Particle Markov Chain Monte Carlo

Bjarke Hautop Kristensen

Student ID: 202005674

Master's Thesis in Statistics

Supervisor: Jan Pedersen

June 5, 2025

Abstract

Particle Markov Chain Monte Carlo (PMCMC) methods combine sequential Monte Carlo (SMC) with Markov chain Monte Carlo (MCMC), enabling fully Bayesian inference in complex state-space models. This thesis begins by reviewing importance sampling, sequential importance sampling, and resampling strategies, which form the core building blocks of a particle filter.

We then develop the theory behind the Particle Marginal Metropolis-Hastings (PMMH) sampler, which targets the joint posterior of both model parameters and latent states. A key insight is that PMMH uses a particle filter to provide an unbiased estimate of the likelihood, ensuring that the correct posterior is preserved despite the intractability of the true likelihood.

To support practical application, we introduce `bayesSSM`, an `R` package offering streamlined implementations of PMMH and supporting routines. The package automates proposal tuning, particle-number selection, and diagnostic reporting, lowering the barrier to entry for applied users.

Finally, we apply these methods to the stochastic modeling of infectious disease dynamics. After deriving stochastic SIR models, we fit a Bayesian negative-binomial observation model to historical influenza outbreak data from a boarding school. This analysis yields probabilistic reconstructions of the epidemic curve and credible intervals for observation noise, highlighting the advantages of stochastic over deterministic modeling in small populations.

Overall, this work integrates literature from multiple fields and illustrates both the theoretical foundations and practical implementation of PMCMC for state-space modeling.

Contents

Glossary

DAG directed acyclic graph. 16

DGP data generating process. 19, 29, 43

FFBSm Forward-Filtering Backward-Sampling with marginalization. 21

IS importance sampling. 1, 3, 4, 6, 8, 10, 11

LLN law of large numbers. 3, 4

MC Monte Carlo. 1, 11, 13, 14, 17, 18, 22, 32, 37

MCMC Markov chain Monte Carlo. 1, 25, 27–29, 32, 34, 55, 56

MCMC ESS MCMC Effective Sample Size. 29, 40, 45, 57

MMH Marginal Metropolis–Hastings. 25

particle ESS particle Effective Sample Size. 13, 14

PMCMC Particle Markov chain Monte Carlo. 1, 2, 32, 50

PMMH Particle Marginal Metropolis-Hastings. 1, 25, 26, 28, 29, 33, 35, 40, 45, 50

SIR Susceptible-Infectious-Recovered. 37, 40, 50

SIS sequential importance sampling. 1, 6, 18, 22

SISAR sequential importance sampling with adaptive resampling. 13, 14, 18, 21, 22, 29

SISR sequential importance sampling with resampling. 9, 11, 13, 14, 18, 22

SMC sequential Monte Carlo. 1, 11

SSM state-space model. 1, 6, 16, 18, 23–25, 28, 30, 31, 34, 38, 50

1 Introduction

Modern statistical inference often involves formulating probabilistic statements about latent processes governed by complex, dynamic systems. In many applied domains, including epidemiology, finance, and ecology, these systems are naturally modeled as state-space models (SSMs), where observed data arise from unobserved, underlying Markov processes. Estimating latent states and parameters in such models remains computationally challenging, especially when the system exhibits non-linearity and non-Gaussianity.

A foundational advance enabling inference in these models was the development of sequential Monte Carlo (SMC) methods, also known as particle filters, which approximate filtering and smoothing distributions by propagating weighted samples over time. However, while SMC methods efficiently estimate states, they do not fully address Bayesian inference when model parameters are unknown.

This limitation was addressed by Particle Markov chain Monte Carlo (PMCMC) methods, introduced by Andrieu et al. [2010], which combine SMC and Markov chain Monte Carlo (MCMC) techniques. PMCMC provides a structured framework for Bayesian inference in SSMs, even when the likelihood function is intractable. A key feature of PMCMC is the use of particle filters to provide unbiased estimates of the likelihood, enabling exact Bayesian inference despite relying on approximate likelihood estimates.

The first part of this thesis provides a theoretical foundation for PMCMC, beginning with a review of Monte Carlo (MC) methods, including importance sampling (IS) [Robert and Casella, 2004] and sequential importance sampling (SIS) [Doucet et al., 2000]. Challenges such as particle degeneracy are discussed, and resampling techniques are introduced to mitigate these issues. We then transition to the formal development of PMCMC algorithm, highlighting its theoretical guarantees and practical implementation.

To bridge the gap between theory and practice, the second part of the thesis introduces `bayesSSM` [Hautop, 2025], an `R` package designed to perform Bayesian inference in SSMs using Particle Marginal Metropolis-Hastings (PMMH). It includes proposal tuning, adaptive selection of the number of particles, and diagnostic evaluation, thereby facilitating practical use.

Finally, we apply PMCMC methods to the stochastic modeling of infectious disease outbreaks. Using historical influenza data from a British boarding school [Communicable Disease Surveillance Centre and Communicable Diseases (Scotland) Unit, 1978], we demonstrate how stochastic SIR models fitted via PMCMC can be used for Bayesian inference. This case study illustrates the practical relevance of PMCMC in small-population epidemic modeling, where stochastic effects play a crucial role.

Overall, this thesis aims to provide a comprehensive treatment of PMCMC methods, balancing theoretical rigor with practical utility, and contributing an open-source `R` package. All code used throughout this thesis is available at: `https://github.com/BjarkeHautop/master-thesis`.

2 Monte Carlo Methods

In this chapter, we provide the theoretical foundation for Particle Markov chain Monte Carlo (PMCMC). We follow the work of Andrieu et al. [2010], Doucet and Johansen [2009], and Kroese et al. [2013].

Let μ be a σ-finite reference measure on the space $\mathcal{X}$ where each x_i takes values, and denote by $\mu^{\otimes n}$ the corresponding product measure on $\mathcal{X}^n$. Let $x_{1:n} = (x_1, x_2, \ldots, x_n)$ and suppose we have a density $\pi_n(x_{1:n})$ (with respect to $\mu^{\otimes n}$) given by

$$\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}, \tag{2.1}$$

where $\gamma_n(x_{1:n})$ is the unnormalized density, and the normalizing constant is given as

$$Z_n = \int_{\mathcal{X}^n} \gamma_n(x_{1:n})\, d\mu^{\otimes n}(x_{1:n}).$$

For notational simplicity for the remainder of this chapter let $\mathcal{X} \subseteq \mathbb{R}^d$ and μ be the Lebesgue measure, in which case Z_n is given by

$$Z_n = \int \gamma_n(x_{1:n})\, dx_{1:n}. \tag{2.2}$$

Let $X_{1:n} \sim \pi_n$, and draw N i.i.d. samples

$$X_{1:n}^{(1)},\, X_{1:n}^{(2)},\, \ldots,\, X_{1:n}^{(N)},$$

and denote their realizations by

$$x_{1:n}^{(1)},\, x_{1:n}^{(2)},\, \ldots,\, x_{1:n}^{(N)}.$$

We can then approximate $\pi_n(x_{1:n})$ by the empirical measure

$$\pi_n^{\mathrm{MC}}(x_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{1:n}^{(i)}}(x_{1:n}),$$

where $\delta_{x_{1:n}^{(i)}}$ denotes the Dirac measure centered at $x_{1:n}^{(i)}$. We can also approximate any marginal $\pi_n(x_k)$ as

$$\pi_n^{\mathrm{MC}}(x_k) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_k^{(i)}}(x_k).$$

The expectation of any function $H_n : \mathcal{X}^n \to \mathbb{R}$ is given by

$$I_n(H_n) \coloneqq \mathbb{E}[H_n(X_{1:n})] = \int H_n(x_{1:n})\, \pi_n(x_{1:n})\, dx_{1:n},$$

and by the law of large numbers (LLN), we can estimate it by

$$I_n^{\mathrm{MC}}(H_n) \coloneqq \int H_n(x_{1:n})\, \pi_n^{\mathrm{MC}}(x_{1:n})\, dx_{1:n} = \frac{1}{N}\sum_{i=1}^{N} H_n(x_{1:n}^{(i)}).$$

However, this requires that we can sample from π_n, which often is not the case when it is a complex high-dimensional distribution. A way to solve this issue is to use importance sampling (IS).

2.1 Importance Sampling

Here we introduce an importance density $q_n(x_{1:n})$ which we can sample from and such that

$$\pi_n(x_{1:n}) > 0 \implies q_n(x_{1:n}) > 0.$$

For the remainder of this chapter, we let

$$X_{1:n} \sim q_n.$$

Suppose we draw i.i.d. samples $X_{1:n}^{(1)}, \ldots, X_{1:n}^{(N)} \sim q_n$, with corresponding realizations $x_{1:n}^{(1)}, \ldots, x_{1:n}^{(N)}$. We then define the *unnormalized importance weight* as

$$w_n(x_{1:n}) \coloneqq \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})}.$$

The normalized importance weight for the ith sample is then

$$W_n^{(i)} \coloneqq \frac{w_n(x_{1:n}^{(i)})}{\sum_{j=1}^{N} w_n(x_{1:n}^{(j)})}.$$

From Equation (2.1) and (2.2), we have

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n}) q_n(x_{1:n})}{Z_n}, \tag{2.3}$$

and

$$Z_n = \int w_n(x_{1:n})\, q_n(x_{1:n})\, dx_{1:n}. \tag{2.4}$$

We then define the IS estimators of $\pi_n(x_{1:n})$ and Z_n as

$$\widehat{\pi}_n(x_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{x_{1:n}^{(i)}}(x_{1:n}), \tag{2.5}$$

$$\widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(x_{1:n}^{(i)}). \tag{2.6}$$

Theorem 2.1 (Relative variance of $\mathrm{Var}(\widehat{Z}_n)$)**.** The relative variance of the IS estimate of the normalizing constant Z_n is given by

$$\frac{\mathrm{Var}(\widehat{Z}_n)}{Z_n^2} = \frac{1}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\, dx_{1:n} - 1\Big).$$

Proof. The variance of $\widehat{Z}_n$ is

$$\begin{aligned}
\operatorname{Var}(\widehat{Z}_n) &= \frac{1}{N}\operatorname{Var}\big(w_n(X_{1:n})\big) \\
&= \frac{1}{N}\Big(\mathbb{E}[w_n^2(X_{1:n})] - \mathbb{E}[w_n(X_{1:n})]^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{\gamma_n^2(x_{1:n})}{q_n^2(x_{1:n})} q_n(x_{1:n})\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{\gamma_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{Z_n^2\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{Z_n^2}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - 1\Big).
\end{aligned}$$

Dividing by Z_n^2 gives the result. □

Furthermore, we can also estimate $I_n(H_n)$ by

$$I_n^{\mathrm{IS}}(H_n) \coloneqq \int H_n(x_{1:n})\widehat{\pi}(x_{1:n})\,dx_{1:n} = \sum_{i=1}^N W_n^{(i)} H_n(X_{1:n}^{(i)}) = \frac{\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})H_n(X_{1:n}^{(i)})}{\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})}.$$

Note, that the numerator is an unbiased estimate of $Z_n I_n(H_n)$, since

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})H_n(X_{1:n}^{(i)})\right] &= \mathbb{E}\left[w_n(X_{1:n})H_n(X_{1:n})\right] \\
&= \int w_n(x_{1:n})H_n(x_{1:n})q(x_{1:n})\,dx_{1:n} \\
&= \int H_n(x_{1:n})\gamma_n(x_{1:n})\,dx_{1:n} \\
&= Z_n I_n(H_n).
\end{aligned}$$

Similar calculations show that the denominator is an unbiased estimate of Z_n. Thus, we have a ratio of unbiased estimates, which is not unbiased. However, it is still consistent, which follows by using the LLN and properties of a.s. convergence.

A natural choice for an importance density $q_n(x_{1:n})$ is one that minimizes the variance of $\widehat{Z}_n$. As shown in Theorem 2.2, this minimum variance is achieved when

$$q_n(x_{1:n}) = \pi_n(x_{1:n}).$$

However, this choice is not feasible in practice, as the entire motivation for using IS is that direct sampling from π_n is not possible. Nonetheless, this result indicates that the importance density should closely resemble the target density.

We could now sample from $\pi_n(x_{1:n})$ using the above method. However, to generate a sequence of samples for each n, each step would grow linearly in n, as generating samples from $\pi_{n+1}(x_{1:n+1})$ depends on the previous samples up to time n. This makes such an approach unfeasible in practice.

Theorem 2.2 (Minimum Variance of IS)**.** The variance $\mathrm{Var}[\widehat{Z}_n]$ is minimized if

$$q_n(x_{1:n}) = \pi_n(x_{1:n}),$$

This can be proven by using the Lagrange multiplier and is inspired by Shimao [2018].

Proof. By independence we have,

$$\mathrm{Var}[\widehat{Z}_n] = \frac{1}{N} \mathrm{Var}\left[w_n(X_{1:n})\right].$$

We then have

$$\mathrm{Var}[w_n(X_{1:n})] = \mathbb{E}\left[\frac{\gamma_n(X_{1:n})^2}{q_n(X_{1:n})^2}\right] - Z_n^2 = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\, dx_{1:n} - Z_n^2.$$

Minimizing $\mathrm{Var}[\widehat{Z}_n]$ is thus equivalent to minimizing

$$J(q_n) = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\, dx_{1:n},$$

subject to the constraint

$$\int q_n(x_{1:n})\, dx_{1:n} = 1.$$

We now introduce a Lagrange multiplier λ and form the Lagrangian

$$L(q_n, \lambda) = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\, dx_{1:n} + \lambda \left(\int q_n(x_{1:n})\, dx_{1:n} - 1 \right).$$

Taking the functional derivative with respect to $q_n(x_{1:n})$ and using chain rule we get

$$\frac{\delta L}{\delta q_n(x_{1:n})} = -\frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})^2} + \lambda = 0.$$

Solving for $q_n(x_{1:n})$ yields

$$q_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{\sqrt{\lambda}}.$$

Enforcing the normalization condition we get

$$\int \frac{\gamma_n(x_{1:n})}{\sqrt{\lambda}}\, dx_{1:n} = 1 \quad \Longrightarrow \quad \frac{Z_n}{\sqrt{\lambda}} = 1,$$

so that $\sqrt{\lambda} = Z_n$. Therefore,

$$q_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n} = \pi_n(x_{1:n}).$$

□

2.2 Sequential Importance Sampling

Sequential importance sampling (SIS) builds upon the basic idea of IS by exploiting a Markov structure in the importance distribution. Instead of sampling the full trajectory at once, SIS extends particle trajectories sequentially, updating the importance weights recursively. This approach is particularly useful when the target distribution itself evolves sequentially, as is common in time series or state-space models (SSMs). By aligning the structure of the proposal and target distributions, SIS. When applying these methods in future chapters, we will refer to this class of algorithms as *particle filters*.

We let our importance distribution have a Markov structure. Specifically, the importance distribution up to time step n is given by

$$q_n(x_{1:n}) = q_1(x_1)q_2(x_2 \mid x_1) \dots q_n(x_n \mid x_{1:n-1}),$$

and each element in the set $\{x_{1:n}\}$ we will refer to as a *particle*. For $n \geq 2$ we define the *incremental importance weight* as

$$\alpha_n(x_{1:n}) \coloneqq \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n \mid x_{1:n-1})}.$$

The unnormalized weights can then be written recursively as

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})} \\ &= \frac{\gamma_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n \mid x_{1:n-1})} \\ &= w_{n-1}(x_{1:n-1}) \cdot \alpha_n(x_{1:n}) \\ &= w_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}). \end{aligned} \tag{2.7}$$

The SIS algorithm is described in Algorithm 2.1. Assuming that sampling from the proposal distribution and evaluating the target density take constant time $O(1)$, the time complexity of the algorithm can be analyzed as follows. The initialization phase samples N particles and computes initial weights, which costs $O(N)$. The main body consists of two nested loops: an outer loop over time steps $n = 2, \dots, T$ and an inner loop over particles $i = 1, \dots, N$. For each particle at each time step, the algorithm samples from the proposal distribution, computes an incremental weight, updates the particle weight, and normalizes the weights. Each of these operations is $O(1)$ per particle, so each time step costs $O(N)$, and the total cost over all time steps is $O(NT)$. Thus, the total time complexity of the algorithm is $O(N) + O(NT) = O(NT)$.

However, this method has a severe drawback: the relative estimated variance $\mathrm{Var}(\widehat{Z}_n)/Z_n^2$ increases exponentially in n even in simple examples. Example 2.3 illustrates this issue.

