

6.2 Model description

The Ricker model is a non-linear difference equation which was devised by [Ricker, 1954] to model stock and recruitment in fisheries. The Ricker model, equation (6.1), can be used to predict the expected number of fish population in a fishery. Due to its non-linearity, it exhibits a chaotic behaviour for some choices of parameter values, and in which case the model is very sensitive to small perturbations of the initial condition.

$$N_{t+1} = rN_t e^{-N_t} \text{ for } t = 0, 1, 2, 3, \ldots \tag{6.1}$$

A noisy version of the Ricker model incorporating a Gaussian error term and a Poisson observation process is considered in [Wood, 2010] and [Fearnhead and Prangle, 2012] where MCMC with synthetic likelihood and semi-automatic ABC, respectively, are used to estimate the model parameters.

The noisy version of the Ricker model is described as follow:

$$\begin{aligned} N_t &= rN_{t-1}e^{-N_{t-1}+e_{t-1}} \\ Y_t &\sim Poisson(\phi N_t) \end{aligned} \tag{6.2}$$

where $e_{t-1} \sim N(0, \sigma^2)$, $\{N_t\}$ is the underlying population dynamics which of primary interest, and $\{Y_t\}$ is the Poissonised observations of the underlying population dynamics $\{N_t\}$. We are interested in estimating the three parameters of the model r, σ, ϕ based on observations $\{Y_t\}$.

The non-linearity in the dynamics of the Ricker model means that applying standard likelihood or data augmentation methods on the plain data would not be appropriate, see Section 6.3 for further details. The problem with the standard likelihood approach is that only $\{Y_t\}$s are observed and the $\{N_t\}$s are not. In order to calculate the likelihood, $\pi(\boldsymbol{Y}|r, \sigma, \phi)$, we are required to calculate $\int \pi(\boldsymbol{Y}|r, \sigma, \phi, \boldsymbol{N})d\boldsymbol{N}$, which is intractable in this case. An initial idea is to apply Monte Carlo integration to estimate the integral,

however, computation of the quantity $\pi(\boldsymbol{Y}|r, \sigma, \phi, \boldsymbol{N})$ is highly variable and often computed to 0 at machine accuracy. The reason for this is because the Ricker model is chaotic. For the same parameters, the overall trajectories between two independently generated $\boldsymbol{N}$ can be very different, again see Section 6.3. The objective of our research is to infer the parameters, $\{r, \sigma, \phi\}$,from the observed $\{y_{T-L+1}, \ldots, y_T\}$, which are assumed to be generated as follow:

Note that, the subscripts of the observed data starts from $T - L + 1$ and ends on T. T is time since reference point time 0 which we assume starts at $N_0 = 1$, and L is the total number of observations made.

Given r, σ, ϕ and initial population size, N_0 (we assume that $N_0 = 1$), the population process, $\{N_t\}$, is generated for T periods with $\boldsymbol{Y}$ observed on the last L time points (the observation period).

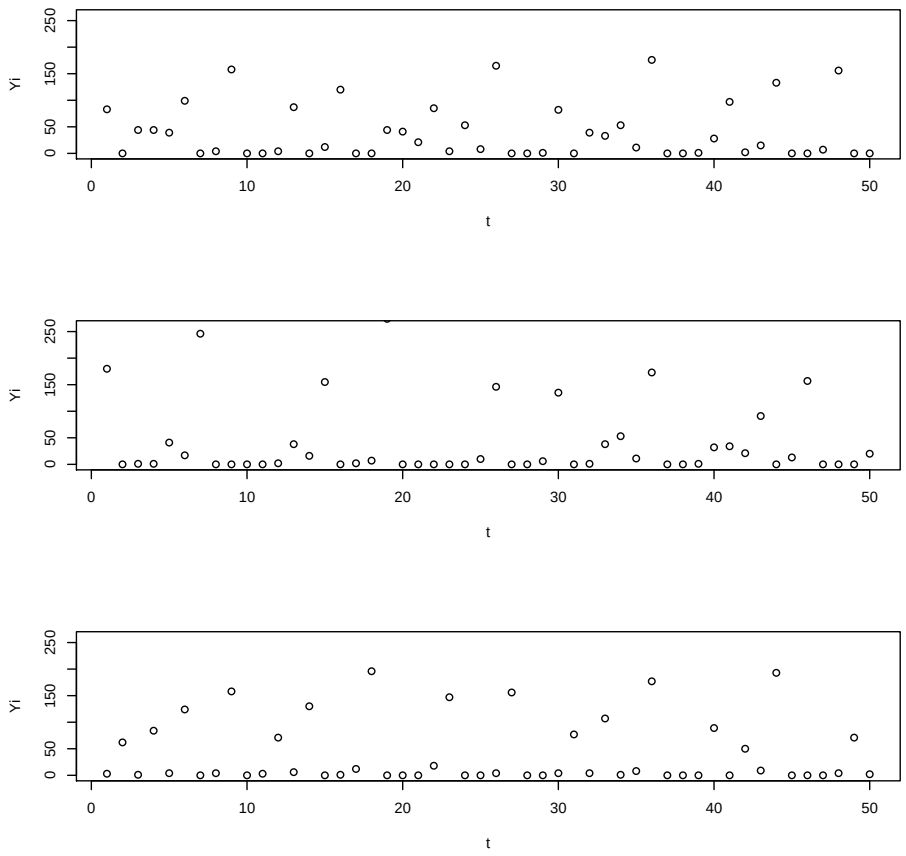
6.3 Exploratory research

Upon examining the model, the standard rsABC algorithm seems straightforward to implement for this model as it is easy to simulate from the model. However, the Ricker model is chaotic, and therefore even given the true parameters, the trajectories of the population in two simulations could be very different. Hence, ABC algorithms based on matching observed data term by term is unlikely to yield fruitful result, and therefore carefully chosen summary statistics is required. We will demonstrate this problem in the graph below.

Figure 6.1 are three sets of observations generated using equation (6.1) with the same parameters and initial conditions. It can be observed that the peaks and troughs in each plot do not coinside at the same time point. The similarity in trajectories is crucial if we were to employ point by point data matching ABC algorithms, as we would expect the outcome of the model

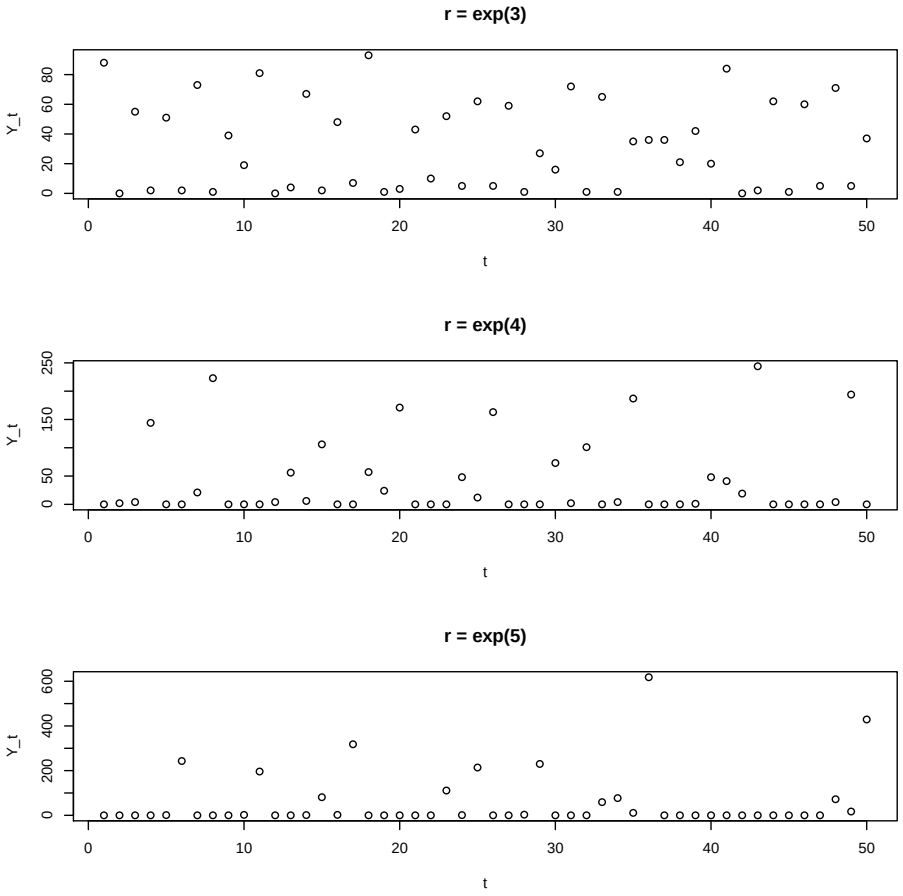
with the same parameters and initial conditions to follow similar trajectories, i.e. the peaks and troughs coinsides, so that algorithms such as rsABC will have a good chance of accepting the parameters when they are close to the true values. However, with the mismatching trajectories, the vanilla application of the rsABC algorithms and alike would be unlikely to yield a meaningful results. Further, let the first plot of Figure 6.1 represent the observed data, and given the $\boldsymbol{N}$ of the first plot, the probabilities of producing samples that matches the other two plots are both 0. This means that even with the true parameters, the importance sampling weight assigned to the parameters is 0, which means that the dcABC algorithm would not work either. This statements will become clear in Section 6.5. The main reason for these failure is because of the correlation disparity between data trajectories and the parameters. The disparity is prevalent in most values of parameters, and only improves if σ is close to 0. In the this section, we will focus on examining the dynamics of the model in the view to find better correlated summary statistics to the model parameters.

Figure 6.1: Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25, \phi = 10$, $N_0 = 1$, T = 100 and L = 50



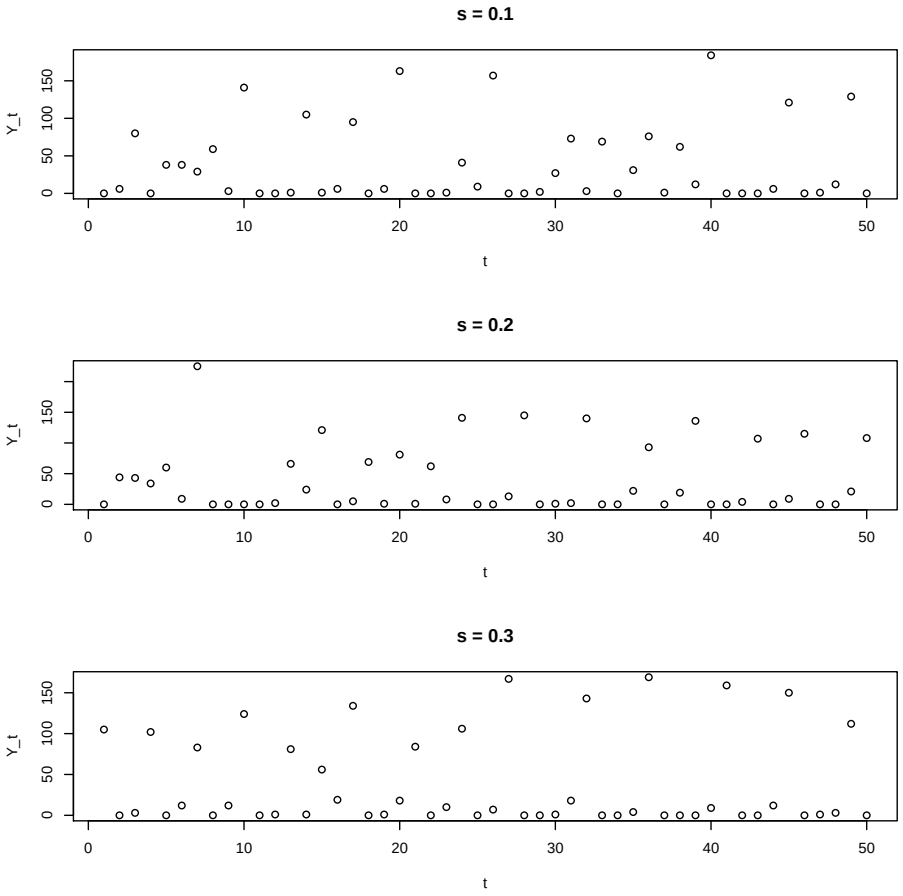
Apart from the disparity between the population trajectories and parameters, there are some common features that can be observed in Figure 6.1. The trajectories stay in roughly the same order of magnitude. The number of peaks and troughs are similar, although some peaks and troughs are less well defined. The total number of 0 observations are similar too. We will next look at three similar graphs (Figures 6.2, 6.3, and 6.4) but each corresponds to varying only one of the three parameters.

Figure 6.2: Three sets of observations simulated with $\sigma = 0.25, \phi = 10$, $N_0 = 1$, T = 100, L = 50, and varying r.



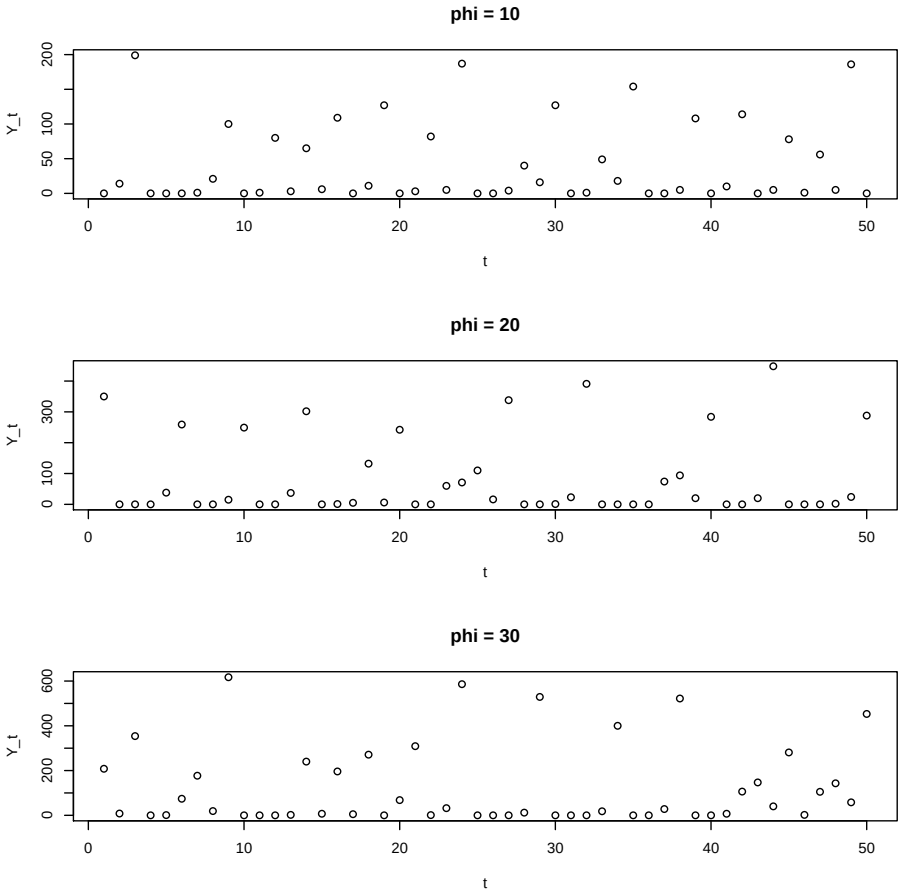
It is clear in Figure 6.2 that as r increases, the number of 0 observations increases and also the length of successive 0 observations. It can also be observed that the as r increases, the maximum of the observations increases.

Figure 6.3: Three sets of observations simulated with $r = e^{3.8}, \phi = 10$, $N_0 = 1$, T = 100, L = 50, and varying σ.



In Figure 6.3, no clear pattern is observed as we vary σ.

Figure 6.4: Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25$, $N_0 = 1$, T = 100, L = 50 and varying ϕ.



In Figure 6.4, the most distinctive feature is the magnitude of the observations generally increases as ϕ increases.

We will investigate these features more formally in Section 6.4, and some of which play an important role in the development of the data conditioned simulation algorithm. From the model there is a clear structure with with dependence between successive observations. i.e. $N_{t+1} = rN_t e^{-N_t + e_t}$ and $Y_t \sim Poisson(\phi N_t)$. It is a first order Markov system in $\{N_t, Y_t\}$.

Because of the first order underlying dynamic structure, we first focus on investigating the pair-wise relationship between the Y_t, and Y_{t-1}. We will attempt to establish if and how each parameter affects the behaviour of the observations in the following sections by firstly investigating the link between the deterministic version of the Ricker model and the noisy version of the Ricker model (Section 6.3.1), and then the Poissonised observations of the Ricker model (Section 6.3.2).

6.3.1 The Ricker model without Poissonised observation

As our main interest is in estimating r and σ, the natural thing to do is to investigate the underlying dynamics first.

Bifurcation Diagram

A bifurcation diagram is a diagram which plots the long run values of a dynamical system against the value of parameters [Arrowsmith and Place, 1990]. Bifurcation diagram provides an insight into the long run behaviour of a dynamical system, and it can help to identify if a certain parameter makes the dynamical system converge, diverge, or be periodic.

Figure 6.5 is the bifurcation diagram of the deterministic Ricker model with starting population size of 1, i.e. $N_0 = 1$. For each value of r, 300 iterations were calculated and the last 100 iterations are used for the plot, assuming that 200 iterations are sufficient for the Ricker model to settle into its long run behaviour.

It can be observed from the diagram that most values of r fall in the chaotic region (many dots scattering in vertical direction), and only small proportion of r is either convergence (only one point) or periodic (a few points).

It is also worth noting that the maximum value that the Ricker model can achieve for any given value of r is r/e, which can be explained by apply simple calculus on the equation $y = rxe^{-x}$.

