Sequential Monte Carlo for Markov jump processes

Matthew Upton

Supervisor: Dr Andrew Golightly

1st May 2015

Abstract

Computational systems biology is concerned with the development of detailed mechanistic models of biological processes. In this project we consider a stochastic kinetic model of a process of interest, defined as a Markov jump process. The problem of performing Bayesian inference for the model parameters will be considered and a sequential Monte Carlo (SMC) scheme developed, that allows our beliefs to be updated as each observation becomes available. For a certain choice of prior distribution and, under the assumption of mass action kinetics, the parameter posterior at each time point can be summarised in terms of a number of (low dimensional) statistics. We aim to exploit these inside the SMC scheme to improve efficiency.

Contents

Chapter 1

Introduction

The standard approach for modelling biochemical networks is to derive ordinary differential equations (ODEs) using the law of mass action and the concentrations of each species. Such an approach, however, assumes that the time-evolution of a system is continuous and deterministic. In reality chemical reactions occur as discrete events as a result of molecular collisions, which are impossible to predict with certainty [1]. Thus for the random processes that underlie systems biology, it is not always possible to satisfactorily implement a deterministic approach.

In order to understand such systems, a stochastic approach is adopted. The resulting stochastic kinetic models are most naturally represented by Markov jump processes (MJPs). These are continuous-time, discrete-valued Markov processes. Given a set of initial conditions and rate constants governing each reaction, exact realisations of the MJP can be achieved through a discrete event simulation technique, known in this context as Gillespie's direct method [1]. Generating forward simulations from the model is therefore straightforward, and can give insight into the behaviour of a system of interest. However, plausible parameter (rate constant) values must be determined from experimental data that may be incomplete and subject to error.

The aim of this project is to perform Bayesian inference for the rate constants of a given reaction network using discrete-time noisy measurements of the system components (species). This problem is challenging since transition probabilities governing the MJP are rarely available in closed form. We will therefore adopt a simulation based Monte Carlo approach to generate samples from a target posterior density. In particular, we will focus on a sequential approach, by updating our beliefs about the rate constants as each observation becomes available. Thus, a sequential Monte Carlo (SMC) scheme will be constructed. Such schemes are typically termed *particle filters*, as they aim to propagate a weighted swarm of parameter samples or particles, according to the information contained in each new observation. Essentially, a weighted resampling scheme is applied iteratively, giving an algorithm known as the bootstrap particle filter [2].

Unfortunately, SMC schemes for static parameter problems can be afflicted by particle degeneracy, which can occur when very few particles have resonable weight, resulting in parameter posteriors collapsing to a point mass (eventually). We therefore aim to mitigate this problem by implementing a filter due to Storvik [3], [4]. Under the assumption of mass action kinetics, the parameter posterior at each time point can be summarised in terms of a number of (low dimensional) statistics. These statistics can be propagated inside an

SMC scheme to give a mechanism for rejuvenating the particle set at each time point. We apply these methods to infer the rate constants governing an immigration-death process and a simple model of predator-prey interaction proposed by Lotka and Volterra [5], [6].

The remainder of this report is organised as follows. In Chapter 2 we give an introduction to stochastic kinetic models. Chapter 3 describes a Bayesian approach to inference, before a sequential implementation is given in Chapter 4. We apply the methodology to two examples in Chapter 5 before drawing conclusions and considering future directions in Chapter 6.

Chapter 2

Stochastic Kinetic Models

Given a system of chemical reactions, their rate constants and associated rate laws, a stochastic kinetic model describes the probabilistic behaviour of each species through time. A Markov jump process (MJP), that is, a continuous time, discrete valued Markov process, provides a natural representation of the biochemical network.

In this chapter, we provide a review of reaction networks and their MJP representation. For further details we refer the reader to [7].

2.1 Reaction networks

Suppose k species $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ are involved in a set of r reactions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_r$, where the rate constant of the i^{th} reaction is denoted by θ_i. Then the reaction network representation of those reactions is

$$\begin{array}{lcccc}
\mathcal{R}_1: & p_{11}\mathcal{X}_1 + p_{12}\mathcal{X}_2 + \ldots + p_{1k}\mathcal{X}_k & \xrightarrow{\theta_1} & q_{11}\mathcal{X}_1 + q_{12}\mathcal{X}_2 + \ldots + q_{1k}\mathcal{X}_k \\
\mathcal{R}_2: & p_{21}\mathcal{X}_1 + p_{22}\mathcal{X}_2 + \ldots + p_{2k}\mathcal{X}_k & \xrightarrow{\theta_2} & q_{21}\mathcal{X}_1 + q_{22}\mathcal{X}_2 + \ldots + q_{2k}\mathcal{X}_k \\
\vdots & \vdots & \vdots & \vdots \\
\mathcal{R}_r: & p_{r1}\mathcal{X}_1 + p_{r2}\mathcal{X}_2 + \ldots + p_{rk}\mathcal{X}_k & \xrightarrow{\theta_r} & q_{r1}\mathcal{X}_1 + q_{r2}\mathcal{X}_2 + \ldots + q_{rk}\mathcal{X}_k
\end{array}$$

where $p_{ij}, q_{ij} \in \mathbb{N}$ are non-negative integers. The species on the left are the reactants and the species on the right are the products. The p_{ij} and q_{ij} are known as the stoichiometric coefficients where

- p_{ij} is the coefficient of reactant j in reaction i, and
- q_{ij} is the coefficient of product j in reaction i.

When a reaction occurs, the number of each reactant reduces by the value of its stoichiometry, whilst the number of each product increases by the value of its stoichiometry. If the reactants destroy each other or are removed from the system, the empty set is used on the right hand side of the reaction. Similarly, the empty set is used on the left hand side of the reaction if the products are intoduced to the system from outside.

A matrix representation can be useful to see how the state of the system is updated as reactions occur. The matrix S, with $(i,j)^{th}$ element equal to $q_{ij} - p_{ij}$, gives the effect of

reaction j on species i. This matrix is known as the stoichiometry matrix and clearly has dimensions $k \times r$.

Consider a reaction

$$\mathcal{R} : \mathcal{X}_1 + \mathcal{X}_2 \xrightarrow{\theta} \mathcal{X}_3.$$

This will occur when a molecule of $\mathcal{X}_1$ collides with a molecule of $\mathcal{X}_2$. Assuming a well stirred container in thermal equilibrium gives a constant collision rate/hazard. Therefore

$$Pr(\text{2 molecules collide in a time interval of length } dt) = \theta dt$$

Supposing there are currently x_1 lots of $\mathcal{X}_1$ and x_2 lots of $\mathcal{X}_2$, then

$$Pr(\text{Reaction in time interval of length } dt) = \theta x_1 x_2 dt$$

This probability only depends on the current system state, giving the Markov property, where the future is only dependent on the past through the present. Also, when a reaction occurs, it's effect is to change the relevant states by an integer amount, meaning this is a discrete valued process. This leads us to model the system dynamics (time course behaviour) with a Markov jump process (MJP), considered in the next section.

2.2 Markov jump process representation

In order to fully specify the Markov process it is also necessary to specify a rate law for each reaction. In the above reaction, $\theta x_1 x_2$ is called the instantaneous rate/hazard. For a general reaction network, we denote the hazard of reaction R_i by

$$h_i(X_t, \theta_i) = \theta_i g(X_t),$$

where $X_t = (X_{1,t}, X_{2,t}, \ldots, X_{k,t})^T$ denotes the state of the system at time t. If the form of each rate is known, then we have a complete specification of a Markov process, and sufficient information to simulate from the process. Under the assumptions of mass action kinetics the hazard is specified by

$$g(X_t) = \prod_{j=1}^{k} \binom{X_{j,t}}{p_{i,j}} = \text{product of binomial coefficients.}$$

For examples of hazard forms under various reaction types, see table 2.1.
Given a set of reactions and their associated rates, we can write down the chemical master equation for the corresponding Markov process and simulate realisations from the process using the Gillespie algorithm (also known as Gillespie's direct method). With the state of the system denoted by $x = (x_1, x_2, \ldots, x_k)^T$ and the reactions $R_1, R_2, \ldots, R_r$ occuring with rates $h_1(x, \theta_1), h_2(x, \theta_2), \ldots, h_r(x, \theta_r)$, then the total rate is

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

When a reaction occurs, the probability that it is of type R_i is $h_i(x, \theta_i)/h_0(x, \theta)$. Moreover, the time between reaction events can be shown to follow an exponential distribution. Thus the algorithm has the following form:

Reaction				Hazard		
$\mathcal{R}_1$:	$\emptyset$	$\xrightarrow{\theta_1}$	$\mathcal{X}_1$	$h_1(X_t, \theta_1)$	=	θ_1
$\mathcal{R}_2$:	$\mathcal{X}_1$	$\xrightarrow{\theta_2}$	$\emptyset$	$h_2(X_t, \theta_2)$	=	$\theta_2 X_{1,t}$
$\mathcal{R}_3$:	$\mathcal{X}_1 + \mathcal{X}_2$	$\xrightarrow{\theta_3}$	$\mathcal{X}_3$	$h_3(X_t, \theta_3)$	=	$\theta_3 X_{1,t} X_{2,t}$
$\mathcal{R}_4$:	$2\mathcal{X}_1$	$\xrightarrow{\theta_4}$	$\mathcal{X}_3$	$h_4(X_t, \theta_4)$	=	$\theta_4 X_{1,t}(X_{1,t} - 1)/2!$

Table 2.1: Example reactions and their hazards under mass action kinetics.