Example 2.3. Consider the case where $\mathcal{X} = \mathbb{R}$ and let the density $\pi_n(x_{1:n})$ be given by

$$\pi_n(x_{1:n}) = \prod_{k=1}^{n} \pi_n(x_k) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x_k^2}{2}\Big).$$

Thus, the unnormalized density $\gamma_n(x_{1:n})$ is given by

$$\gamma_n = \prod_{k=1}^{n} \exp\Big(-\frac{x_k^2}{2}\Big),$$

Algorithm 2.1 Sequential Importance Sampling (SIS)

Input: Number of particles N, proposal distributions $q_n(x_n \mid x_{1:n-1})$, unnormalized target densities $\gamma_n(x_{1:n})$
for each particle $i = 1, \ldots, N$ **do**
    Generate initial particles $x_1^{(i)} \sim q_1(x_1)$
    Compute initial weights: $w_1^{(i)} \leftarrow \frac{\gamma_1(x_1^{(i)})}{q_1(x_1^{(i)})}$
    Normalize: $W_1^{(i)} \leftarrow \frac{w_1^{(i)}}{\sum_{j=1}^N w_1^{(j)}}$
end for
for each time step $n = 2, \ldots, T$ **do**
    for each particle $i = 1, \ldots, N$ **do**
        Generate particles $x_n^{(i)} \sim q_n(x_n \mid x_{1:n-1}^{(i)})$
        Compute the incremental importance weight:

$$\alpha_n^{(i)} \leftarrow \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)}) q_n(x_n^{(i)} \mid x_{1:n-1}^{(i)})}$$

        Update the particle weight: $w_n^{(i)} = w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
    end for
    Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{j=1}^N w_n^{(j)}}$
end for
Output: Particle trajectories and weights:

$$X_{1:T} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_T^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_T^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \ldots & x_T^{(N)} \end{bmatrix}, \quad W_{1:T} = \begin{bmatrix} W_1^{(1)} & W_2^{(1)} & \ldots & W_T^{(1)} \\ W_1^{(2)} & W_2^{(2)} & \ldots & W_T^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ W_1^{(N)} & W_2^{(N)} & \ldots & W_T^{(N)} \end{bmatrix}$$

and the normalizing constant Z_n is

$$Z_n = (2\pi)^{n/2}.$$

Ignoring that we could easily sample from π_n, we select the importance distribution q_n to sample from as

$$q_n(x_{1:n}) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{x_k^2}{2\sigma^2}\Big).$$

Using q_n the goal is to estimate the normalizing constant Z_n. Recall from Theorem 2.1 that the relative variance of $\widehat{Z}_n$ is given by

$$\frac{\mathrm{Var}(\widehat{Z}_n)}{Z_n^2} = \frac{1}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\, dx_{1:n} - 1\Big).$$

In our case, this factors over the coordinates and we have

$$\begin{aligned}\int \frac{\pi_n(x_{1:n})^2}{q_n(x_{1:n})}\,dx_{1:n} &= \prod_{k=1}^{n}\int \frac{1/(2\pi)\exp(-x^2)}{1/\sqrt{2\pi\sigma^2}\exp(-x^2/(2\sigma^2))}\,dx_{1:k} \\ &= \left[\frac{\sqrt{2\pi\sigma^2}}{2\pi}\int \exp\Big(-x^2+\frac{x^2}{2\sigma^2}\Big)\,dx\right]^n \\ &= \left[\frac{\sqrt{2\pi\sigma^2}}{2\pi}\int \exp\Big(-\Big(1-\frac{1}{\sigma^2}\Big)x^2\Big)\,dx\right]^n.\end{aligned}$$

The integral is finite if and only if $1-1/(2\sigma^2)>0$, that is $\sigma^2>1/2$. In this case, it is equal to

$$\int_{-\infty}^{\infty}\exp\Big(-\Big(1-\frac{1}{\sigma^2}\Big)x^2\Big)\,dx = \sqrt{\frac{\pi}{1-1/(2\sigma^2)}}.$$

Thus, for $\sigma^2>1/2$ we have

$$\begin{aligned}\int \frac{\pi_n(x_{1:n})^2}{q_n(x_{1:n})}\,dx_{1:n} &= \left(\frac{\sqrt{2\pi\sigma^2}}{2\pi}\sqrt{\frac{\pi}{1-1/(2\sigma^2)}}\right)^n \\ &= \left(\sqrt{\frac{\sigma^4}{2\sigma^2-1}}\right)^n \\ &= \Big(\frac{\sigma^4}{2\sigma^2-1}\Big)^{n/2}.\end{aligned}$$

Finally, we can conclude that for $\sigma^2>1/2$ the relative variance of $\widehat{Z}_n$ is

$$\frac{\mathrm{Var}[\widehat{Z}_n]}{Z_n^2} = \frac{1}{N}\left[\Big(\frac{\sigma^4}{2\sigma^2-1}\Big)^{n/2}-1\right].$$

Note that for any $1/2<\sigma^2\neq 1$ we have that $\sigma^4/(2\sigma^2-1)>1$, and thus the variance increases exponentially with n.

For example, choosing $\sigma^2=1.2$ then we have a reasonably good importance distribution as $q_n(x_{1:n})\approx\pi_n(x_{1:n})$. However,

$$N\,\mathrm{Var}[\widehat{Z}_n]/Z_n^2 \approx (1.103)^{n/2},$$

which for $n=1000$ is roughly equal to $1.9\cdot 10^{21}$. So we would need to use $N\approx 2\cdot 10^{23}$ particles to obtain a relative variance of 0.01.

2.3 Resampling

Over time, many particles receive negligible weight, leading to a situation where only a few particles dominate the estimate. This phenomenon, known as weight degeneracy, can degrade the performance of the estimator [Doucet et al., 2000]. The idea for resampling is to get rid of particles with low weights with a high probability, so the focus is spent on high-probability regions instead of carrying forward particles with very low weights. This typically reduces the variance, see for instance Example 2.5.

The IS approximation $\widehat{\pi}_n(x_{1:n})$ of the target distribution $\pi_n(x_{1:n})$ is constructed using weighted samples drawn from $q_n(x_{1:n})$. Resampling is then used, where each particle

$X_{1:n}^{(i)}$ is selected with probability proportional to its normalized weight $W_n^{(i)}$. A simple resampling strategy is multinomial resampling, where the number of offspring $N_n^{(i)}$ assigned to each particle $X_{1:n}^{(i)}$ follows a multinomial distribution:

$$N_n^{(1:N)} = (N_n^{(1)}, \ldots, N_n^{(N)}) \sim \text{Multinomial}(N, W_n^{(1:N)}).$$

That is, each particle is independently resampled N times, with probabilities given by the normalized weights $W_n^{(i)}$.

After resampling, all particles are assigned equal weight, since the resampled particles are now an unweighted representation of the target distribution. Each selected particle appears with frequency $N_n^{(i)}$, and thus the empirical distribution assigns equal mass to all N retained particles. That is, the original weighted particle approximation of the target distribution is

$$\hat{p}(x) = \sum_{i=1}^{N} W_n^{(i)} \delta_{X_{1:n}^{(i)}}(x),$$

and after resampling, the new approximation is

$$\hat{p}^*(x) = \sum_{i=1}^{N} \frac{N_n^{(i)}}{N} \delta_{X_{1:n}^{(i)}}(x).$$

Taking expectations we obtain

$$\mathbb{E}\big[\hat{p}^*(x)\big] = \sum_{i=1}^{N} \mathbb{E}\left[\frac{N_n^{(i)}}{N} \delta_{X_{1:n}^{(i)}}(x)\right] = \sum_{i=1}^{N} W_n^{(i)} \delta_{X_{1:n}^{(i)}}(x) = \hat{p}(x),$$

showing that the resampled particle system is unbiased.

Using the recursive structure of the unnormalized weights from Equation (2.7), a natural way to estimate Z_n is to define

$$\widetilde{Z}_1 \coloneqq \frac{1}{N} \sum_{i=1}^{N} w_1(x_1^i),$$

and for $n \geq 2$ estimate Z_n recursively by

$$\widetilde{Z}_n \coloneqq \widetilde{Z}_{n-1} \alpha_n^{\text{MC}}, \tag{2.8}$$

where α_n^{MC} is the standard MC estimate of α_n.

A generic sequential importance sampling with resampling (SISR) algorithm is given in Algorithm 2.2. The time complexity of this algorithm remains $O(NT)$ since the resampling step, which involves drawing N indices according to the particle weights, is done within the T-loop and has a time complexity of $O(N)$.

While resampling effectively eliminates low-weight particles, it also causes many distinct trajectories to vanish over successive iterations. In effect, resampling resets the system by providing a reliable approximation of the current state's marginal distribution, albeit at the cost of losing detailed ancestral information. The deeper problem of weight degeneracy is that trying to represent a high-dimensional distribution with a finite number of samples will inevitably fail. An alternative would be to increase the number of particles at each iteration; however, this approach quickly becomes infeasible due to the exponential growth in the required number of particles [Doucet and Johansen, 2009].

Now we provide a result similar to Theorem 2.1 for the case with resampling.

Algorithm 2.2 Sequential Importance Sampling with Resampling (SISR)

Input: Number of particles N, proposal distributions $q_n(x_n \mid x_{1:n-1})$, unnormalized target densities $\gamma_n(x_{1:n})$
for each particle $i = 1, \ldots, N$ **do**
    Generate initial particles $x_1^{(i)} \sim q_1(x_1)$
    Compute initial weights: $w_1^{(i)} \leftarrow \frac{\gamma_1(x_1^{(i)})}{q_1(x_1^{(i)})}$
    Normalize: $W_1^{(i)} \leftarrow \frac{w_1^{(i)}}{\sum_{j=1}^N w_1^{(j)}}$
end for
for each time step $n = 2, \ldots, T$ **do**
    for each particle $i = 1, \ldots, N$ **do**
        Generate particles $x_n^{(i)} \sim q_n(x_n \mid x_{1:n-1}^{(i)})$
        Compute the incremental importance weight:

$$\alpha_n^{(i)} \leftarrow \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)}) q_n(x_n^{(i)} \mid x_{1:n-1}^{(i)})}$$

        Update the particle weight: $w_n^{(i)} \leftarrow w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
    end for
    Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{i=1}^N w_n^{(i)}}$
    Draw N indices $\{a^{(i)}\}_{i=1}^N$ from $\{1, \ldots, N\}$ according to the probabilities $\{W_n^{(i)}\}$
    Set $x_{1:n}^{(i)} \leftarrow x_{1:n}^{(a^{(i)})}$ for all i
    Reset weights: $w_n^{(i)} \leftarrow 1$ and $W_n^{(i)} \leftarrow 1/N$
end for
Output: Particle trajectories:

$$X_{1:T} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_T^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_T^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \ldots & x_T^{(N)} \end{bmatrix}$$

Theorem 2.4 (Relative Asymptotic Variance with Resampling)**.** The relative asymptotic variance of the IS estimate of the normalizing constant Z_n with resampling at every time step is

$$\frac{\mathrm{Var}(\widetilde{Z}_n)}{Z_n^2} = \frac{1}{N}\left[\left(\int \frac{\pi_1^2(x_1)}{q_1(x_1)}\, dx_1 - 1\right) + \sum_{k=2}^{n}\left(\int \frac{\pi_k^2(x_{1:k})}{\pi_{k-1}(x_{1:k-1})\, q_k(x_k \mid x_{1:k-1})}\, dx_{k-1:k} - 1\right)\right].$$

A proof is omitted but follows from the Feynman–Kac framework; see, e.g., Chapter 9 in Moral [2004] for a proof.

Notably, comparing Theorem 2.4 with Theorem 2.1 reveals that while resampling introduces additional variance, the resampling has a resetting property of the systems. Thus, we get that the associated errors accumulate linearly rather than multiplicatively. This becomes important when the dimension becomes large. We continue Example 2.3 using Theorem 2.4 to highlight this.

Example 2.5 (Example 2.3 continued)**.** Using Theorem 2.4 and using the derivation done in Example 2.3 we have that it is finite for $\sigma^2 > 1/2$ and the relative variance using resampling at every time step is approximately equal to

$$\begin{aligned}\frac{\operatorname{Var}(\widetilde{Z}_n)}{Z_n^2} &\approx \frac{1}{N}\left[\left(\int \frac{\pi_1^2(x_1)}{q_1(x_1)}\,dx_1 - 1\right) + \sum_{k=2}^{n}\left(\int \frac{\pi_k^2(x_{1:k})}{\pi_{k-1}(x_{1:k-1})\,q_k(x_k \mid x_{1:k-1})}\,dx_{k-1:k} - 1\right)\right] \\ &= \frac{n}{N}\left[\left(\frac{\sigma^4}{2\sigma^2 - 1}^{1/2}\right) - 1\right]\end{aligned}$$

which is linear in n in contrast to the exponential growth of the IS estimate of the relative variance. If we again select $\sigma^2 = 1.2$ then to obtain a relative variance of 0.01 we only need $N \approx 10^4$ particles instead of the $N \approx 2 \cdot 10^{23}$ particles that were needed for the IS estimate to obtain the same precision. That is, we obtain an improvement by 19 orders of magnitude.

This setup favors sequential Monte Carlo (SMC) massively since the density $\pi_n(x_{1:n})$ factorizes. A more realistic example is given in Example 3.1.

Reducing variance of Resampling

While resampling helps mitigate weight degeneracy by focusing computational effort on high-probability regions, it introduces additional variance into the algorithm. We describe two distinct techniques that can reduce this extra variance, thereby improving the overall efficiency of the SISR algorithm. Notably, these approaches are complementary and can be applied simultaneously.

We can reduce the variance introduced during resampling by changing the sampling scheme. Several methods exist, but we will focus on stratified resampling, a method often used in survey sampling [Kiderlen, 2022]. In stratified resampling, the interval $[0, 1]$ is divided into N equal strata, and one uniform random number is drawn from each sub-interval. That is, for $i = 1, \ldots, N$, we draw

$$u_i \sim \text{Uniform}\left(\frac{i-1}{N}, \frac{i}{N}\right).$$

Each u_i is then used to select a particle based on the cumulative normalized weights. In Douc et al. [2005] it was shown that stratified resampling always gives lower variance than multinomial resampling when used in a particle filter. We provide a proof of this result below.

Theorem 2.6. Let $\{x_{1:n}^{(i)}, W_n^{(i)}\}_{i=1}^N$ be the particle set and normalized weights at time n. Suppose we draw N resampled particles using either multinomial or stratified resampling. Let $\widetilde{Z}_n$ denote the SMC estimate of the normalizing constant, updated as

$$\widetilde{Z}_n = \widetilde{Z}_{n-1}\,\alpha_n^{\text{MC}},$$

where α_n^{MC} is a Monte Carlo (MC) estimate of the incremental weight factor.

Then,

$$\operatorname{Var}\big(\widetilde{Z}_n \mid \{x_{1:n}^{(i)}, W_n^{(i)}\}\big)_{\text{stratified}} \leq \operatorname{Var}\big(\widetilde{Z}_n \mid \{x_{1:n}^{(i)}, W_n^{(i)}\}\big)_{\text{multinomial}}.$$

Proof. Conditional on $\{x_{1:n}^{(i)}, W_n^{(i)}\}$, the previous estimate $\widetilde{Z}_{n-1}$ is fixed. Hence, it suffices to compare $\mathrm{Var}(\alpha_n^{\mathrm{MC}})$ under the two resampling schemes.

Let $g_n^{(i)}$ denote the incremental weight for particle i. The update based on the resampled particles is

$$\alpha_n^{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} N_n^{(i)}\, g_n^{(i)},$$

where $N_n^{(i)}$ is the number of times particle i is selected.

For multinomial resampling, where

$$\big(N^{(1)}, \ldots, N^{(N)}\big) \sim \mathrm{Multinomial}\big(N, W^{(1)}, \ldots, W^{(N)}\big)$$

then we have the following properties

$$\begin{aligned}\mathrm{Var}(N^{(i)}) &= NW^{(i)}(1 - W^{(i)}),\\ \mathrm{Cov}(N^{(i)}, N^{(j)}) &= -NW^{(i)}W^{(j)}, \quad i \neq j.\end{aligned}$$

Using these properties, we compute the variance as follows.

$$\begin{aligned}\mathrm{Var}(\alpha_n^{\mathrm{MC}}) &= \mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N} N_n^{(i)} g_n^{(i)}\right)\\ &= \frac{1}{N^2}\,\mathrm{Var}\left(\sum_{i=1}^{N} N_n^{(i)} g_n^{(i)}\right)\\ &= \frac{1}{N^2}\left(\sum_{i=1}^{N} \mathrm{Var}\big(N_n^{(i)}\big)\big(g_n^{(i)}\big)^2 + \sum_{i\neq j} \mathrm{Cov}\big(N_n^{(i)}, N_n^{(j)}\big) g_n^{(i)} g_n^{(j)}\right)\\ &= \frac{1}{N^2}\left(\sum_{i=1}^{N} NW_n^{(i)}\big(1 - W_n^{(i)}\big)\big(g_n^{(i)}\big)^2 - \sum_{i\neq j} NW_n^{(i)} W_n^{(j)} g_n^{(i)} g_n^{(j)}\right)\\ &= \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)}\big(g_n^{(i)}\big)^2 - \sum_{i=1}^{N}\big(W_n^{(i)}\big)^2\big(g_n^{(i)}\big)^2 - \sum_{i\neq j} W_n^{(i)} W_n^{(j)} g_n^{(i)} g_n^{(j)}\right)\\ &= \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)}\big(g_n^{(i)}\big)^2 - \left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2\right).\end{aligned}$$

For stratified resampling, consider each draw as coming from a separate stratum $j = 1, \ldots, N$, with within-stratum weights $\{W_{i,j}\}_i$, where $W_{i,j}$ is the probability of selecting particle i in stratum j. These satisfy $\sum_i W_{i,j} = 1$ for each j, and $\sum_j W_{i,j} = NW_n^{(i)}$. Let

$$a_j = \sum_{i=1}^{N} W_{i,j}\, g_n^{(i)}$$

denote the expected value of the draw from stratum j. Then

$$\begin{aligned}\mathrm{Var}(\alpha_n^{\mathrm{MC}})_{\mathrm{stratified}} &= \frac{1}{N^2}\sum_{j=1}^{N}\left(\sum_{i=1}^{N} W_{i,j}\big(g_n^{(i)}\big)^2 - a_j^2\right)\\ &= \frac{1}{N}\sum_{i=1}^{N} W_n^{(i)}\big(g_n^{(i)}\big)^2 - \frac{1}{N^2}\sum_{j=1}^{N} a_j^2.\end{aligned}$$

Comparing the two expressions gives

$$\mathrm{Var}(\alpha_n^{\mathrm{MC}})_{\mathrm{multinomial}} - \mathrm{Var}(\alpha_n^{\mathrm{MC}})_{\mathrm{stratified}} = \frac{1}{N^2}\sum_{j=1}^{N} a_j^2 - \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2.$$

Since $x \mapsto x^2$ is convex, Jensen's inequality implies

$$\frac{1}{N}\sum_{j=1}^{N} a_j^2 \geq \left(\frac{1}{N}\sum_{j=1}^{N} a_j\right)^2,$$

and thus

$$\begin{aligned}
\mathrm{Var}(\alpha_n^{\mathrm{MC}})_{\mathrm{multinomial}} - \mathrm{Var}(\alpha_n^{\mathrm{MC}})_{\mathrm{stratified}} &= \frac{1}{N^2}\sum_{j=1}^{N} a_j^2 - \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2 \\
&\geq \frac{1}{N}\left(\frac{1}{N}\sum_{j=1}^{N} a_j\right)^2 - \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2 \\
&= \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2 - \frac{1}{N}\left(\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}\right)^2 \\
&= 0,
\end{aligned}$$

where we used that

$$\sum_{j=1}^{N} a_j = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{i,j}\, g_n^{(i)} = N\sum_{i=1}^{N} W_n^{(i)} g_n^{(i)}.$$

□

Another way to reduce the variance of the SISR algorithm is to perform the resampling step only when many particles have low weights. We will refer to this approach as sequential importance sampling with adaptive resampling (SISAR). A common metric used to determine when to trigger a resampling step is the particle Effective Sample Size (particle ESS), first introduced in Liu and Chen [1995]. The particle ESS at time n is defined as:

$$\mathrm{ESS}_n^{\mathrm{PF}} := \frac{1}{\sum_{i=1}^{N}\left(W_n^{(i)}\right)^2}.$$

The particle ESS takes values between 1 and N. When the particle ESS falls below a predetermined threshold, N_τ, often chosen as $N_\tau = N/2$, it indicates that most of the weight is carried by only a few particles, signaling that resampling is warranted.