Figure 6.5: Bifurcation diagram of the deterministic Ricker model, $r \in [1, 60], N_0 = 1$

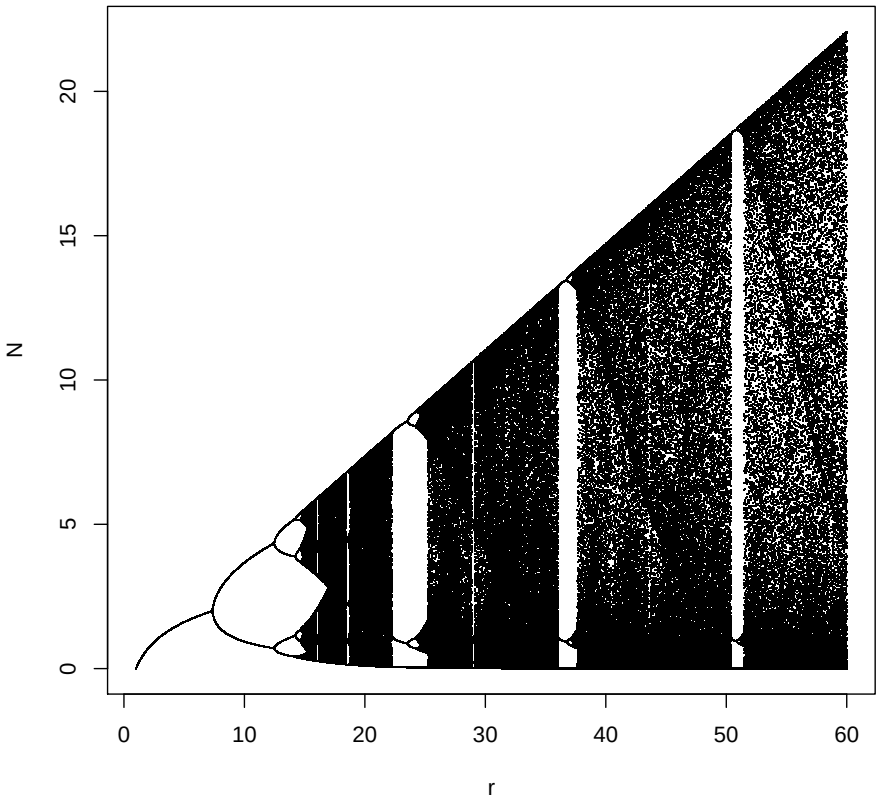
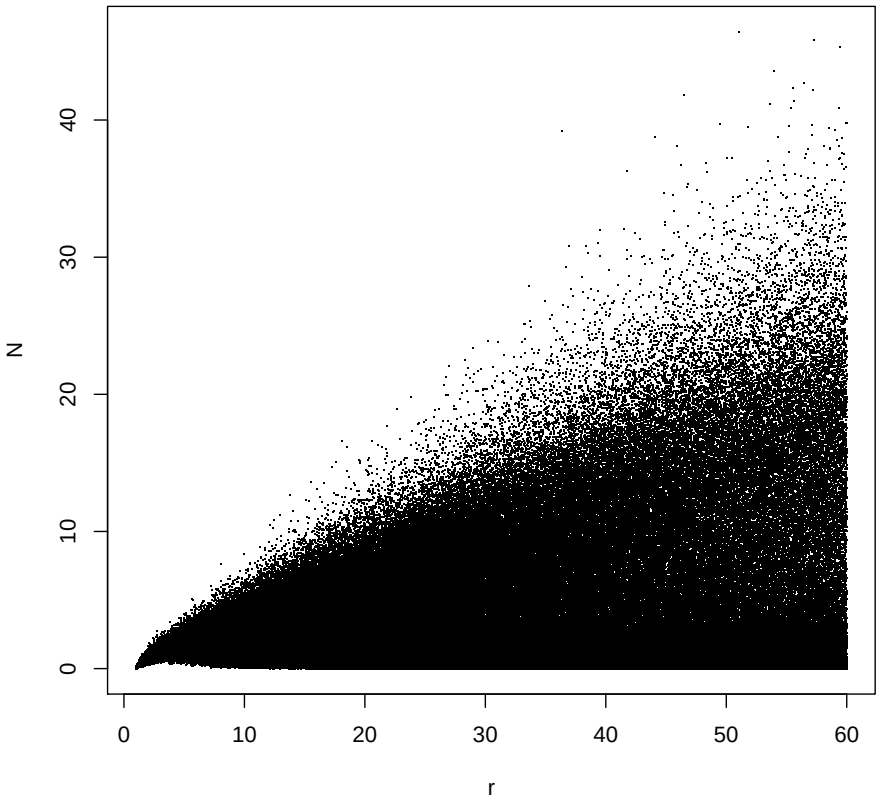


Figure 6.6 is the bifurcation diagram of the stochastic Ricker model. It can be observed that the transition from periodicity of chaos is no longer clear and everything just merged into one. However, an obvious trend is that as r increases, the maximum value of N increases. This is expected, as the random error term introduced forces the dynamical system to break out from

its long run behaviour, and therefore values which do not belong to the long run behaviour of a given r, can appear in the cycle which explains the 0's we are getting in the Monte Carlo integration in Section 6.2

Figure 6.6: Bifurcation diagram of the Ricker model with a Gaussian error term, $r \in [1, 60], \sigma = 0.25$



Phase space

As the bifurcation diagram has provided limited information as to how the Gaussian error term affects the undying dynamics, we turn to the phase space

of the Ricker model.

A phase space is a space which represents all possible states of a dynamical system, and in this case, the phase space of the deterministic Ricker model can be represented by $y = rxe^{-x}$, i.e. the Ricker model with the subscript taken out. Figure 6.7 show how the space would look like given various values of r. Combined with figure 6.6, we can conclude that when $r = e^{1.61}, e^{3.22}$, the deterministic Ricker model is periodic, and when $r = e^{3.81}, e^{4.17}$ the Ricker model is most likely to be chaotic (we cannot be certain until verified with the formal definition of a chaotic system). As well as the changes in the dynamics, different values of r change the shape of the phase diagram.

Figure 6.7: Phase diagram of the deterministic Ricker model with various values of r and $N_0 = 1$

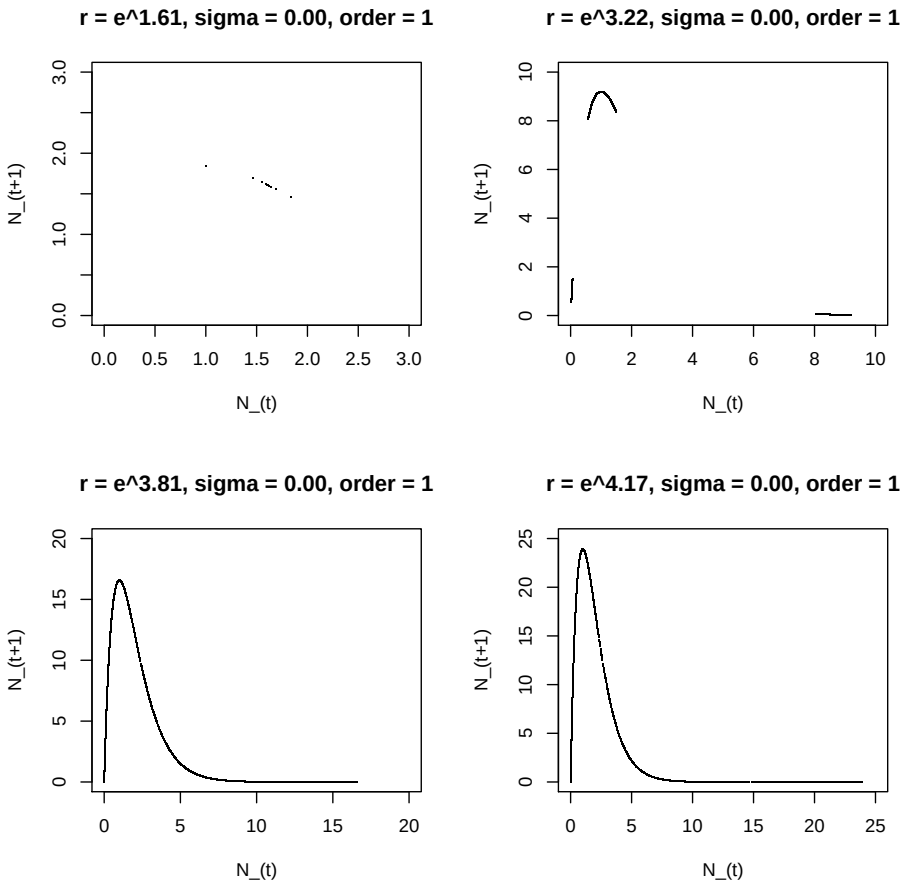


Figure 6.8: Phase diagram of the stochastic Ricker model with various value of r with $\sigma = 0.5$ and $N_0 = 1$

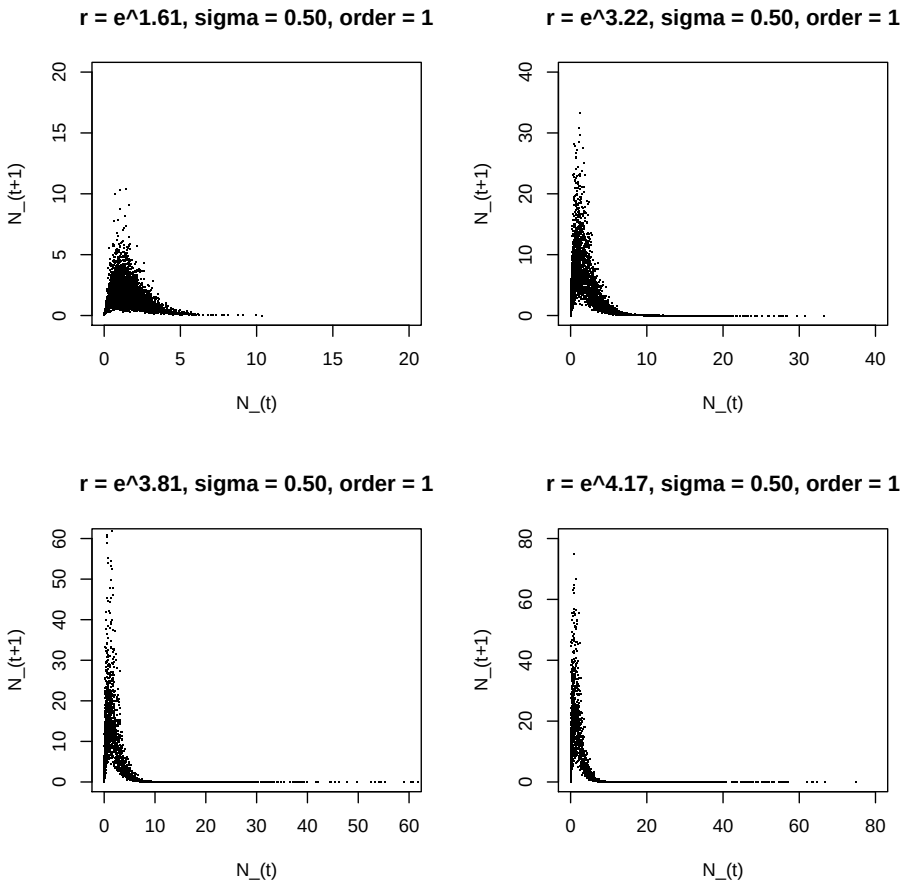


Figure 6.9: Phase diagram of the stochastic Ricker model with various value of σ with $r = 45$ and $N_0 = 1$

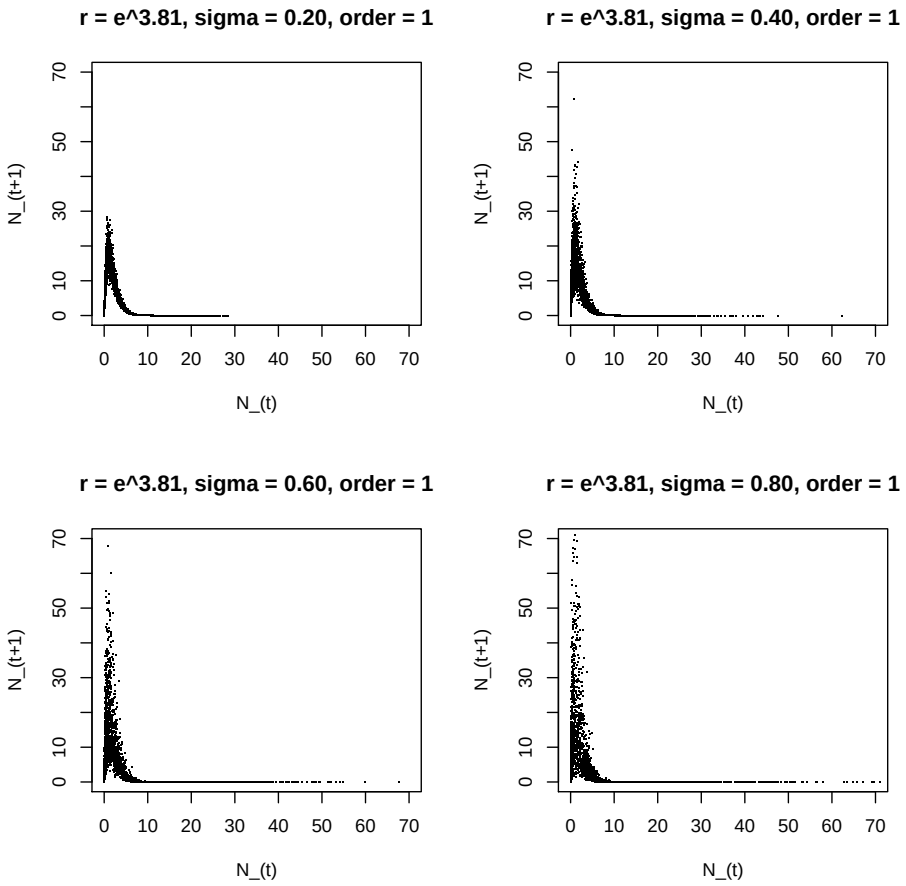


Figure 6.8 and Figure 6.9 are phase diagrams of the stochastic Ricker model for various values of r and σ. Visual inspection of the shape of the plots are informative of how changes in r and σ affects the dynamics. The notable differences in the two groups of plots is that as σ increases, the gap under the graph closes up whereas as r increases, the gap under the graph expands, even though both increase in r and σ increases the range of the values in the plot.

On the log scale

Inspecting the Ricker model in log scale seems natural because of the presence of the exponential function and the multiplication only formulation.

If we take log of the equation $y = rxe^{-x}$, and rename $\log(y)$ as z we get:

$$z = \log(r) + \log(x) - x \tag{6.3}$$

Differentiate with respect to x:

$$\frac{dz}{dx} = \frac{1}{x} - 1 \tag{6.4}$$

Equation (6.3) and (6.4) show that:

- The maximum value of z is $\log(r) - 1$ and occurs at $x = 1$
- $\frac{dz}{dx}$ is independent of r.

Equation (6.4) is particularly important, because we know that the shape of the log of the Ricker model is independent of r, and therefore we can isolate the effect of the error term by examining the shape of the log of the Ricker model. Also, we know that $\log(r)$ is linked to the maximum value of z.

Figure 6.10: Phase diagram of the stochastic Ricker model on the log scale for fixed $\sigma = 0.5$ and $N_0 = 1$, and various value of r

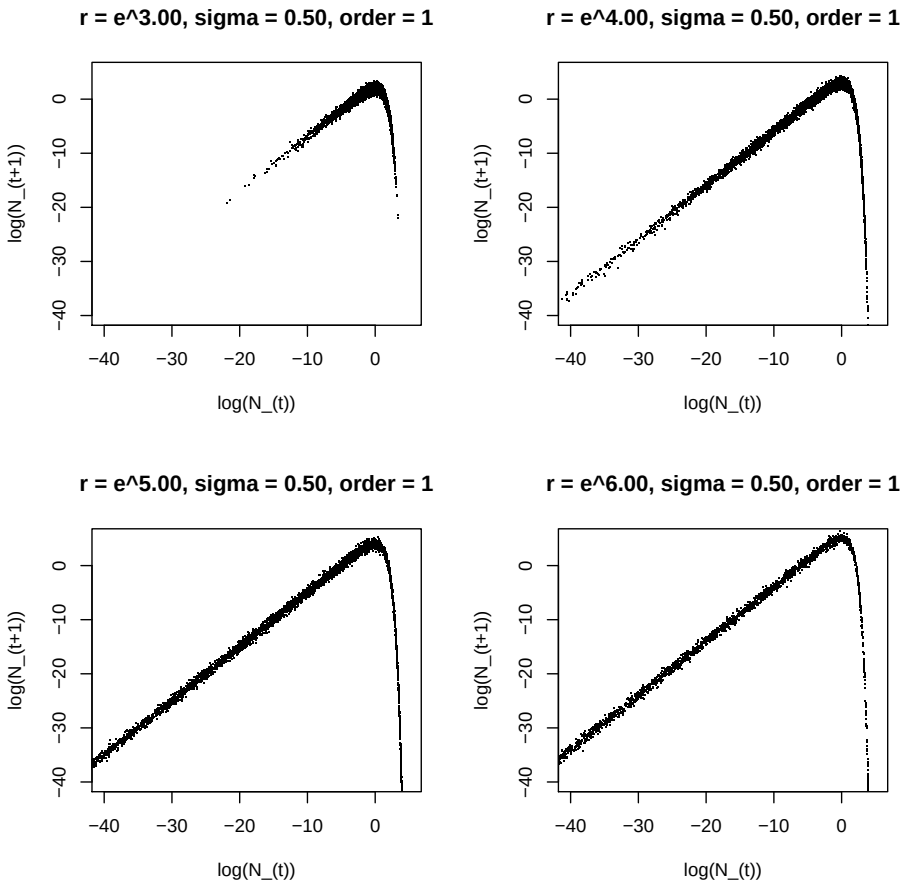
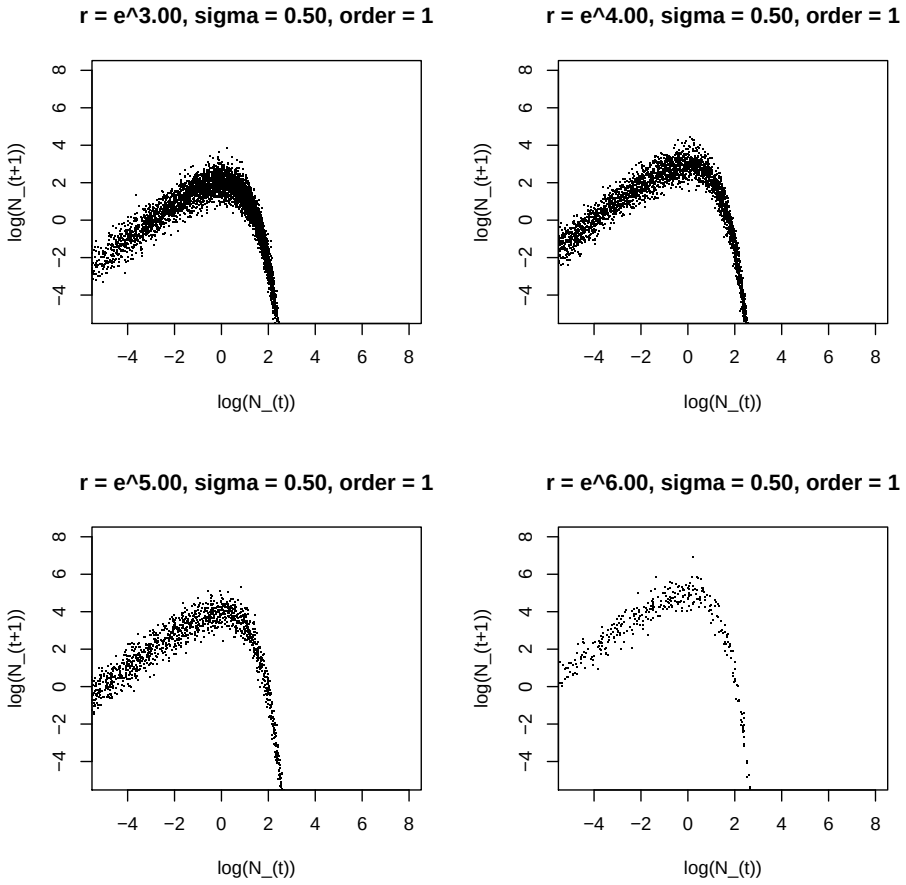


Figure 6.11: Phase diagram of the stochastic Ricker model on the log scale with various value of r with $\sigma = 0.5$ and $N_0 = 1$ zoomed in at the turning point.



Figures 6.10 and 6.11 both demonstrate how r affects the phase diagram on the log scale. It can be observed in Figure 6.10 that as r increases, the "tails" extend further into the negative value, i.e. z or $\log(N_t)$ is closer to 0. In Figure 6.11, it can be observed that as r increases the maximum value at the turning point also increases as predicted by our previous assertion. Note that, the width of the "band" stays roughly the same. Here, width of the band is used to mean the vertical spread at a given value of N_t.