1. Set $t = 0$. Initialise $\theta_1, \theta_2, \ldots, \theta_r$ and $x = (x_1, x_2, \ldots, x_r)^T$.
2. Calculate $h_i(x, \theta_i), i = 1, 2, \ldots, r$ and the combined hazard $h_0(x, \theta) = \sum_{i=1}^{r} h_i(x, \theta_i)$.
3. Simulate the time to the next reaction $\tau \sim Exp(h_0)$ and set $t := t + \tau$.
4. Simulate a reaction index j as a discrete random quantity with probability $h_j(x, \theta_i)/h_0(x, \theta)$.
5. Update x accordig to reaction R_j. That is, set x to $x + s_j$ where s_j is the j^{th} column of the stoichiometry matrix.
6. Output (or store) the current state and time, x and t.
7. If $t \leq T$, some max time, go to step 2. Otherwise, stop.

2.3 Examples

In this section we consider two toy reaction networks that will be used to illustrate the inference schemes described in Chapters 3 and 4.

2.3.1 Immigration-Death

In the immigration-death model, there are 2 reactions (immigration and death) for a single species. The immigrations come from an external source, thus the model has the form:

$$\begin{aligned} \mathcal{R}_1 : &\quad \emptyset \xrightarrow{\theta_1} \mathcal{X} \quad \text{i.e. } (x) \mapsto (x+1) \\ \mathcal{R}_2 : &\quad \mathcal{X} \xrightarrow{\theta_2} \emptyset \quad \text{i.e. } (x) \mapsto (x-1) \end{aligned}$$

where x represents the population level for the species. Reaction 1 ($\mathcal{R}_1$) is immigration, a zeroth-order reaction. Reaction 2 ($\mathcal{R}_2$) is death, a first order reaction. Thus the reaction hazards are

$$\begin{aligned} h_1(X_t, \theta_1) &= \theta_1 \\ h_2(X_t, \theta_2) &= \theta_2 x \end{aligned}$$

with stoichiometry matrix

$$S = \begin{pmatrix} 1 & -1 \end{pmatrix}.$$

The above description gives sufficient information for the system to be simulated, upon

specifying an initial condition and values of the rate constants. Below is a typical realisation for $(\theta_1, \theta_2) = (0.8, 0.1)^T$:

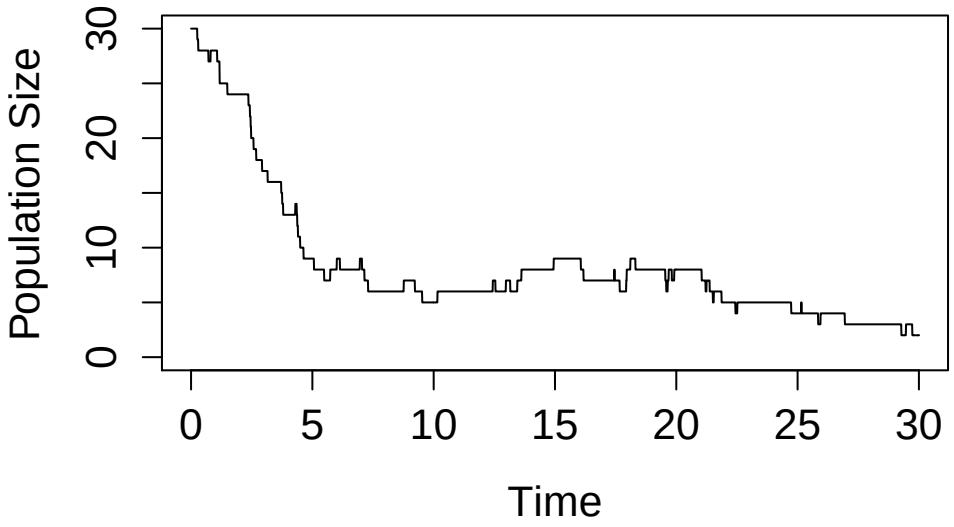

Figure 2.1: A realisation of population size for the immigration-death model using Gillespie's direct method with $\theta = (0.8, 0.1)^T$, initial population size of 30 and max time of 30.

2.3.2 Lotka-Volterra

Suppose $X(t) = (X_1(t), X_2(t))^T$ represents the numbers of individuals in two species in a predator-prey network with prey X_1 and predators X_2 (e.g. foxes and rabbits). Increases in prey population translates to an abundance in food for predators, so the predator population will tend to increase. Over-predation can then occur which causes a drop in prey numbers. The interaction between predators and prey, together with how the two populations breed to produce offspring, is clearly a random process with population levels varying over continuous time. The Lotka-Volterra predator-prey model has the form:

$$\begin{array}{lccll} \mathcal{R}_1: & \mathcal{X}_1 & \xrightarrow{\theta_1} & 2\mathcal{X}_1 & \text{i.e. } (x_1, x_2) \mapsto (x_1 + 1, x_2) \\ \mathcal{R}_2: & \mathcal{X}_1 + \mathcal{X}_2 & \xrightarrow{\theta_2} & 2\mathcal{X}_2 & \text{i.e. } (x_1, x_2) \mapsto (x_1 - 1, x_2 + 1) \\ \mathcal{R}_3: & \mathcal{X}_2 & \xrightarrow{\theta_3} & \emptyset & \text{i.e. } (x_1, x_2) \mapsto (x_1, x_2 - 1) \end{array}$$

where x_1, x_2 represent population levels for prey and predators (rabbits and foxes). Reaction 1 ($\mathcal{R}_1$) is prey reproduction, reaction 2 ($\mathcal{R}_2$) is prey death and predator reproduction, and reaction 3 ($\mathcal{R}_3$) is predator death. Therefore, the reaction hazards are

$$\begin{aligned} h_1(X_t, \theta_1) &= \theta_1 x_1 \\ h_2(X_t, \theta_2) &= \theta_2 x_1 \times x_2 \\ h_3(X_t, \theta_3) &= \theta_3 x_2 \end{aligned}$$

with stoichiometry matrix

$$S = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}.$$

As for the Immigration-Death model, the above description gives sufficient information for the system to be simulated. Below is a typical realisation for $(\theta_1, \theta_2, \theta_3)^T = (0.5, 0.0025, 0.3)^T$:

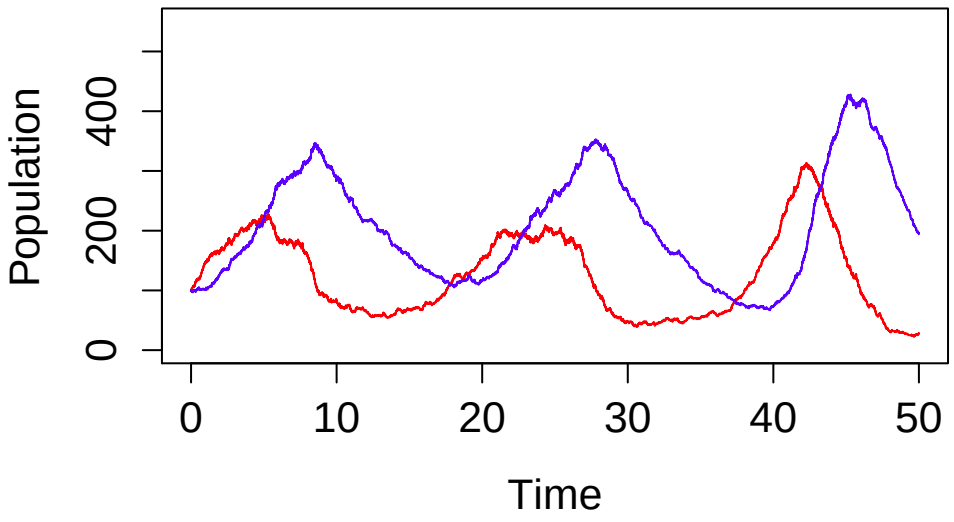

Figure 2.2: A realisation of population sizes for the Lotka-Volterra model using Gillespie's direct method with $\theta = (0.5, 0.0025, 0.3)^T$, initial population sizes of 100 and max time of 50. The red line represents the prey population, and the blue line represents the predator population.

Chapter 3

Bayesian Inference

When working within the Bayesian paradigm, we typically consider a parameter vector $\theta = (\theta_1, \theta_2, \ldots, \theta_p)^T$ and data $x = (x_1, x_2, \ldots, x_n)^T$. Given these data, we can then make inferences about the parameter values in a chosen model. A key ingredient of the Bayesian approach is the specification of a prior distribution for θ. Associated with this distribution is the *prior density* $\pi(\theta)$. We formulate a model that defines the distribution of X given the parameters, i.e. we specify a density $f_{X|\Theta}(x|\theta)$. This can then be regarded as a function of θ when we have got some fixed observed data x, called the *likelihood*.

The prior and likelihood determine the full *joint density* over data and parameters:

$$f_{\Theta,X}(\theta, x) = \pi(\theta) f_{X|\Theta}(x|\theta).$$

Given the joint density, we are then able to compute its marginal and conditional distributions:

$$f_X(x) = \int_\Theta \pi(\theta) f_{\Theta,X}(\theta, x)\ d\theta = \int_\Theta \pi(\theta) f_{X|\Theta}(x|\theta)\ d\theta$$

and

$$f_{\Theta|X}(\theta|x) = \frac{f_{\Theta,X}(\theta, x)}{f_X(x)} = \frac{\pi(\theta) f_{X|\Theta}(x|\theta)}{\int_\Theta \pi(\theta) f_{X|\Theta}(x|\theta)\ d\theta}.$$

The term $f_{\Theta|X}(\theta|x)$ is known as *the posterior density*, and is usually denoted $\pi(\theta|x)$, leading to the continuous version of Bayes theorem:

$$\pi(\theta|x) = \frac{\pi(\theta) f_{X|\Theta}(x|\theta)}{\int_\Theta \pi(\theta) f_{X|\Theta}(x|\theta)\ d\theta}$$

Now, the denominator is not a function of θ, so we can in fact write this as

$$\pi(\theta|x) \propto \pi(\theta) f_{X|\Theta}(x|\theta)$$

where the constant of proportionality is chosen to ensure that the density integrates to one. Hence, the posterior is proportional to the prior times the likelihood.

In what follows, we consider the task of formulating the posterior density for the rate constants governing a stochastic kinetic model.