The motivation behind this metric is that the MC estimator of a function f, based on a weighted sample, is

$$\hat{I}_w = \sum_{i=1}^{N} W_n^{(i)} f(x^{(i)}),$$

and the variance of this estimator is

$$\mathrm{Var}(\hat{I}_w) = \sum_{i=1}^{N} (W_n^{(i)})^2 \,\mathrm{Var}[f(x^{(i)})] = \sigma^2 \sum_{i=1}^{N} (W_n^{(i)})^2,$$

where $\sigma^2 = \text{Var}[f(x^{(i)})]$ is the variance of the function f. In contrast, the unweighted MC estimator based on M independent samples is

$$\hat{I}_u = \frac{1}{M} \sum_{i=1}^{M} f(x^{(i)}),$$

with variance

$$\text{Var}[\hat{I}_u] = \frac{\sigma^2}{M}.$$

By setting the variances of the weighted and unweighted estimators equal to each other, we find that the particle ESS can be interpreted as the number of equally weighted particles that would produce the same variance as the current weighted estimator. When all particles have equal weights we obtain $\text{ESS}_n^{\text{PF}} = N$, while when only a few particles carry most of the weight, ESS_n^{PF} becomes much smaller, approaching 1 in the extreme case.

Thus, the particle ESS serves as a diagnostic for particle degeneracy, where a low value indicates that few particles are effectively contributing to the estimate, which can lead to high-variance estimators. This observation motivates the use of resampling to focus computational effort on the more informative particles and maintain diversity within the particle population.

In conclusion, the SISAR algorithm is just the SISR algorithm described in Algorithm 2.2, where the resampling is only done when the resampling condition is met. A generic algorithm incorporating adaptive resampling is described in Algorithm 2.3. The time complexity remains $O(NT)$, assuming the resampling condition can be computed in $O(N)$, which holds for the particle ESS with a predetermined threshold.

Algorithm 2.3 Sequential Importance Sampling with Adaptive Resampling (SISAR)

Input: Number of particles N, proposal distributions $q_n(x_n \mid x_{1:n-1})$, unnormalized target densities $\gamma_n(x_{1:n})$
for each particle $i = 1, \ldots, N$ **do**
    Generate initial particles $x_1^{(i)} \sim q_1(x_1)$
    Compute initial weights: $w_1^{(i)} \leftarrow \frac{\gamma_1(x_1^{(i)})}{q_1(x_1^{(i)})}$
    Normalize: $W_1^{(i)} \leftarrow \frac{w_1^{(i)}}{\sum_{j=1}^{N} w_1^{(j)}}$
end for
for each time step $n = 2, \ldots, T$ **do**
    for each particle $i = 1, \ldots, N$ **do**
        Generate particles $x_n^{(i)} \sim q_n(x_n \mid x_{1:n-1}^{(i)})$
        Compute the incremental importance weight:

$$\alpha_n^{(i)} = \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)}) q_n(x_n^{(i)} \mid x_{1:n-1}^{(i)})}$$

        Update the particle weight: $w_n^{(i)} = w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
    end for
    Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{i=1}^{N} w_n^{(i)}}$
    if Resampling condition is met **then**
        Draw N indices $\{a^{(i)}\}_{i=1}^{N}$ from $\{1, \ldots, N\}$ according to the probabilities $\{W_n^{(i)}\}$
        Set $x_{1:n}^{(i)} \leftarrow x_{1:n}^{(a^{(i)})}$ for all i
        Reset weights: $w_n^{(i)} \leftarrow 1$
    end if
end for
Output: Particle trajectories and weights:

$$X_{1:T} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_T^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_T^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \ldots & x_T^{(N)} \end{bmatrix}, \quad W_{1:T} = \begin{bmatrix} W_1^{(1)} & W_2^{(1)} & \ldots & W_T^{(1)} \\ W_1^{(2)} & W_2^{(2)} & \ldots & W_T^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ W_1^{(N)} & W_2^{(N)} & \ldots & W_T^{(N)} \end{bmatrix}$$

3 State-Space Models

This chapter builds on the work of Doucet and Johansen [2009] and Kantas et al. [2015]. Suppose we have a state-space model (SSM), also called a hidden Markov model. Specifically, we consider a discrete-time Markov process $\{X_n; n \geq 0\}$, where each X_n takes values in $\mathcal{X}$. This process is characterized by an initial distribution

$$X_0 \sim \mu_\theta(x_0)$$

and the transition density

$$X_{n+1} \mid (X_n = x_n) \sim f_\theta(x_{n+1} \mid x_n) \tag{3.1}$$

for some parameter $\theta \in \Theta$. Our goal is to infer the latent states $\{X_n\}$ (and θ if it is unknown), given a sequence of noisy observations $\{Y_n; n \geq 1\}$, where each Y_n takes values in the space $\mathcal{Y}$. The observation Y_n is assumed to be conditionally independent of all other observations and states, given the latent state X_n, and is characterized by

$$Y_n \mid X_n = x_n \sim g_\theta(y_n \mid x_n). \tag{3.2}$$

The structure of the SSM is also visualized in Figure 3.1 as a directed acyclic graph (DAG). For simplicity, we consider only the homogeneous case, where the transition and observation densities are time-invariant. The extension to the inhomogeneous case follows naturally.

Let $y_{1:T} = (y_1, \ldots, y_T)$ denote the sequence of observations up to time $T \geq 1$. For the rest of this chapter let $\mathcal{X} \subseteq \mathbb{R}^{d_x}$, $\mathcal{Y} \subseteq \mathbb{R}^{d_y}$, and let the densities be with respect to the corresponding Lebesgue measure.

The likelihood function p_θ is given by

$$p_\theta(y_{1:n}) = \int p_\theta(x_{0:n}, y_{1:n})\, dx_{0:n},$$

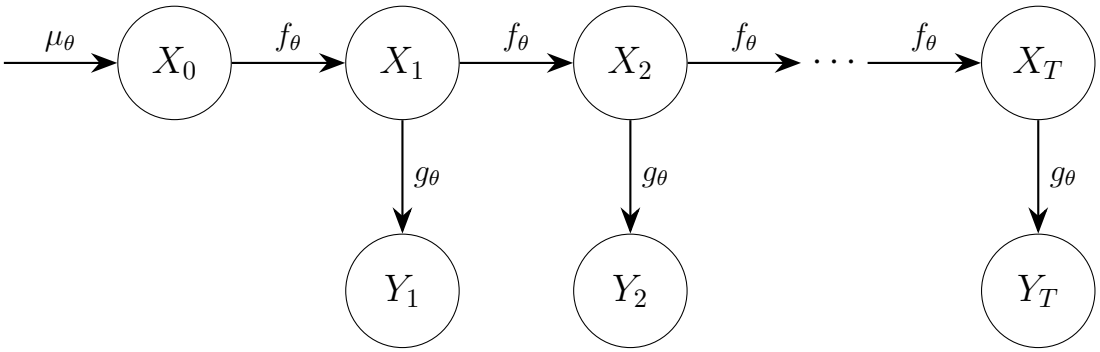

Figure 3.1: The structure of a SSM visualized in a DAG.

where $p_\theta(x_{0:n}, y_{1:n})$ is the joint density of $(X_{0:n}, Y_{1:n})$ which by Equations (3.1)-(3.2) is given by

$$p_\theta(x_{0:n}, y_{1:n}) = \mu_\theta(x_0) \prod_{k=1}^{n} f_\theta(x_k \mid x_{k-1}) \prod_{k=1}^{n} g_\theta(y_k \mid x_k).$$

Here, the product over the transition densities $f_\theta(x_k \mid x_{k-1})$ captures the evolution of the latent states, while the product over the likelihoods $g_\theta(y_k \mid x_k)$ incorporates the information provided by the observations.

However, the likelihood is often intractable, necessitating Monte Carlo (MC) methods. The literature splits this setup up into two problems, filtering and smoothing. We define them as follows:

- *Filtering:* At each time step n, the goal is to sequentially approximate: The joint conditional distribution of the latent states given the observations,

  $$p_\theta(x_{0:n} \mid y_{1:n}),$$

  and the marginal likelihood,

  $$p_\theta(y_{1:n}).$$

  So, at time $n = 1$, we approximate $p_\theta(x_{0:1} \mid y_1)$ and $p_\theta(y_1)$; at time $n = 2$, we approximate $p_\theta(x_{0:2} \mid y_{1:2})$ and $p_\theta(y_{1:2})$, and so on. This sequential framework aligns directly with the setup described in the previous chapter.

- *Smoothing:* The objective in smoothing is to estimate the latent states by using the entire sequence of observations $y_{1:T}$. In particular, approximating the joint conditional distribution

  $$p_\theta(x_{0:n} \mid y_{1:T}), \quad n = 0, \ldots, T,$$

  and the marginal conditional distributions

  $$p_\theta(x_n \mid y_{1:T}), \quad n = 0, \ldots, T.$$

  Because smoothing uses future observations, the resulting state estimates are generally more accurate and smoother than those obtained via filtering. This improvement is particularly beneficial when real-time estimation is not required.

That is, the difference between filtering and smoothing is whether the inference is carried out on-line (filtering) or off-line (smoothing). On-line inference is done sequentially when observations become available, and off-line inference uses a fixed number of observations.

For the remainder of this chapter, we suppose that θ is known.

3.1 Filtering

We can write the density $p_\theta(x_{0:n} \mid y_{1:n})$ and the likelihood $p_\theta(y_{1:n})$ recursively for $n \geq 1$. For notational convenience, we define $y_{1:0}$ to denote the empty set. We can then write $p_\theta(x_{0:n} \mid y_{1:n})$ as:

$$p_\theta(x_{0:n} \mid y_{1:n}) = p_\theta(x_{0:n-1} \mid y_{1:n-1}) \frac{f_\theta(x_n \mid x_{n-1}) g_\theta(y_n \mid x_n)}{p_\theta(y_n \mid y_{1:n-1})}$$

Algorithm	SIS	SISR	SISAR
RMSE	0.87 (0.09)	0.76 (0.08)	0.79 (0.09)

Table 3.1: RMSE comparison for SIS, SISR, and SISAR, based on 10,000 MC replications for Example 3.1. Values shown are the means with standard deviations in parentheses.

and

$$p_\theta(y_{1:n}) = p_\theta(y_{1:n-1}) p_\theta(y_n \mid y_{1:n-1}),$$

where the predictive likelihood $p_\theta(y_n \mid y_{1:n-1})$ is given by

$$\begin{aligned} p_\theta(y_n \mid y_{1:n-1}) &= \int p_\theta(y_n, x_n \mid y_{1:n-1})\, dx_n \\ &= \int g_\theta(y_n \mid x_n)\, p_\theta(x_n \mid y_{1:n-1})\, dx_n \\ &= \int g_\theta(y_n \mid x_n) f_\theta(x_n \mid x_{n-1})\, p_\theta(x_{n-1} \mid y_{1:n-1})\, dx_{n-1:n}. \end{aligned}$$

Here we are in the setup discussed in Chapter 2, and we can for example use the SISAR algorithm where the proposal distribution is chosen as the transition density, i.e. as $f_\theta(x_n \mid x_{n-1})$. From now on, we refer to a particle filter that uses the transition density as the proposal distribution as a *bootstrap particle filter*.

Example 3.1 (SSM with known θ)**.** Consider the following non-linear Gaussian SSM, where the parameter vector $\theta = (\phi, \sigma_x, \sigma_y)$ is assumed to be known:

$$\begin{aligned} &X_0 \sim N(0,1), \\ &X_t = \phi X_{t-1} + \sin(X_{t-1}) + \sigma_x V_t, \quad V_t \sim N(0,1), \quad t \geq 1, \\ &Y_t = X_t + \sigma_y W_t, \quad W_t \sim N(0,1), \quad t \geq 1. \end{aligned}$$

We set $\phi = 0.7$, $\sigma_x = 1$, and $\sigma_y = 1$, with a time horizon of $T = 50$, and employ $N = 1000$ particles. A single realization of this SSM is illustrated in Figure 3.2.

Our objective is to compare the performance of three particle filtering methods—sequential importance sampling (SIS), sequential importance sampling with resampling (SISR), and sequential importance sampling with adaptive resampling (SISAR)—for estimating the latent state X_t. All methods are implemented using the bootstrap particle filter framework. Performance is assessed using the root mean squared error (RMSE), computed over 10,000 MC replications. For both SISR and SISAR, we use stratified resampling. The `R` code used for this simulation is available in the accompanying GitHub repository.

Figure 3.3 shows the particle filter estimates of the latent state for the same realized trajectory as in Figure 3.2. Summary statistics from the full set of 10,000 replications are reported in Table 3.1, presented as means with standard deviations in parentheses.

Among the three methods, SIS exhibits the highest RMSE due to weight degeneracy, where only a few particles carry most of the weight. SISR achieves the lowest RMSE, marginally outperforming SISAR. However, SISAR offers greater computational efficiency by avoiding resampling at every time step.

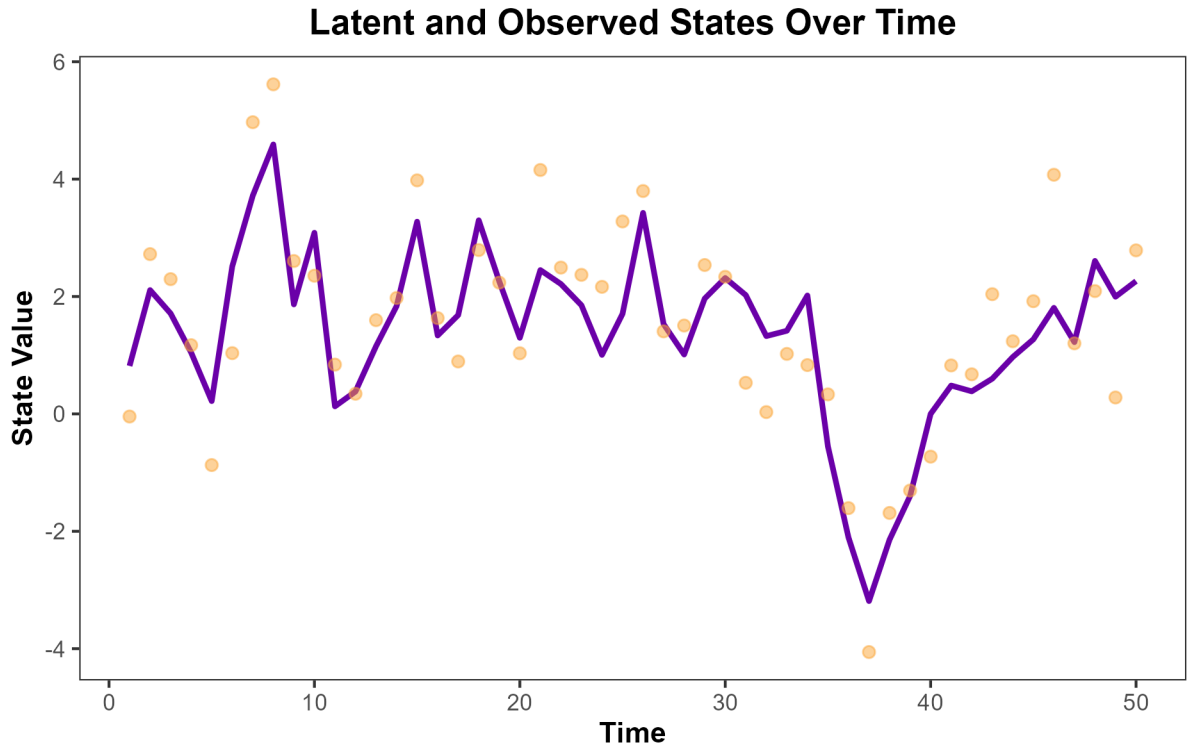

Figure 3.2: A simulated trajectory of the data generating process (DGP) in Example 3.1. The line represents the latent state, while the points denote the corresponding observations.