Figure 6.12: Phase diagram of the stochastic Ricker model on the log scale for various value of σ and fixed $r = e^4$, $N_0 = 1$

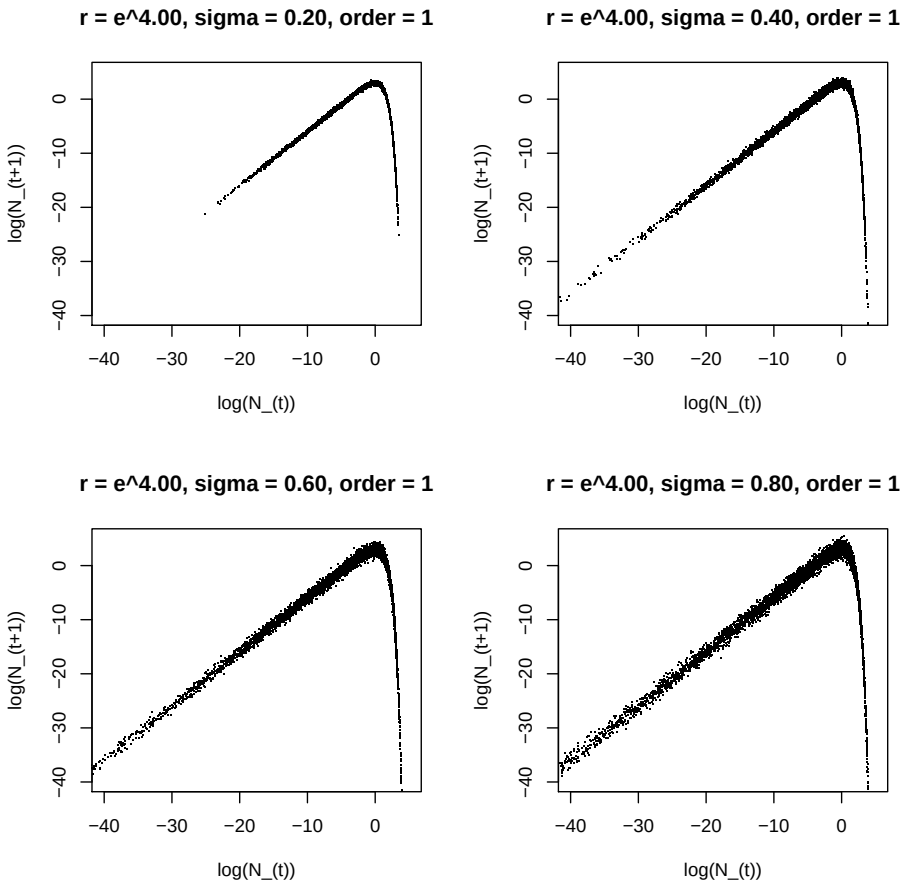
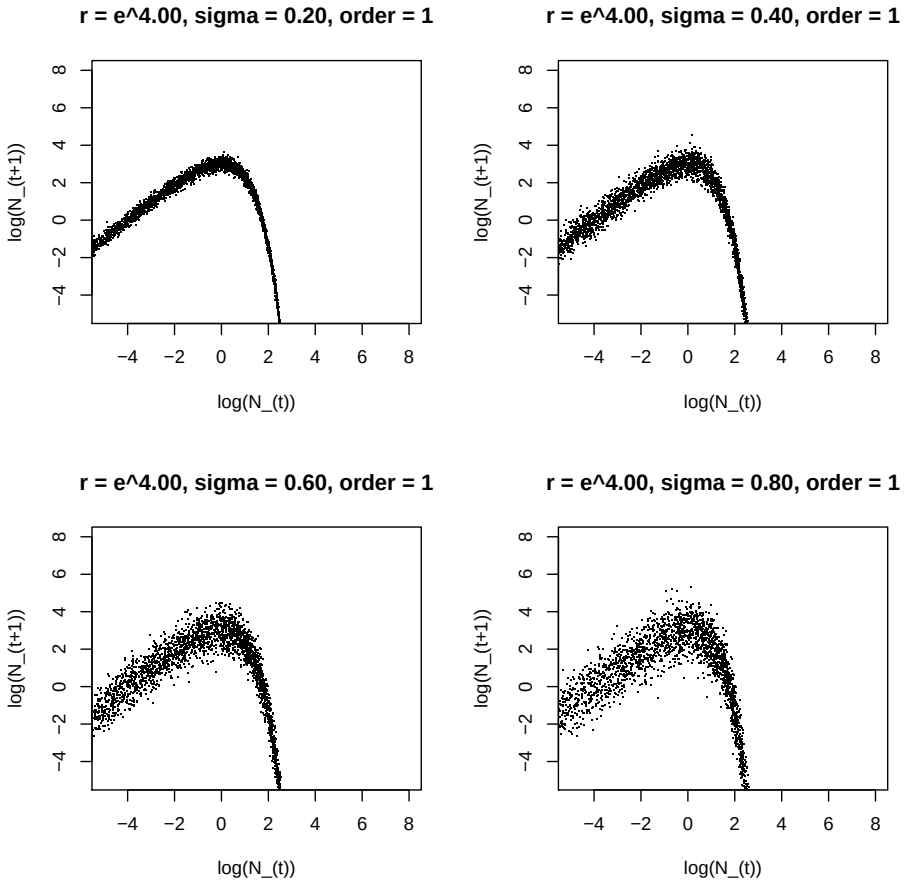


Figure 6.13: Phase diagram of the stochastic Ricker model on the log scale for various value of σ and fixed $r = e^4$, $N_0 = 1$ zoomed in at the turning point.



Figures 6.12 and 6.13 both demonstrate how σ affects the phase diagram on the log scale. In Figure 6.12, increases in σ also extend the "tails" of the diagram. However, in Figure 6.13, it shows a more fundamental difference. The turning point stays at roughly the same level, but the band widens as σ increases, which reflecting greater variability.

These observations pave the way to successfully developing ABC algorithm.

However, we do not observe $\{N_t\}$ but a Poissonised version. Therefore we will see how these features are translated or lost through Poissonised observation. We will go back to the higher order behaviour again in the Poissonised observation section.

6.3.2 The Ricker model with Poissonised observation on the log scale

In order to avoid the singularity point, log(0), we consider $Y_t + 1$ in plotting the diagrams (the Poissonised observation + 1). Let $Z_t = log(Y_t + 1)$, and we also compare the plot of Z_t against its corresponding N_t.

Figure 6.14: Z_{t+1} plotted against Z_t for various values of r, σ with $\phi = 10$

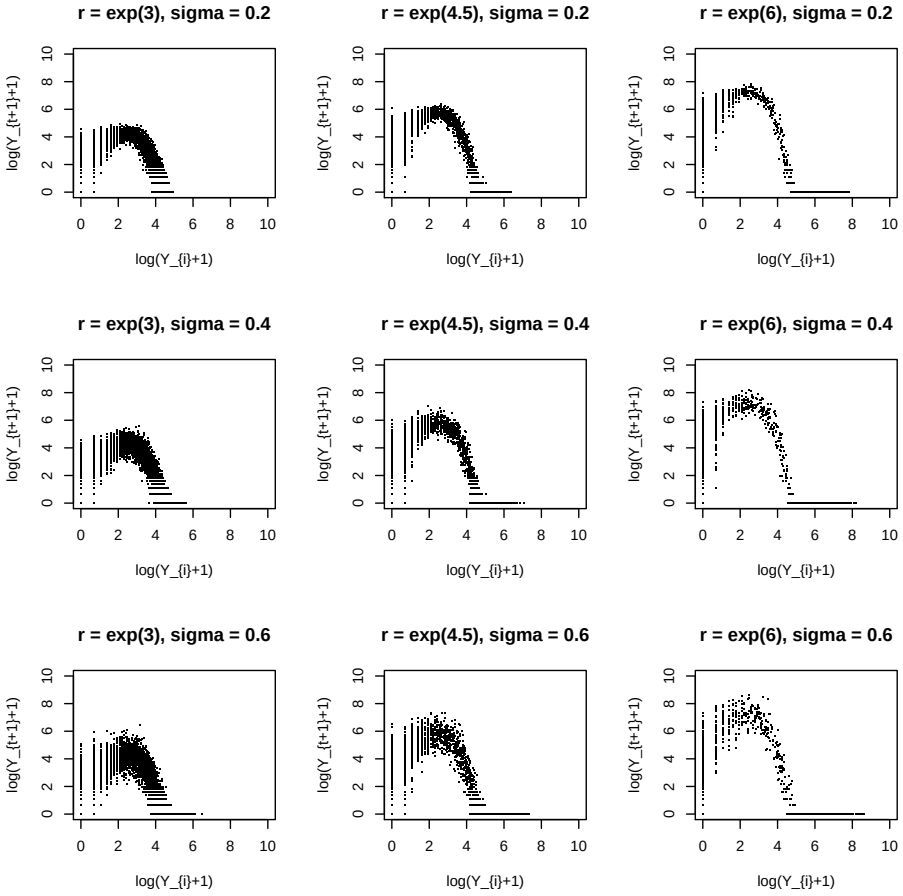
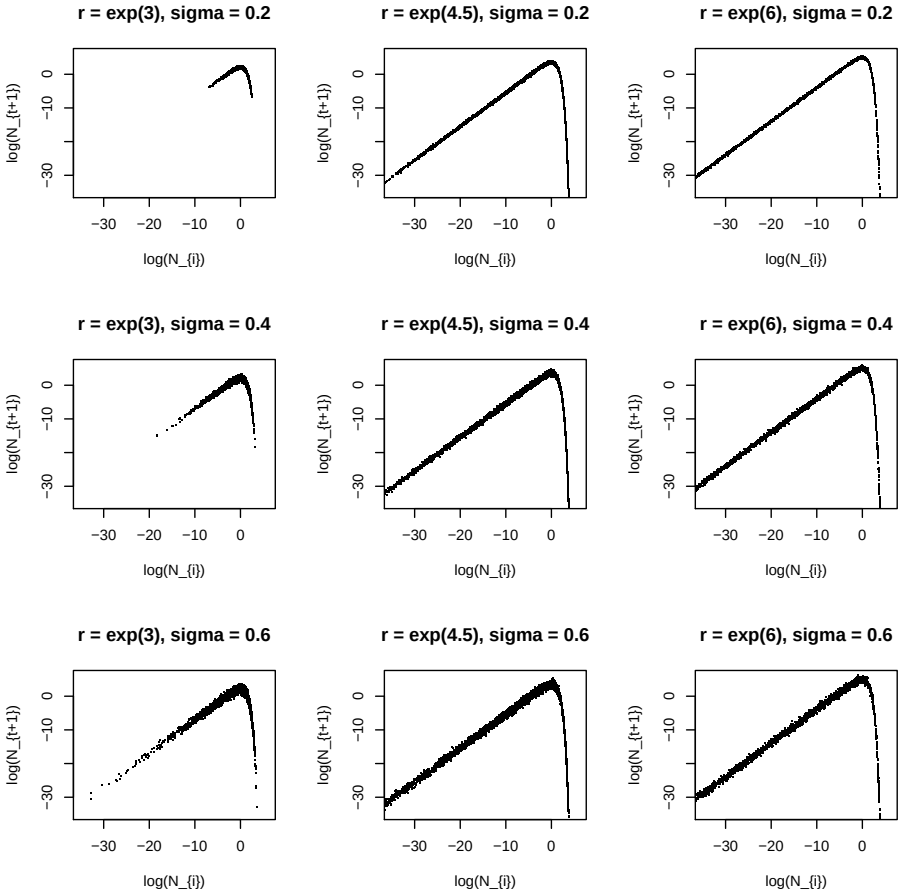


Figure 6.15: Corresponding $log(N_{t+1})$ plotted against $log(N_t)$ for Figure 6.14



Comparing between Figure 6.14 and Figure 6.15, we can see that the 0's in Y_t corresponds to small N_t's. However, we lose information on the magnitude of N_t.

From Figure 6.14, we can observe the following features:

As r increases,

1. The maximum value of Z_{t+1} at $Z_t = 0$ increases.

2. The maximum value of Z_t at $Z_{t+1} = 0$ increases.

3. The minimum value of $Z_t > \log(\phi + 1)$ at $Z_{t+1} = 0$ increases.

As σ increases,

1. The maximum value of Z_{t+1} increases.

2. The maximum value of Z_t at $Z_{t+1} = 0$ increases.

3. The minimum value of $Z_t > \log(\phi + 1)$ at $(Z_{t+1} = 0)$ increases.

4. The range of values of Z_{t+1} near $Z_t = \log(\phi + 1)$ increases.

$Z_t = \log(\phi + 1)$ is where the expected theoretical maximum should occur. This is a direct result from Section 6.3.1, which shows that the expected theoretical maximum of z occurs at $x = 1$, i.e. $N_t = 1$. We also know that $\mathbb{E}(Y_t) = \phi N_t$. Therefore, the expected maximum of $\mathbb{E}(Y_{t+1})$ occurs when $Y_t = \phi$. Both log and $+1$ are monotonic functions, which preserves the order of the function domain in the range. Therefore, when Y_{t+1} reaches its maximum, both $Y_{t+1}+1$ and Z_{t+1} are at their maximums too. The maximum value of Y_{t+1} can be obtained by taking the expectation of $Poisson(\phi N_{t+1})$ when $N_t = 1$, which is $\phi r e^{-1}$.

We are interested in the behaviour of Z_{t+1} when $Z_t > \log(\phi + 1)$, because before this point the relationship in $\log(N)$ is close to linear, and after this point the graph N_{t+1} plotted against N_t goes into a sharp decline. As it can be observed in Figures 6.10 and 6.12, the magnitude of the decline depends on both r and σ.

Behaviour of Z_{t+1} near the turning point shows that larger r implies a consistently bigger fall, whereas larger σ, implies irregular, steeper, and premature falls in $\{N_t\}$.

Figure 6.16: Z_{t+1} against Z_t for various values of r, σ with $\phi = 30$

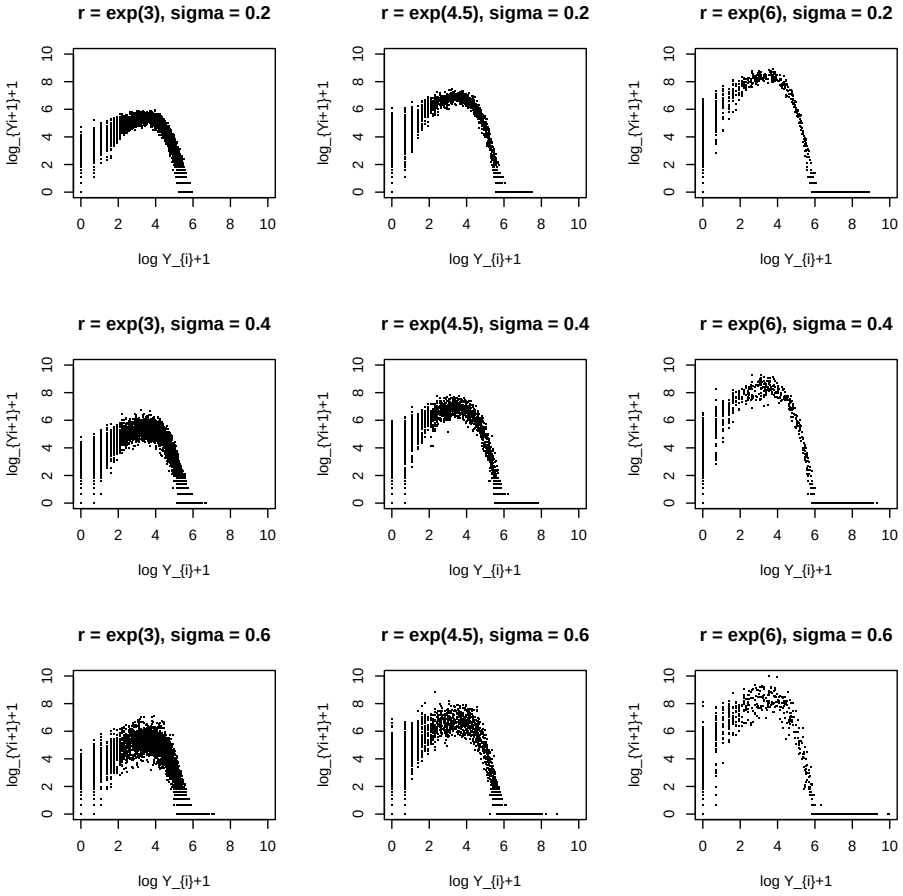
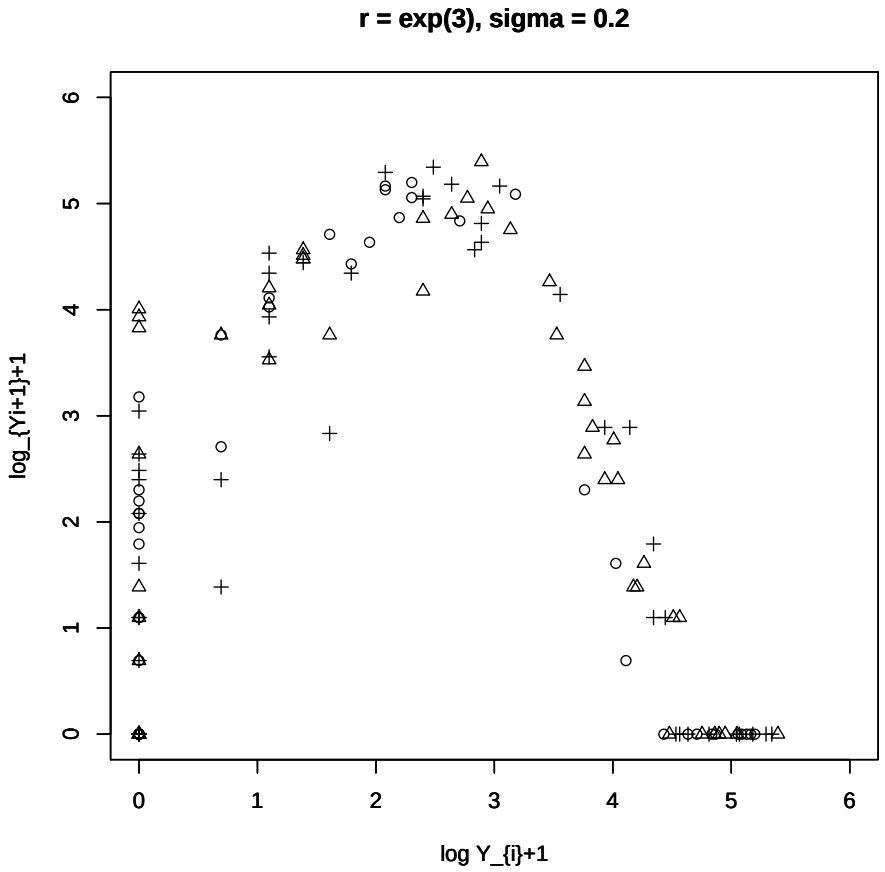


Figure 6.16 has the same combinations of r and σ, but with $\phi = 30$. Theoretically, increase in ϕ will increase the mean of $\boldsymbol{Y}$ given the same $\boldsymbol{N}$, reduce the number of 0 observations in Y as the result a better reflection on the underlying dynamics of $\boldsymbol{N}$. This conjecture can be observed by comparing between Figure 6.14 and Figure 6.16.

Finally, Figure 6.17 has the same data sets as Figure 6.1, but plotted on the log scale as described here. It can be observed that the three data sets do indeed closely match each other.

Figure 6.17: Three sets of observations simulated with $r = e^{3.8}, \sigma = 0.25, \phi = 10$, $N_0 = 1$, T = 100 and L = 50



6.3.3 Higher order dynamics

We have studied the first order dynamic structure of the Ricker Model. We will now inspect the higher order behaviour as implied in [Wood, 2010] and [Fearnhead and Prangle, 2012] to gain further insight of the model. We start by looking at the general behaviour of $N_{t+2}, N_{t+3}, N_{t+4}$ and N_{t+5} against N_t in both deterministic and stochastic case, and then we will explore deeper

into second order dynamics.

Figure 6.18: Phase diagram of the deterministic Ricker model with different orders and fixed r.