3.1 Using complete data

We consider first a special case where all times and types of reaction are observed. Consider an interval $[0, T]$ and let r_j denote the number of reaction events of type $\mathcal{R}_j$, $j = 1, 2, \ldots, r$, and define $n = \sum_{j=1}^{r} r_j$ as the total number of reaction events over the interval. Reaction times (assumed to be in increasing order) and types are denoted by (t_i, ν_i), $i = 1, 2, \ldots, n$, $\nu_i \in \{1, 2, \ldots, r\}$ and we take $t_0 = 0$ and $t_{n+1} = T$. Let x be all reaction times and types in the interval. The *complete data likelihood* $\pi(x|\theta)$ is given in [7] as

$$\begin{aligned}\pi(x|\theta) &= \left\{\prod_{i=1}^{n} h_{\nu_i}\left(x_{t_{i-1}}, \theta_{\nu_i}\right)\right\} \exp\left\{-\sum_{i=1}^{n} h_0\left(x_{t_i}, \theta\right)\left[t_{i+1} - t_i\right]\right\} \\ &= \left\{\prod_{i=1}^{n} h_{\nu_i}\left(x_{t_{i-1}}, \theta_{\nu_i}\right)\right\} \exp\left\{-\int_0^T h_0\left(x_t, \theta\right)\, dt\right\}.\end{aligned}$$

For mass action kinetics, $h_j(x_t, \theta_j) = \theta_j g_j(x_t)$ and therefore

$$\begin{aligned}\pi(x|\theta) &= \prod_{j=1}^{r} \theta_j^{r_j} \exp\left\{-\sum_{i=0}^{n} \theta_j g_j(x_{t_i})\left[t_{i+1} - t_i\right]\right\} \\ &= \prod_{j=1}^{r} \pi_j(x|\theta_j).\end{aligned}$$

A conjugate analysis is now possible by taking independent Gamma priors for each rate constant. That is, $\theta_j \sim \text{Gamma}(a_j, b_j)$, $j = 1, 2, \ldots, r$. We obtain the posterior as

$$\theta_j|x \sim \text{Gamma}\left(a_j + r_j\,,\; b_j + \sum_{i=0}^{n} g_j(x_{t_i})\left[t_{i+1} - t_i\right]\right).$$

3.1.1 Example: Immigration-Death

Figure 3.1 shows a sample path for an immigration-death process with $\theta = (0.5, 0.1)^T$, initial population size of 30, and max time of 30. The marginal posteriors for each parameter, assuming observations on all reaction times and types, are also given. The prior distributions used for the parameters are

$$\theta_j \sim \text{Gamma}(0.2, 0.2), \qquad \text{for both } j = 1, 2.$$

The posteriors are highly peaked about the true parameter values (0.5 and 0.1), showing that using complete data allowed accurate inference about the rate constants to be made.

3.2 Using data at discrete times

Consider a time interval $[0, T]$ over which a Markov jump process $X = \{X_t \,|\, 0 \le t \le T\}$ is not observed directly, but observations (on a regular grid) $y = \{y_t \,|\, t = 0, 1, \ldots, T\}$ are

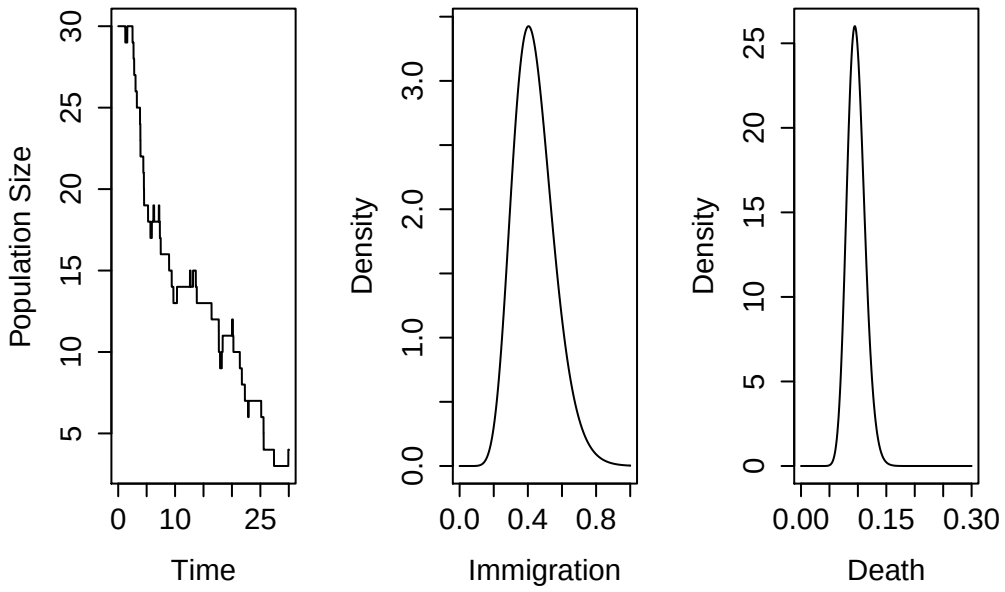

Figure 3.1: Marginal parameter posteriors assuming observations on all reaction times and types. The left hand plot shows the data used to construct the posteriors.

available and assumed conditionally independent (given X) with conditional probability distribution obtained via the observation equation,

$$Y_t = X_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathrm{N}\left(0, \Sigma\right), \qquad t = 0, 1, \ldots, T.$$

Bayesian inference may then proceed through the joint posterior density

$$\begin{aligned} \pi(\theta, x|y) &\propto \pi(\theta)\, \pi(x|\theta)\, f(y|x, \theta) \\ &\propto \pi(\theta)\, \pi(x_0)\, \pi(x|x_0, \theta) \prod_{t=0}^{T} f(y_t|x_t, \theta) \\ &\propto \text{prior} \times \text{likelihood} \end{aligned} \tag{3.1}$$

where $\pi(x|x_0, \theta)$ is the probability associated with the MJP (conditional on x_0) and can be sampled from by executing Gillespie's direct method. Since the posterior in (3.1) will be intractable in practice, samples are usually generated from (3.1) via a Monte Carlo scheme. We now review such a scheme.

3.2.1 Weighted resampling

There are a number of tools available for generating draws from distributions whose densities are only known up to proportionality e.g. Markov chain Monte Carlo (MCMC), rejection sampling and weighted resampling. We focus on weighted resampling, also known as *importance resampling*, as it can be used iteratively to give a simple sequential Monte Carlo scheme. We provide the algorithm here before considering a simple example.

Consider some target density $f(\theta)$. At the first step of the algorithm, N points or particles $\{\theta^{(1)}, \theta^{(2)},$
$\ldots, \theta^{(N)}\}$ are sampled from some *proposal* density $g(\cdot)$. For each $\theta^{(i)}$, a (normalised) weight $w^{(i)}$ is constructed as

$$w^{(i)} = \frac{f(\theta^{(i)})/g(\theta^{(i)})}{\sum_{j=1}^{N} f(\theta^{(j)})/g(\theta^{(j)})}\,, \; i = 1, 2, \ldots, N\,.$$

Finally, a second sample of size M is drawn from the discrete distribution on $\{\theta^{(1)}, \theta^{(2)}, \ldots,$ $\theta^{(N)}\}$ with probabilities $w^{(1)}, w^{(2)}, \ldots, w^{(N)}$. The resulting sample $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}\}$ has approximate distribution $f(\cdot)$. Note that $f(\cdot)$ need only be known up to a normalising constant. For example, if only $f^*(\theta) = kf(\theta)$ is available without knowledge of the constant k, each $w^{(i)}$ remains unchanged. The complete weighted resampling algorithm can therefore be written as follows.

1. Initialise: generate a sample of size N, $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$, from the prior $\pi(\cdot)$.

2. Construct the normalised weights

$$w^{(i)} = \frac{f(\theta^{(i)})/g(\theta^{(i)})}{\sum_{j=1}^{N} f(\theta^{(j)})/g(\theta^{(j)})}\,, \; i = 1, 2, \ldots, N\,.$$

3. Resample M times amongst the $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$ using the weights as probabilities. Denote the resulting sample by $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}\}$.

4. Take $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}\}$ as an approximate sample from $f(\cdot)$.

To justify the method we refer the reader to [8]. Note the the CDF under the approximation tends to the target CDF as $N \to \infty$. For a posterior density $\pi(\theta|x)$, provided that the prior $\pi(\theta)$ is straightforward to sample from, and the likelihood $f_{X|\Theta}(x|\theta)$ is straightforward to evaluate, a weighted resampling scheme can be implemented by using the prior as a proposal density. Naturally, if the prior is relatively uninformative and the likelihood is particularly peaked, this scheme will be very inefficient.