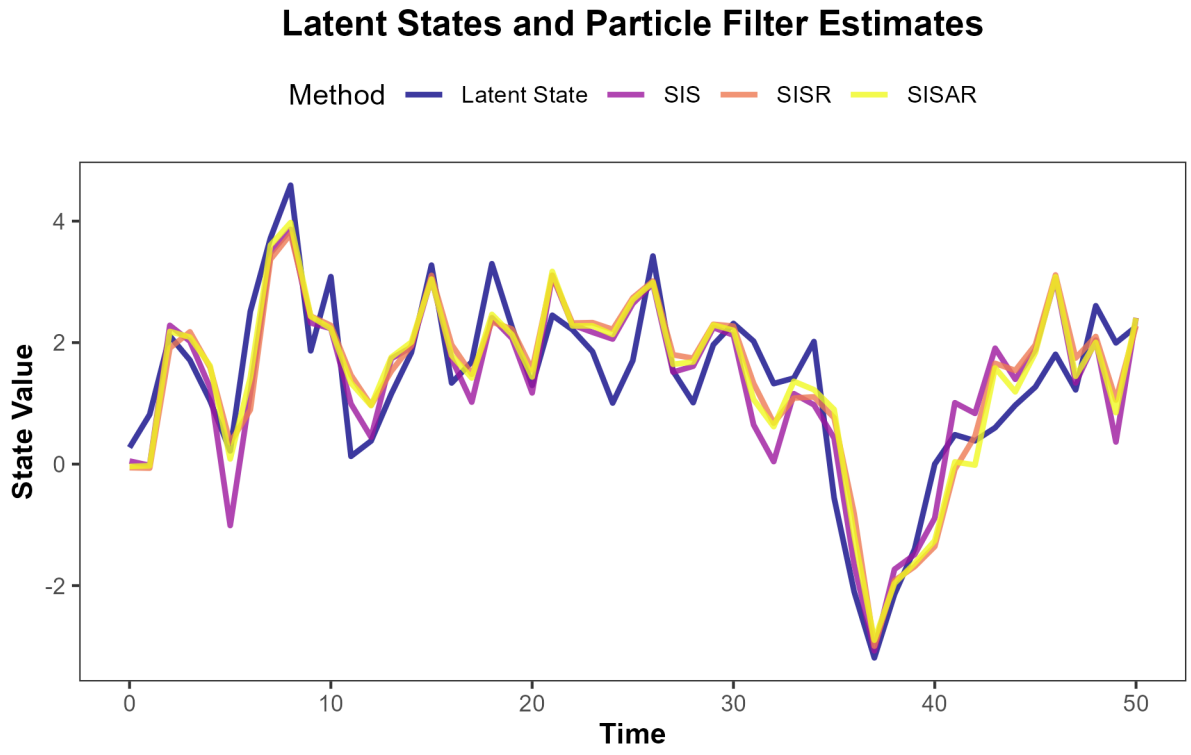

Figure 3.3: The particle filter estimates of the latent state of the DGP in Example 3.1.

3.2 Smoothing

Forward-Backward Recursions

By doing a decomposition of the joint distribution $p_\theta(x_{1:T} \mid y_{1:T})$ we see that conditional on $y_{1:T}$ that $\{X_n\}$ is a non-homogeneous Markov process

$$\begin{aligned} p_\theta(x_{0:T} \mid y_{1:T}) &= p_\theta(x_T \mid y_{1:T}) \prod_{n=0}^{T-1} p_\theta(x_n \mid x_{n+1}, y_{1:T}) \\ &= p_\theta(x_T \mid y_{1:T}) \prod_{n=0}^{T-1} p_\theta(x_n \mid x_{n+1}, y_{1:n}), \end{aligned} \tag{3.3}$$

where we in the 2nd equality used the Markov property. Using Bayes' Theorem, we can write the marginal $p_\theta(x_n \mid x_{n+1}, y_{1:n})$ as

$$p_\theta(x_n \mid x_{n+1}, y_{1:n}) = \frac{f_\theta(x_{n+1} \mid x_n) p_\theta(x_n \mid y_{1:n})}{p_\theta(x_{n+1} \mid y_{1:n})}. \tag{3.4}$$

By integrating out $(x_{1:n-1}, x_{n+1:T})$ in Equation (3.3) we have that the marginal distribution $p_\theta(x_n \mid y_{1:T})$ is given by

$$\begin{aligned} p_\theta(x_n \mid y_{1:T}) &= \int p_\theta(x_{1:T} \mid y_{1:T}) \, dx_{0:n-1} \, dx_{n+1:T} \\ &= \int p_\theta(x_T \mid y_{1:T}) \prod_{k=0}^{T-1} p_\theta(x_k \mid x_{k+1}, y_{1:k}) \, dx_{0:n-1} \, dx_{n+1:T} \\ &= \int p_\theta(x_T \mid y_{1:T}) \prod_{k=0}^{T-1} \frac{f(x_{k+1} \mid x_k) p_\theta(x_k \mid y_{1:k})}{p_\theta(x_{k+1} \mid y_{1:k})} \, dx_{0:n-1} \, dx_{n+1:T} \\ &= p_\theta(x_n \mid y_{1:n}) \int \frac{f_\theta(x_{n+1} \mid x_n)}{p_\theta(x_{n+1} \mid y_{1:n})} p_\theta(x_{n+1} \mid y_{1:T}) \, dx_{n+1}. \end{aligned} \tag{3.5}$$

This is referred to as forward-backward smoothing, as a forward pass yields $\{p_\theta(x_n \mid y_{1:n})\}_{n=0}^T$, which can then be used in a backward pass to obtain $\{p_\theta(x_n \mid y_{1:T})\}_{n=0}^T$.

Forward-Filtering Backward-Sampling

A common method for drawing samples from the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$ is the Forward-Filtering Backward-Sampling with marginalization (FFBSm) method. This method proceeds in two steps:

1. **Forward filtering:** The filtering distributions $\{p_\theta(x_n \mid y_{1:n})\}_{n=0}^T$ are approximated using a filtering method, such as those described in Chapter 2. In this step, particles are used to approximate the filtering distributions, with each particle $x_n^{(j)}$ associated with a weight $W_n^{(j)}$, where $W_n^{(j)}$ is the normalized weight of the particle $x_n^{(j)}$. The weight reflects how well the particle approximates the target distribution $p_\theta(x_n \mid y_{1:n})$. These weights are essential for guiding the importance of each particle in subsequent sampling steps.

2. **Backward sampling:**

- First, we sample the final state X_T from the distribution $p_\theta(x_T \mid y_{1:T})$, which is proportional to the forward-filtered particles at time T. Since the forward filter gives a weighted sample $\{x_T^{(j)}\}$ we sample $X_T^{(i)}$ with probability $W_T^{(j)}$.
- For each time step $n = T-1, T-2, \ldots, 0$, we sample the state X_n given the sampled state X_{n+1} and the observations $y_{1:n}$. The sampling procedure is based on the backward conditional distribution $p_\theta(x_n \mid X_{n+1}, y_{1:n})$, which is proportional to the joint likelihood of the transition from x_n to x_{n+1} and the forward filtering distribution. More formally, we have:

$$p_\theta(x_n \mid X_{n+1}, y_{1:n}) = \frac{f_\theta(X_{n+1} \mid x_n) p_\theta(x_n \mid y_{1:n})}{p_\theta(X_{n+1} \mid y_{1:n})},$$

  where $f_\theta(X_{n+1} \mid x_n)$ is the state transition density, and $p_\theta(x_n \mid y_{1:n})$ is the forward filtering distribution for state x_n. To sample $X_n^{(i)}$, we first compute the backward weights $\tilde{w}_n^{(j)}$, which adjust the forward weights by the transition probability:

$$\tilde{w}_n^{(j)} = W_n^{(j)} f_\theta(X_{n+1}^{(j)} \mid x_n^{(j)}).$$

- Finally, we normalize the backward weights:

$$\tilde{W}_n^{(j)} = \frac{\tilde{w}_n^{(j)}}{\sum_{j=1}^N \tilde{w}_n^{(j)}}.$$

  Then, we sample an index k from $\{1, 2, \ldots, N\}$ with probability $\tilde{W}_n^{(k)}$, and set $X_n^{(i)} = x_n^{(k)}$. This step ensures that the sampled trajectory is consistent with the joint smoothing distribution $p_\theta(x_{0:T} \mid y_{1:T})$.

In summary, the forward weights $W_n^{(j)}$ capture the likelihood of each particle given the observations up to time n, while the backward weights $\tilde{w}_n^{(j)}$ adjust for the transition probabilities between states. The pseudocode for the FFBSm algorithm is provided in Algorithm 3.1. The time complexity of the FFBSm algorithm is $O(N^2T)$ since the backward sampling step involves three nested loops. Hence, the total time complexity is $O(N^2T)$. This is significantly slower than the algorithms in the previous chapter that had a complexity of $O(NT)$.

Another approach for smoothing is the Generalised Two-Filter Formula, as discussed in BRESLER [1986].

Example 3.2 (Example 3.1 continued)**.** We continue from Example 3.1, using the same model and simulation setup. In this extension, we apply smoothing using the FFBSm algorithm. The forward filtering step is performed with the bootstrap particle filter implemented via SISAR, incorporating stratified resampling.

Due to the computational complexity of the FFBSm algorithm—particularly the $O(N^2T)$ cost of the backward sampling step—the implementation was carried out in `Julia` for computational efficiency. The corresponding `Julia` code is available in the accompanying GitHub repository.

Figure 3.4 displays a representative trajectory of the latent state X_t, along with the filtering estimate obtained using SISAR and the smoothed estimate from the FFBSm algorithm.

Algorithm 3.1 Forward-Filtering Backward-Sampling with marginalization (FFBSm)

Input: Observations $y_{1:T}$, number of particles N, initial distribution $\mu(x_0)$, state transition density $f_\theta(x_{n+1} \mid x_n)$, observation density $g_\theta(y_n \mid x_n)$
Forward Filtering:
Use a forward filtering algorithm (described in Chapter 2) to obtain, for each $n = 0, \ldots, T$, the particle approximation $\{x_n^{(j)}, W_n^{(j)}\}_{j=1}^N$ of $p_\theta(x_n \mid y_{1:n})$, where $W_n^{(j)}$ are the normalized weights
for $i = 1, \ldots, N$ **do**
    Backward Sampling:
    Sample index k from $\{1, \ldots, N\}$ with probability $W_T^{(k)}$
    Set $X_T^{(i)} \leftarrow x_T^{(k)}$
    for $n = T-1, T-2, \ldots, 0$ **do**
        for $j = 1, \ldots, N$ **do**
            Compute backward weights:

$$\tilde{w}_n^{(j)} \leftarrow W_n^{(j)} \, f_\theta\Big(X_{n+1}^{(i)} \mid x_n^{(j)}\Big)$$

        end for
        Normalize: $\tilde{W}_n^{(j)} \leftarrow \frac{\tilde{w}_n^{(j)}}{\sum_{j=1}^N \tilde{w}_n^{(j)}}$
        Sample index k from $\{1, \ldots, N\}$ with probability $\tilde{W}_n^{(k)}$
        Set $X_n^{(i)} \leftarrow x_n^{(k)}$
    end for
end for
Output: N sampled trajectories $\{X_{0:T}^{(i)}\}_{i=1}^N$

Algorithm	SIS	SISR	SISAR	FFBSm
RMSE	0.87 (0.09)	0.76 (0.08)	0.79 (0.09)	0.70 (0.08)

Table 3.2: Comparison of RMSE performance for SIS, SISR, SISAR, and FFBSm algorithms based on 10,000 MC replications for Example 3.2. Reported values are means with standard deviations in parentheses.

To evaluate performance, we compute the RMSE and standard deviation over 10,000 MC iterations. Table 3.2 summarizes the results, including the previously reported filtering methods for comparison. As expected, the smoothing method yields the best performance, though the improvement over the filtering algorithms is modest.

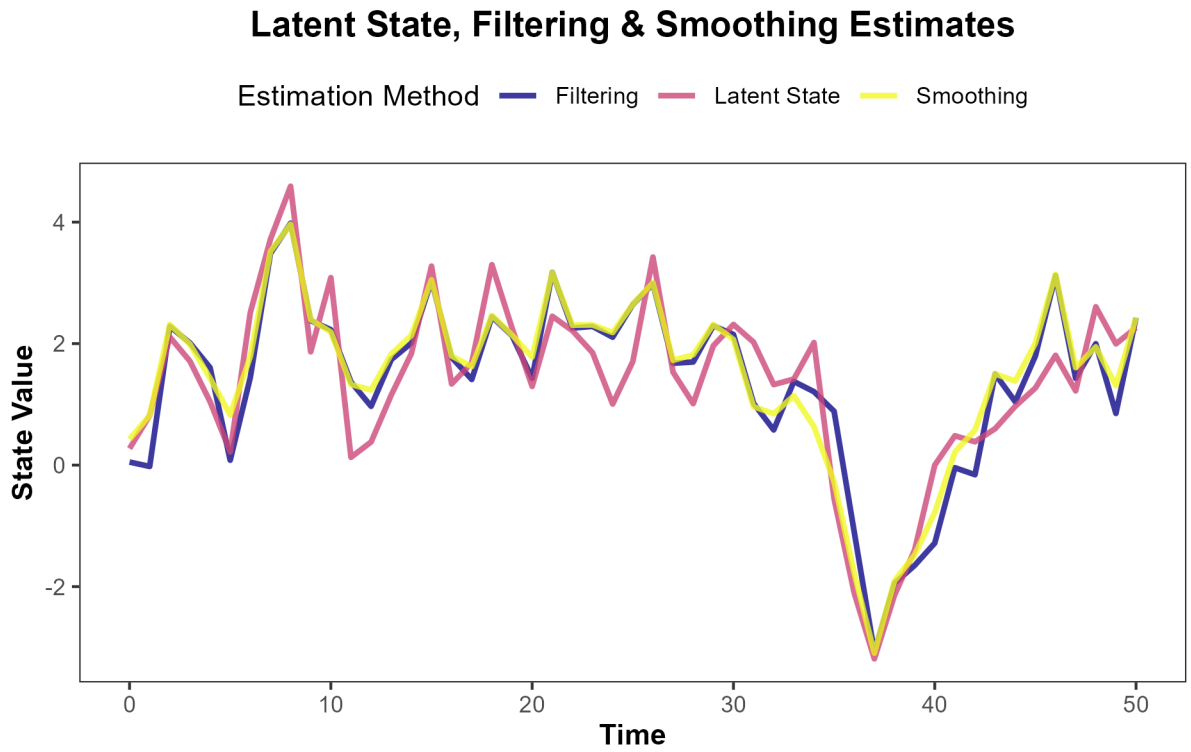

Figure 3.4: A simulated trajectory of the SSM in Example 3.2, along with filtering and smoothing estimates.

4 Bayesian Inference for State-Space Models

As in Chapter 3, we consider a state-space model (SSM) characterized by an intial distribution $\mu(x_0 \mid \theta)$, a transition density $f(x_{n+1} \mid x_n, \theta)$, and an observation density $g(y_n \mid x_n, \theta)$. In the previous chapter, we used the shorthand notation $h_\theta(\cdot)$ for densities depending on θ. Here, we instead adopt the more explicit notation $h(\cdot \mid \theta)$, which aligns better with Bayesian conventions. Let

$$y_{1:T} = (y_1, \ldots, y_T)$$

denote the sequence of observations up to time $T \geq 1$. We assume that $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ and $\mathcal{Y} \subseteq \mathbb{R}^{d_y}$, and that all densities are defined with respect to the corresponding Lebesgue measures. Furthermore, we assume that the parameter θ admits a density with respect to the Lebesgue measure.

In many state-space models, both the parameters θ and the latent states $x_{0:T}$ are unknown and must be inferred simultaneously. In the Bayesian setting, inference is carried out on the joint posterior distribution

$$\begin{aligned} p(x_{0:T}, \theta \mid y_{1:T}) &= \frac{p(y_{1:T} \mid x_{0:T}, \theta) p(x_{0:T}, \theta)}{p(y_{1:T})} \\ &= \frac{p(y_{1:T} \mid x_{0:T}, \theta) \pi(\theta) p(x_{0:T} \mid \theta)}{p(y_{1:T})} \\ &\propto \pi(\theta) p(y_{1:T} \mid x_{0:T}, \theta) p(x_{0:T} \mid \theta), \end{aligned} \tag{4.1}$$

where $\pi(\theta)$ denotes the prior on θ, and

$$p(y_{1:T}) = \int \int \pi(\theta) p(y_{1:T} \mid x_{0:T}, \theta) p(x_{0:T} \mid \theta) \, dx_{0:T} \, d\theta$$

is the marginal likelihood, which is intractable in general and treated as a normalizing constant. Using the Markov structure of the model, the full posterior can also be expressed as

$$p(x_{0:T}, \theta \mid y_{1:T}) \propto \pi(\theta) \mu(x_0 \mid \theta) \prod_{n=0}^{T-1} f(x_{n+1} \mid x_n, \theta) \prod_{n=1}^{T} g(y_n \mid x_n, \theta).$$

When the latent states are not of primary interest, they can be integrated out to obtain the marginal posterior for the parameters:

$$\begin{aligned} p(\theta \mid y_{1:T}) &= \int p(x_{0:T}, \theta \mid y_{1:T}) \, dx_{0:T} \\ &\propto \pi(\theta) p(y_{1:T} \mid \theta), \end{aligned} \tag{4.2}$$

where the marginal likelihood is

$$p(y_{1:T} \mid \theta) = \int p(x_{0:T}, y_{1:T} \mid \theta)\, dx_{0:T}.$$

Since neither Equation (4.1) nor (4.2) is generally tractable, Markov chain Monte Carlo (MCMC) methods are widely used to approximate the posterior distributions. However, standard techniques such as one-variable-at-a-time Gibbs sampling often perform poorly for non-linear, non-Gaussian SSMs [Kantas et al., 2015].

4.1 Particle Marginal Metropolis-Hastings (PMMH)

A Marginal Metropolis–Hastings (MMH) sampler would sample from the joint posterior $p(x_{0:T}, \theta \mid y_{1:T})$ by using the proposal density

$$q\big((x'_{0:T}, \theta') \mid x_{0:T}, \theta\big) = q(\theta' \mid \theta) p(x'_{0:T} \mid y_{1:T}, \theta'), \tag{4.3}$$

where $q(\theta' \mid \theta)$ is a proposal density to obtain a candidate parameter θ' from θ, a candidate latent trajectory $x'_{0:T}$ given $y_{1:T}$ and θ'. The acceptance probability is given by

$$\min\left\{1,\ \frac{p(y_{1:T} \mid \theta')\pi(\theta')q(\theta \mid \theta')}{p(y_{1:T} \mid \theta)\pi(\theta)q(\theta' \mid \theta)}\right\}. \tag{4.4}$$

However, we can generally neither sample from $p(x'_{0:T} \mid y_{1:T}, \theta')$ nor compute the likelihood terms $p(y_{1:T} \mid \theta')$ and $p(y_{1:T} \mid \theta)$ from the acceptance probability. Thus, an implementation of MMH is generally impossible.