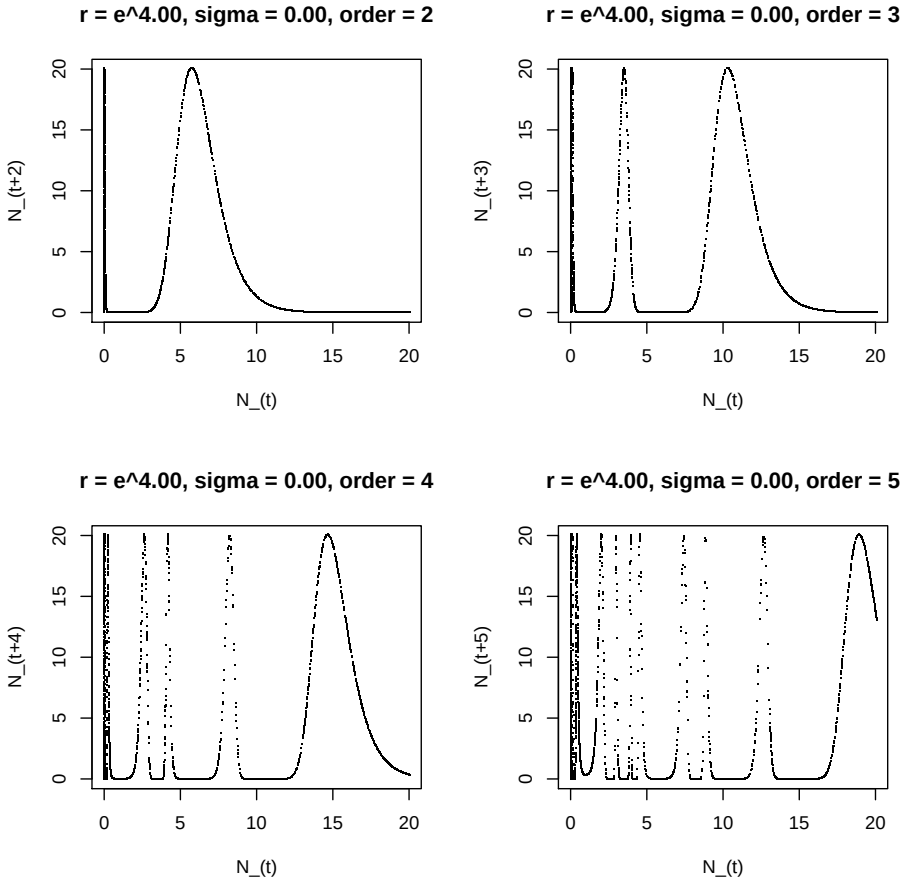
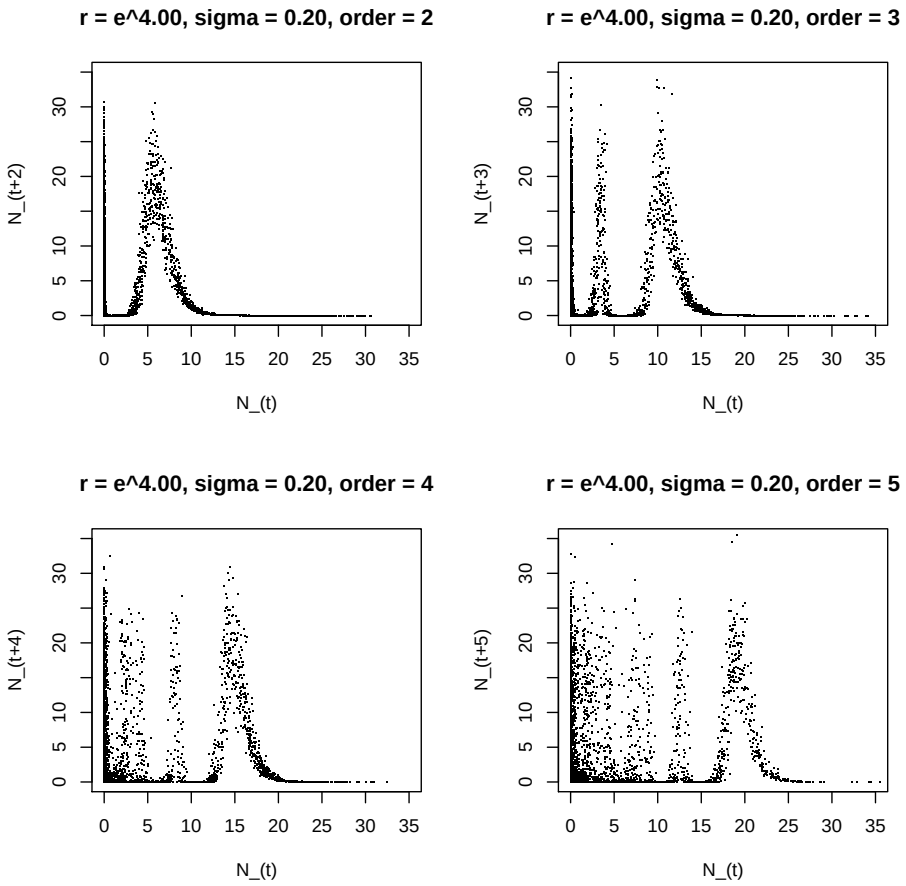


Figure 6.19: Phase diagram of the stochastic Ricker model with different orders and fixed r and σ.



It can be observed in Figure 6.18 that increase in order creates more peaks in the phase diagram in the deterministic Ricker model. We observe similar behaviour in Figure 6.19 for the stochastic model. We also observe that in the stochastic model that the line is more diffused similar to what we have observed before in the first order case, and as the order increases, it becomes harder to distinguish between peaks, see the 5^{th} order case in Figure 6.19.

We will now look at the second order dynamics in more details. We will

proceed as before by writing down the general equation which represents the deterministic second order dynamics and then we will also find its first order derivative. Recall that $N_{t+1} = rN_t e^{-N_t}$, then

$$\begin{aligned} N_{t+2} &= rN_{t+1}e^{-N_{t+1}} \\ &= r(rN_t e^{-N_t})e^{-(rN_t e^{-N_t})} \\ &= r^2 N_t e^{-N_t(1+re^{-N_t})}. \end{aligned} \tag{6.5}$$

To make the algebra easier to read, again, we rewrite the above equation as: $y = r^2 x e^{-x(1+re^{-x})}$, and then we differentiate to get the first derivative:

$$\begin{aligned} \frac{dy}{dx} =& \frac{d}{dx} r^2 x e^{-x(1+re^{-x})} \\ =& r^2 e^{-x(1+re^{-x})} + r^2 x \frac{d}{dx} e^{-x(1+re^{-x})} \\ =& r^2 e^{-x(1+re^{-x})} + r^2 x e^{-x(1+re^{-x})}\left(1 + re^{-x} - rxe^{-x}\right) \\ =& r^2 e^{-x(1+re^{-x})}\left(1 - x - rxe^{-x} + rx^2 e^{-x}\right). \end{aligned} \tag{6.6}$$

In order to find the turning points of y, we solve $\frac{dy}{dx} = 0$:

$$\begin{aligned} & r^2 e^{-x(1+re^{-x})}\left(1 - x - rxe^{-x} + rx^2 e^{-x}\right) = 0 \\ \Rightarrow & e^{-x(1+re^{-x})}\left(1 - x - rxe^{-x} + rx^2 e^{-x}\right) = 0 \\ \Rightarrow & e^{-x(1+re^{-x})}(1-x)(1-rxe^{-x}) = 0 \end{aligned} \tag{6.7}$$

$x = 1$ is always a solution regardless the value of r. Although we cannot solve this completely, we know that for $r \leq e$ there are 2 solutions, and for $r > e$, there are 3 solutions. The above tells us that the locations of the remaining two solutions are directly linked to the magnitude of r, which is a fact worth exploring further. We can extrapolate this to higher order cases and state that the number of turning points will also be linked to the magnitude of r. However, we do not consider higher order cases in this thesis because implementing numerical algorithms to find the location of the turning points soon becomes computationally expensive and does not

yield much more insight than the first order dynamics, and also performing data conditioned simulation constrained by autocorrelation [Wood, 2010] is challenging. We now turn to the Poissonised observation of the second order model.

6.3.4 Posterior distribution of ϕ

Let's now consider the posterior distribution of ϕ in the stochastic Ricker model (6.2). Given $\boldsymbol{Y}, \boldsymbol{N}, r$ and σ we can calculate the posterior distribution of ϕ exactly. We utilise the fact that the conjugate prior of the Poisson distribution is the Gamma distribution. Let $\pi(\phi)$, $\pi(\boldsymbol{Y}, \boldsymbol{N}|\phi, r, \sigma)$, and $\pi(\phi|\boldsymbol{Y}, \boldsymbol{N}, r, \sigma)$ denote the prior distribution, the likelihood, and the conditional posterior distribution respectively.

Let $\pi(\phi) \sim \Gamma(\alpha, \beta)$, where α is the shape parameter and β is the rate parameter. Then the conditional posterior distribution given $\boldsymbol{Y}, \boldsymbol{N}, r, \sigma, \alpha, \beta$ is $\pi(\phi|\boldsymbol{Y}, \boldsymbol{N}, r, \sigma, \alpha, \beta) \sim \Gamma\Big(\alpha + \sum Y_t^*, \beta + \sum N_t\Big)$.

This will become useful later as the simulation uses a two stage process, i.e. simulate $\{N_t\}|r, \sigma$ and then simulate $\{Y_t\}|\phi, \boldsymbol{N}, r, \sigma$. By choosing ϕ based on $\Gamma\Big(\alpha + \sum Y_t^*, \beta + \sum N_t\Big)$, where $\boldsymbol{Y^*}$ is the actual observed data, it should allow us to make an informed choice of ϕ which leads to a higher probability of matching the summary statistics of the simulated $\boldsymbol{Y}$ and $\boldsymbol{Y^*}$. The idea of making an informed choice of a parameter based on some latent variable is related to the coupled ABC algorithm in [Neal, 2012]. We will see how this can be utilised in section 6.6.

6.3.5 Exploratory research conclusion

We have explored aspects of the underlying dynamics of the model and identified possible ideas to exploit as summary statistics in analysing the data.

We will summarise and investigate the observed summary statistics in the next section.

6.4 Summary statistics selection

In this section, we will take what we observed in Section 6.3 and refine these visual observations into quantifiable summary statistics so that they can be utilised in the later analysis.

Here are the two requirements that will help us narrow down the choices of summary statistics:

- There are three parameters, r, σ, ϕ, to be estimated, and therefore we require at least three summary statistics to enable us to uniquely identify the three parameters.
- The summary statistics should also have a strong relationship with the parameters.

6.4.1 Choices of summary statistics

In [Wood, 2010], he considered autocovariances to lag 5; the coefficients of cubic regression of the order differences $T_t - Y_{t-1}$; the coefficients, β_1 and β_2, of the autoregression $Y_{t+1}^{0.3} = \beta_1 Y_t^{0.3} + \beta_2 Y_t^{0.6} + \epsilon_t$, where ϵ_t denote the error; the mean of the population $\frac{1}{n}\sum_{t=1}^{n} T_t$; and the total number of zeros observed for summary statistics. By considering the 'maneuverability' of these statistics and observations we made in section 6.3.2, we have a list of potentially viable summary statistics to be used for the data conditioned simulation below:

1. The average length of consecutive run of 0's on Y_t.

2. The total number of 0's in Y_t.

3. The maximum value of Y_t.

4. The maximum value of $\log(Y_t+1) > \log(\phi+1)$ when $\log(Y_{t+1}+1) = 0$.

5. The minimum value of $\log(Y_t+1) > \log(\phi+1)$ when $\log(Y_{t+1}+1) = 0$.

6. The difference between summary statistics 4. and 5.

7. The vertical width near the theoretical turning point on the graph $\log Y_{t+1} + 1$ plotted against $\log Y_t + 1$. (To be defined later).

8. The average value $\overline{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$.

Summary statistic 1. is observed by plotting $\{Y_t\}$ against time for various choice of r, σ, ϕ, see Figure 6.2. We see that there is a clear change in the length of consecutive 0's as r varies.

Summary statistic 2. can be observed in Figures 6.2 and 6.4. It is strongly correlated to summary statistic 1, but much cheaper in terms of computation.

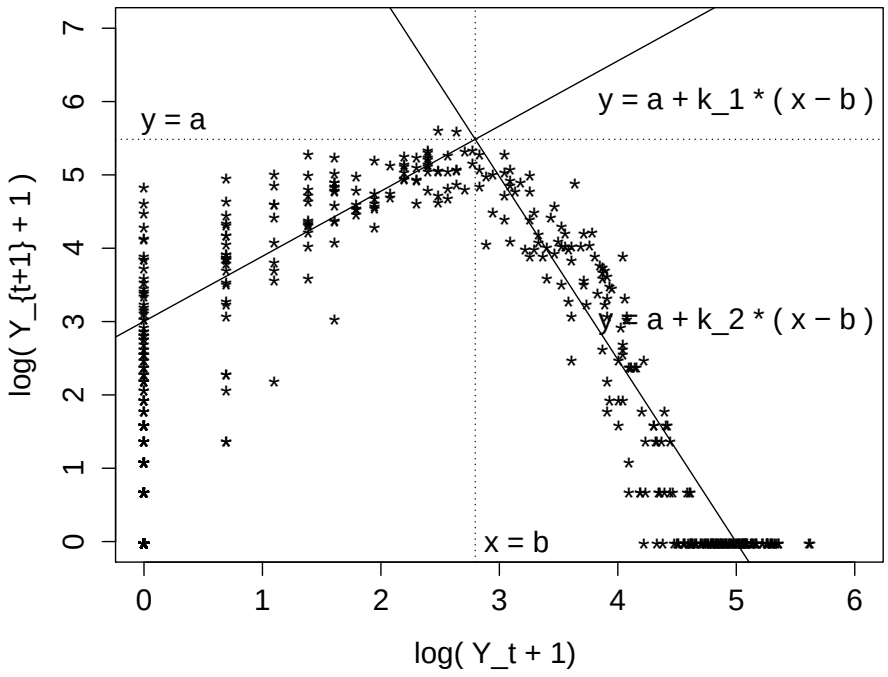
Summary statistic 3. and 4, can be observed in Figures 6.14 and 6.16. As ϕ increases from 10 to 30, the maximum value increases. Most of the time summary statistic 3 and 4 are the same with the rare exception that when the maximum occurs at the very last observation, and therefore we could eliminate summary statistic 4, because computing the maximum of $\{Y_t\}$ is fast.

Summary statistics 5. and 6. can be observed in Figure 6.16. There is a clear positive relationship between summary statistics 5. and 6. and r.

Summary statistic 7. requires explicit definition as the theoretical turning point is not given in the observation, and also, even if we were able to determine the turning point, it is possible that there is not enough data near the point to provide useful information. Here, we address the first

problem by fitting a mixture of two straight line model, and use the break point (where two straight lines meet), $(b, \log y_{t_b} + 1)$ as the empirical turning point for the observed data (illustrated in Figure 6.20). The point $x = b$ is determined by fitting a least square method. Before fitting the mixture of two straight lines model, we clean up the data by deleting duplicated data points. Without doing so, we are likely to place unwanted weight at certain points (most likely at $(0, 0)$) and hence distort the estimation of the turning point. From there, instead of looking at exactly on $\log y_t + 1 = b$, we take a band, $\log y_t + 1 \in (b - h, b + h)$, around the turning point and calculate the range of $\{\log y_{t+1} + 1\}$ which lie within of the band. We choose h based on the observed data. h is chosen such that $2h = 0.13\big(max(\log y_t + 1) - min(\log y_t + 1)\big)$, i.e. the band $(b - h, b + h)$ is 13% of the horizontal axis' range. We tested the size of band ranging from 1% to 20% with 1% increment. For each size of band, we generated 500 distinct sets of parameters and data of size 500, and calculate the correlation between σ and $width.at.turning$. The band size of 13% achieved the highest correlation of 0.7452.

Figure 6.20: Fitting the mixture of two straight lines model.



Summary statistic 8. It is easy to see why it is worth investigating given that there is a clear relationship between the mean of $\boldsymbol{N}$ and ϕ and the mean of $\boldsymbol{Y}$. This can be observed in Figure 6.4.

It is worth mentioning that the total number of 0 observations and the mean value of observations are also used in [Fearnhead and Prangle, 2012]. Before we will look at how each of the summary statistics reflects the change of each parameter in the model, we firstly examine collinearity between the summary statics and some of their variants. In Table 6.1 we check for collinearity between summary statistics 1. to 8. with the addition of maximum value of $log(Y_t + 1)$ and the mean value of $log(Y_t + 1)$. We abbreviate the summary statistics as follow $mean.0.length$, $total.0$, $max.y$, $max.0.tran$, $min.0.tran$,

$diff.0.tran$, $width.at.turning$, $mean.y$, $max.log.y$, and $mean.log.y$, and they are: the mean length of consecutive runs of 0s, total number of 0s, maximum of $\boldsymbol{Y}$, the maximum value of $log(Y_t+1) > log(\phi+1)$ when $log(Y_{t+1}+1) = 0$, the minimum value of $log(Y_t+1) > log(\phi+1)$ when $log(Y_{t+1}+1) = 0$, the difference between the previous two summary statistics, the vertical width near the theoretical turning point on the graph $log(Y_{t+1}+1)$ against $log(Y_t+1)$, the mean value of $\boldsymbol{Y}$, the maximum value of $log(Y_t+1)$, and the mean value of $log(Y_t+1)$, respectively.

Table 6.1: Correlations between all summary statistics based on 5000 data sets each with 500 observations.

	mean.0.length	total.0	max.y	max.log.y	max.0.tran	min.0.tran	diff.0.tran	width.at.turning	mean.log.y	mean.y
mean.0.length	1.0000	0.6443	0.6814	0.6141	0.6138	0.2767	0.6194	0.1077	-0.6505	0.1990
total.0	0.6443	1.0000	0.4636	0.6996	0.7002	0.2378	0.7801	0.1044	-0.9757	0.2709
max.y	0.6814	0.4636	1.0000	0.7931	0.7922	0.5313	0.6353	0.0456	-0.4302	0.5596
max.log.y	0.6141	0.6996	0.7931	1.0000	0.9998	0.7199	0.7553	0.0416	-0.5980	0.7109
max.0.tran	0.6138	0.7002	0.7922	0.9998	1.0000	0.7195	0.7558	0.0416	-0.5984	0.7102
min.0.tran	0.2767	0.2378	0.5313	0.7199	0.7195	1.0000	0.0892	-0.3536	-0.0695	0.9143
diff.0.tran	0.6194	0.7801	0.6353	0.7553	0.7558	0.0892	1.0000	0.3931	-0.7928	0.1566
width.at.turning	0.1077	0.1044	0.0456	0.0416	0.0416	-0.3536	0.3931	1.0000	-0.1782	-0.3371
mean.log.y	-0.6505	-0.9757	-0.4302	-0.5980	-0.5984	-0.0695	-0.7928	-0.1782	1.0000	-0.1236
mean.y	0.1990	0.2709	0.5596	0.7109	0.7102	0.9143	0.1566	-0.3371	-0.1236	1.0000

We can observe some highly correlated (more than 0.9 or less than -0.9) pairs:

- $total.0$ and $mean.log.y$,
- $max.log.y$ and $max.0.tran$,
- $min.0.tran$ and $mean.y$.

We choose $total.0$, $max.log.y$, and $mean.y$ from the three pairs above, because they are computationally less intensive. The rest of the summary statistics although exhibit some correlation, they are less prominent. We will examine the relationships between the summary statistics and each parameter below.

6.4.2 For r

Figure 6.21 are plots of all summary statistics mentioned above except summary statistic 4 plotted against r. There are 5000 data sets, and each data set was generated by $r \in [e^2, e^6]$, $\sigma \in [0.1, 1]$, and $\phi \in [5, 50]$ sampled uniformly in the range specified, and each with total time points of 1000, observation points 500, and $N_0 = 1$. This parameter space is chosen so that it is in line with [Wood, 2010] and [Fearnhead and Prangle, 2012].