Example: Binomial model

In this example, n data points, x, are obtained from independent binomial distributions i.e. $X_i \sim \text{Bin}(k, \theta), i = 1, 2, \ldots, n$, with k known and θ unknown. Combined with a Beta distribution for θ *a priori*, it is possible to obtain a tractable posterior distribution for θ, as shown below.

$$\begin{aligned} f(x|\theta) = L(\theta|x) &\propto \theta^{n\bar{x}}(1-\theta)^{n(k-\bar{x})} \\ \pi(\theta) &\propto \theta^{a-1}(1-\theta)^{b-1}, \qquad 0 < \theta < 1 \\ \Rightarrow \pi(\theta|x) &\propto \theta^{a+n\bar{x}-1}(1-\theta)^{b+n(k-\bar{x})-1} \end{aligned}$$

$$\Rightarrow \theta|x \sim \text{Beta}\left(a + n\bar{x}, b + n(k - \bar{x})\right)$$
$$\theta|x \sim \text{Beta}\left(A_n, B_n\right)$$

Now, after observing a new data point x_{n+1}

$$\begin{aligned}\pi(\theta|x, x_{n+1}) &\propto \pi(\theta) f(x, x_{n+1}|\theta) \\ &\propto \pi(\theta) f(x|\theta) f(x_{n+1}|\theta) \\ &\propto \pi(\theta|x) f(x_{n+1}|\theta).\end{aligned}$$

So

$$\begin{aligned}\pi(\theta|x, x_{n+1}) &\propto \theta^{a+n\bar{x}-1}(1-\theta)^{b+n(k-\bar{x})-1} \times \theta^{x_{n+1}}(1-\theta)^{k-x_{n+1}} \\ &\propto \theta^{a+n\bar{x}+x_{n+1}-1}(1-\theta)^{b+n(k-\bar{x})+k-x_{n+1}-1} \\ &\propto \theta^{a+\sum_{i=1}^{n+1} x_i - 1}(1-\theta)^{b+(n+1)k-\sum_{i=1}^{n+1} x_i - 1}\end{aligned}$$

$$\Rightarrow \theta|x, x_{n+1} \sim \text{Beta}\left(a + \sum_{i=1}^{n+1} x_i, b + (n+1)k - \sum_{i=1}^{n+1} x_i\right)$$
$$\theta|x \sim \text{Beta}\left(A_{n+1}, B_{n+1}\right).$$

Therefore

$$A_{n+1} = A_n + x_{n+1}, \qquad B_{n+1} = B_n + k - x_{n+1}.$$

Upon receipt of a new observation, a batch analysis would compute A_{n+1} and B_{n+1} from scratch, requiring the entire dataset to be stored. A sequential analysis would compute A_{n+1} and B_{n+1} using A_n and B_n, only storing the 'current' A_n and B_n. This illustrates the advantages of a sequential approach to inference, namely, a reduction in storage cost and a reduction in computational cost.

In the plots shown in Figure 3.2, we 'pretend' that $\pi(\theta|x)$ is intractable and implement a weighted resampling scheme (that uses the prior as a proposal density) to generate a sample of θ values approximately distributed according to $\pi(\theta|x)$. A dataset of length $n = 10$ is generated with $\theta = 0.25$ and $k = 20$. The prior parameters are set to $a = 1$ and $b = 4$. Note that the histograms are very close to the theoretical densities for large numbers of particles, which is to be expected given that the weighted resampling scheme is exact as $N \to \infty$.

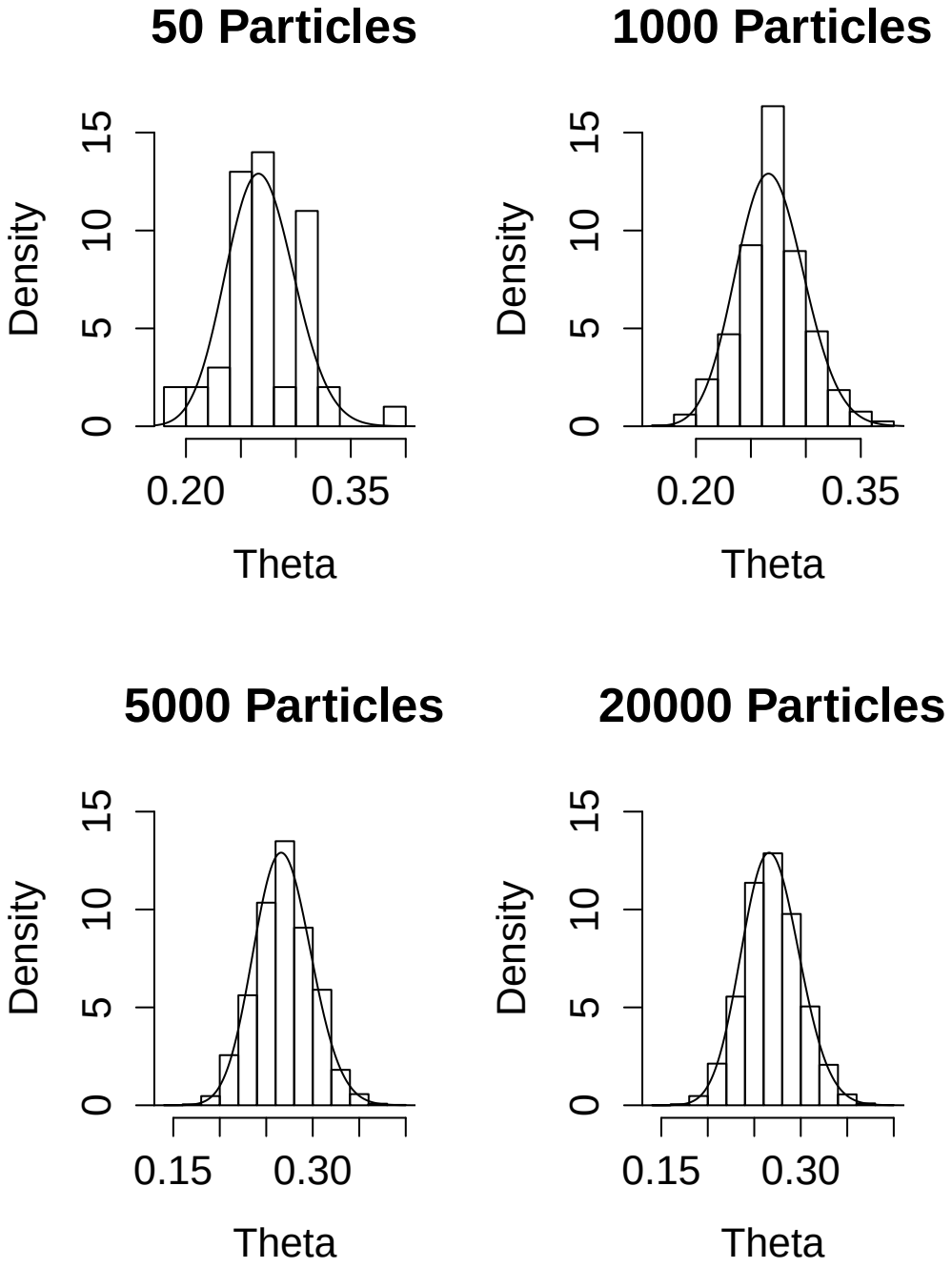

Figure 3.2: Comparison of sample posteriors with theoretical densities for different numbers of particles.

Chapter 4

Sequential Monte Carlo Schemes

Often, observations arrive sequentially in time and one is interested in performing inference on-line. From a Bayesian perspective, it is therefore necessary to update the posterior distribution as data become available. Since the posterior distribution is typically intractable, Monte Carlo schemes are required for generating posterior samples. Sequential Monte Carlo schemes aim to approximate the posterior through a sequence of weighting steps. In this chapter we will consider two such schemes.

4.1 Bootstrap particle filter

Recall that the posterior in (3.1) is only available up to proportionality, and we wish to generate samples using a suitable Monte Carlo method. Suppose that we use weighted resampling and consider the following strategy.

- Observe y_0. Run a weighted resampling scheme targeting $\pi(\theta, x_0|y_0)$.
- Observe y_1. Discard all previous samples from $\pi(\theta, x_0|y_0)$. Run a weighted resampling scheme targeting $\pi(\theta, x_0, x_1|y_0, y_1)$.
- Observe y_2 ... etc.

This is a batch analysis applied sequentially, not a truly sequential analysis! This requires us to store all observations up to and including the current time point t. Moreover, since the computational cost of sampling the posterior is likely to increase as t increases, the computational cost of each step in the above algorithm will also increase with t. However, if we appropriately reweight each sample rather than discard it, we can obatin a more efficient algorithm, the bootstrap particle filter.

Let $x_{0:j} = \{X_t \,|\, 0 < t \leq j\}$ and $y_{0:j} = \{y_t \,|\, t = 0, 1, \ldots, j\}$. Suppose that an equally weighted sample of size N is available from $\pi(\theta, x_{0:j}|y_{0:j})$ and denote this sample by $\{(\theta^{(i)}, x_{0:j}^{(i)}), i = 1, 2, \ldots, N\}$. The next observation y_{j+1} becomes available and we wish to sample the posterior

$$\begin{aligned}\pi(\theta, x_{0:j+1}|y_{0:j+1}) &\propto \pi(\theta)\,\pi(x_0)\,\pi(x_{0:j+1}|x_0, \theta)\prod_{t=0}^{j+1} f(y_t|x_t, \theta) \\ &\propto \pi(\theta, x_{0:j}|y_{0:j})\pi(x_{j:j+1}|x_j, \theta)f(y_{j+1}|x_{j+1}, \theta).\end{aligned}$$

Using the sample from $\pi(\theta, x_{0:j}|y_{0:j})$ as a proposal mechanism, together with the ability to sample from $\pi(x_{j:j+1}|x_j, \theta)$ via Gillespie's direct method gives the following strategy for generating a sample of size N approximately distributed according to $\pi(\theta, x_{0:j+1}|y_{0:j+1})$:

1. For $i = 1, 2, \ldots, N$ draw $x_{j:j+1}^{(i)} \sim \pi(x_{j:j+1}|x_j^{(i)}, \theta^{(i)})$ using Gillespie's direct method.
2. For $i = 1, 2, \ldots, N$ evaluate the weights
$$w^{(i)} \propto f(y_{j+1}|x_{j+1}^{(i)}, \theta^{(i)}).$$
3. Resample N times amongst the set $\{(\theta^{(i)}, x_{0:j+1}^{(i)}), i = 1, 2, \ldots, N\}$ using the weights as probabilities.

Hence, after initialising with a sample from the prior, the above steps are executed for each observation $y_0, y_1, \ldots, y_T$.

4.1.1 Particle degeneracy

Often, only a small fraction of sampled θ values will have significant weight. Consequently, after resampling, the set of sampled values $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$ may only contain a few distinct values. For datasets consisting of many observations, we might then expect the SMC posteriors to collapse to a point mass. This is known as sample impoverishment or *particle degeneracy*.