A Particle Marginal Metropolis-Hastings (PMMH) sampler is an approximation of the MMH sampler where particle methods are used to approximate these intractable terms. The pseudocode for the algorithm is given in Algorithm 4.1. We will now show that the PMMH algorithm has the target density

$$\psi(x_{0:T}, \theta) \propto \pi(\theta) p(x_{0:T} \mid \theta) p(y_{1:T} \mid x_{0:T}, \theta)$$

stationary distribution. In the following Theorem and proof we let z denote the auxiliary variables used to estimate the likelihood, which in the case of a particle filter is the particles and weights. The theorem below, inspired by Theorem 2 in Andrieu et al. [2010], which states a similar result, demonstrates that the PMMH algorithm, despite using an unbiased likelihood estimator instead of the true likelihood, preserves the correct joint posterior distribution as its stationary distribution under mild assumptions.

Theorem 4.1 (PMMH Correctness)**.** Suppose that the ideal marginal Metropolis-Hastings (MMH) sampler targeting the joint posterior

$$\psi(x_{0:T}, \theta) \propto \pi(\theta) p(x_{0:T} \mid \theta) p(y_{1:T} \mid x_{0:T}, \theta)$$

has stationary distribution $\psi(x_{0:T}, \theta)$. Assume that for each pair $(x_{0:T}, \theta)$ there exists a non-negative, unbiased estimator $\widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z)$ of $p(y_{1:T} \mid x_{0:T}, \theta)$, where z denotes the auxiliary variables generated from a density $h(z \mid x_{0:T}, \theta)$ whose support is independent of $(x_{0:T}, \theta)$, i.e.,

$$\operatorname{supp}\big(h(\cdot \mid x_{0:T}, \theta)\big) = \operatorname{supp}\big(h(\cdot \mid x'_{0:T}, \theta')\big)$$

for all $(x_{0:T}, \theta)$ and $(x'_{0:T}, \theta')$. Define the extended target on the augmented space $(x_{0:T}, \theta, z)$ by

$$\widetilde{\psi}(x_{0:T}, \theta, z) \propto \pi(\theta) p(x_{0:T} \mid \theta) \widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z) h(z \mid x_{0:T}, \theta).$$

Then, the PMMH algorithm that uses the proposal density

$$q\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) = q(\theta' \mid \theta)\, h(z' \mid x'_{0:T}, \theta')$$

hence accepts a move from $(x_{0:T}, \theta, z)$ to $(x'_{0:T}, \theta', z')$ with probability

$$\begin{aligned}
&\alpha\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) \\
&\quad = \min\left\{1,\ \frac{\pi(\theta')\, p(x'_{0:T} \mid \theta')\, \widehat{p}(y_{1:T} \mid x'_{0:T}, \theta, z')\, h(z' \mid x'_{0:T}, \theta')\, q(\theta \mid \theta')}{\pi(\theta)\, p(x_{0:T} \mid \theta)\, \widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z)\, h(z \mid x_{0:T}, \theta)\, q(\theta' \mid \theta)}\right\},
\end{aligned}$$

and has the extended target $\widetilde{\psi}(x_{0:T}, \theta, z)$ as its stationary distribution. Consequently, its marginal distribution on $(x_{0:T}, \theta)$ is $\psi(x_{0:T}, \theta)$.

Proof. First, note that integrating the extended target over the auxiliary variables z yields

$$\begin{aligned}
\int \widetilde{\psi}(x_{0:T}, \theta, z)\, dz &\propto \pi(\theta) p(x_{0:T} \mid \theta) \int \widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z) h(z \mid x_{0:T}, \theta)\, dz \\
&= \pi(\theta) p(x_{0:T} \mid \theta) p(y_{1:T} \mid x_{0:T}, \theta),
\end{aligned}$$

where the equality follows from the unbiasedness of $\widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z)$. Thus, the marginal target on $(x_{0:T}, \theta)$ is

$$\psi(x_{0:T}, \theta) \propto \pi(\theta) p(x_{0:T} \mid \theta) p(y_{1:T} \mid x_{0:T}, \theta).$$

Next, we show that the Markov chain on the extended space $(x_{0:T}, \theta, z)$ satisfies detailed balance with respect to $\widetilde{\psi}(x_{0:T}, \theta, z)$. The PMMH algorithm proposes a move from $(x_{0:T}, \theta, z)$ to $(x'_{0:T}, \theta', z')$ using the proposal density

$$q\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) = q(\theta' \mid \theta) h(z' \mid x'_{0:T}, \theta').$$

Define the ratio

$$r := \frac{\pi(\theta') p(x'_{0:T} \mid \theta') \widehat{p}(y_{1:T} \mid x'_{0:T}, \theta, z') h(z' \mid x'_{0:T}, \theta') q(\theta \mid \theta')}{\pi(\theta) p(x_{0:T} \mid \theta) \widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z) h(z \mid x_{0:T}, \theta) q(\theta' \mid \theta)},$$

so that the acceptance probability is

$$\alpha\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) = \min\{1, r\}.$$

By symmetry, the acceptance probability for the reverse move is

$$\alpha\big((x'_{0:T}, \theta', z'), (x_{0:T}, \theta, z)\big) = \min\{1, r^{-1}\}.$$

We wish to show the detailed balance equation is satisfied, that is for all $(x_{0:T}, \theta, z)$ and $(\theta', x'_{0:T}, z')$ we have

$$\begin{aligned}
&\widetilde{\psi}(x_{0:T}, \theta, z) q\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) \alpha\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) \\
&\quad = \widetilde{\psi}(x'_{0:T}, \theta', z') q\big((x'_{0:T}, \theta', z'), (x_{0:T}, \theta, z)\big) \alpha\big((x'_{0:T}, \theta', z'), (x_{0:T}, \theta, z)\big).
\end{aligned}$$

Without loss of generality, assume that $r \leq 1$. Then the product

$$\widetilde{\psi}(x_{0:T}, \theta, z) q\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big) \alpha\big((x_{0:T}, \theta, z), (x'_{0:T}, \theta', z')\big)$$

can be written as (up to the normalizing constant)

$$\begin{aligned} &\pi(\theta)\, p(x_{0:T} \mid \theta) \widehat{p}(y_{1:T} \mid x_{0:T}, \theta, z) h(z \mid x_{0:T}, \theta) q(\theta' \mid \theta) h(z' \mid x'_{0:T}, \theta') r \\ &\quad = \pi(\theta') p(x'_{0:T} \mid \theta') \widehat{p}(y_{1:T} \mid x'_{0:T}, \theta, z') h(z' \mid x'_{0:T}, \theta') q(\theta \mid \theta'). \end{aligned}$$

Similarly, the reverse move yields (up to the normalizing constant)

$$\begin{aligned} &\widetilde{\psi}(x'_{0:T}, \theta', z') q\big((x'_{0:T}, \theta', z'), (x_{0:T}, \theta, z)\big) \alpha\big((x'_{0:T}, \theta', z'), (x_{0:T}, \theta, z)\big) \\ &\quad = \pi(\theta') p(x'_{0:T} \mid \theta') \widehat{p}(y_{1:T} \mid x'_{0:T}, \theta, z') h(z' \mid x'_{0:T}, \theta') q(\theta \mid \theta') h(z \mid x_{0:T}, \theta). \end{aligned}$$

Thus, the two sides are equal, and the detailed balance equation is satisfied.

Since the detailed balance equation holds, the extended Markov chain has stationary distribution $\widetilde{\psi}(x_{0:T}, \theta, z)$ [Bhattacharya and Waymire, 2009]. Consequently, its marginal over $(x_{0:T}, \theta)$ is $\psi(x_{0:T}, \theta)$, which completes the proof. □

Note that Theorem 4.1 established that any unbiased estimator of the likelihood, under the given assumptions, ensures that

$$\psi(x_{0:T}, \theta) = p(\theta \mid y_{1:T})$$

is a stationary distribution. In particular, when using a particle filter to estimate the likelihood, the resulting Markov chain maintains $\psi(x_{0:T}, \theta)$ as a stationary distribution for any number of particles N.

The proposal density $q(\theta' \mid \theta)$ is often chosen as a random walk, that is θ'

$$q(\theta' \mid \theta) \sim N(\theta, \hat{\mathcal{P}}),$$

where $\hat{\mathcal{P}}$ is the estimated posterior covariance based on a pilot run [Dahlin and Schön, 2019].

Recall, that the particle filter algorithms discussed in Chapter 2 had time complexity $O(NT)$, and thus this algorithm has time complexity $O(MNT)$, where M is the number of MCMC iterations. Even though any choice of N leads to $\psi(x_{0:T}, \theta)$ being a stationary distribution, the parameter N directly impacts the variance of the likelihood estimator $\widehat{p}(y_{1:T} \mid \theta)$. Increasing N reduces this variance, leading to a more stable acceptance probability. In practice, N is typically determined via pilot runs that monitor the variance of the log-likelihood estimate. The common guideline based on simulation studies is to choose N so that this variance is approximately around 1-1.7 when evaluated at the posterior mean of θ [Pitt et al., 2012, Dahlin and Schön, 2019].

The number of MCMC iterations M determines the overall length of the Markov chain and hence the quality of the posterior approximation. A larger M facilitates a more thorough exploration of the posterior distribution, allowing for greater precision in the resulting parameter estimates after discarding an appropriate burn-in period. In practice, we often run several independent chains and use convergence diagnostics such as the potential scale reduction statistic, $\hat{R}$, and the effective sample size of each parameter, see also Appendix A.

Algorithm 4.1 Particle Marginal Metropolis-Hastings (PMMH)

1: **Input:** Observation sequence $y_{1:T}$; number of MCMC iterations M; number of particles N; prior density $\pi(\theta)$; proposal density $q(\theta' \mid \theta)$; and initial parameter value $\theta^{(0)}$.

2: **Initialization:** Run a particle filter with parameter $\theta^{(0)}$ to obtain a likelihood estimate $\widehat{p}(y_{1:T} \mid \theta^{(0)}, x_{0:T}^{(0)})$ and a latent state sample $x_{0:T}^{(0)}$.

3: **for** $m = 1, \ldots, M$ **do**

4: Propose a new parameter: $\theta' \sim q(\theta' \mid \theta^{(m-1)})$.

5: Run a particle filter with θ' to obtain the likelihood estimate $\widehat{p}(y_{1:T} \mid \theta', x'_{0:T})$ and a latent state sample $x'_{0:T}$.

6: Compute the acceptance probability

$$\alpha \leftarrow \min\left\{1,\ \frac{\pi(\theta')\,\widehat{p}(y_{1:T} \mid \theta', x'_{0:T})\,q(\theta^{(m-1)} \mid \theta')}{\pi(\theta^{(m-1)})\,\widehat{p}(y_{1:T} \mid \theta^{(m-1)}, x_{0:T}^{(m-1)})\,q(\theta' \mid \theta^{(m-1)})}\right\}.$$

7: With probability α, set

$$\theta^{(m)} \leftarrow \theta', \quad x_{0:T}^{(m)} \leftarrow x'_{0:T}, \quad \widehat{p}(y_{1:T} \mid \theta^{(m)}, x_{0:T}^{(m)}) \leftarrow \widehat{p}(y_{1:T} \mid \theta', x'_{0:T}).$$

8: Otherwise, set

$$\theta^{(m)} \leftarrow \theta^{(m-1)}, \quad x_{0:T}^{(m)} \leftarrow x_{0:T}^{(m-1)}, \quad \widehat{p}(y_{1:T} \mid \theta^{(m)}, x_{0:T}^{(m)}) \leftarrow \widehat{p}(y_{1:T} \mid \theta^{(m-1)}, x_{0:T}^{(m-1)}).$$

9: **end for**

10: **Return:** The chain $\{\theta^{(m)}, x_{0:T}^{(m)}\}_{m=0}^{M}$.

We will refer throughout to the *standard settings* for PMMH as using the settings described in Table 4.1. The pilot proposal distribution used is

$$q_{\text{pilot}}(\theta' \mid \theta) \sim N\big(\theta, (0.1)^2 \cdot I\big),$$

where I is the identity matrix. From the pilot run, we estimate both the posterior covariance matrix, $\hat{\mathcal{P}}$, and the posterior mean, $\hat{\theta}$. Next, using the estimated posterior mean, we calculate an estimate of the variance of the log-likelihood, denoted by $\widehat{\text{Var}}(\ell)$, by performing 100 repetitions with $N_{\text{pilot}} = 100$ particles each time. Finally, we set

$$N = \max\Big(N_{\text{pilot}} \cdot \widehat{\text{Var}}(\ell),\ 50\Big),$$

so that the log-likelihood has approximately unit variance at the estimated posterior while ensuring that N is at least 50. A pilot run is done for each chain. See Appendix C for further implementation details.

Example 4.2 (SSM with unknown θ)**.** We consider the same model from Example 3.1 and Example 3.2, that is we have $\theta = (\phi, \sigma_x, \sigma_y)$ and we have the following SSM

$$\begin{aligned}
&X_0 \sim N(0,1),\\
&X_t = \phi X_{t-1} + \sin(X_{t-1}) + \sigma_x V_t, \quad V_t \sim N(0,1), \quad t \geq 1,\\
&Y_t = X_t + \sigma_y W_t, \quad W_t \sim N(0,1), \quad t \geq 1.
\end{aligned}$$

Parameter	Value	Description
`algorithm`	SISAR	Algorithm used
`resample_fn`	Stratified	Resampling method
`pilot_proposal_sd`	0.5	Initial proposal standard deviation
`pilot_n`	100	Number of particles in pilot run
`pilot_m`	2,000	Number of MCMC iterations in pilot run
`pilot_burn_in`	500	Burn-in amount in pilot run
`pilot_target_var`	1	Target variance of the log-likelihood at posterior mean
`pilot_reps`	100	Repetitions for estimating variance of the log-likelihood
`num_chains`	4	Number of MCMC chains
`m`	40,000	Number of MCMC iterations
`burn_in`	500	Burn-in amount

Table 4.1: Standard PMMH settings.

We generate data with the true parameters $\phi = 0.7$, $\sigma_x = 1$, $\sigma_y = 1$ over $T = 50$ time steps, but treat θ as unknown. We place weakly informative priors

$$\begin{aligned} \phi &\sim \text{Uniform}(-1, 1), \\ \sigma_x &\sim \text{Normal}^+(0, 1), \\ \sigma_y &\sim \text{Normal}^+(0, 1). \end{aligned}$$

All assumptions of Theorem 4.1 are satisfied, since the bootstrap particle filter provides a non-negative, unbiased estimator of the likelihood whose auxiliary variables (particles and weights) have support independent of $(x_{0:T}, \theta)$, and all transition and observation densities have full support regardless of θ. We perform a prior predictive check, in which data are generated from the model using parameters drawn from the prior distribution. This allows us to assess whether the prior places reasonable mass on data that resemble the observed data. In Figure 4.1, we show data simulated from the data generating process (DGP) using the true parameter values $\theta = (0.7, 1, 1)$, alongside 100 datasets simulated from the prior.

We run the PMMH algorithm using the standard settings given in Table 4.1, only changing the number of MCMC samples to 50,000 to ensure reliable inference. The `R` code for this analysis is available in the accompanying GitHub repository.

To ensure that the posterior distribution produces data consistent with the observed data, we perform a posterior predictive check, shown in Figure 4.2. The results appear reasonable, with the posterior values aligning well with the observed data.

We assess the reliability of the inference by computing the split-$\widehat{R}$ statistic and the MCMC Effective Sample Size (MCMC ESS) (see Appendix A for details). As summarized in Table 4.2, the MCMC ESS are above the recommended minimum of 400, and all split-$\widehat{R}$ values are below the threshold of 1.01. These diagnostics confirm that the MCMC samples are suitable for reliable inference. The estimated posterior distribution for ϕ is shown in Figure 4.3. The posterior predictive check showed that the posterior distribution can generate data consistent with the observed data, even though the posterior mean of σ_y is 0.37, which is far from the true value of 1.0. This suggests potential identifiability issues in the model.

The estimated posterior distributions for σ_x and σ_y are available in Appendix D (Figures D.1 and D.2, respectively).

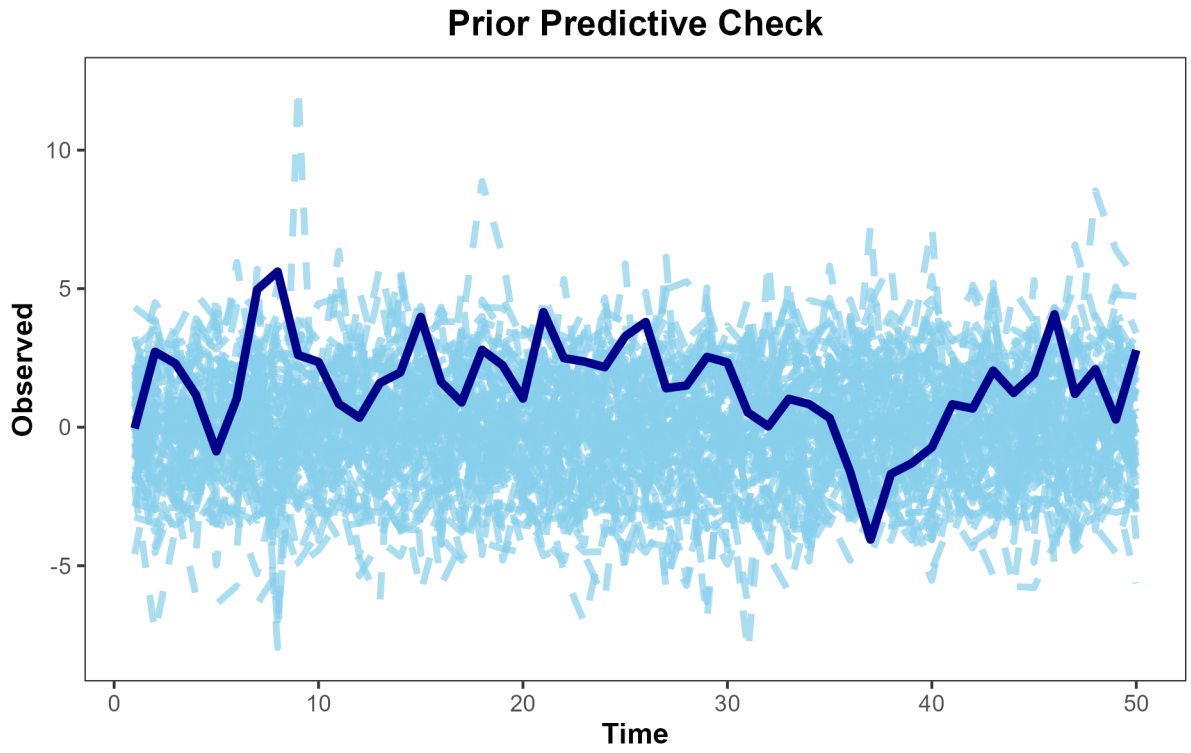

Figure 4.1: The bold dark blue line represents a simulated trajectory of the SSM in Example 4.2, while the light blue lines show 100 outbreaks simulated from the prior distribution.