Figure 6.21: All summary statistics plotted against $log(r)$

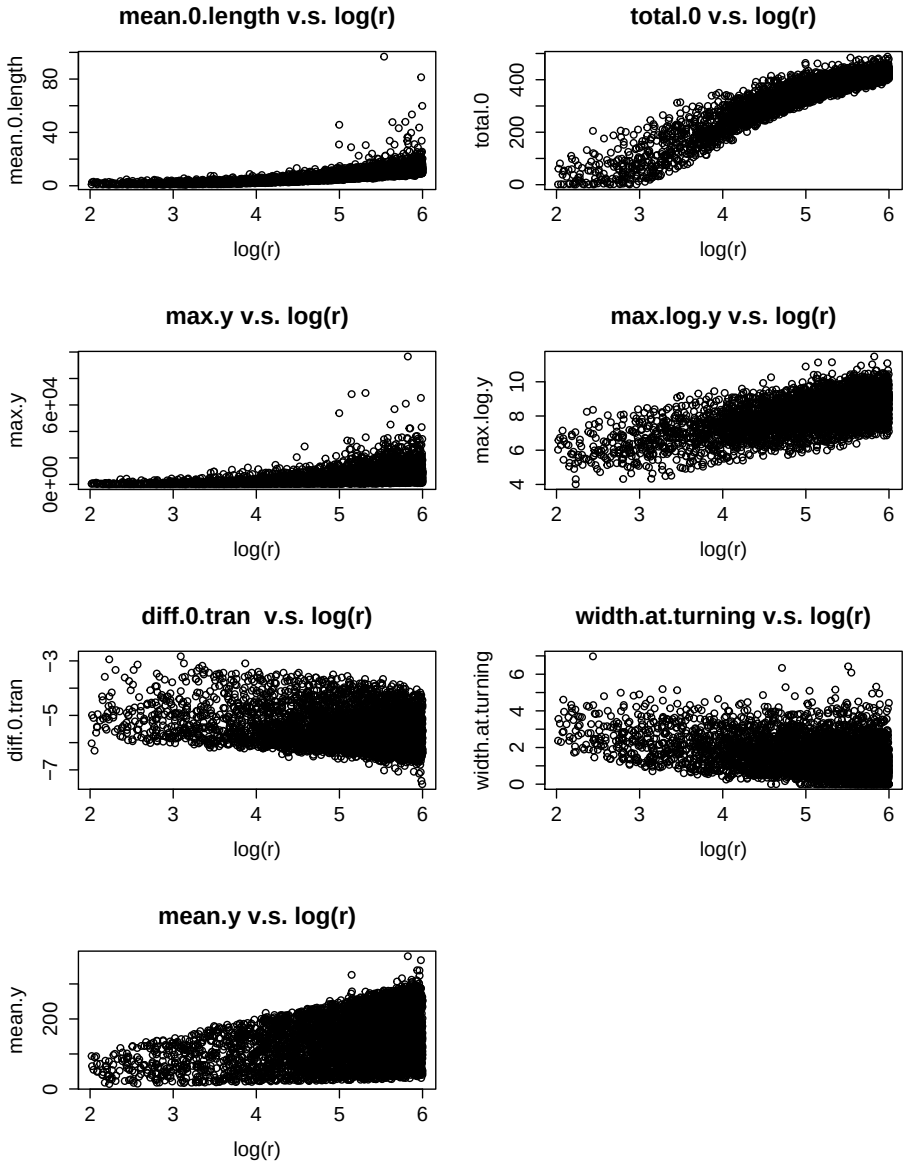


Table 6.2: Linear correlations between r, $\log(r)$ and the summary statistics.

	mean.0.length	total.0	max.y	max.log.y	diff.0.tran	width.at.turning	mean.y
$\log(r)$	0.6775	0.9404	0.4100	0.6355	0.6316	-0.3729	0.3291

From Figure 6.21, $total.0$ v.s. $log(r)$ looks to have a strong positive correlation. It is further confirmed in Table 6.2, for which the coefficient of correlation between $total.0$ and $log(r)$ is 0.94, an almost perfectly linear relationship.

6.4.3 For σ

We will investigate the relationships between the summary statistics and σ in this section. A preliminary study shows that log transformation does not improve the correlation and therefore we will use the original data in this section.

Figure 6.22: All summary statistics plotted against $log(\sigma)$

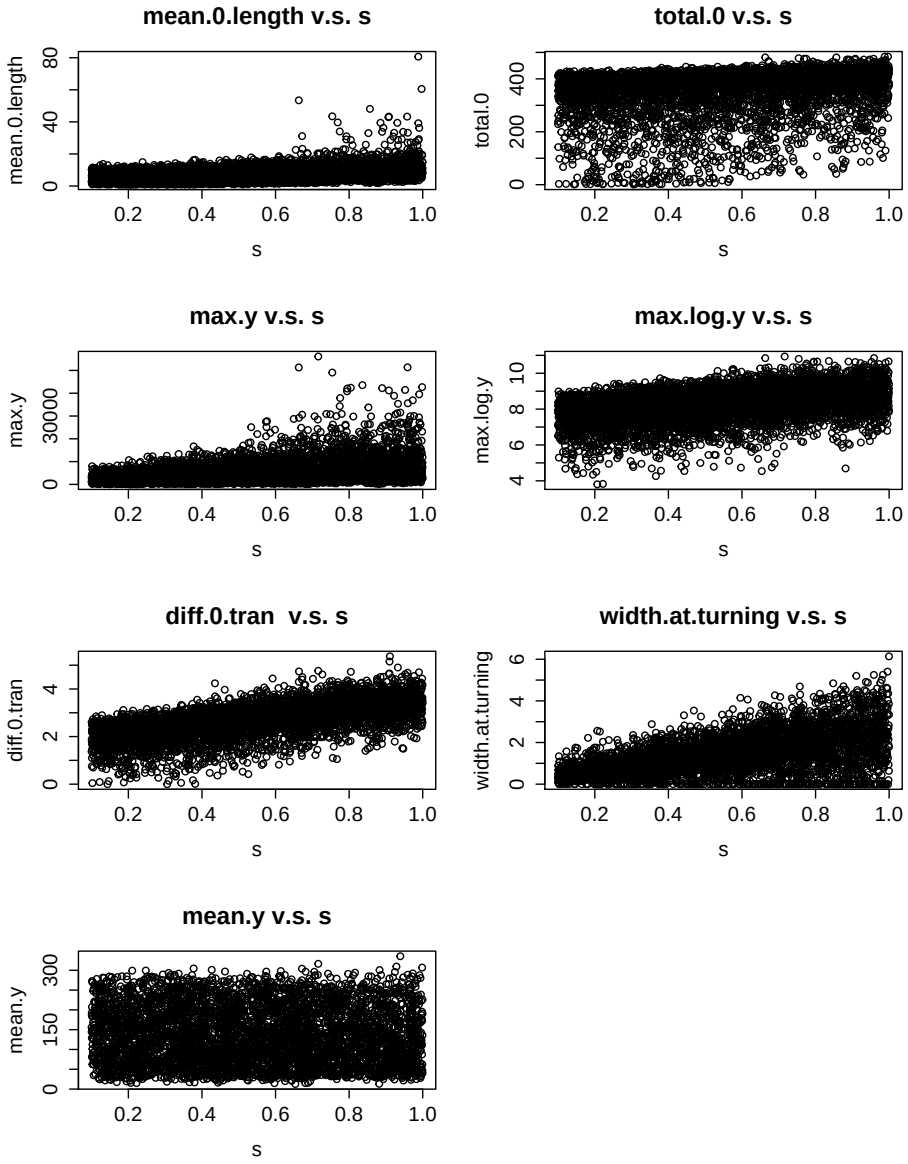


Table 6.3: Correlations between $\log(\sigma)$ and the summary statistics.

	mean.0.length	total.0	max.y	max.log.y	diff.0.tran	width.at.turning	mean.y
$log(\sigma)$	0.3103	0.1759	0.3833	0.4071	0.5947	0.5315	-0.0326
σ	0.3474	0.1948	0.4127	0.4208	0.6167	0.5437	-0.0368

It can be observed in Figure 6.22 and Table 6.3 that $log(\sigma)$ and the summary statistics show little correlation or any obvious relationship as suspected in Section 6.3. The reason being that before introducing the random error (e_t), even though the system is chaotic, for any given r, the underlying dynamic is still bounded by a cyclic structure, although some cycles may have very long or even infinite periods (chaotic). We can observe this in the bifurcation diagram in Figure 6.5. For example when $r = 10$, we can see that the long term value of N switches between two values and when $r = 20$, the long run value of N can be anything between 0 and 7.5.

With the introduction of the random error, the underlying dynamic allows to break out from its original cycle and hop into other cycles and as the result N would exhibit values from long run cycle of various rs. The bifurcation diagram of the stochastic version of the ricker model in Figure 6.6 demonstrates such property. Now, if we look at $r = 10$, the long run value of N looks to contain all long run value of N for $r < 10$, and similarly for $r = 20$.

From the exploratory research, we learned that r and σ could have similar effects on some summary statistics, and therefore we hope to improve the correlations by keeping r fixed and indeed we see a noticeable improvements in the overall correlations in Figure 6.23, and in particular $mean.0.length$, $total.0$, $diff.0.tran$, and $width.at.turning$.

Figure 6.23: All summary statistics plotted against σ with fixed r

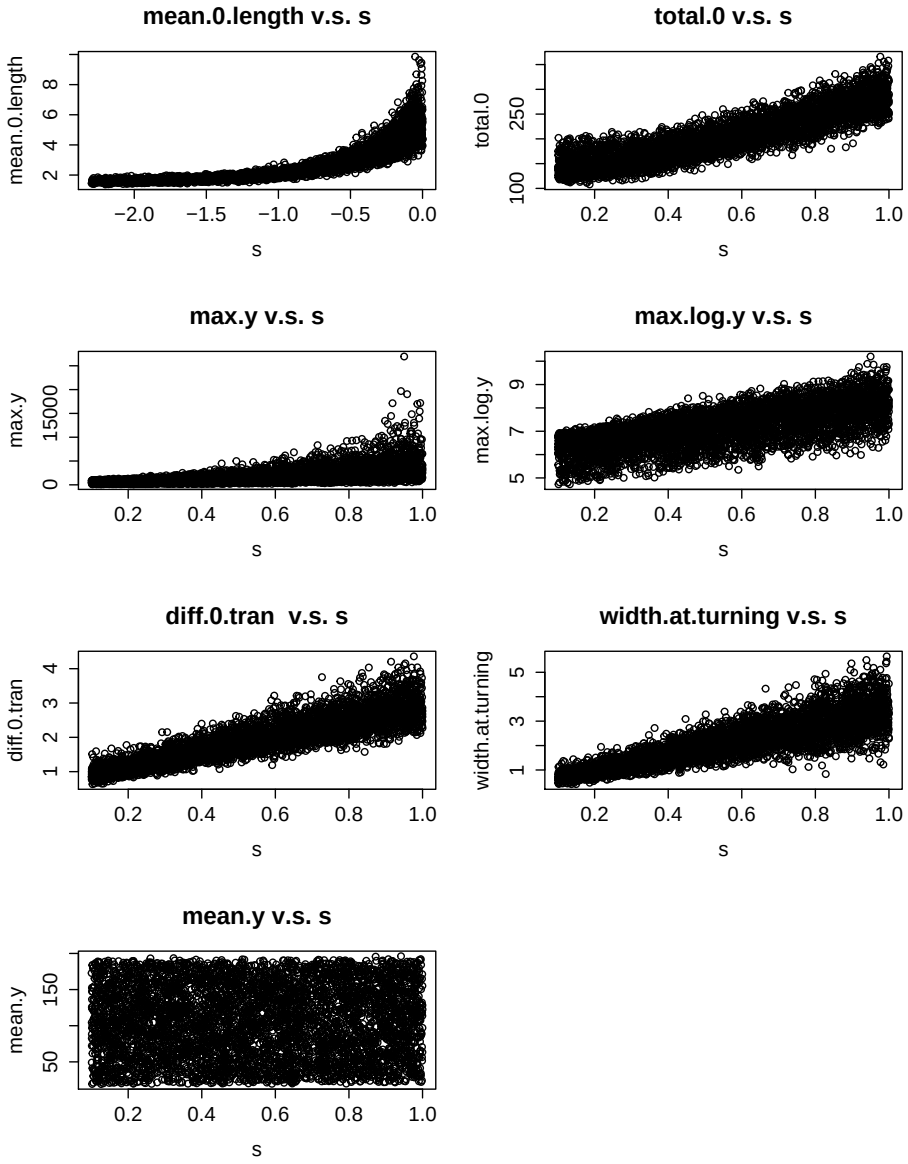


Table 6.4: Correlations between σ, $log(\sigma)$ and the summary statistics with fixed $r = e^{3.8}$.

	mean.0.length	total.0	max.y	max.log.y	diff.0.tran	width.at.turning	mean.y
$log(\sigma)$	0.8180	0.8246	0.5373	0.6646	0.8624	0.8340	0.0139
σ	0.9021	0.8914	0.6002	0.6899	0.8945	0.8657	0.0152

We see a much improved correlation between σ and all summary statistics when r is fixed in Figure 6.23 and Table 6.4. Specifically, $mean.0.length$, $total.0$, $diff.0.tran$ and $width.at.turning$ have close to 0.9 correlation. We will be able to implement a two stage algorithm by exploiting this observation later.

6.4.4 For ϕ

We will now look at the relationships between the summary statistics and ϕ. Again, a preliminary analysis showed that the *log* transformation does not improve the correlations and therefore we will use the original data in this section.

Figure 6.24: All summary statistics plotted against ϕ

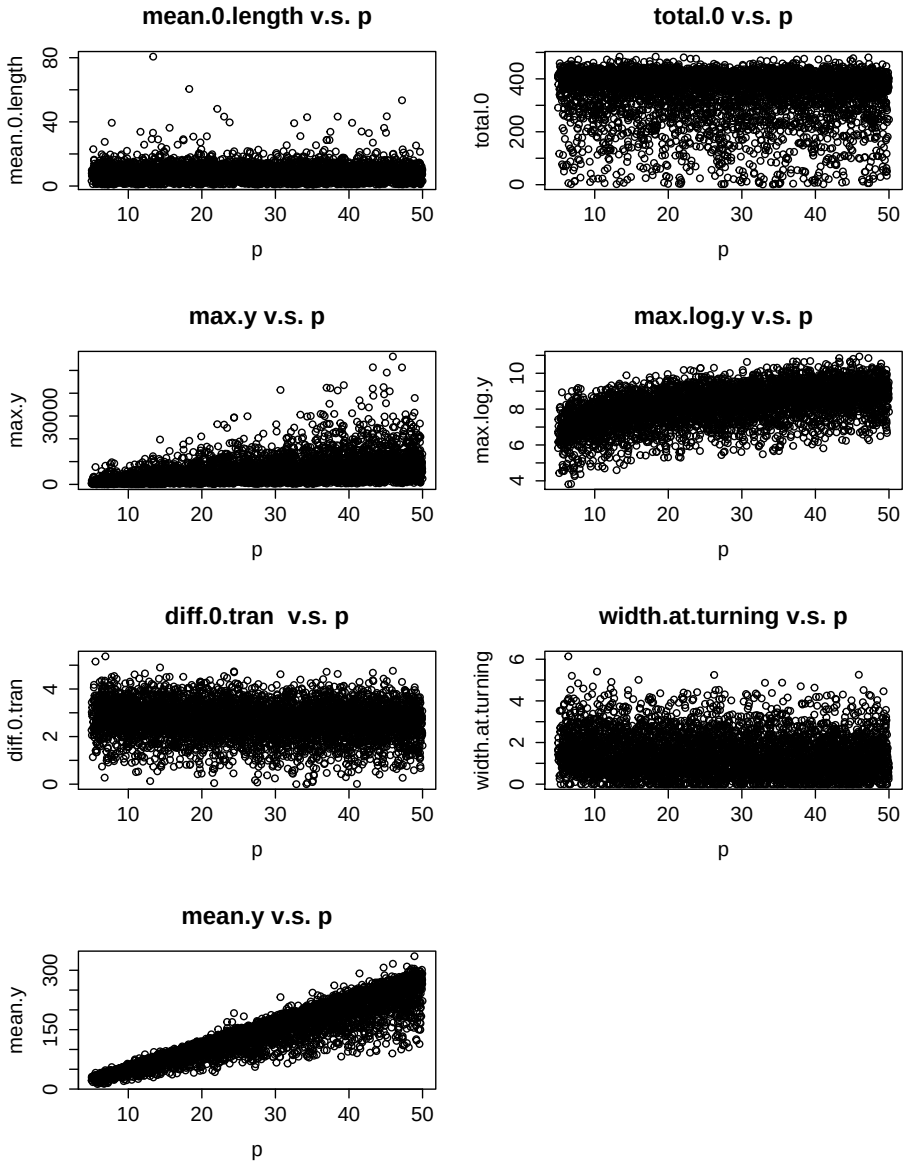


Table 6.5: Correlations between ϕ, $\log(\phi)$ and the summary statistics.

	mean.0.length	total.0	max.y	max.log.y	diff.0.tran	width.at.turning	mean.y
$log(\phi)$	0.0002	-0.0735	0.4492	0.5547	-0.1222	-0.1165	0.8979
ϕ	-0.0034	-0.0705	0.4639	0.5342	-0.1172	-0.1092	0.9313

In Figure 6.24 and Table 6.5, it can be observed that the effect of the *log* transformation on correlation is somewhat mixed. It's worth noting that he highest correlated summary statistic, $mean.y$, is in favour of the untransformed data.

However, as we can calculate the posterior distribution of ϕ given $\{\boldsymbol{Y}, \boldsymbol{N}, r, \sigma\}$ analytically, the choice of a summary statistic reflecting the changes in ϕ is somewhat redundant.

6.4.5 Summary statistics conclusion

With the combination of the discussion above and the computation intensity of each summary statistic, we narrow down the initial list of summary statistics to the following three:

1. The total number of 0's in the observation. ($total.0$)
2. The maximum value of $log(Y_i + 1)$. ($max.log.y$)
3. The mean value of $\boldsymbol{Y}$. ($mean.y$)

The above summary statistics are chosen not only based on their correlations to each parameters, but also their ease for implementing the data conditioned simulation. If we were to choose summary statistics purely based their correlations with r, σ and ϕ, then we would have $total.0$ (0.9404 correlation with

$log(r)$ and 0.9021 correlation with σ given r), $diff.0.tran$ (0.8945 correlation with σ), and $mean.y$ (0.9313 correlation with ϕ). However, $diff.0.tran$ is difficult to implement when it comes to the data condition simulation and therefore we opt for using $max.log.y$, which has 0.6355 correlation with $log(r)$ and 0.6899 correlation with σ given r.

6.5 Data simulation

6.5.1 Unconditioned Data Simulation

We wish to generate a sample of $\{Y_t\}$ with size L. Given r, σ, ϕ, the process is straight forward. To start, we utilise (6.2) and generate $N_1, \ldots, N_T$. The reason for generating extra $T - L$ number of N_ts is because $N_0, \ldots, N_{T-L}$ terms are used as burn in to ensure that the dynamical process settles into its long term behaviour. We then sample Y_i from $Poisson(\phi N_{T-L+i})$ for $i = 1, \ldots, L$. Then $\{y_1, \ldots, y_T\}$ is a simulation based on the parameters r, σ and ϕ.