A number of *ad hoc* approaches have been proposed to overcome degeneracy. Prior boosting (suggested by [10]) is where the prior sample used is larger than the posterior sample size. Another simple method is to *jitter* each particle before calculating the weights (see for examples [2], [9]). If a particular value is replicated in the sample a number of times, it will be replaced by distinct (but similar) values. Hence, prior to calculating the weights, a step is included in the bootstrap particle filter algorithm which sets

$$\theta^{(i)} := \theta^{(i)} + \epsilon^{(i)}, \qquad \epsilon^{(i)} \sim \mathrm{N}\left(0, h^2 \widehat{Var}(\theta|y_{1:j})\right)$$

for $i = 1, 2, \ldots, N$.

Note that h is an arbitrarily chosen smoothing parameter, and that the variance of the resulting sample is $(1 + h^2)Var(\theta|y_{1:j})$. Therefore, large values of h can lead to posterior samples that are overdispersed. Standard rules of thumb can be used to choose h. For example, for a univariate θ with a Gaussian target, the optimal choice *Silverman's rule of thumb*, which, for our problem is

$$h = 1.06 N^{-\frac{1}{5}} \sqrt{\widehat{Var}(\theta|y_{1:j})}.$$

In practice, choosing a slightly larger value of h may be of benefit. Jittering is the approach we will adopt in Chapter 5 to overcome this problem of particle degeneracy.

4.2 Storvik filter

The Storvik filter [3] uses a small number of sufficient statistics to rejuvenate the particle set by using a conjugate gamma prior. Suppose we have observed as far as time j, then we have the set of observations $y_{0:j} = \{y_t : t = 0, 1, \ldots, j\}$ of the latent stochastic process $x_{0:j} = \{x_t : t \in (0, j]\}$. Suppose that we observe y_{j+1} and wish to assimilate the information contained in this observation. Note that associated with the new observation y_{j+1} is the latent path $x_{j:j+1}$. The posterior using all observations (including y_{j+1}) is

$$\pi(\theta, x_{0:j+1}|y_{0:j+1}) \propto \pi(\theta, x_{0:j}|y_{0:j})\pi(x_{j:j+1}|x_j, \theta)f(y_{j+1}|x_{j+1})$$

Now, $\pi(\theta, x_{0:j}|y_{0:j})$ factorises as

$$\pi(\theta|x_{0:j}, y_{0:j})\pi(x_{0:j}|y_{0:j}) = \pi(\theta|x_{0:j})\pi(x_{0:j}|y_{0:j}).$$

For mass action kinetics, and (independent) gamma priors for each θ_i, $\pi(\theta|x_{0:j})$ is tractable. $\pi(x_{0:j}|y_{0:j})$ however is intractable, so a particle approximation is used. Assuming an equally weighted sample $\{x_{0:j}^{(i)}, i = 1, 2, \ldots, N\}$ from $\pi(x_{0:j}|y_{0:j})$, the Storvik filter targets the mixture

$$\hat{\pi}(\theta, x_{0:j+1}|y_{0:j+1}) \propto \sum_{i=1}^{N} \frac{1}{N}\pi(\theta|x_{0:j}^{(i)})\pi(x_{j:j+1}|x_j^{(i)}, \theta)f(y_{j+1}|x_{j+1}).$$

Note that not all particle paths need to be stored. In fact the density $\pi(\theta|x_{0:j})$ can be constructed using a low dimensional set of sufficient statistics. That is $\pi(\theta|x_{0:j}) = \pi(\theta|z_j)$, where $z_j = z(z_0, x_{0:j})$ is a vector of sufficient statistics (e.g. number of reactions of each type up to time j). Also note that z_0 contains the prior hyper parameters. As seen in the conjugate analysis discussed in § 3.1, the sufficient statistics take the form r_k, denoting the number of reaction events of type $\mathcal{R}_k$, and $\sum_{i=0}^{n} g_k(x_{t_i})\,[t_{i+1} - t_i]$, the integrated hazard.

This setup allows us to construct the following algorithm for a Storvik filter. Suppose at time j, we have a sample $\{(z_j^{(i)}, x_j^{(i)}), i = 1, 2, \ldots, N\}$, then, applying weighted resampling:

1. Draw $\theta^{(i)} \sim \pi(\cdot|z_j^{(i)})$ for $i = 1, 2, \ldots, N$.
2. Draw $x_{j:j+1}^{(i)} \sim \pi(\cdot|x_j^{(i)})$ for $i = 1, 2, \ldots, N$ (using Gillespie's direct method).
3. Generate and normalise weights $w_{j+1}^{(i)} \propto f(y_{j+1}|x_{j+1}^{(i)})$ for $i = 1, 2, \ldots, N$.
4. Set $z_{j+1}^{(i)} = z(z_j^{(i)}, x_{j:j+1}^{(i)})$ i.e. update sufficient statistics.
5. Resample amongst the $(z_{j+1}^{(i)}, x_{j+1}^{(i)})$ using weights as probabilities.

Note that Step 1 of the scheme automatically rejuvenates the parameter set. No jittering is required. In the next chapter we will apply both the bootstrap and Storvik filters to the immigration-death model of § 2.3.1.

Chapter 5

Simulation Studies

5.1 Bootstrap filter

We will consider the immigration-death model of § 2.3.1. Data were simulated from the model with $\theta = (0.5, 0.1)^T$, by running Gillespie's direct method over [0, 50]. For simplicity, we assume that x_0 is a fixed and known quantity, in this case setting $x_0 = 30$. We then took the data at discrete times $\{0, 1, \ldots, 50\}$ and corrupted each value via the observation model

$$Y_t|X_t \sim \begin{cases} \text{Poisson}(X_t), & X_t > 0, \\ \text{Bern}(0.25), & X_t = 0. \end{cases}$$

We assumed *a priori* that each $\theta_i \sim \text{Gamma}(a_i, b_i)$ with $a_1 = 0.5, b_1 = 1, a_2 = 0.1, b_2 = 1; i = 1, 2$. We then applied the bootstrap filter of § 4.1. To find the 'best' choice for the smoothing parameter h, the posterior distributions, for both parameters, after running the bootstrap filter with $N = 5000$ particles for different values of h were examined. Plots of these distributions are given in Figures 5.1 and 5.2. Clearly, when the smoothing parameter is too small, the effect of jittering can not overcome the effect of particle degeneracy. Choosing a large value of h over-smooths the resulting posteriors. These plots suggest that an h value of about 0.04 yields the best results, thus that is the value we use in the rest of the study. Note that the posteriors reported in Figures 5.1 and 5.2 show sampled parameter values that are consistent with the true values that produced the data. Using the jittering approach is undesirable, however, $h = 0$ would require many thousands of particles at great computational cost. We therefore focus on a more efficient scheme in the next section.

5.2 Storvik filter

Using the same data and priors as § 5.1, we applied the Storvik filter with $N = 5000$ particles. The output is summarised in Figure 5.3. These posterior distributions are similar to those shown in § 5.2, consistent with the true values that produced the data. In the next section we shall compare the results from both particle filters, for different numbers of particles.

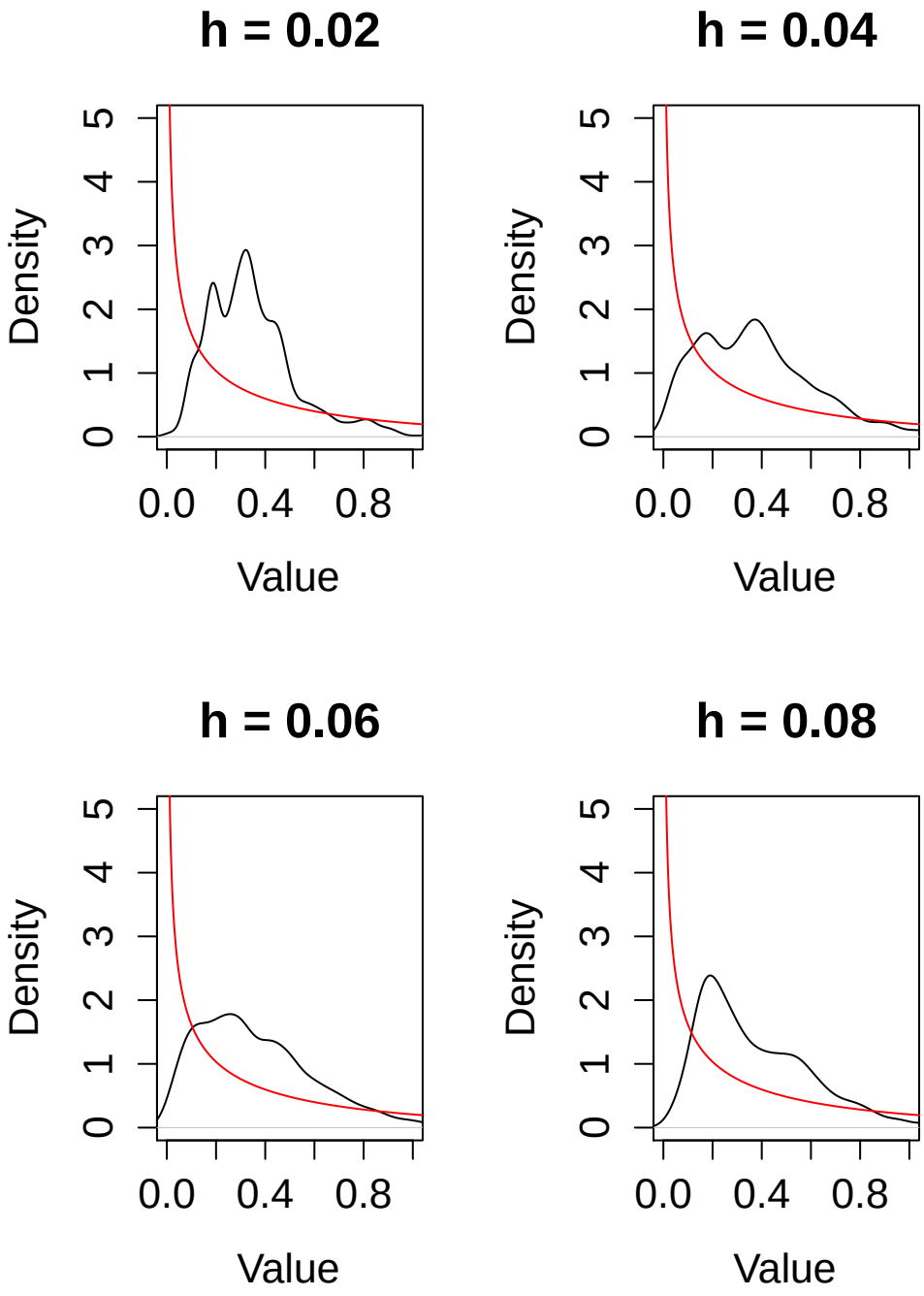

Figure 5.1: Posterior distributions of $\theta_1|x$ from a bootstrap filter with $N = 5000$ particles for different values of h. Prior distribution shown in red.