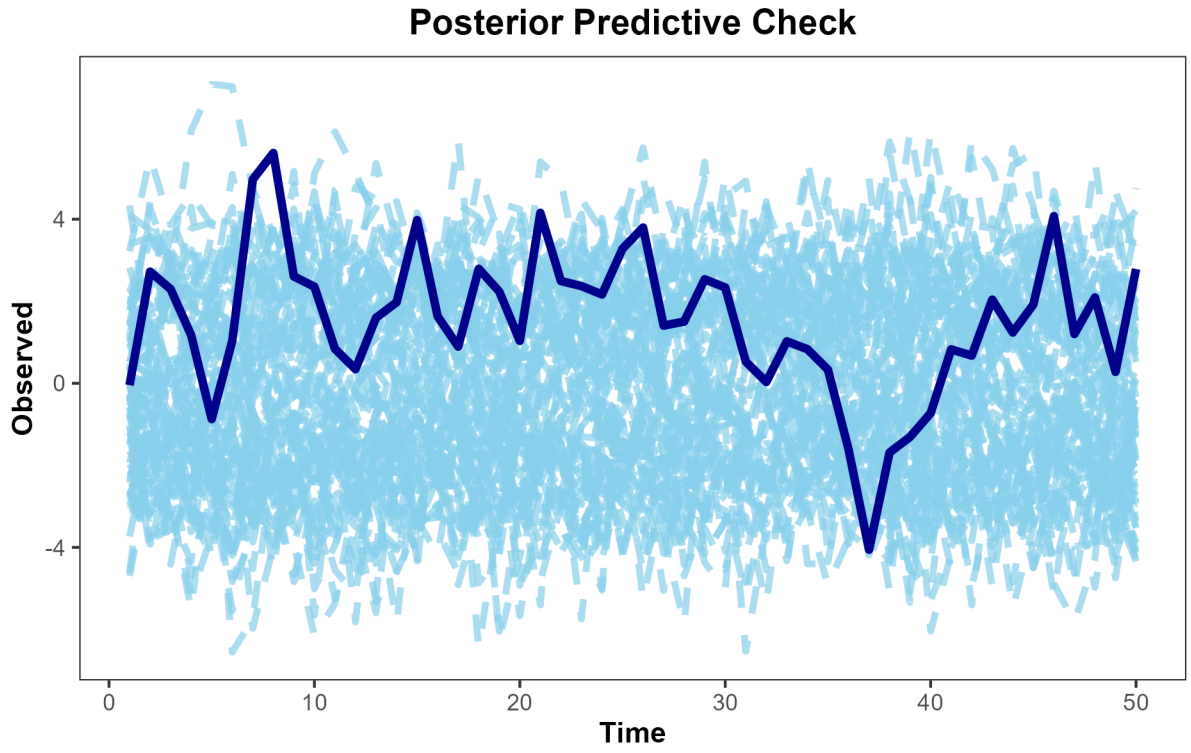

Figure 4.2: The bold dark blue line represents a simulated trajectory of the SSM in Example 4.2, while the light blue lines show 100 outbreaks simulated from the posterior distribution.

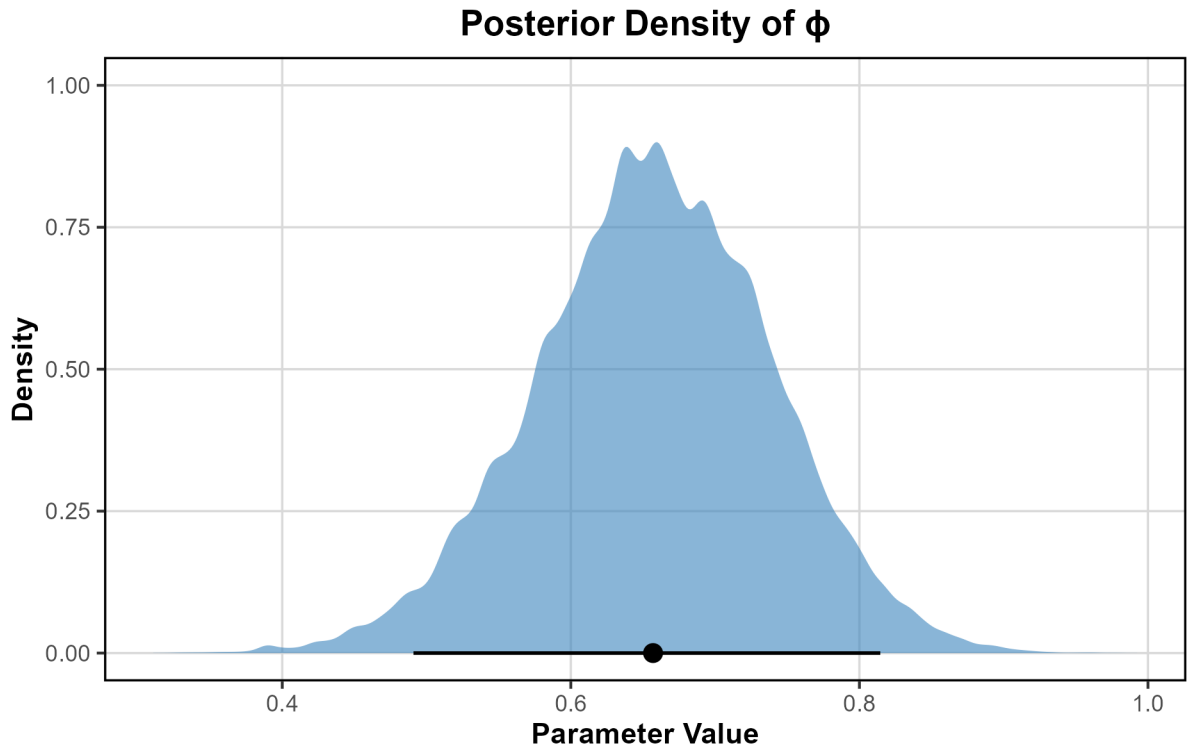

Figure 4.3: Posterior density estimate for ϕ in Example 4.2. The black dot indicates the posterior mean, and the horizontal line represents the 95% credible interval.

Parameter	True Value	ESS	split-$\widehat{R}$	Mean	95% Credible Interval
ϕ	0.7	5151	1.00	0.66	[0.49, 0.81]
σ_x	1	2565	1.00	1.28	[0.90, 1.62]
σ_y	1	432	1.00	0.37	[0.05, 0.80]

Table 4.2: Diagnostics, mean, and 95% credible interval for parameters.

The RMSE of the latent state based on this one simulated dataset is 0.80, only slightly worse than the result of the filtering and smoothing algorithms from Table 3.2, where the parameters were known.

4.2 Comparison with Alternative Methods for Intractable Likelihoods

In many practical settings, the likelihood function $p(y \mid \theta)$ is analytically or computationally intractable. This thesis has focused on performing Bayesian inference by using an unbiased estimator of the likelihood, particularly through particle filters in SSMs. Here we briefly mention some alternative methods for Bayesian inference with intractable likelihoods.

Approximate Bayesian Computation (ABC)

ABC methods avoid explicit likelihood evaluation by comparing observed data to simulated data through summary statistics. The simplest rejection ABC algorithm proceeds as:

1. Sample $\theta \sim \pi(\theta)$.
2. Simulate $x \sim p(x \mid \theta)$.
3. Accept θ if $\rho(S(x), S(y)) < \epsilon$,

where $S(\cdot)$ is a summary statistic, ρ is a discrepancy measure (not necessarily a metric), and $\epsilon > 0$ is a tolerance. The resulting approximate posterior is

$$\pi_\epsilon(\theta \mid y) \propto \pi(\theta) \int \mathbb{I}\big(\rho(S(x), S(y)) < \epsilon\big) p(x \mid \theta)\, dx.$$

More generally, the indicator function $\mathbb{I}(\cdot < \epsilon)$ can be replaced by a smoothing kernel K_ϵ, giving

$$\pi_\epsilon(\theta \mid y) \propto \pi(\theta) \int K_\epsilon\big(\rho(S(x), S(y))\big) p(x \mid \theta)\, dx,$$

which can reduce Monte Carlo (MC) variance.

As $\epsilon \to 0$, and if S is sufficient for θ, $\pi_\epsilon(\theta \mid y)$ converges to the true posterior. However, for fixed $\epsilon > 0$ or non-sufficient S, ABC provides only an approximate posterior due to information loss and tolerance bias [Beaumont, 2010, Marin et al., 2012, Sunnåker et al., 2013]. Thus, the performance of ABC depends strongly on the choice of summary statistics S, discrepancy ρ, and tolerance ϵ.

Synthetic Likelihood

The synthetic likelihood approach, introduced by Wood [2010], assumes that the summary statistics $S(x) \mid \theta$ are approximately multivariate Gaussian

$$S(x) \mid \theta \sim \mathcal{N}(\mu_\theta, \Sigma_\theta).$$

The synthetic likelihood at parameter θ is estimated by simulating m independent datasets $x^{(1)}, \ldots, x^{(m)} \sim p(x \mid \theta)$, computing the sample mean $\hat{\mu}_\theta$ and covariance $\hat{\Sigma}_\theta$ of the summaries, and evaluating the Gaussian density at the observed summaries $S(y)$:

$$\mathcal{L}_{\mathrm{SL}}(\theta) \approx \mathcal{N}(S(y); \hat{\mu}_\theta, \hat{\Sigma}_\theta).$$

This plug-in estimator of the synthetic likelihood can then be used within e.g., a Metropolis–Hastings framework to perform Bayesian inference.

The synthetic likelihood approach is generally more stable and easier to tune than ABC because it avoids choosing a discrepancy metric and tolerance. However, it relies critically on the Gaussianity assumption of the summaries, which may not hold in all models, potentially leading to biased or overconfident inference [Wood, 2010, Price et al., 2018]. Furthermore, the computational cost per likelihood evaluation is higher than ABC, since each θ requires simulating m datasets.

Summary

In summary, Particle Markov chain Monte Carlo (PMCMC) provides the most principled and exact Bayesian inference when computational resources allow, as it uses unbiased estimates of the likelihood within a MCMC scheme. In contrast, ABC and synthetic likelihood methods provide approximate posteriors but are generally computationally cheaper, as they avoid explicit likelihood estimation. ABC relies heavily on summary statistics and tolerance choices, while synthetic likelihood trades this for an approximate Gaussian likelihood assumption and the cost of repeated simulations.

5 Software implementation

5.1 Overview and Motivation

The `R` package `bayesSSM` [Hautop, 2025], developed as part of this thesis, provides a lightweight and user-friendly toolkit for performing Bayesian inference in state-space models using particle methods. The most widely used `R` packages for Bayesian inference in state-space models are `pomp` [King et al., 2016] and `NIMBLE` [de Valpine et al., 2017]. These are generally considered the primary tools in the field [Endo et al., 2019]. However, their extensive feature sets and high degree of flexibility can pose a steep learning curve for new users.

In contrast, `bayesSSM` offers a streamlined, accessible interface, enabling users to get started quickly with minimal overhead. `bayesSSM` automatically tunes the number of particles and the proposal distribution using a pilot run as described in Chapter 4. As opposed to this, `pomp` does not support automatic tuning for the number of particles, and `NIMBLE` only provides an experimental feature for tuning the number of particles. It is important to note, however, that `bayesSSM` is implemented purely in `R`, whereas both `pomp` and `NIMBLE` rely on `C++` for core computations. As a result, `bayesSSM` is generally slower than the alternatives.

All functionality in the package was implemented from scratch using what was described in Chapter 3 and Chapter 4. Every code example using a particle filter or Particle Marginal Metropolis-Hastings (PMMH) was done using `bayesSSM`. The package adheres to best practices outlined in Wickham and Bryan [2025], including continuous integration and deployment, ensuring a robust development cycle.

5.2 Core Functionality

The package exports several functions, with the main one being `pmmh()`. The function `pmmh()` implements the PMMH algorithm. This function automatically tunes the number of particles and proposal covariance, as described in the previous chapter. Its signature is:

```
pmmh(
  y,
  m,
  init_fn,
  transition_fn,
  log_likelihood_fn,
  log_priors,
  pilot_init_params,
  burn_in,
  ... # Additional optional arguments
```

```
)
```

Listing 5.1: Function signature of `pmmh()`.

We briefly describe the required arguments and refer to the package documentation for more details.

`y` A numeric vector or matrix of observations.

`m` The total number of Markov chain Monte Carlo (MCMC) iterations per chain.

`init_fn` Function to initialize particle states. Must accept `num_particles` and return a vector or matrix of initial states. Model parameters can be passed as additional named arguments.

`transition_fn` State–transition function. Must accept the current `particles` and return propagated particles. Model parameters can be passed as additional named arguments. The function can optionally depend on time by including a time step argument `t`.

`log_likelihood_fn` A function that computes the log-likelihoods for the particles. Must accept `y` (current observation), `particles`, and return a vector of log-likelihoods. Model parameters can be passed as additional named arguments. The function can optionally depend on time by including a time step argument `t`.

`log_priors` A named list of functions mapping each parameter to its log-prior.

`pilot_init_params` A list of named vectors, one per chain, specifying initial parameter values for the pilot chains.

`burn_in` The number of samples to discard as burn-in.

5.3 Example usage

We now provide a minimal example for performing Bayesian Inference for the following state-space model (SSM)

$$\begin{aligned} X_0 &\sim N(0,1), \\ X_t &= \phi X_{t-1} + \sin(X_{t-1}) + \sigma_x V_t, \quad V_t \sim N(0,1), \quad t \geq 1, \\ Y_t &= \cos(X_t) + \sigma_y W_t, \quad W_t \sim N(0,1), \quad t \geq 1, \end{aligned}$$

with the following priors

$$\begin{aligned} \phi &\sim \text{Uniform}(0,1), \\ \sigma_x &\sim \text{Exp}(1), \\ \sigma_y &\sim \text{Exp}(1). \end{aligned}$$

This can be accomplished with the `pmmh()` function, as demonstrated in Listing 5.2.

```r
library(bayesSSM)
set.seed(1405)

init_fn <- function(num_particles) {
  rnorm(num_particles, mean = 0, sd = 1)
}
transition_fn <- function(particles, phi, sigma_x) {
  phi * particles + sin(particles)
  + rnorm(length(particles), 0, sigma_x)
}
log_likelihood_fn <- function(y, particles, sigma_y) {
  dnorm(y, mean = cos(particles), sd = sigma_y, log = TRUE)
}

phi_prior <- function(phi) {
  dunif(phi, 0, 1, log = TRUE)
}
sigma_x_prior <- function(sigma_x) {
  dexp(sigma_x, 1, log = TRUE)
}
sigma_y_prior <- function(sigma_y) {
  dexp(sigma_y, 1, log = TRUE)
}

log_priors <- list(
  phi = phi_prior,
  sigma_x = sigma_x_prior,
  sigma_y = sigma_y_prior
)

pilot_init_params <- list(
  c(phi = 0.8, sigma_x = 1, sigma_y = 0.5),
  c(phi = 0.4, sigma_x = 0.5, sigma_y = 1.0)
)

y <- rnorm(50) # Dummy data

pmmh_result <- pmmh(
  y = y,
  m = 1000,
  init_fn = init_fn,
  transition_fn = transition_fn,
  log_likelihood_fn = log_likelihood_fn,
  log_priors = log_priors,
  pilot_init_params = pilot_init_params,
  burn_in = 100,
  num_chains = 2
)
```

Listing 5.2: R code for PMMH setup.

The `pmmh()` function automatically tunes the number of particles via a pilot run as described in Chapter 4. The output of running Listing 5.2 is shown in Listing 5.3. It includes progress messages, parameter summaries, and warnings if applicable.

```
Running chain 1...
Running pilot chain for tuning...
Using 50 particles for PMMH:
Running Particle MCMC chain with tuned settings...
Running chain 2...
Running pilot chain for tuning...
Using 50 particles for PMMH:
Running Particle MCMC chain with tuned settings...

PMMH Results Summary:
Parameter  Mean   SD   Median   2.5%  97.5%   ESS   Rhat
phi        0.54   0.30 0.57     0.03  0.98    82    1.001
sigma_x    2.10   0.67 1.94     1.31  4.08    48    1.026
sigma_y    0.70   0.12 0.71     0.48  0.95    65    1.053

Warning messages:
1: In pmmh(y = y, m = 1000, init_fn = init_fn, transition_fn =
   transition_fn,  :
Some ESS values are below 400, indicating poor mixing.
Consider running the chains for more iterations.
2: In pmmh(y = y, m = 1000, init_fn = init_fn, transition_fn =
   transition_fn,  :
Some Rhat values are above 1.01, indicating that the chains have
   not converged.
Consider running the chains for more iterations and/or increase
   burn_in.
```

Listing 5.3: Output of running Listing 5.2.