6.5.2 Data conditioned simulation

Data conditioned simulation is more complex in this example, because we need to satisfy three summary statistics. Recall Section 6.4.5, the three summary statistics are: $total.0, max.log.y$, and $mean.y$. We wish to generate a sample $\{Y_t\}$ of size L, which satisfies the summary statistics. Given r, σ, ϕ, we generate $N_1, \ldots, N_T$ as in Section 6.5.1 and relabel $N_{T-L+1}, \ldots, N_T$ as $N_1, \ldots, N_L$. We then steer the simulation of Y_is, so that the simulated $\boldsymbol{Y}$ has the same summary statistics as the observed data. We start by renaming $total.0, max.log.y$, and $mean.y$ of $\boldsymbol{Y}$ as $s_1(\boldsymbol{Y}), s_2(\boldsymbol{Y})$ and $s_3(\boldsymbol{Y})$ respectively and let s_1^*, s_2^* and s_3^* denote $s_1(\boldsymbol{Y^*}), s_2(\boldsymbol{Y^*})$ and $s_3(\boldsymbol{Y^*})$ respectively. Our aim

is to estimate the likelihood $\pi(s_1^*, s_2^*, s_3^*|r, \sigma, \phi)$. We will utilise a combination of data augmentation and importance sampling to help us achieve an estimation through simulation. We start by considering the unobserved underlying dynamics $\boldsymbol{N}$. The model construction allows us to apply the standard data augmentation easily, see below:

$$\pi(s_1^*, s_2^*, s_3^*|r, \sigma, \phi) = \int_{\forall \boldsymbol{N}} \pi(s_1^*, s_2^*, s_3^*|\boldsymbol{N}, r, \sigma, \phi)\pi(\boldsymbol{N}|r, \sigma)d\boldsymbol{N}.$$

we can estimate the above integral using:

$$\frac{1}{k}\sum_{i=1}^{k} \pi(s_1^*, s_2^*, s_3^*|\boldsymbol{N_i}, r, \sigma, \phi), \tag{6.8}$$

where $\boldsymbol{N_i}$s are samples generated from $\pi(\boldsymbol{N}|r, \sigma)$. We will refer to k in (6.8) as the $\boldsymbol{N}$ augmentation size, which is the number of simulations we make for a given set of parameters. Generating samples from $\pi(\boldsymbol{N}|r, \sigma)$ is straightforward. The main challenge is in estimating $\pi(s_1^*, s_2^*, s_3^*|\boldsymbol{N_i}, r, \sigma, \phi)$. The principal idea we will use to estimate $\pi(s_1^*, s_2^*, s_3^*|\boldsymbol{N_i}, r, \sigma, \phi)$ is the same as the one used in Section 3.3.2, even though the model is more complex, and that is through steered simulation. We consider the estimation through simulating $\boldsymbol{Y}|\boldsymbol{N}, r, \sigma, \phi$ as a data augmentation algorithm. We further apply importance sampling algorithm in the simulation of $\boldsymbol{Y}$ by carefully choosing the importance sampling proposal distribution $q(\cdot)$ such that $s_1(\boldsymbol{Y}) = s_1^*, s_2(\boldsymbol{Y}) = s_2^*, s_3(\boldsymbol{Y}) = s_3^*$, see below:

$$\begin{aligned}
&\pi(s_1^*, s_2^*, s_3^*|\boldsymbol{N_i}, r, \sigma, \phi) \\
&\quad = \int_{\forall \boldsymbol{Y}} \pi(s_1^*, s_2^*, s_3^*|\boldsymbol{Y}, \boldsymbol{N_i}, r, \sigma, \phi)\pi(\boldsymbol{Y}|\boldsymbol{N_i}, r, \sigma, \phi)d\boldsymbol{Y} \qquad (6.9) \\
&\quad = \int_{\forall \boldsymbol{Y}} \pi(s_1^*, s_2^*, s_3^*|\boldsymbol{Y}, \boldsymbol{N_i}, r, \sigma, \phi)\frac{\pi(\boldsymbol{Y}|\boldsymbol{N_i}, r, \sigma, \phi)}{q(\boldsymbol{Y}|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)} \\
&\qquad q(\boldsymbol{Y}|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)d\boldsymbol{Y}. \qquad (6.10)
\end{aligned}$$

In (6.9), we use data augmentation to introduce the simulation of $\boldsymbol{Y}$. In (6.10), we apply the importance sampling algorithm to achieve "steering".

The integral above can be estimated using:

$$\frac{1}{k}\sum_{j=1}^{k}\Big(1 \times \frac{\pi(\boldsymbol{Y_j}|\boldsymbol{N_i}, r, \sigma, \phi)}{q(\boldsymbol{Y_j}|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)}\Big), \tag{6.11}$$

where $q(\boldsymbol{Y_j}|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)$ represents a distribution of $\boldsymbol{Y}$'s that has the same summary statistics as the observed data given $\boldsymbol{N_i}, r, \sigma, \phi$, and therefore $\pi(s_1^*, s_2^*, s_3^*|\boldsymbol{Y}, \boldsymbol{N_i}, r, \sigma, \phi) = 1$, which is written out explicitly in (6.11). We will refer to the k in (6.11) as the parameter importance sampling size, or the group size in the context of group independence Metropolis Hastings algorithm. Not to be confused with the k in (6.8). k is just a generic variable given to indicate looping. Our next step is to construct $q(\boldsymbol{Y_j}|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)$. For clarity, we will drop the subscript on $\boldsymbol{N_i}$ and $\boldsymbol{Y_j}$ during the construction of $q(\cdot)$, because we are only considering one instance at a time. We will use Y_i to denote the ith component of $\boldsymbol{Y}$. We further let p_i to denote the corresponding importance sampling weight of Y_i and $p_i = \dfrac{\pi(Y_i|\boldsymbol{N_i}, r, \sigma, \phi)}{q(Y_i|\boldsymbol{N_i}, r, \sigma, \phi, s_1^*, s_2^*, s_3^*)}$ where it's appropriate. The use of p_i assumes that the simulation mechanism is carried out component by component. To construct $q(\cdot)$, we begin by ensuring s_1^* ($total.0$) is met, and follow by s_2^* ($max.log.y$), and finally s_3^* ($mean.y$). In this example, we will also demonstrate that we have some freedom in choosing the importance sampling proposal distribution $q(\cdot)$ by describing two methods of sampling for s_3^*.

For s_1 ($total.0$) We want to ensure that there are exactly s_1^* 0's in $\boldsymbol{Y}$. We know that $\mathbb{P}(Y_i = 0) = e^{-\phi N_i}$. To find the location of which Y_i should be 0, we sample s_1^* samples from $\{1, \ldots, L\}$ without replacement according to weights $\{e^{-\phi N_1}, \ldots, e^{-\phi N_L}\}$. The outcome is the positions of Y_i's that should be 0. We label these indices as $I_1^{(s_1)}, \ldots, I_{s_1^*}^{(s_1)}$, then $Y_{I_j^{(s_1)}} = 0$ and $p_{I_j^{(s_1)}} = e^{-\phi N_{I_j^{(s_1)}}}$ for $j = 1, \ldots, s_1^*$. For the subsequent simulation, we need to ensure $Y_i \geq 1$ for the remaining Y_i's.

For s_2 ($max.log.y$) Since $max.log.y$ and $max.y$ represent the same observation, we will use $max.y$ for better accuracy (potential rounding error) and computational efficiency. We first calculate $\mathbb{P}(Y_i = s_2^*) = \frac{(\phi N_i)^{s_2^*} e^{-\phi N_i}}{s_2^*!}$ for

all i. We also know that $max.y$ only occurs immediately before a 0 observation or at the very last observation. Suppose that there are h qualifying positions, and let $\{I_1^{(s_2)}, \ldots, I_h^{(s_2)}\}$ denote the indices of these position. We take a sample of size 1 on $\{I_1^{(s_2)}, \ldots, I_h^{(s_2)}\}$ with sampling weight $\left\{\frac{\left(\phi N_{I_1^{(s_2)}}\right)^{s_2^*} e^{-\phi N_{I_1^{(s_2)}}}}{s_2^*!}, \ldots, \frac{\left(\phi N_{I_h^{(s_2)}}\right)^{s_2^*} e^{-\phi N_{I_h^{(s_2)}}}}{s_2^*!}\right\}$. Let $I^{(s_2)}$ denote the outcome, then $N_{I^{(s_2)}} = s_2^*$ and $p_{I^{(s_2)}} = \frac{\left(\phi N_{I^{(s_2)}}\right)^{s_2^*} e^{-\phi N_{I^{(s_2)}}}}{s_2^*!}$.

For s_3 (mean.y), method 1 We now just need to fill in the blank spaces of Y_i's in a way so that the mean of $\boldsymbol{Y}$ is s_3^* without affecting the other two summary statistics. We will discuss two methods of simulating for s^3. One of which is mathematically more rigorous, and the other one is adjusted for practicality. We will discuss the more mathematically rigorous method here. Let $I_1^{(s_3)}, \ldots, I_{L-s_1^*-1}^{(s^3)}$ denote the indices of all the blank Y_i's. We do not want to have any more 0's and therefore the minimum value of the remaining slots is 1 because of s_1. We suppose that the overall maximum value of Y_i's is unique, then the maximum possible value of the unfilled Y_i's is $s_2^* - 1$. We can treat this process of filling in the Y_i's as a multinomial model. In order to satisfy $s_3 = s_3^*$, we view the empty Y_i's as boxes, and we have in total $s_3^* L - s_2^*$ balls to allocate with probabilities being the normalised $\left\{\phi N_{I_1^{(s_3)}}, \ldots, \phi N_{I_{L-s_1^*-1}^{(s^3)}}\right\}$. We will denote the probabilities as $\left\{w_{I_1^{(s_3)}}, \ldots, w_{I_{L-s_1^*-1}^{(s^3)}}\right\}$. We also let p_{s_3} denote the amount of steering we do for the multinomial allocation, and we initialise $p_{s_3} = 1$. In this method, we treat the remaining Y_i's collectively in a multinomial distribution unlike earlier cases for s_1 and s_2, where Y_i's are determined component by component. Therefore, we declare a new variable, p_{s_3}, to record the steering.

The basic idea is that during the allocation we need to exclude those Y_i's that hit $s_2^* - 1$, if they do. We will also need to make sure that Y_i's with indices $I_1^{(s_3)}, \ldots, I_{L-s_1^*-1}^{(s^3)}$ are at least 1. We will put the rest of the algorithm in a pseudo-code below for easier presentation. In the pseudo code, any summation over Y_i's only considers already filled Y_i's and ignores the ones that are blank. Y_i can contain 0 and it is not considered blank. For example,

all the Y_i's that have been selected in the simulation of s_1 will have value 0 but they are not blank. We will call any Y_i full if $Y_i = s_2^* - 1$.

1: **BEGIN**

2: **while** $\Big(\sum_{i=1}^{L} Y_i <$ ($s_3^* L -$ number of blank Y_i's) $\Big)$ **do** ▷ this while statement ensures that we don't over allocate and we can make sure that all blank Y_i's will finish with at least 1

3: Let $\mathbb{I}$ denote the set of blank Y_i's indices $\big\{I_1^{(s_3)}, \ldots, I_{L-s_1^*-1}^{(s^3)}\big\}$.

4: J $\leftarrow$ a sample of size one from $\{I_{\mathbb{J}}\}$ with sampling weight $\{w_{\mathbb{J}}\}$, where $\mathbb{J} = \big\{i : \forall i \in \mathbb{I} \cap Y_i \text{ is not full}\big\}$; ▷ we exclude those Y_i's that are full from the sample.

5: $Y_J = Y_J + 1$; ▷ adding 1 count to the sampled position.

6: $p_{s_3} = p_{s_3} \times \sum_{j \in \mathbb{J}} w_j$; ▷ accounting for the steering. The steering comes from excluding the full Y_i's. $\sum_{j \in \mathbb{J}} w_j$ is the normalising term for the truncated multinomial distribution.

7: **end while**

8: $Y_j \leftarrow 1$ for $\big\{j : j \in \mathbb{I} \cap Y_j \text{ is blank}\big\}$; ▷ assign 1 to all the remaining blank Y_i's.

9: $p_{s_3} = p_{s_3} \times \prod_j \mathbb{P}(Y_j = 1)$ for $\big\{j : j \in \mathbb{I} \cap Y_j \text{ is blank}\big\}$;

10: **return** p_{s_3};

11: **END**

Following a similar process to that used in Section **??**, we can establish that the final importance sampling weight for matching all three summary statistics is $P = \prod_{i \notin \mathbb{I}} p_i \times p_{s_3}$. Method 1 discussed here is said to be mathematically more rigorous because the method did not include extra human intervention when allocating the blank Y_i's in order for the algorithm to produce viable outcome. Whereas in the method 2, which will be discussed next, extra human intervention had to be made in order to avoid 0 probabilities (extremely small probabilities that are beyond the accuracy level of the computer).

For s_3 (mean.y), method 2 The goal is the same as method 1, in that we are looking fill in the blank Y_i's in a way so that $s_3(\boldsymbol{Y}) = s_3^*$ and $s_1(\boldsymbol{Y})$ and $s_2(\boldsymbol{Y})$

remain unchanged. This time we take a more intuitive view and consider the blank Y_i's one by one. For each blank Y_i, we sample from a truncated Poisson distribution, $Poisson(\phi N_i)$, with some lower and upper bounds. We initially use 1 and $min\Big(s_2^* - 1, s_3^* L - \sum_{\text{not blank}} Y_j - \text{ number of blank } Y_i\text{'s}\Big)$ as the lower and upper bound respectively. This does indeed generate a sample with required property. However, the problem arises when calculating calculating the normalising weight for the truncated Poisson distribution. It is often possible that ϕN_i is much greater than upper bound, and therefore calculating the probability $\mathbb{P}(\text{lower bound} \leq Y_i \leq \text{upper bound})$ result in extreme small values that is beyond the accuracy level of the computer. The reverse scenario could also pose an issue, i.e. ϕN_i is much smaller than the lower bound, but it is less common. If ϕN_i were very small, it is likely that they have been picked to be 0 in simulation of s_1. In order to reduce the occurence of these extreme probabilities, we intervene by choosing the order that we fill in the blank Y_i's and adapting the lower bounds to the number of blank Y_i's left.

Again, let $I_1^{(s_3)}, \ldots, I_{L-s_1^*-1}^{(s^3)}$ denote the indices of all the blank Y_i's. All blank Y_i's should have values between 1 and $s_2^* - 1$. Suppose $g = L - s_1^* - 1$, and let $\{J_1, \ldots, J_g\}$ denote the ordered indices by the size of N_i, such that $N_{J_1} > N_{J_2} > \ldots > N_{J_g}$. We will fill in Y_{J_i}'s by sampling from a series of truncated Poisson distributions. For $i = 1, \ldots, g$, we let $upper = min\Big(s_2^* - 1, s_3^* L - \sum_{\text{not blank}} Y_j - \text{ number of blank } Y_i\text{'s}\Big)$ and $lower = min\Big(\frac{s_3^* L - \sum_{\text{not blank}} Y_j}{g-i+1}, s_3^* L - \sum_{\text{not blank}} Y_j - \text{ number of blank } Y_i\text{'s}\Big)$. Then:

$$Y_{J_i} \sim Poisson(\phi N_{J_i}) | lower \leq Y_{J_i} \leq upper \tag{6.12}$$

$$p_{J_i} = \mathbb{P}(lower \leq Y_{J_i} \leq upper) \tag{6.13}$$

The upper bound ensures that the maximum value does not exceed s_2^*. The lower bound ensures that at each iteration Y_{J_i} is of reasonable size so that at the end of the loop we don't need to force a large Y_i on a Poisson distribution with a small parameter, which is likely to produce a near zero probability.

Again, following a similar process to that used in Section **??**, we can establish that the final importance sampling weight for matching all three summary statistics is $P = \prod_{i=1}^{L} p_i$. In our preliminary research, we observed that although both methods perform well when estimating s and ϕ, method 2 performs significantly better when estimating σ. Method 1 also takes longer than method 2. Therefore, method 2 will be used in the algorithm implementation later. In the next section, we will look at how the unconditioned simulation and the data conditioned simulation can be used in the algorithm implementation.

6.6 Algorithm implementation

In this section, we will look at implementation of the dcABC algorithm, and the GIMH algorithm for the Ricker model. We leave out the discussion of the rsABC algorithm here as it was not possible to produce useful results within reasonable computation time.

6.6.1 The dcABC algorithm

In this section we will describe the details of the implementation of the dcABC algorithm for the Ricker model. It is a slightly different algorithm to the equivalent algorithms in Chapters 4 and 5. An important difference for this algorithm is that instead of sampling ϕ independently from its prior distribution as we have done so far with the SIR model and time inhomogeneous Markov chain model, we choose ϕ depending on $\boldsymbol{Y}^*$ and $\boldsymbol{N}$. The reason for that is to reduce the occurrence of 0 sampling weight, which we will come back to after stating the algorithm. One major difference in the algorithm is that for each set of (r, σ), we will firstly generate multiple $\boldsymbol{N}$'s and corresponding ϕ's (this corresponds to the N augmentation size in Section 6.5.2), and then for each combination of $(r, \sigma, \phi, \boldsymbol{N})$, we will perform

multiple data conditioned simulations (this corresponds to the group size in Section 6.5.2).

The algorithm

Algorithm 23 (dcABC algorithm for the Ricker model)**.**

1. Sample r, σ according to $log(r) \sim U(2,6), log(\sigma) \sim U(log(0.1), 0)$ respectively, where $U(a,b)$ denotes an uniform distribution with lower bound a and upper bound b.
2. Let $N_0 = 1$. Generate $\boldsymbol{N}$ with twice as many terms as the observation. For example, if the observation has 500 terms, then we will generate 1000 N_i's.
3. Recall that the prior distribution $\phi \sim \Gamma(7.5, 1)$, then the posterior distribution $\phi|\boldsymbol{N}, \boldsymbol{Y^*}, r, \sigma \sim \Gamma\big(7.5 + \sum_i Y_i^*, 1 + \sum_i N_i\big)$. We take a sample from the posterior distribution.
4. Given $r, \sigma, \phi, \boldsymbol{N}$, we use the data conditioned simulation described in Section 6.5.1 to estimate the importance sampling weight P.
5. Record $\{r, \sigma, \phi, \frac{\pi(\phi)}{\pi(\phi|\boldsymbol{N}, \boldsymbol{Y^*}, r, \sigma)} P\}$, where $\pi(\phi)$ is the density function of $\Gamma(7.5, 1)$ evaluated at the sampled ϕ and $\pi(\phi|\boldsymbol{N}, \boldsymbol{Y^*}, r, \sigma)$ is the density function of $\Gamma\big(7.5 + \sum_i Y_i^*, 1 + \sum_i N_i\big)$ evaluated at the sampled ϕ. The additional factor before the sampling weight P is to account for the sampling of ϕ. In effect, we performed an additional importance sampling step in the sampling of ϕ.
6. Repeat 4. and 5. for $k^{(group)}$ number of times, and record $\left\{r_i, \sigma_i, \phi_i, Q_i = \frac{\sum_{l=1}^{k^{(group)}} P_{il}}{k^{(group)}} \frac{\pi(\phi)}{\pi(\phi|\boldsymbol{N}, \boldsymbol{Y^*}, r, \sigma)}\right\}$. $k^{(group)}$ represents the group size.
7. Repeat 2. to 6. for $k^{(\phi)}$ number of times, and record $\left\{r_i, \sigma_i, \phi_{ij}, Q_{ij}\right\}$ for $j = 1, \ldots, k^{(\phi)}$. $k^{(\phi)}$ represents the N augmentation size.
8. Repeat 1. to 7. for $k^{(r\sigma)}$ number of times. The final sample size is then $k^{(r\sigma)} \times k^{(\phi)}$. There will be a total of $k^{(r\sigma)}$ pairs of (r, σ). Each pair of (r, σ) will correspond to $k^{(\phi)}$ number of ϕ's from step 7.