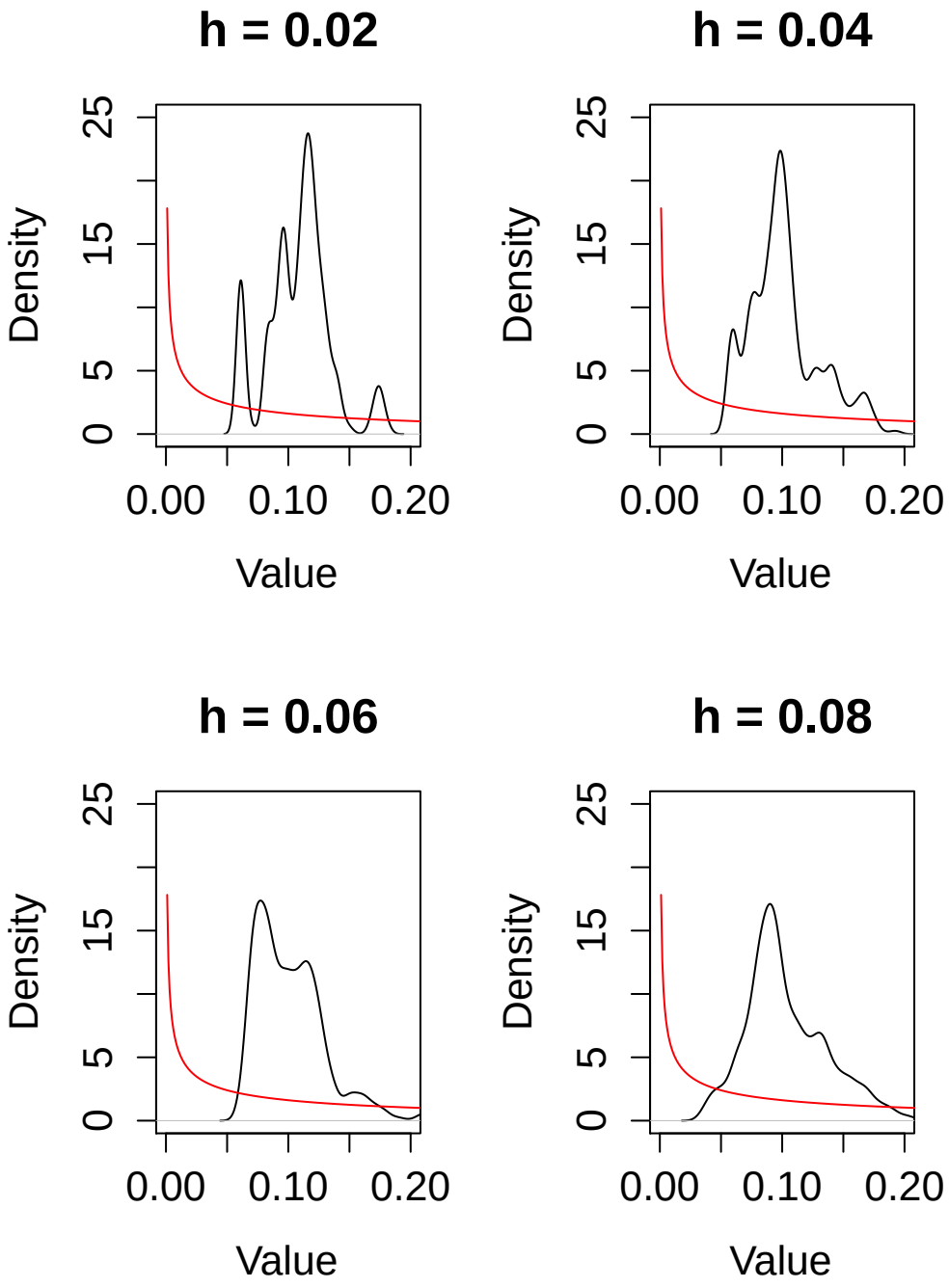

Figure 5.2: Posterior distributions of $\theta_2|x$ from a bootstrap filter with $N = 5000$ particles for different values of h. Prior distribution shown in red.

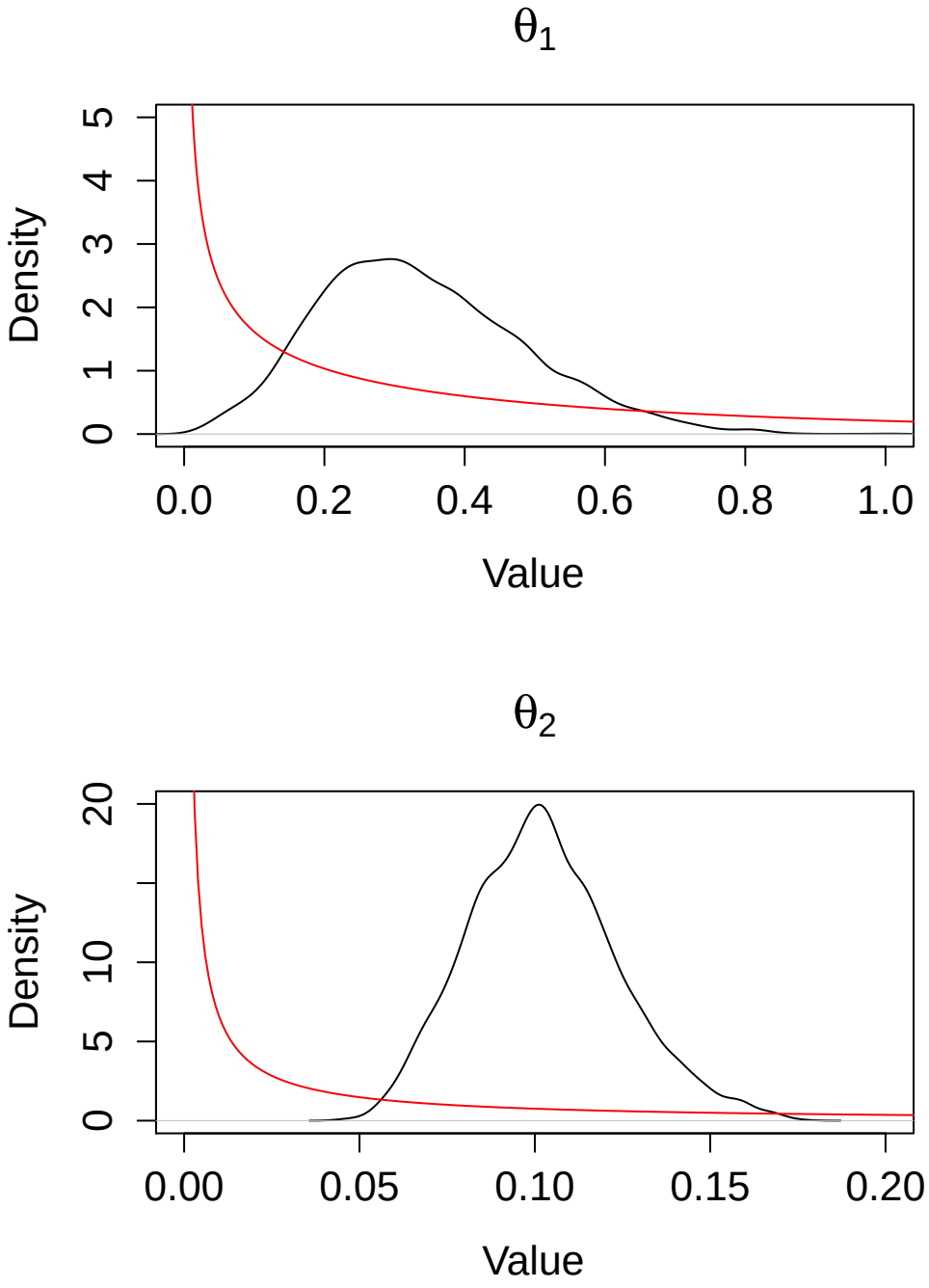

Figure 5.3: Posterior distributions of $\theta|x$ from a Storvik filter with $N = 5000$ particles. Prior distribution shown in red.

5.3 Comparison

The same data is used when applying both the bootstrap and Storvik filter - the data from § 5.1. The output is summarised in Figures 5.4 and 5.5, and in Table 5.1. The plots show that the output is fairly similar, other than perhaps smoother posterior distributions from the Storvik filter. Most of the values shown in Table 5.1 are fairly similar also, however, in most cases the Storvik filter has a smaller standard deviation and a smaller 95% credible interval.