6 Stochastic Modeling of Disease Outbreaks

In this chapter, we introduce the theoretical framework for stochastic modeling of disease outbreaks, following the approach in Andersson and Britton [2000].

Most of the literature on epidemic modeling describes the system using either ordinary differential equations (ODEs) (e.g., Chen et al. [2020], Giraldo and Palacio [2008], Carcione et al. [2020]) or stochastic differential equations (SDEs) (e.g., Dureau et al. [2013], Liu et al. [2020]). These models treat the states as continuous and assume that the system evolves according to smooth dynamics, which are deterministic in the case of ODEs, and include continuous stochastic perturbations in the case of SDEs.

In contrast, our approach models the epidemic process as a series of events, where the time between events (such as infections or recoveries) is exponentially distributed. By performing inference on the rate parameters using Monte Carlo (MC) simulations, we capture the inherent randomness in the transmission process more accurately, especially in scenarios involving small populations or the early stages of an outbreak.

Given the often limited number of observations and the wealth of prior knowledge about similar diseases, a Bayesian framework is particularly well-suited for modeling disease outbreaks.

6.1 SIR Model

We consider a *closed* population of size N, where births, natural deaths, and migrations are neglected. Initially, every individual (who is not infected) is assumed to be *susceptible*. Once an individual becomes *infected*, they enter an infectious state during which they may transmit the disease. After the infectious period, the individual *recovers* (or is removed) and no longer contributes to disease transmission. Hence, this model comprises three compartments and is commonly referred to as a Susceptible-Infectious-Recovered (SIR) model.

At time $t = 0$, suppose there are i infectious individuals (who have just been infected), s susceptible individuals, and $r = 0$ recovered. We model the duration of the infectious period as a non-negative random variable I with finite mean μ and variance σ^2. We assume that the infectious periods are i.i.d. across individuals.

Suppose that during an individual's infectious period, contacts with any given member of the population occur according to a time-homogeneous Poisson process with rate λ/N, where $\lambda > 0$. Thus, each infectious individual makes contact with any other individual at an overall rate of λ, independent of the population size. When a susceptible individual is contacted, they become infected instantaneously and subsequently follow the same

infectious process. The epidemic terminates once there are no infectious individuals remaining.

Some key quantities of interest are:

- The *final size* of the epidemic, Z, is defined as the total number of individuals that eventually become infected.
- The *effective reproduction number*. We define the basic reproduction number by

$$R_0 = \lambda\mu,$$

  which represents the expected number of secondary infections produced by a single infectious individual in an entirely susceptible population. The effective reproduction number is then defined as

$$R_e = R_0\frac{s}{N},$$

  which represents the expected number of secondary infections produced by an infectious individual when only s individuals are susceptible. An epidemic is likely to occur if $R_e > 1$, whereas it is likely to die out quickly if $R_e < 1$.

Since the population size N is fixed it is sufficient to only track the number of susceptible and the number of infected. Let $S(t)$ denote the number of susceptible individuals, $I(t)$ denote the number of infectious individuals, and $R(t)$ the number of recovered individuals at time $t \geq 0$. Since population size N is assumed fixed, we have

$$N = S(t) + I(t) + R(t),$$

and it is sufficient to only keep track of $S(t)$ and $I(t)$ through time.

Markov Process

To integrate this model into our framework for state-space models (SSMs), we require the process

$$\{\big(S(t),\, I(t)\big) : t \geq 0\}$$

to be a Markov process. For it to be a Markov process the infection and removal events must be *memoryless*—a property unique to the exponential distribution among continuous distributions. The infection event is inherently memoryless, as it corresponds to the time until the next event in a Poisson process. For the removal process, we further assume that

$$I \sim \text{Exp}(\gamma),$$

which implies that the mean infectious period is $\mu = 1/\gamma$ and, consequently, the basic reproduction number becomes

$$R_0 = \frac{\lambda}{\gamma}.$$

Under these assumptions, the Markov process $\big(S(t),\, I(t)\big)$ is characterized by the following transition rates:

- **Infection Event:** The transition
$$(s, i) \to (s - 1, i + 1)$$
occurs at rate
$$\frac{\lambda}{N} s\, i,$$
for $s > 0$ and $i > 0$.
- **Removal Event:** The transition
$$(s, i) \to (s, i - 1)$$
occurs at rate
$$\gamma\, i,$$
for $i > 0$.

Thus, the time until the next event (of any type) follows an exponential distribution with rate
$$\frac{\lambda}{N} si + \gamma i.$$
Conditional on an event occurring, the probability that it is an infection event is given by
$$\frac{\lambda/N \cdot si}{\lambda/N \cdot si + \gamma i},$$
with the complementary probability corresponding to a removal event.

Partial Observations

In practice, epidemic data are often collected only in aggregated form; for example, we might only know the initial state, $(I(0), S(0))$ and either the number of infectious individuals, $I(t)$, at each observation time $t = 1, \ldots, T$ for some $T \geq 1$, or the daily number of new infections. Since individual infection and recovery events are not recorded, the corresponding number of susceptible individuals, $S(t)$, remains unobserved and is treated as a latent variable.

Noisy Measurements

In many real-world scenarios, the observed data are affected by measurement errors. The true number of infectious individuals, $I(t)$, might be misrepresented due to factors such as limited testing capacity or diagnostic inaccuracies, leading to observed counts $Y(t)$ that differ from the true values.

To capture this uncertainty, the observation process can be modeled probabilistically. One common approach is to assume a Poisson measurement model:
$$Y(t) \mid I(t), r \sim \text{Poisson}\big(rI(t)\big), \quad t = 1, \ldots, T,$$
where $r > 0$ is a scaling parameter that adjusts for under- or over-reporting.

Alternatively, to allow for possible over-dispersion, a Negative Binomial model may be more appropriate:
$$Y(t) \mid I(t), \phi, r \sim \text{NegBinomial}\big(\phi,\, rI(t)\big), \quad t = 1, \ldots, T,$$
where $\phi > 0$ is the dispersion parameter, $r > 0$ is the scaling parameter, and the mean is given by $rI(t)$.

6.2 Limitations of Standard Markov Chain Monte Carlo

If one wanted to use standard Markov chain Monte Carlo (MCMC) tools in this model the exact transition probabilities of the underlying continuous-time Markov process are needed. These can be obtained through matrix exponentiation. Let Q denote the generator matrix of the Markov process, where each entry Q_{ij} corresponds to the rate of transitioning from state i to state j. The probability distribution of the system state at time t is given by:

$$P(t) = \exp(Qt),$$

where $\exp(Qt)$ denotes the matrix exponential.

Transition Probabilities for Small Populations

To illustrate this, we consider a small population of size N and construct the full state space of all possible (s, i) pairs with $s + i \leq N$. The generator matrix Q is built using the infection and removal transition rates described in Section 6.1. The diagonal entries are set to ensure that each row of Q sums to zero.

Given a state (s_t, i_t) at time t, the transition probabilities at time $t + 1$ are then obtained from the corresponding row of the matrix $\exp(Qt)$:

$$\mathbb{P}\big((S(t+1), I(t+1)) = (s, i) \mid (S(t), I(t)) = (s_t, i_t)\big) = \big[\exp(Qt)\big]_{(s_t,i_t),(s,i)}.$$

Computational Complexity

While exact, this approach scales poorly with increasing population size. The number of valid states in the (S, I) space grows quadratically with N:

$$\#\text{states} = \binom{N+2}{2}.$$

For instance, when $N = 100$, there are already over 5,000 states; for $N = 200$, this increases to over 20,000. Thus, this approach becomes infeasible even for moderate N.

This motivates the use of simulation-based approaches such as particle MCMC, which we demonstrate in the following example.

6.3 Example of Bayesian Inference in a SIR model

Example 6.1 (SIR Bayesian Inference)**.** We start by simulating some data from a SIR model. Let there at $t = 0$ be $i = 10$ infected and $s = 490$ susceptible. Let the infection rate be $\lambda = 0.5$, the removal rate be $\gamma = 0.2$, and the dispersion parameter be $\phi = 3.5$. We have observations $t = 0, \ldots, 10$ for the first 10 days of the epidemic.

We suppose that we only observe the initial state and $I(t)$ as a noisy measurement, that is we let our observations be

$$\begin{aligned} Y(0) &= \big(S(0),\, I(0)\big) \\ Y(t) \mid I(t), \phi &\sim \text{NegBinomial}\big(\phi,\, I(t)\big), \quad t = 1, \ldots, 10. \end{aligned}$$

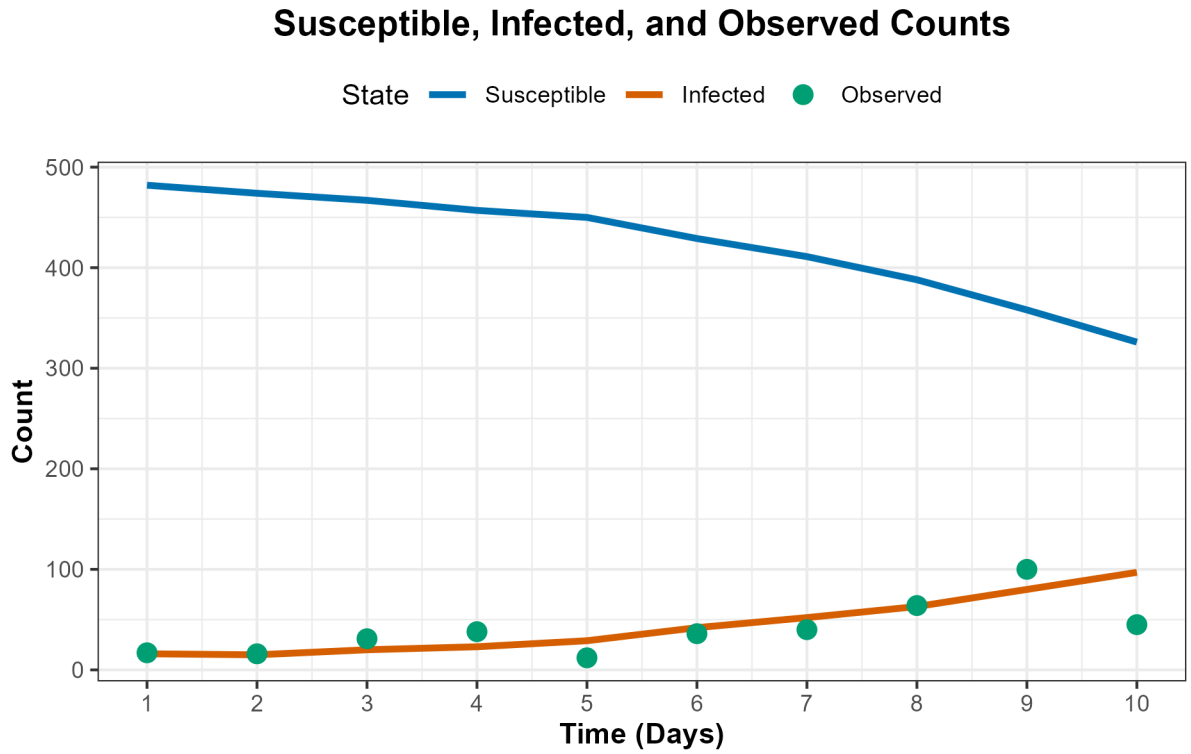

Figure 6.1: A simulated dataset of the epidemic in Example 6.1 alongside our noisy observations.

We define the priors for λ, γ and ϕ as

$$\begin{aligned} \lambda &\sim \text{Normal}^+(0, 1), \\ \gamma &\sim \text{Normal}^+(0, 1), \\ \frac{1}{\sqrt{\phi}} &\sim \text{Normal}^+(0, 1), \end{aligned}$$

where the prior for the over-dispersion parameter ϕ follows the recommendation in Stan Development Team [2025b], which also gives a lot of mass on low amounts of over-dispersion. Our simulated data set is shown in Figure 6.1. A prior predictive check is shown in Figure 6.2. The prior predictive check indicates that the chosen priors lead to reasonable and plausible simulated data, consistent with the observed patterns in the data. We run the Particle Marginal Metropolis-Hastings (PMMH) algorithm using the standard settings given in Table 4.1. The `R` code for this analysis is available in the accompanying GitHub repository.

Figure 6.3 shows a posterior predictive check for the observed number of infected, verifying that the posterior distribution generates data consistent with the observations. Table 6.1 presents the estimated parameters, including their posterior means, 95% credible intervals and diagnostic measures. The MCMC Effective Sample Size (MCMC ESS) are all above the recommendation of 400 and the split-$\widehat{R}$ are all below 1.01, so we can reliably use the samples for inference. As shown in the table, the model accurately recovers the true parameter values of λ and γ used to generate the simulated data.

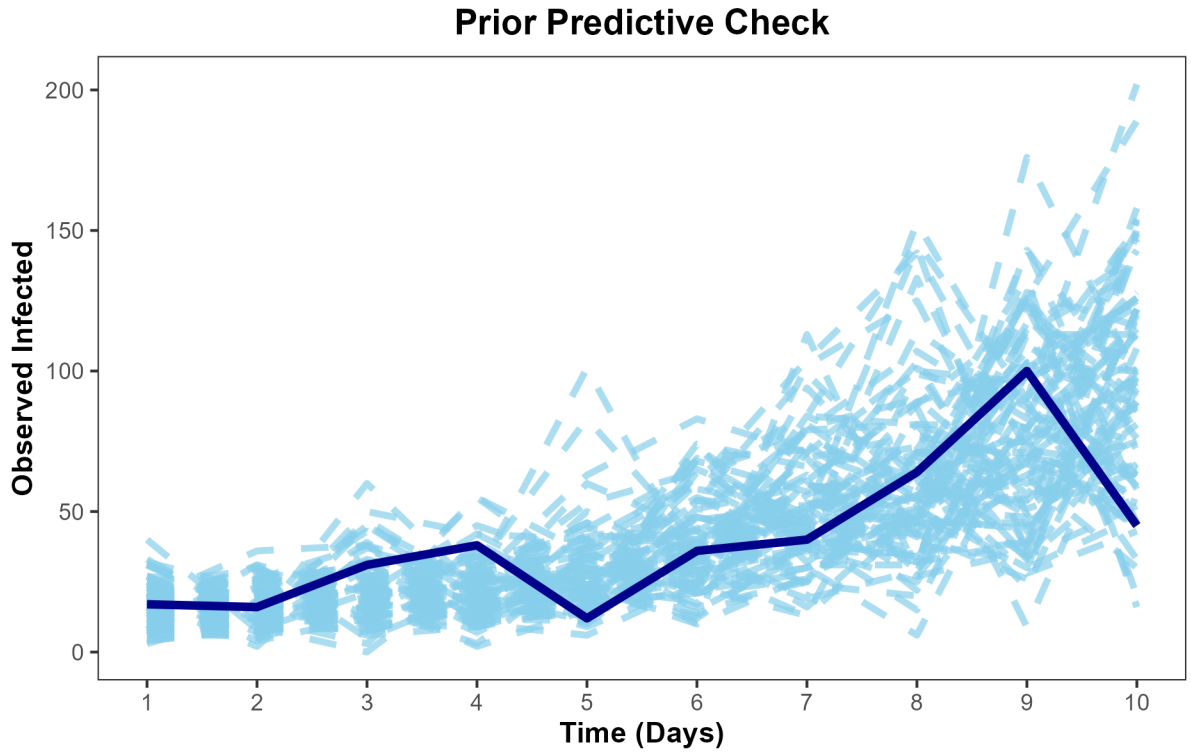

Figure 6.2: The bold dark blue line represents the observed outbreak for Example 6.1, while the light blue lines show 100 outbreaks simulated from the prior distribution.

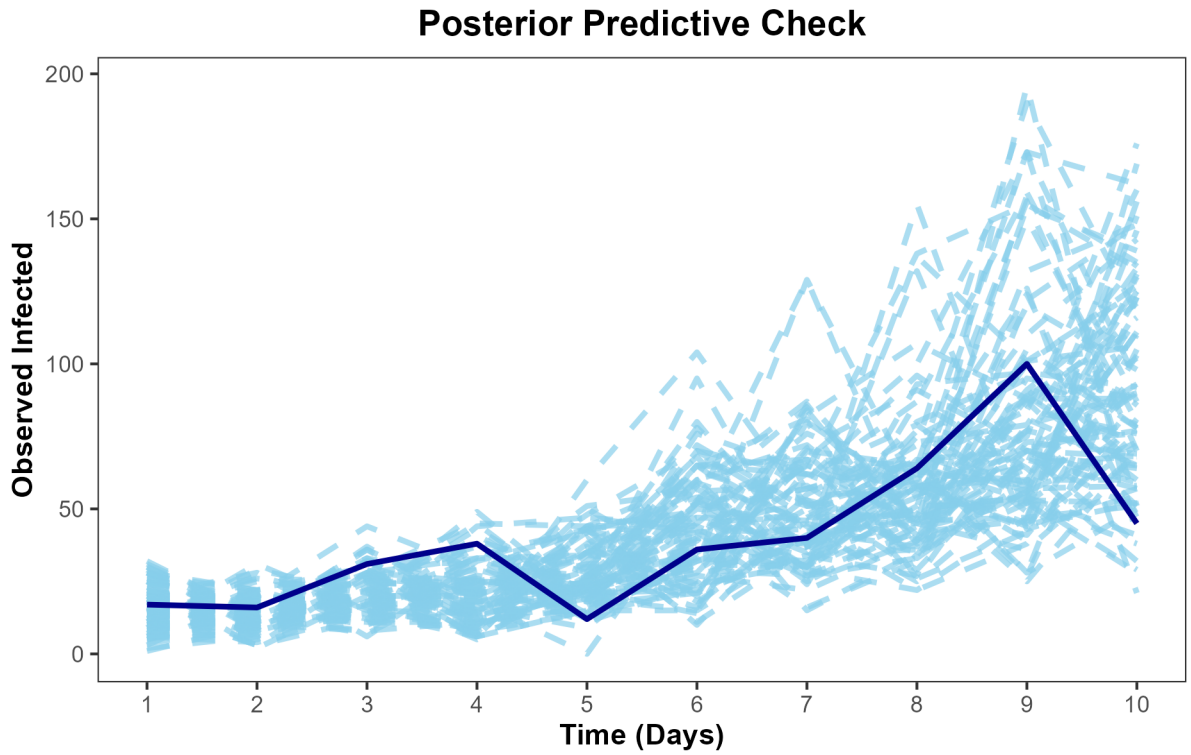

Figure 6.3: The bold dark blue line represents the observed outbreak for Example 6.1, while the light blue lines show 100 outbreaks simulated from the posterior distribution.