An output from the above algorithm would look like:

$$\{r_1, \sigma_1, \phi_{1,1}, Q_{1,1}\}, \ldots, \{r_1, \sigma_1, \phi_{1,k^{(\phi)}}, Q_{1,k^{\phi}}\},$$

$$\vdots$$

$$\{r_{k^{(r\sigma)}}, \sigma_{k^{(r\sigma)}}, \phi_{k^{(r\sigma)},1}, Q_{k^{(r\sigma)},1}\}, \ldots, \{r_{k^{(r\sigma)}}, \sigma_{k^{(r\sigma)}}, \phi_{k^{(r\sigma)},k^{\phi}}, Q_{k^{(r\sigma)},k^{\phi}}\}.$$

Step 5. is where we make an informed choice of ϕ. Without the modification, one would simply sample from its prior distribution, $\Gamma(7.5, 1)$. However, as the size of the observation increases, calculation of the sampling weight quickly reaches the accuracy limit of the computer. The multiplications in the calculation of the sampling weight also means that a small change in each term (p_i), would cause changes in orders of magnitude in the final sampling weight (P). Therefore, a good choice of ϕ is important. We utilise the posterior distribution of ϕ as the importance sampling proposal distribution, see Section 6.3.4. The final output of the algorithm is a weighted sample of the approximate posterior distribution, and inferences can therefore be made using weighted mean method.

In order to take advantage of the fact that the σ has a better correlation with the summary statistics when r and ϕ are known, we adapt this algorithm into a two stage algorithm. We start by allocating $\frac{2}{3}$ of the total sample size for the full algorithm above to reach an estimate for r and ϕ. We then use the remaining sample size with the above algorithm but fixing r and ϕ and only varying σ. And since we no longer propose ϕ base on $\boldsymbol{N}$, the importance sampling weight for the second stage of the process is just P without the fraction factor of the posterior density of ϕ.

6.6.2 The GIMH algorithm

We will now look at the implementation of the GIMH algorithm for the Ricker model. Again, modifications are made to the GIMH algorithm in order to account for the choice of ϕ. For the proposal distributions, we use folded Normal distributions $\left|N(\mu, 4)\right|$ and $-\left|N(\mu, 0.01)\right|$ for $log(r)$ and $log(\sigma)$ respectively. In effect, we are only performing the GIMH algorithm to r and σ, and ϕ is treated as a latent variable that we record. Let $q_r(a, b)$ and $q_\sigma(a, b)$ denote the density functions of the proposal distributions of r and σ respectively. Then $q_r(a, b) = f_{\mu=b,\sigma^2=4}(a) + f_{\mu=b,\sigma^2=4}(-a)$, where $f_{\mu=b,\sigma^2=4}(x)$ is the density function of $N(b, 4)$. $q_\sigma(a, b)$ can be defined similarly.

The algorithm

Algorithm 24 (dcABC algorithm for the Ricker model)**.**

1. We use the dcABC algorithm to perform a pilot run. The estimate of $\{r, \sigma\}$ from the pilot run is then used as the initial value for the GIMH algorithm.
2. Use $\left|N(\mu, 2^2)\right|$ and $-\left|N(\mu, 0.15^2)\right|$ to propose new values of $log(r)$ and $log(\sigma)$ respectively. We then exponentiate them to recover the new proposed parameters (r', σ').
3. Generate $2L$ terms of $\boldsymbol{N}$, and take only the last L terms.
4. Sample ϕ_i from $\Gamma\big(7.5 + \sum_j Y_j^*, 1 + \sum_j N_j\big)$.
5. Use the data conditioned simulation to estimate the likelihood, P_i. Set $\widehat{\pi(\boldsymbol{Y^*}|r_i, \sigma_i, \phi_i)} = P_i \frac{\pi(\phi)}{\pi(\phi|\boldsymbol{N}, \boldsymbol{Y^*}, r, \sigma)}$. We will denote this quantity Q_i as in the dcABC algorithm. Note that in GIMH algorithm the N augmentation size is set to 1.
6. Set $\{r_{i+1}, \sigma_{i+1}\} = \{r', \sigma'\}$ and $Q_{i+1} = Q'$ with probability $\min\Big(1, \frac{q_r(r_i, r')q_\sigma(\sigma_i, \sigma')Q'\pi(r')\pi(\sigma')}{q_r(r', r_i)q_\sigma(\sigma', \sigma_i)Q_i\pi(r_i)\pi(\sigma_i)}\Big)$, or else $\{r_{i+1}, \sigma_{i+1}, \phi_{i+1}\} = \{r_i, \sigma_i, \phi_i\}$ and $Q_{i+1} = Q_i$.
7. To achieve a sample of size m, we repeat steps 2. to 6. for m number of times.

In the GIMH algorithm, we don't sample multiple $\boldsymbol{N}$ for a given set of r, σ, and therefore the $\boldsymbol{N}$ augmentation size is 1.

We make similar adaptation to the GIMH algorithm as the dcABC algorithm. We allocate $\frac{2}{3}$ of the sample size to the original algorithm and we then use the estimations of r, σ, and ϕ as the initial point of the second stage calculation. We then perform the GIMH algorithm by fixing r and ϕ. For the second stage of the GIMH algorithm, we make the following changes to the original GIMH algorithm:

- For step 1, we use the estimation from the stage 1 as the initial value.
- Replace step 2, we propose r' using $-log(r) \sim N(r, 0.15^2)$.
- Skip step 4.
- For step 5, set $Q_i = P_i$.
- For step 6, the acceptance probability: $\min\left(1, \frac{q_\sigma(\sigma_i, \sigma')Q'\pi(\sigma')}{q_\sigma(\sigma', \sigma_i)Q_i\pi(\sigma_i)}\right)$.

We are expecting to see less improvements in estimating σ in the second stage than the dcABC algorithm, because GIMH proposes new parameters based on previously accepted value whereas the dcABC algorithm proposes new parameters uniformly across the parameter space.

While we were testing the algorithm, we noticed the presence of some wild outliers. Upon investigation, it was the result of the calculation of the acceptance probability. Since we often reach the accuracy limit of the system with the likelihood estimation, the numerator and the denominator could both be 0. The problem arises when the denominator is 0 and both the numerator and the denominator are 0. We resolve the problem by accepting the proposal if the denominator is 0 unless both the numerator and the denominator are both 0, then we reject. The problem of 0 denominator only occur at the beginning stage of the algorithm when the pilot run outputs an initial parameter that has close to 0 likelihood estimate.

6.6.3 Results

We compare the performance of the dcABC algorithm and the GIMH algorithm to the SMCMC algorithm proposed in [Wood, 2010]. We will also compare the estimates in two different ways. The first way, we compare how the algorithms perform across data generated from 250 distinct parameters. This should give us an insight to how the algorithm estimation reflects the true parameters. The second way, we use the algorithms to repeatedly analyse the same data for 250 times. We are looking to assess the consistency of the estimation of both algorithms. We set all three algorithms to have iteration size equivalent to 1,000,000. The dcABC algorithm has parameter sample size of $40,000$, $\boldsymbol{N}$ augmentation size of 5, and group size of 5. The GIMH algorithm has parameter sample size of $40,000$, $\boldsymbol{N}$ augmentation size of 1, and the group size of 25. In order to have a fair comparison, the SMCMC algorithm has a parameter size of $40,000$ and the synthetic likelihood simulation size is set to 25. We also report the estimation of σ from the two stages of the algorithm for comparison for the dcABC and the GIMH algorithms. Simple linear models (an intercept term and one coefficient) are used as a way of assessing the performance of the estimated values against the true values. For a perfectly estimated parameter, we should see the intercept being 0, the single coefficient being 1, and $R^2 = 1$.

Results from 250 distinct parameter sets

Figure 6.25: dcABC estimations of the Ricker Model

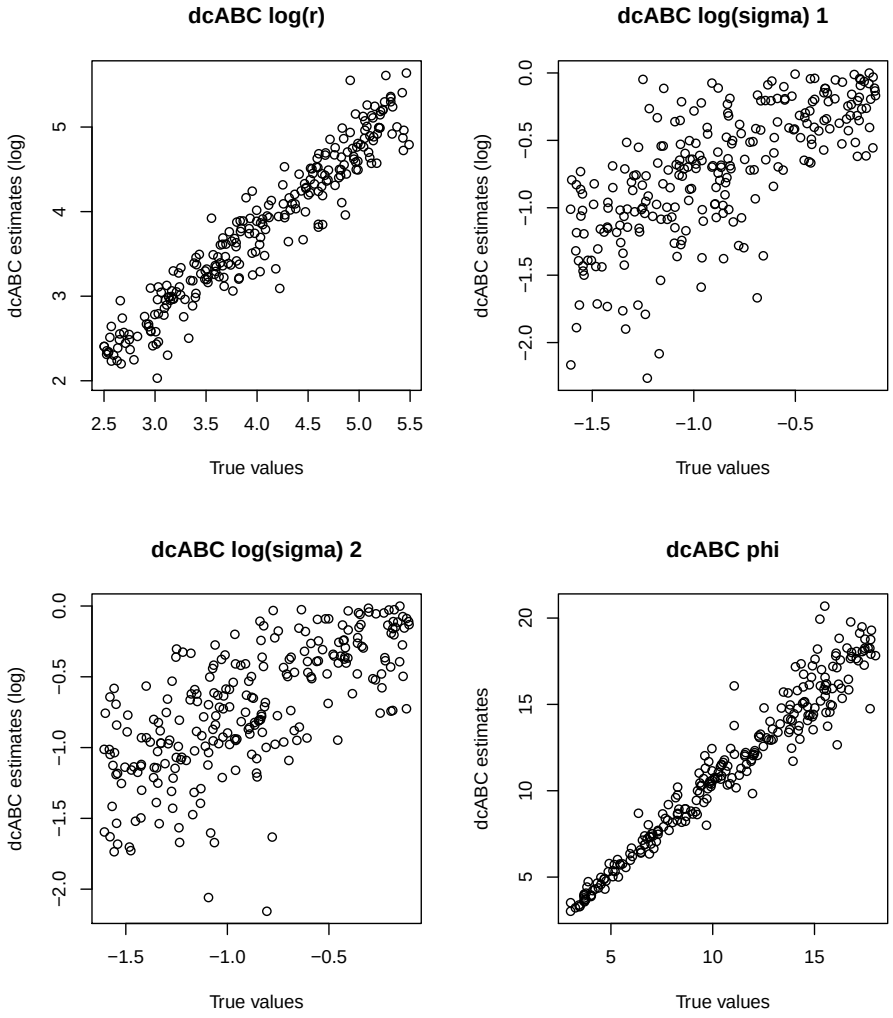


Figure 6.25 shows the the dcABC estimations of the three parameters against their corresponding true values. For r and σ, the plots are based on log of the values because of the way the prior distributions were designed. $log(r)$ and ϕ show almost perfectly linear relationships with little variance. The plot of σ

is less defined, which was expected due its lower correlations with all of the summary statistics. Nonetheless, we can see a general positive correlation in both stages of the estimations of the σ but with large variance. A somewhat surprising observation is that there seems to be little difference between the first stage and the second stage of the estimations of σ.

Figure 6.26: GIMH estimations of the Ricker Model

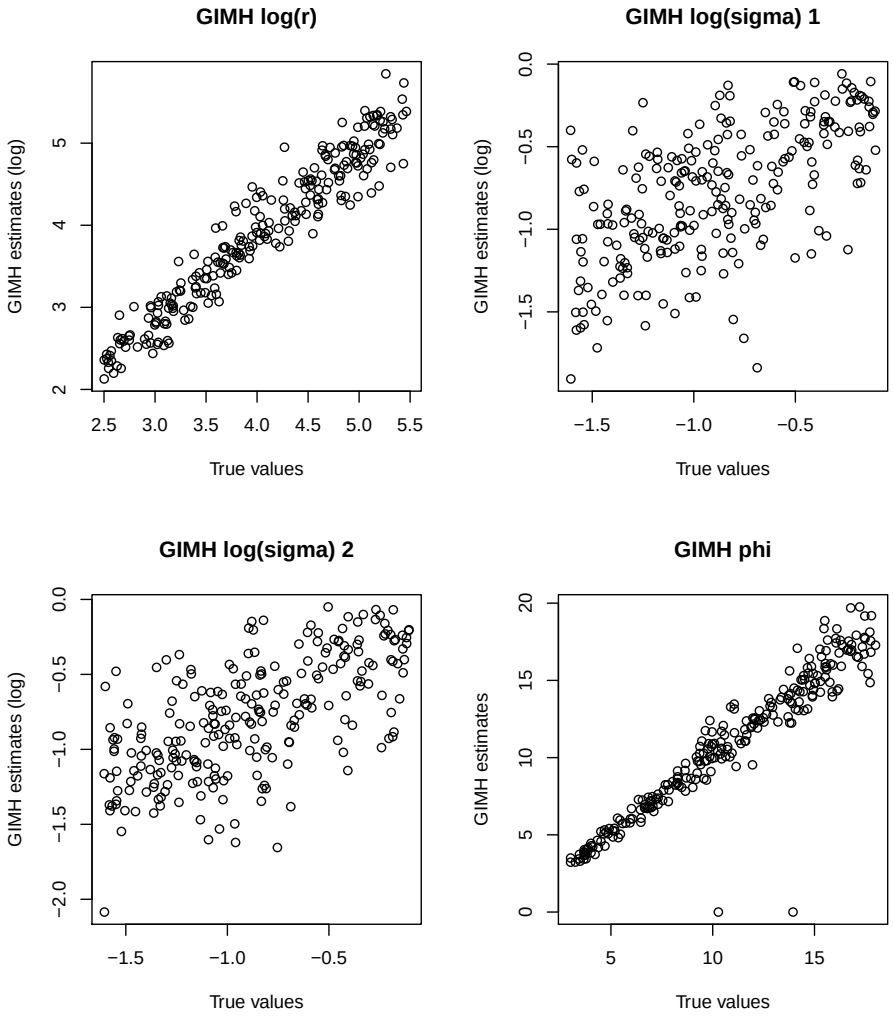
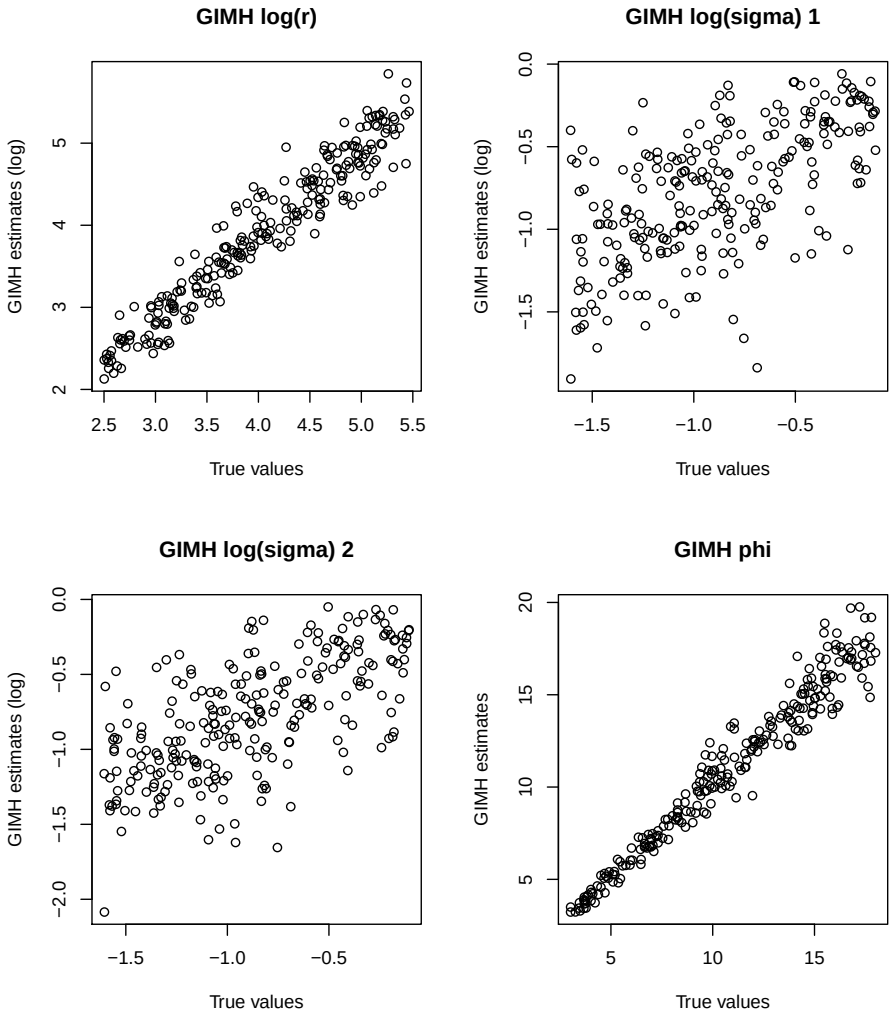


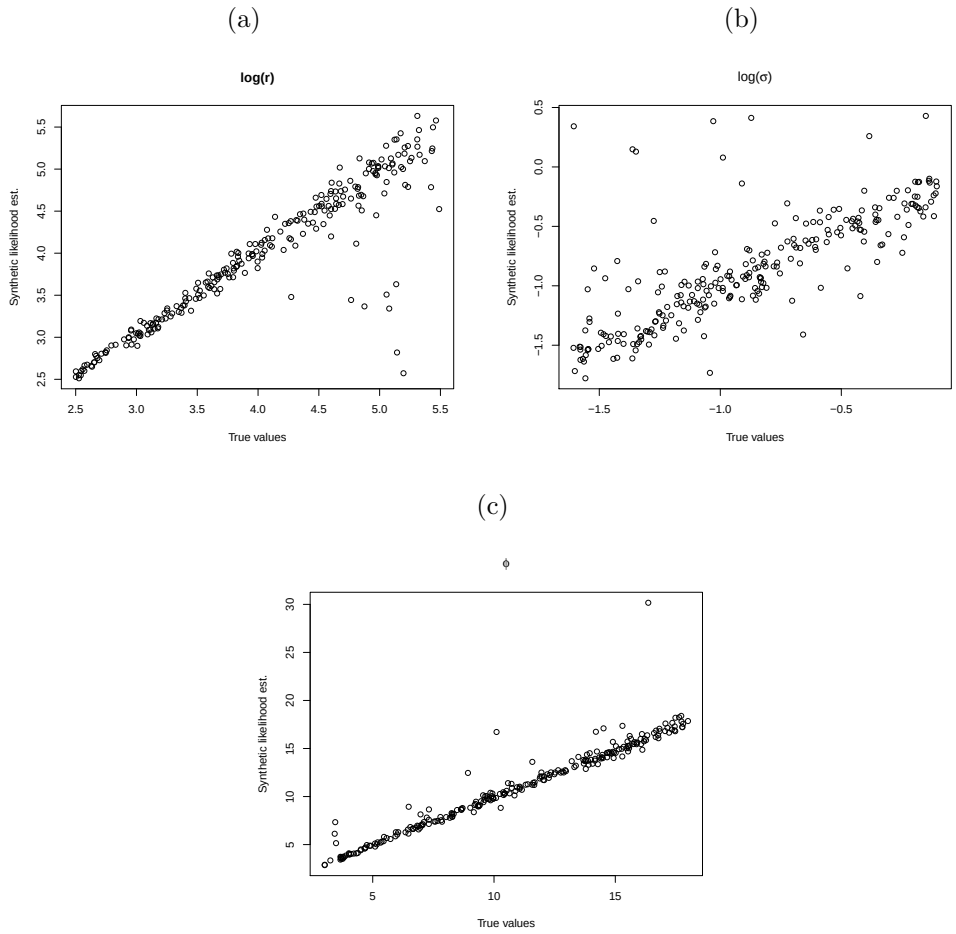
Figure 6.26 shows the GIMH estimations of the three parameters against their corresponding true values. We can observe two clear outliers from the estimation of ϕ. Further investigation suggests that these outliers were caused by the pilot run failed to find a reasonable initial point and triggered the fail safe mechanism which outputs 0's for all three parameters.