N	Parameter (True Value)	Filter	Post. mean	Post. s.d.	95% cred. int.
5000	θ_1 (0.5)	Bootstrap	0.388	0.252	(0.041, 1.007)
		Storvik	0.339	0.146	(0.095, 0.661)
	θ_2 (0.1)	Bootstrap	0.101	0.028	(0.058, 0.168)
		Storvik	0.102	0.021	(0.065, 0.149)
10000	θ_1 (0.5)	Bootstrap	0.274	0.160	(0.057, 0.695)
		Storvik	0.334	0.173	(0.069, 0.720)
	θ_2 (0.1)	Bootstrap	0.096	0.024	(0.066, 0.155)
		Storvik	0.102	0.024	(0.061, 0.154)
20000	θ_1 (0.5)	Bootstrap	0.366	0.219	(0.064, 0.896)
		Storvik	0.321	0.173	(0.065, 0.728)
	θ_2 (0.1)	Bootstrap	0.107	0.030	(0.058, 0.183)
		Storvik	0.101	0.025	(0.061, 0.157)
50000	θ_1 (0.5)	Bootstrap	0.339	0.215	(0.057, 0.876)
		Storvik	0.302	0.168	(0.060, 0.706)
	θ_2 (0.1)	Bootstrap	0.101	0.027	(0.059, 0.168)
		Storvik	0.098	0.024	(0.060, 0.154)

Table 5.1: Posteriors means, standard deviations, and 95% credible intervals for the sampled parameter values.

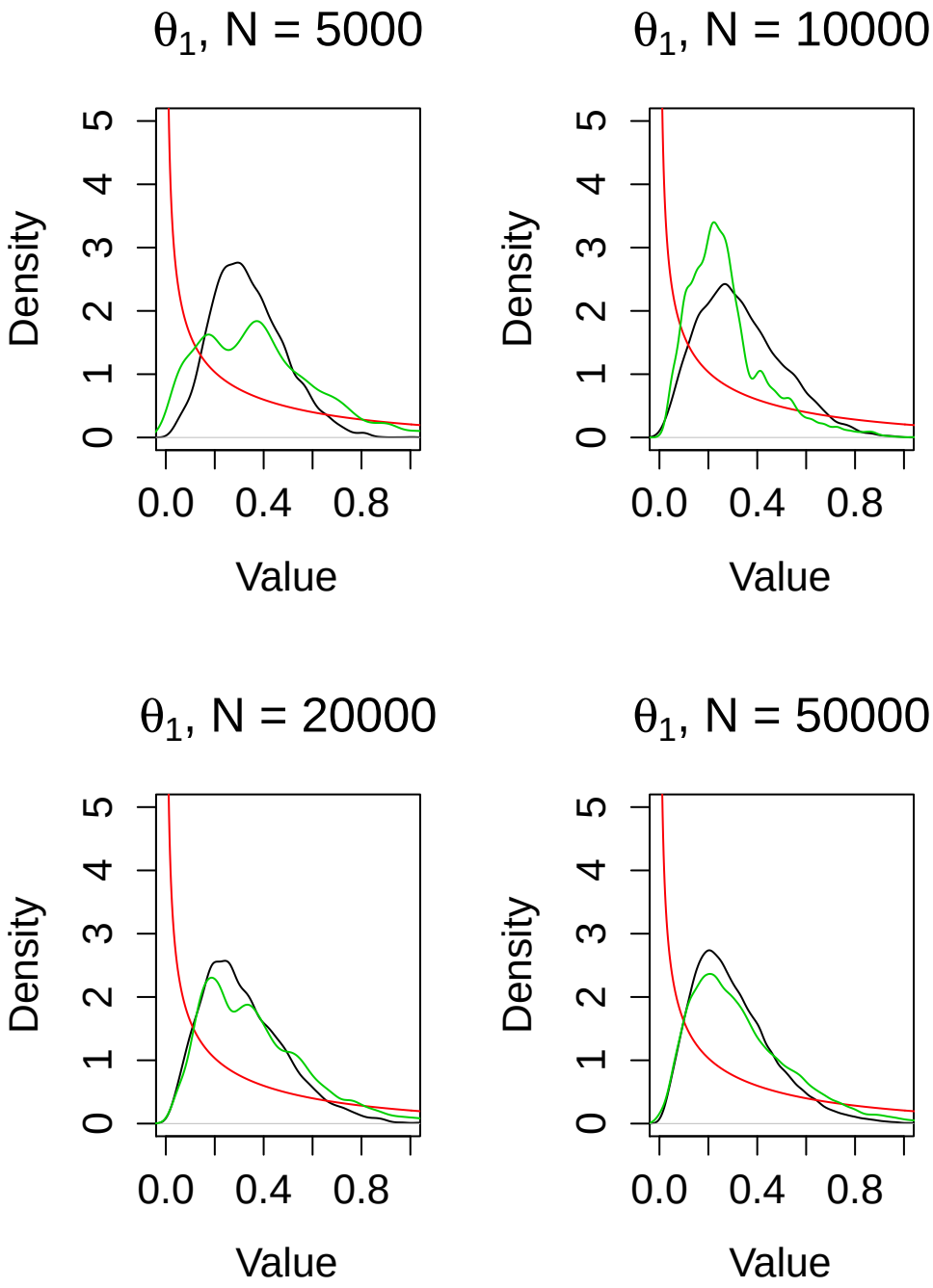

Figure 5.4: Posterior distributions of $\theta_1|x$ for varying N. Posteriors from Storvik filter in black. Posteriors from bootstrap filter in green. Prior distribution shown in red.

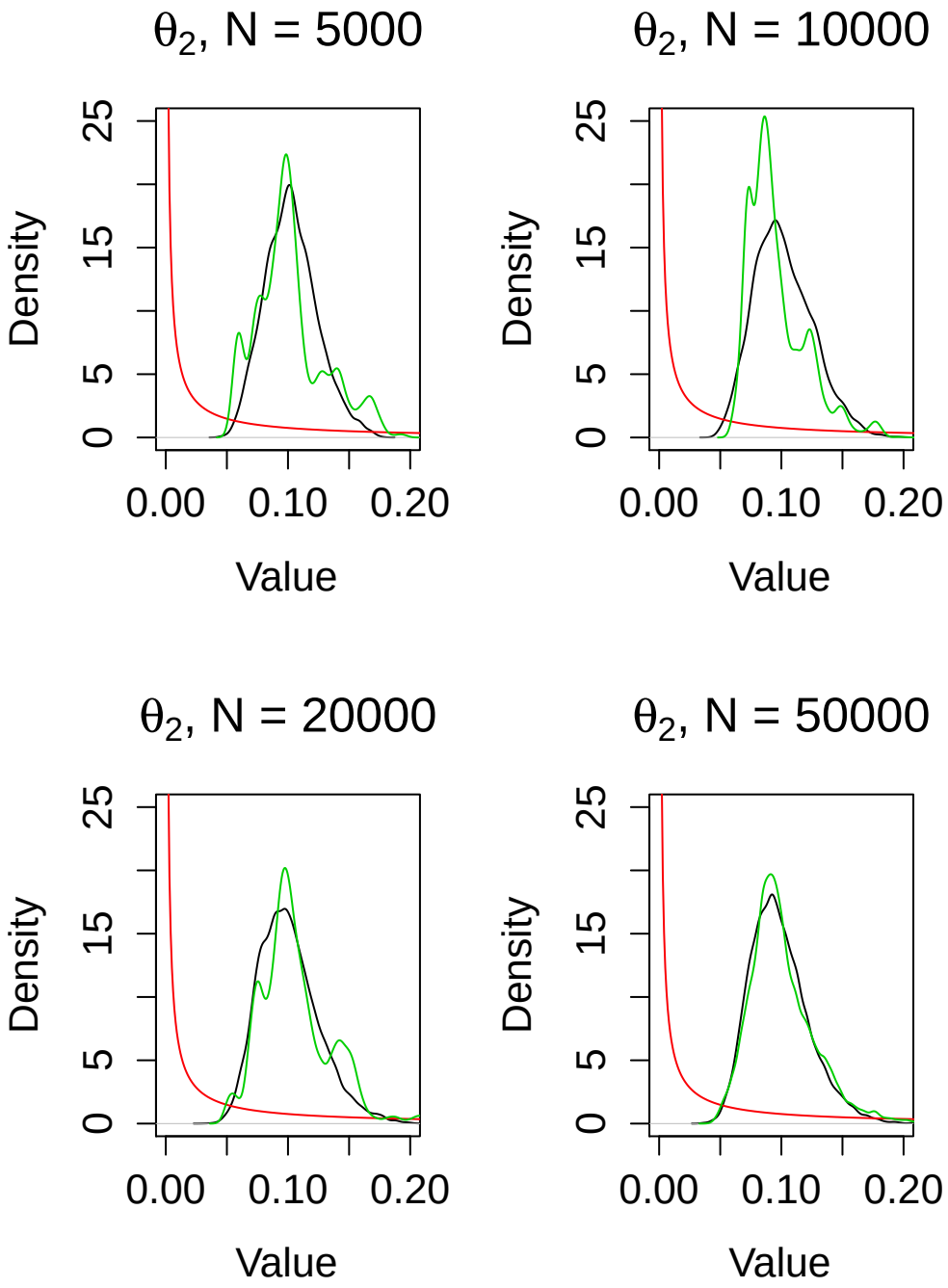

Figure 5.5: Posterior distributions of $\theta_2|x$ for varying N. Posteriors from Storvik filter in black. Posteriors from bootstrap filter in green. Prior distribution shown in red.

Chapter 6

Discussion and Further Work

This report has considered the task of performing fully Bayesian inference for the rate constants governing reaction networks. Species dynamics were modelled using a Markov jump process and Bayesian inference was performed based on observations at discrete times, assumed to be subject to measurement error. A sequential approach was adopted whereby we updated our beliefs about each parameter as each observation became available. The intractability of the parameter posterior necessitated the use of Monte Carlo methods. A weighted resampling scheme was therefore applied iteratively, giving the so called bootstrap particle filter. Unfortunately, sequential Monte Carlo schemes for static parameter problems are known to suffer from sample impoverishment. To alleviate this problem we considered a particle filter proposed by Storvik [3]. The assumption of mass action kinetics leads to a tractable parameter posterior (for a particular choice of prior) when complete information on all reaction times and types is available. Consequently, by updating and storing this information (contained in a set of low dimensional statistics) when running an SMC scheme, the parameter particle set can be rejuvenated by sampling from the tractable parameter posterior conditional on a particular statistic. The methods were used to infer the rate constants governing an immigration-death process.