Figure 6.27: GIMH estimations of the Ricker Model excluding the outliers.



We replotted Figure 6.26 in Figure 6.27 with the outliers removed. Again, r and ϕ show a good positive linear relationships between the estimated values and the true values. σ is less correlated. However, the second stage estimation of σ appears to have a better positive linear relationship.

Figure 6.28: SMCMC estimates



In Figure 6.28, we plot the SMCMC estimates of the three parameters of the Ricker model. From the plot, it can be observed that SMCMC estimates exhibit similar trend to the dcABC and the GIMH algorithm in that r and ϕ

are better estimated than σ. Although, it appears that SMCMC σ estimates has better resemblence to the 45° line, except a few outliers. We also observe comparable number of outliers between the SMCMC estimate and the GIMH algorithms.

Figure 6.29: SMCMC estimates

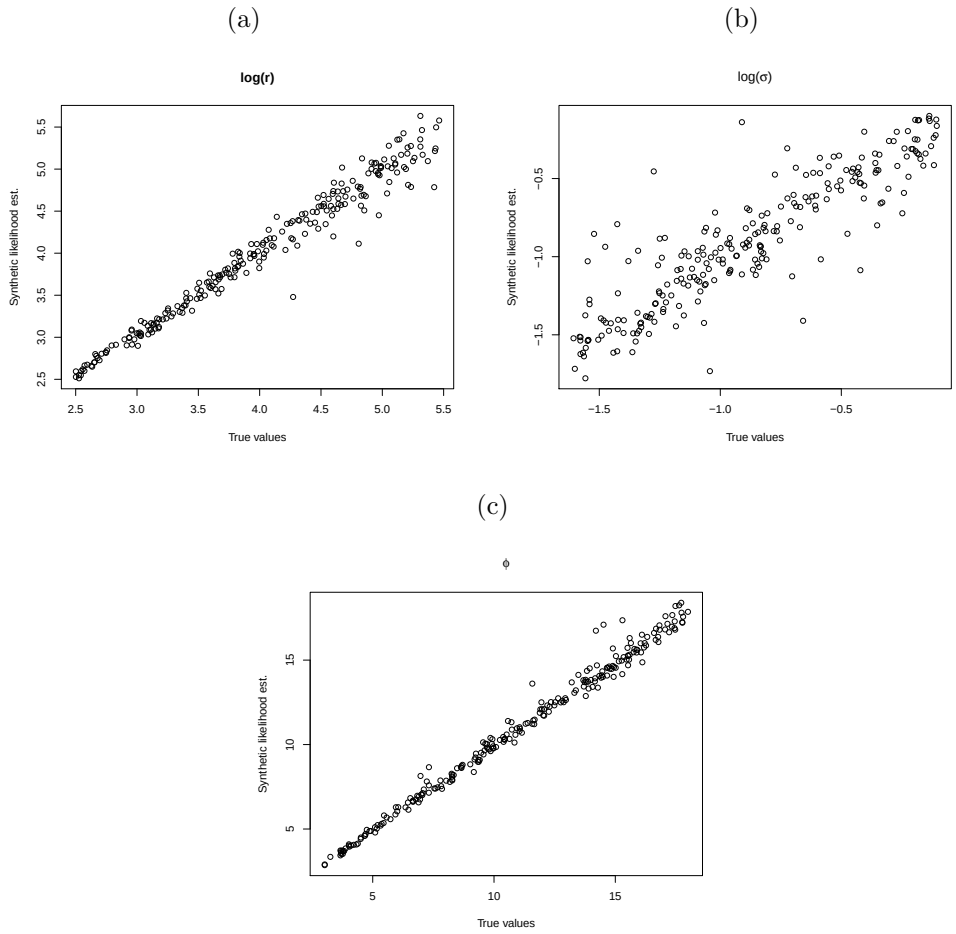


Figure 6.29 is the same estimate from Figure 6.28 with the outliers removed. An obvious improvement is that σ estimates appear to show a better fit in Figure 6.29.

Table 6.6: Simple linear models fitted for the estimations of the dcABC algorithm again the true values. β_0 denotes the intercept term and β_1 denotes the coefficient term.

	β_0	β_1	adj. R^2
$log(r)$	-0.2429	1.0043	0.9189
$log(\sigma)$ stg. 1	-0.0470	0.7651	0.4887
$log(\sigma)$ stg. 2	-0.0837	0.7159	0.4716
ϕ	0.1417	1.0315	0.9458

Table 6.7: Linear models fitted for the estimations of the GIMH algorithm again the true values after removing the outliers. β_0 denotes the intercept term and β_1 denotes the coefficient term.

	β_0	β_1	adj. R^2
$log(r)$	-0.2445	1.0309	0.9264
$log(\sigma)$ stg. 1	-0.2664	0.5765	0.3792
$log(\sigma)$ stg. 2	-0.2661	0.5888	0.4168
ϕ	-0.1514	1.0038	0.9570

Table 6.8: Linear models fitted for the estimations of the SMCMC algorithm again the true values after removing the outliers. β_0 denotes the intercept term and β_1 denotes the coefficient term.

	β_0	β_1	adj. R^2
$log(r)$	0.1664	0.9590	0.9691
$log(\sigma)$	-0.1301	0.8745	0.7963
ϕ	0.0365	0.9921	0.9881

Comparing Tables 6.6 and 6.9, both algorithms perform similarly with mariginal differences in estimating r and ϕ. However, the dcABC algorithm

performs slightly better when estimating σ, because the coefficient, β_1, is closer to 1 and the R^2 value is closer to 1. In contrary to our earlier prediction with regard to the estimation of σ between the two stages, the GIMH algorithm show an improvement with higher β_1 and larger R^2, whereas the dcABC algorithm shows the opposite. After the removal of the outliers in the SMCMC estimates, the SMCMC estimates has closer to 1 β_1 and closer to 1 adjusted R^2 value for all three parameters than both data conditioned algorithms. We will look at how both algorithms perform on repeatedly estimating the same data.

Results from estimating the same data 250 times

Figure 6.30: Box plot of GIMH estimations and dcABC estimations

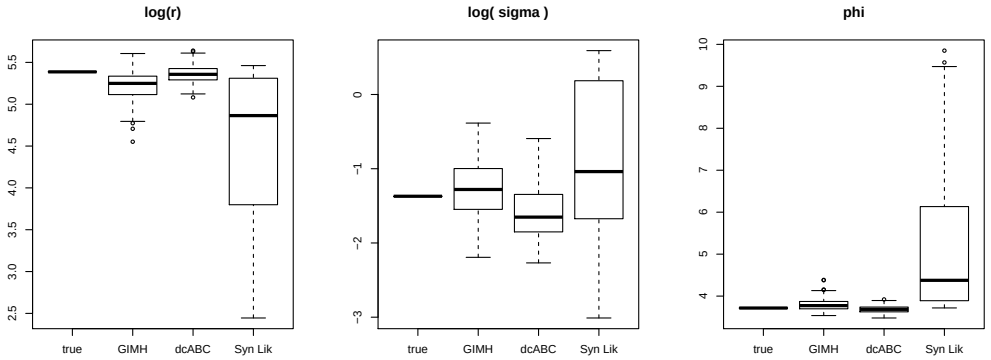


Table 6.9: Mean estimates of GIMH, dcABC, and SMCMC

	$log(r)$	$log(\sigma)$	ρ
True values	5.3868	-1.3708	3.7156
GIMH	5.2344	-1.1904	3.7961
dcABC	5.3647	-1.5823	3.6861
SMCMC	4.5372	-0.8591	4.9146

The box plots in Figure 6.30 shows that the dcABC algorithm has the smallest interquartile range and the true values are all within the interquartile range for all three parameters. The GIMH algorithm has slightly larger interquartile ranges and the true value of $log(r)$ lies outside the interquartile range of the GIMH estimates but within the estimate's range. The SMCMC algorithm has significantly larger interquartile range than both data conditioned simulation related algorithms which indicates larger Monte Carlo error. On surface it seems that the dcABC algorithm gives the most consistent estimates, but the computation time of SMCMC is roughly 6.5 times faster than both the dcABC and the GIMH algorithms without parallelisation.

6.7 Chapter Conclusion

We have shown that dcABc and GIMH are both affective in estimating r and ϕ. However, both of the algorithms come short in estimating σ with the dcABC algorithm performs slightly better. A surprising observation is that the second stage of both algorithm did not show a clear improvement in the estimation of σ. However, considering the weaker correlations between σ and the summary statistics, both algorithm performed better than anticipated. However, with the presence of outliers and the larger inter quartile range, the GIMH algorithm is less practical than the dcABC algorithm.

When comparing dcABC and GIMH to SMCMC, SMCMC appears to estimate all three parameters better and especially in estimating σ. However, when we repeatedly apply all three algorithms on a like for like basis (40000 iterations on parameter sampling and 25 samples for each likelihood estimation) to the same set of data, the SMCMC appears to have significantly larger variation in its estimates. Although, the SMCMC is about 6.5 times faster than the dcABC and the GIMH, it is possible to speedup both algorithms by implementing them in C for example, where iteration is significantly faster than in R, however, such exercise is beyond the scope of this thesis. We also tested SMCMC with double of synthetic likelihood sampling size (50),

the outcome shows improved interquartile ranges as well as the number of outliers on estimation of all three parameters.

Chapter 7

Conclusions

7.1 Discussion

At the beginning of this research, we set out to explore the possibilities of the model simulation aspect of the ABC algorithm as we realised that little research has been done in this particular area. The idea of data condition simulation is introduced as a more efficient and robust alternative to the unconditioned data simulation when it comes to estimating the likelihood. Although in this thesis, we discuss the data conditioned simulation and the dcABC algorithm together, the main contribution of the research is in the data conditioned simulation. We achieved the data conditioned simulation by using a mixture of the data augmentation algorithm and importance sampling algorithm, so that we can ensure that the simulated data closely matches its summary statistics to the observed data, and in our examples, we matched the summary statistics exactly. The idea of data conditioned simulation itself is not completely original (see [Beaumont, 2003]). It was mainly featured as a by-product of the grouped independence Metropolis-Hastings algorithm. However, our research provides a practical guide to utilise this feature. We introduced the idea of data conditioned simulation in the context of the

dcABC algorithm and the GIMH algorithm, but one of the major advantages of the data conditioned simulation is not restricted to the dcABC algorithm and the GIMH algorithm. The data conditioned simulation can be used in many other ABC related algorithms.

In the three examples presented in this thesis, the dcABC algorithm performs consistently and gives robust estimates when applied to complex models (such as time-inhomogeneous Markov chain model, and the Ricker model). When compared to the GIMH algorithm, the dcABC algorithm gives just as good parameter estimates and with lower possibility of producing outliners. Although the GIMH algorithm appears to be slightly more time efficient in complex models, it is only so when compared to the un-parallelised version of the dcABC algorithm. The dcABC algorithm is classified as an embarrassingly parallel problem which has no serial dependence between iterations, whereas the GIMH algorithm is serially dependent and therefore parallelisation would be challenging. From the examples we presented, we demonstrate the flexibility and adaptability of the algorithm.

From a computation point of view, even though the GIMH algorithm is slightly better in terms of its computation time in our examples, the dcABC algorithm has the advantage of being easily parallelisable as well as the implementation of dcABC algorithm is simpler.

To conclude, the dcABC algorithm and the GIMH algorithm show little difference in terms of their estimation. The dcABC algorithm demonstrates the idea of the data conditioned simulation in its pure form and proves that it is a viable solution to approach complex models. The GIMH algorithm demonstrates how the data conditioned simulation can substitute the likelihood estimate in other ABC related algorithms, such as the rejection sampling based method. We believe the data conditioned simulation is a robust and practical method to estimating the likelihood of complex models.

7.2 Further research

We will discuss some of the common problems that we found during the research and areas that we think are worth future exploration.

One of the main points of failure during the computation is that the importance sampling weight for the data conditioned simulation reaches the precision limit of the computer. Although, there are measures can be taken such as using a constant factor or using log(), they may not work so well in practice. A careful choice of the constant factor is required, otherwise the sampling weight may reach the upper limit of the system accuracy. Choosing the optimal factor for the data conditioned simulation is an interesting topic for further research. The main issue with using the log probability is in the calculation of the weighted mean. To calculate the weight mean, we still require the raw weight, since

$$\frac{\sum x_i p_i}{\sum p_i} \neq \frac{\sum x_i \log(p_i)}{\sum \log(p_i)}.$$

Although there will be definitely be a strong correlation between the two estimates, and we may be able infer the raw weighted mean from the log weighted mean, however, we cannot be certain until further research is done.

The data conditioned simulation relies heavily on the use of truncated distributions when sampling. That is the feature that provides the steering we need. In theory, the data conditioned simulation is rigorous, and indeed the data conditioned simulation in the SIR model and the time inhomogeneous Markov chain model has a firm mathematical reasoning. Whereas in the Ricker model case, a more practical approach was required in both the summary statistics selection and the data conditioned simulation, especially in the data conditioned simulation, some ad-hoc adjustments were made based purely on practicality, so that the computation will produce a valid output. It would be an interesting exercise to compare the effects on the parameter estimation between these adjustments.

Finally, it would certainly be interesting to see how other ABC related algorithms compare to their data conditioned adapted counter parts.

Bibliography

Pierre Alquier, Nial Friel, Richard Everitt, and Aidan Boland. Noisy monte carlo: Convergence of markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47, 2016.

Gidi Amir. Continuous time markov chains. URL `http://u.math.biu.ac.il/~amirgi/CTMCnotes.pdf`.

Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient monte carlo computations. *The Annals of Statistics*, pages 697–725, 2009.

Christophe Andrieu and Matti Vihola. Establishing some order amongst exact approximations of mcmcs. *arXiv preprint arXiv:1404.6909*, 2014.

Christophe Andrieu, Matti Vihola, et al. Convergence properties of pseudo-marginal markov chain monte carlo algorithms. *The Annals of Applied Probability*, 25(2):1030–1077, 2015.

David K Arrowsmith and Colin M Place. *An introduction to dynamical systems*. Cambridge University Press, 1990.

Bailey. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

Frank Ball. A unified approach to the distribution of total size and total area under the trajectory of infectives in epidemic models. *Advances in Applied Probability*, pages 289–310, 1986.

Mark A Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003.

Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

Alexandros Beskos, Dan Crisan, Ajay Jasra, and Nick Whiteley. Error bounds and normalizing constants for sequential monte carlo in high dimensions. *arXiv preprint arXiv:1112.1544*, 2011.

Fernando V Bonassi, Mike West, et al. Sequential monte carlo with adaptive weights for approximate bayesian computation. *Bayesian Analysis*, 10(1): 171–187, 2015.

Paola Bortot, Stuart G Coles, and Scott A Sisson. Inference for stereological extremes. *Journal of the American Statistical Association*, 102(477):84–92, 2007.

Tom Britton. Stochastic epidemic models: a survey. *Mathematical biosciences*, 225(1):24–35, 2010.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential monte carlo method for approximate bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

Klaus Dietz. The estimation of the basic reproduction number for infectious diseases. *Statistical methods in medical research*, 2(1):23–41, 1993.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2011.01010.x. URL `http://dx.doi.org/10.1111/j.1467-9868.2011.01010.x`.

William Feller. *An introduction to probability theory and its applications, Volume 2*. Wiley, New York, 1957. ISBN 978-0-471-25709-7.

Sarah Filippi, Chris P Barnes, Julien Cornebise, and Michael PH Stumpf. On optimality of kernels for approximate bayesian computation using sequential monte carlo. *Statistical applications in genetics and molecular biology*, 12(1):87–107, 2013.

Ian Foster. Designing and building parallel programs, 1995.

Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

Donald Ludwig. Final size distribution for epidemics. *Mathematical Biosciences*, 23(1):33–46, 1975.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

Felipe J Medina-Aguayo, Anthony Lee, and Gareth O Roberts. Stability of noisy metropolis–hastings. *Statistics and Computing*, pages 1–25, 2015.

Alexandru Mereacre. Transition probabilities for inhomogeneous continuous time markov chains. URL `http://www-i2.informatik.rwth-aachen.de/i2/fileadmin/user_upload/documents/Mereacre/ictmc.pdf`.

Peter Neal. Efficient likelihood-free bayesian computation for household epidemics. *Statistics and Computing*, pages 1–18, 2012. ISSN 0960-3174. URL `http://dx.doi.org/10.1007/s11222-010-9216-x`. 10.1007/s11222-010-9216-x.

Peter Neal and Chien Lin Huang. Forward simulation mcmc with applications to stochastic epidemic models. *Scandinavian Journal of Statistics*, 2015.

Peter Neal and Gareth Roberts. Optimal scaling for random walk metropolis on spherically constrained target densities. *Methodology and Computing in Applied Probability*, 10(2):277–297, 2008.

J.R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997. ISBN 9781107393479. URL `http://books.google.co.uk/books?id=xbmtAQAAQBAJ`.

Philip D O'Neill, David J Balding, Niels G Becker, Mervi Eerola, and Denis Mollison. Analyses of infectious disease data from household outbreaks by markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 49(4):517–542, 2000.

Gareth W Peters, Yanan Fan, and Scott A Sisson. On sequential monte carlo, partial rejection control and approximate bayesian computation. *Statistics and Computing*, 22(6):1209–1222, 2012.

J K Pritchard, M T Seielstad, A Perez-Lezaun, and M W Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, 1999. URL `http://mbe.oxfordjournals.org/content/16/12/1791.abstract`.

Oliver Ratmann, Ole Jørgensen, Trevor Hinkley, Michael Stumpf, Sylvia Richardson, and Carsten Wiuf. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of h. pylori and p. falciparum. *PLoS Computational Biology*, 3(11):e230, 2007.

W. E. Ricker. Stock and recruitment. *Journal of the Fisheries Research Board of Canada*, 11(5):559–623, 1954. doi: 10.1139/f54-039. URL `http://www.nrcresearchpress.com/doi/abs/10.1139/f54-039`.

Christian P Robert, Mark A Beaumont, Jean-Michel Marin, and Jean-Marie Cornuet. Adaptivity for abc algorithms: the abc-pmc scheme. *arXiv preprint arXiv:0805.2256*, 2008.

Thomas Sellke. On the asymptotic distribution of the size of a stochastic epidemic. *Journal of Applied Probability*, 20(2):pp. 390–394, 1983. ISSN 00219002. URL `http://www.jstor.org/stable/3213811`.

Daniel Silk, Saran Filippi, and Michael PH Stumpf. Optimizing threshold-schedules for approximate bayesian computation sequential monte carlo samplers: applications to molecular systems. *arXiv preprint arXiv:1210.3296*, 2012.

Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104 (6):1760–1765, 2007.

David Thompson, William H Foege, World Health Organization, et al. *Faith Tabernacle smallpox epidemic, Abakaliki, Nigeria*. World Health Organization, 1968.

Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

Simon R White, Theodore Kypraios, and Simon P Preston. Fast approximate bayesian computation for discretely observed markov models using a factorised posterior distribution. *arXiv preprint arXiv:1301.2975*, 2013.

Ward Whitt. Continuous-time markov chains. URL `http://www.columbia.edu/~ww2040/6711F13/CTMCnotes120413.pdf`.

S.N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.