There are a number of ways in which this work could be extended. The methodology could be applied to other more complicated systems e.g. a Lotka-Volterra system. An auxiliary particle filter can be implemented which, rather than extending the latent path blindly (myopic of the observation), has a proposal mechanism that pushes the path towards the observation (see [11], [12]). Incomplete data scenarios could also be examined where observations are only on one species or a subset of species components.

Appendix A

R code

This function returns the final population size of a simulation of an immigration death model after an amount of time, `maxtime`. `cvec` is a vector containing the rate constants for the reactions. `x` is the initial population size.

```
id = function(maxtime = 10, cvec = c(0.5, 0.1), x = 20){
  r = length(cvec)  #no. reactions
  h = rep(0, r)    #vector of hazards
  t = 0
  while(t < maxtime){
    h[1] = cvec[1]
    h[2] = cvec[2] * x #defines hazards
    h0 = sum(h)

    tau = rexp(1, h0)
    t = t + tau #simulates time to reaction

    #Don't go by maxtime
    if(t >= maxtime){
      break
    }

    u = runif(1, 0, 1)

    if (u < h[1] / h0){
      x = x + 1 #immigration
    }
    else{
      x = x - 1 #death
    }
  }
  return(x)  #return final population size
}
```

The function `lv` lists a matrix that contains the population sizes of both populations of a Lotka-Volterra predator-prey model at all time points examined, using similar arguments

to the `id` function above. It also lists a vector containing the times of the changes in population sizes.

```
lv = function(maxtime = 50, cvec = c(0.5, 0.0025, 0.3), x = c(100, 100))
{
  r = length(cvec)  #no. reactions
  k = length(x)     #no. species
  h = rep(0, r)    #vector of hazards
  t = 0
  xmat = x  #initialise
  tvec = t
  while(t < maxtime)
  {
    h[1] = cvec[1] * x[1]
    h[2] = cvec[2] * x[1] * x[2]
    h[3] = cvec[3] * x[2]
    h0 = sum(h)
    if(h0 == 0)
    {
      #All dead
      t = maxtime
    }
    else
    {
      tau = rexp(1, h0)
      t = t + tau
    }
    #Don't go by maxtime
    if(t >= maxtime)
      break

    u = runif(1, 0, 1)
    if (u < h[1] / h0)
    {
      x[1] = x[1] + 1 #prey reproduction
    }else if (u < (h[1] + h[2]) / h0)
    {
      x[1] = x[1] - 1 #prey death
      x[2] = x[2] + 1 #predator reproduction
    } else {
      x[2] = x[2] - 1 #predator death
    }
    xmat = rbind(xmat, x)
    tvec = c(tvec, t)
  }
  #Append final state and time
  tvec = c(tvec, maxtime)
```

```
  xmat = rbind(xmat, x)
  list(tvec, xmat)  #return these as a list
}
```

`smcid` runs a bootstrap particle filter of an immigration death process, for `N` particles, `x0` initial population size, smoothing parameter `h`, with time length `dt` between observations and other arguments for the prior parameters. It returns a matrix for each parameter where the i^{th} column of that matrix is the set of particles at time point i.

```
smcid = function(N,data,x0 = 30,h = 0.1,dt = 0.5,a1 = 1,b1 = 1,a2 = 1,b2 = 1){
  n = length(data) #no. of data points
  #sample prior, assumed priors
  c1 = rgamma(N, a1, b1); c2 = rgamma(N, a1, b1)
  c1mat = matrix(0, ncol = (n - 1), nrow = N)
  c2mat = matrix(0, ncol = (n - 1), nrow = N)
  #c1mat is a matrix where the ith column is the set of particles at time point i
  #similarly for c2mat
  xvec = rep(x0, N) #store current latent population
  xvecprop = rep(0, N) #vector for storing proposals
  wts = rep(0, N) #empty vector for weights
  #assimilate info in each observation
  for(i in 1:(n - 1)){ #loop over time points
    print(i)
    #sample latent path up to next time for each theta value
    sdc1 = sd(log(c1))
    sdc2 = sd(log(c2))
    for(j in 1:N){ #loop over number of particles
      #jitter
      c1[j] = exp(log(c1[j]) + rnorm(1, 0, h * sdc1))
      c2[j] = exp(log(c2[j]) + rnorm(1, 0, h * sdc2))
      #go forward
      x = id(dt, c(c1[j], c2[j]), xvec[j])
      xvecprop[j] = x
      #calculate weights
      if(x > 0){
        wts[j] = exp(dpois(data[i + 1], x, log = T))
      }
      else{
        wts[j] = exp(dbinom(data[i + 1], 1, 0.25, log = T))
      }
    }
    #resample
    index = sample(seq(1:N), N, TRUE, wts)
    c1 = c1[index]; c2 = c2[index]
    c1mat[ , i] = c1; c2mat[ , i] = c2
    xvec = xvecprop[index]
  }
```

```
  list(c1mat, c2mat)
}
```

The function `sfid` is similar to `smcid` but runs a Storvik filter instead of a bootstrap filter. It does however use a modified version of the `id` function, `idmod`, so that the parameter posteriors can be summarised in terms of the low dimensional statistics the Storvik filter uses.

```
sfid = function(N, data, x0 = 30, dt = 0.5, a = c(0.5, 0.1), b = c(1, 1)){
  n = length(data) #no. of data points
  c1 = rep(0,N); c2 = rep(0,N)
  c1mat = matrix(0, ncol = (n - 1), nrow = N)
  c2mat = matrix(0, ncol = (n - 1), nrow = N)
  #c1mat is a matrix where the ith column is the set of particles at time point i
  #similarly for c2mat
  xvec = rep(x0, N) #store current latent population
  xvecprop = rep(0, N) #vector for storing proposals
  wts = rep(0, N) #empty vector for weights
  suffmat = matrix(rep(c(a[1],a[2],b[1],b[2]),N),ncol = 4,nrow = N,byrow = T)
  #ordered as r[1], r[2], g[1], g[2]
  suffmatprop = matrix(0, ncol = 4, nrow = N)

  #assimilate info in each observation
  for(i in 1:(n - 1)){ #loop over time points
    #extend latent path up to next time for each theta value
    print(i)
    for(j in 1:N){ #loop over number of particles
      #draw parameter value using sufficient statistics
      c1[j] = rgamma(1, suffmat[j, 1], suffmat[j, 3])
      c2[j] = rgamma(1, suffmat[j, 2], suffmat[j, 4])
      #go forward
      x = idmod(dt, c(c1[j], c2[j]), xvec[j])
      xvecprop[j] = x[[1]] #update path
      suffmatprop[j, 1:2] = suffmat[j, 1:2] + x[[2]]
      suffmatprop[j, 3:4] = suffmat[j, 3:4] + x[[3]]
      #adds sufficient statistics

      #calculate weights
      if(x[[1]] > 0){
        wts[j] = exp(dpois(data[i + 1], x[[1]], log = T))
      }
      else{
        wts[j] = exp(dbinom(data[i + 1], 1, 0.25, log = T))
      }
    }
    #resample
    index = sample(seq(1:N), N, TRUE, wts)
```

```
    c1 = c1[index]; c2 = c2[index]
    c1mat[ , i] = c1; c2mat[ , i] = c2
    xvec = xvecprop[index]
    #resample amongst suff. statistics -- pick out appropriate rows of suffmatprop, s
    suffmat = suffmatprop[index, ]
  }
  list(c1mat, c2mat)
}
```

Bibliography

[1] Gillespie, D. T. (1977), *Exact stochastic simulation of coupled chemical reactions*, Journal of physical chemistry **81**, 2340 - 2361.

[2] Gordon, N. J., Salmond, D. J. & Smith, A. F. M. (1993), *Novel approach to nonlinear/non-Gaussian Bayesian state estimation*, IEE Procedings F **140**, 107 - 113.

[3] Storvik, G. (2002), *Particle Filters for State-Space Models With the Presence of Unknown Static Parameters*, IEEE Transactions on **50**, 281 - 289.

[4] Fearnhead, P. (2001), *Perfect simulation of population genetic models with selection*, Genetics **174**, 1397 - 1406.

[5] Lotka, A.J. (1925), *Elements of physical Biology*, Williams and Wilkens, Baltimore.

[6] Volterra, V. (1926), *Fluctuations in the abundance of a species considered mathematically*, Nature **118**, 558 - 560.

[7] Wilkinson, D. J. (2011), *Stochastic Modelling for Systems Biology*, CRC Press, Boca Raton.

[8] Gelfand, A. E. & Smith, A. F. M. (1990), *Sampling-Based Approaches to Calculating Marginal Densities*, Journal of the American Statistical Association **85**, 398 - 409.

[9] Liu, J. & West, M. (2001), *Combined parameter and state estimation in simulation-based filtering*, in A. Doucet, N de Freitas & N. Gordon eds, Sequential Monte Carlo Methods in Practice, 197 - 233, Springer-Verlag, New York.

[10] Gordon, N., Salmond, D. & Ewing, C. (1995), *Bayesian state estimation for tracking and guidance using the bootstrap filter*, Journal of Guidance, Control, and Dynamics **18**, 1434 - 1443.

[11] Pitt, M. K. & Shephard, N. (1999), *Filtering via Simulation: Auxiliary Particle Filters*, Journal of the American Statistical Association **94**, 590 - 599.

[12] Golightly, A. & Wilkinson, D. J. (2015), *Bayesian Inference for Markov Jump Processes with Informative Observations*, to appear in SAGAMB.