Parameter	True Value	ESS	split-$\widehat{R}$	Mean	95% Credible Interval
λ	0.5	919	1.00	0.61	[0.26, 1.07]
γ	0.2	751	1.00	0.29	[0.02, 0.60]

Table 6.1: Diagnostics, posterior mean, and 95% credible intervals for parameters.

6.4 Comparison to ODE

In this section, we provide a simple example to demonstrate how the ODE approach can differ substantially from our stochastic approach in small populations. The ODE model, derived by considering the system's average behavior over time [Hethcote, 2000], is given by the following set of differential equations:

$$\begin{aligned}\frac{dS}{dt} &= -\frac{\lambda}{N}SI,\\ \frac{dI}{dt} &= \frac{\lambda}{N}SI - \gamma I,\\ \frac{dR}{dt} &= \gamma I.\end{aligned}$$

This deterministic model assumes that the state variables are continuous and approximates the mean behavior of the epidemic. This model is coded in Stan [Stan Development Team, 2025a].

We initialize the system at time $t = 0$ with $i = 3$ infected individuals and $s = 47$ susceptible individuals. We use a smaller population than in Example 6.1 to highlight the differences between the stochastic and deterministic models more clearly. The infection rate is set to $\lambda = 0.5$ and the removal rate to $\gamma = 0.2$. We consider observations at times $t = 0, \ldots, 15$, corresponding to the first 15 days of the epidemic. Our observations are modeled as:

$$\begin{aligned}Y(0) &= \big(S(0), I(0)\big),\\ Y(t) \mid I(t) &\sim \text{Poisson}(I(t)), \quad t = 1, \ldots, 15,\end{aligned}$$

and the data is generated using the stochastic approach described in Section 6.1. We now want to compare the parameter estimations for λ and γ using the ODE approach and using the stochastic approach. We define generic priors for λ and γ for both approaches as

$$\begin{aligned}\lambda &\sim \text{Normal}^{+}(0,\, 1),\\ \gamma &\sim \text{Normal}^{+}(0,\, 1).\end{aligned}$$

We generate 100 datasets and compute the RMSE of the posterior mean of λ and γ. The `R` code for this analysis is available in the accompanying GitHub repository.

The results are summarized in Table 6.2. We see, that the ODE approach has a much higher RMSE than using the true data generating process (DGP) with the stochastic approach. This shows, that the inference can change substantially depending on which method is used. Note though, that the ODE approach is faster.

Parameter	ODE RMSE	Stochastic RMSE
λ	0.29	0.23
γ	0.34	0.22

Table 6.2: Comparison of ODE and stochastic model for parameter estimation. The values are the estimated RMSE.

7 Application to Epidemiological Data

In this chapter, we present Bayesian inference performed on a real disease outbreak.

7.1 Data and Model

In 1978, an influenza outbreak occurred at a boarding school in England, affecting 763 boys. The outbreak lasted 14 days. The data, originally reported in Communicable Disease Surveillance Centre and Communicable Diseases (Scotland) Unit [1978] and available in the R package `outbreaks` Jombart et al. [2022], includes daily counts of students in bed. It is believed that one student initiated the infection. The `R` code for this analysis is available in the accompanying GitHub repository.

Figure 7.1 illustrates the daily counts of students in bed during the outbreak. The goal of this analysis is to estimate the number of infected students each day and estimate key quantities such as the basic reproduction number, R_0, and mean recovery time of the disease.

A naive way to estimate the number of infected students each day would be to use the observed number of students in bed as an estimate for the number of infected students.

Similarly to Example 6.1, our model is specified as follows. At time $t = 0$, there is $i = 1$ infected student and $s = 763 - 1$ susceptible students. We have observations at times $t = 0, 1, \ldots, 14$ for the epidemic. Since we only observe students confined to bed based on symptoms, we assume that the true number of infected is a latent state. Thus, we define our model as

$$\begin{aligned} Y(0) &= \big(S(0), I(0)\big), \\ Y(t) \mid I(t), \phi &\sim \text{NegBinomial}\big(\phi,\, I(t)\big), \quad t = 1, \ldots, 14, \end{aligned}$$

where $Y(t)$ represents the observed number of students in bed and $I(t)$ is the latent number of infected students.

For the infection period, parameterized by the recovery rate γ, we specify a truncated normal prior to ensure that γ is positive. Based on reports from the CDC [Centers for Disease Control and Prevention, 2025] we decide that the infectious period should have a mean of 3 days (i.e., $1/\gamma = 3$ days). Additionally, we incorporate the belief that the probability of the mean infectious period being less than 1 day is 0.1, i.e., $P(\gamma > 1) = 0.1$. Solving for the parameters of a truncated normal distribution, we obtain

$$\gamma \sim \text{Normal}^{+}(0, 0.41^2).$$

Similarly, for the infection rate λ (the rate at which an infected individual transmits the disease to a susceptible individual), we use a truncated normal distribution. For a disease to persist, the infection rate λ should typically exceed the recovery rate γ.

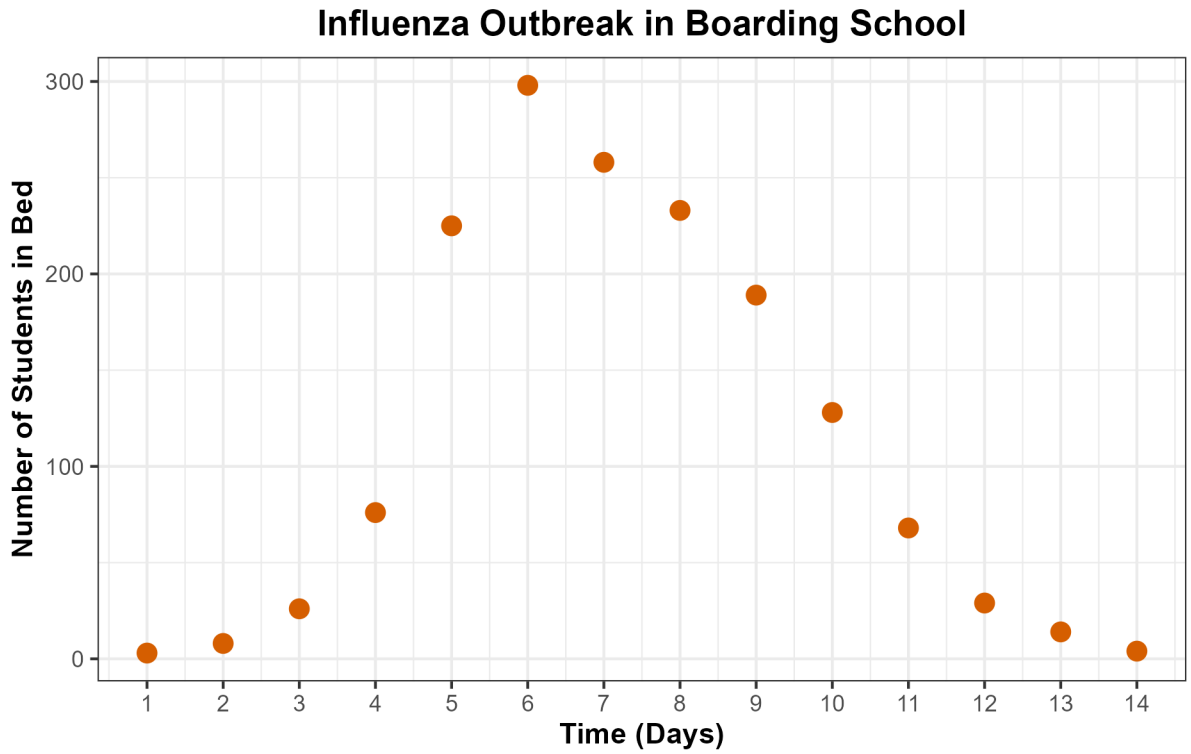

Figure 7.1: Daily counts of students in bed during the 1978 influenza outbreak at a boarding school in England.

Assuming a mostly susceptible population at the start of the epidemic, the effective transmission rate per infected individual is approximately λ, and we model the average time between new infections as $1/\lambda = 2$ days. Additionally, we incorporate the belief that the probability of the mean infection interval being shorter than 1 day is 0.2; that is, $P(1/\lambda < 1) = P(\lambda > 1) = 0.2$.

Solving for the parameters, we get

$$\lambda \sim \text{Normal}^+(0, 0.63^2).$$

For ϕ we choose a generic prior

$$\frac{1}{\sqrt{\phi}} \sim \text{Normal}^+(0, 1),$$

as recommended in Stan Development Team [2025b], which also gives a lot of mass on low amounts of over-dispersion.

7.2 Prior and Posterior Predictive Checks

We start by performing a prior predictive check based on 100 samples, which can be seen in Figure 7.2. We see that most of the outbreaks seem to peak later than in our case, but overall the priors are reasonable.

We run the Particle Marginal Metropolis-Hastings (PMMH) algorithm using the standard settings given in Table 4.1. All MCMC Effective Sample Size (MCMC ESS) are well above 400, and all split-$\widehat{R}$ values are below the threshold of 1.01, so we can reliably

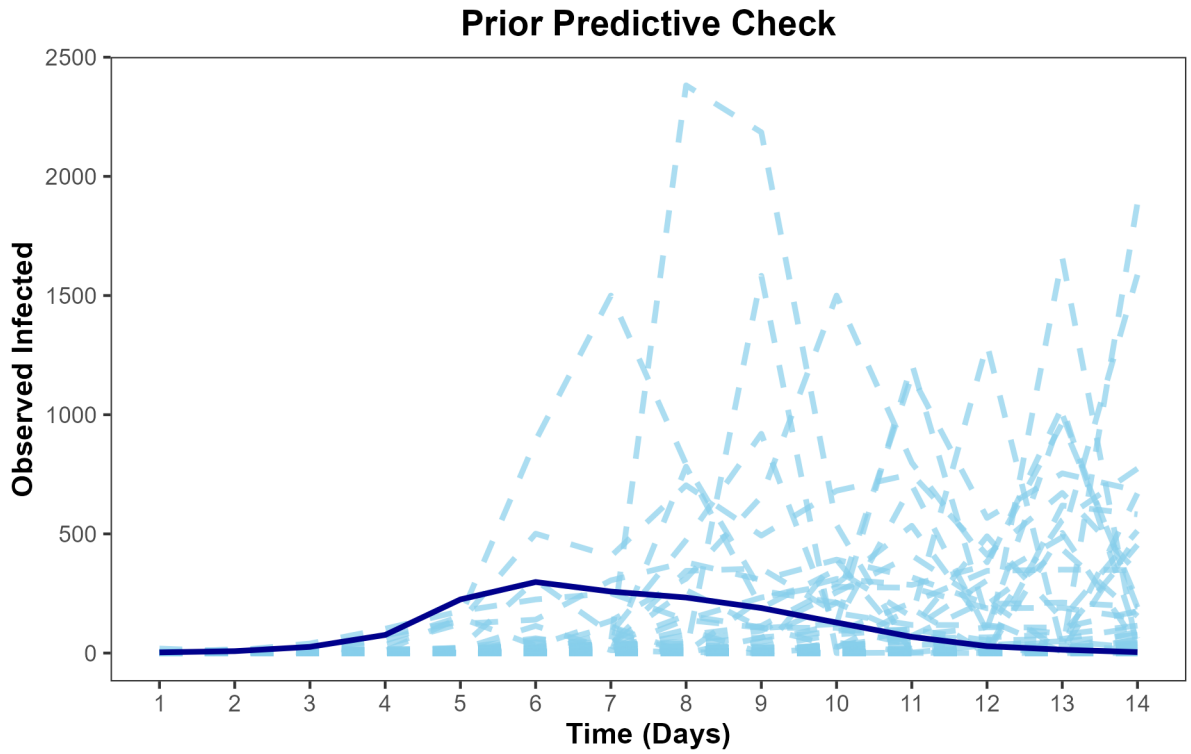

Figure 7.2: The bold dark blue line represents the observed outbreak for the 1978 influenza outbreak at a boarding school in England, while the light blue lines show 100 outbreaks simulated from the prior distribution.

use our posterior samples for inference. We start by performing a posterior predictive check, which is shown in Figure 7.3. The posterior predictive check looks good with it aligning well with the observed data.

7.3 Results

Posterior means and 95% credible intervals are shown for the Bayesian models (with informative and vague priors), while maximum likelihood estimates and 95% confidence intervals are shown for the frequentist model. Table 7.1 presents a side-by-side comparison of parameter estimates under all three approaches.

- **Informative priors (base)**: As described in Section 7.1.
- **Vague priors**: Priors for λ and γ are changed to Normal$^+(0, 1)$.
- **Frequentist**: Parameters estimated using maximum likelihood estimation (MLE).

The model with vague priors was fitted by re-weighting the samples, as outlined in Appendix B. The MLE was obtained by numerically optimizing the likelihood using Iterated Filtering 2 (IF2) [Ionides et al., 2015], which refines parameter estimates through a sequence of perturbation, filtering, and resampling steps.

To quantify uncertainty around the MLE, we used a percentile bootstrap procedure [Efron and Tibshirani, 1994]. After obtaining the MLE, we simulated 200 bootstrap

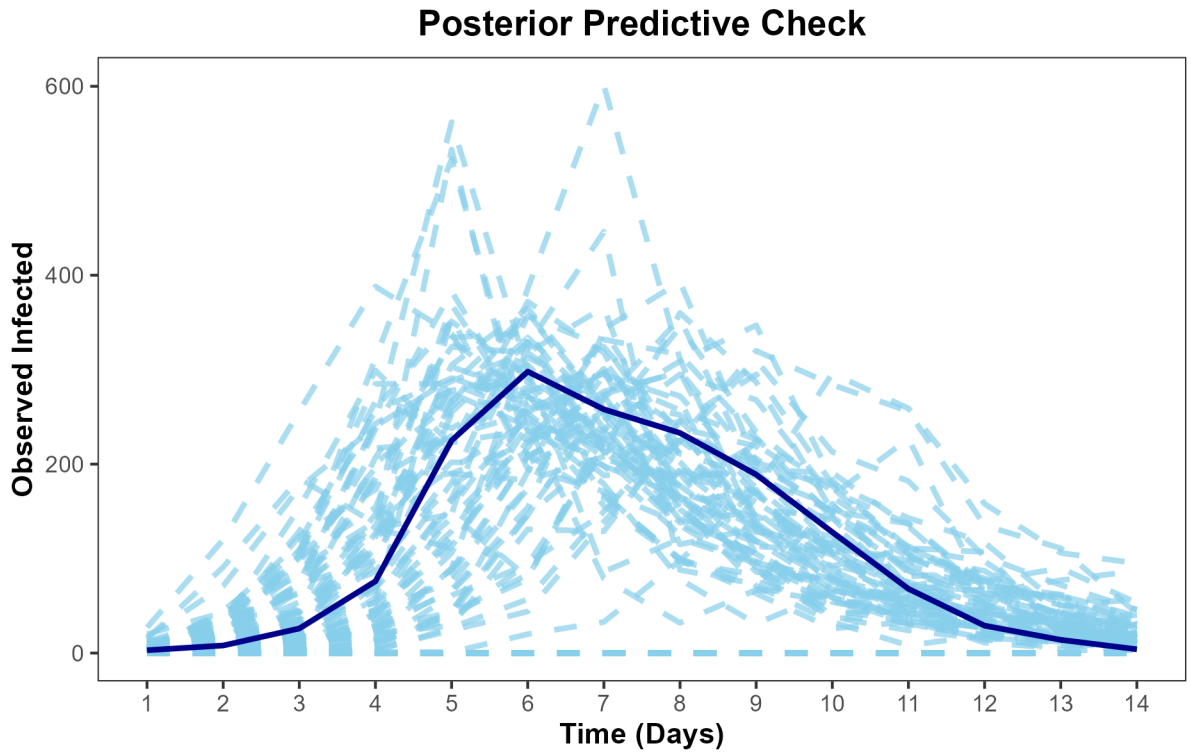

Figure 7.3: The bold dark blue line represents the observed outbreak for the 1978 influenza outbreak at a boarding school in England, while the light blue lines show 100 outbreaks simulated from the posterior distribution.

	Informative Priors		Vague Priors		Frequentist	
Parameter	Mean	95% CI	Mean	95% CI	Estimate	95% CI
λ	1.80	[1.58, 2.05]	1.84	[1.61, 2.13]	1.90	[1.61, 2.28]
γ	0.49	[0.44, 0.58]	0.50	[0.44, 0.59]	0.50	[0.45, 0.55]
R_0	3.67	[2.93, 4.46]	3.74	[2.96, 4.60]	3.80	[3.23, 4.67]
Mean Recovery Time	2.04	[1.73, 2.29]	2.03	[1.70, 2.29]	2.00	[1.80, 2.23]

Table 7.1: Comparison of parameter estimates under different methods.

datasets from the model and re-estimated parameters using IF2. Confidence intervals were constructed by taking quantiles of the resulting bootstrap distributions.

Because our model is stochastic and starts with a single infected individual, there is a nonzero probability that the outbreak will die out. Specifically, since the time to transmission and removal are exponentially distributed with rates $\lambda = 1.90$ and $\gamma = 0.50$, the probability of a removal occurring before any transmission is

$$\mathbb{P}(\text{removal before transmission}) = \frac{\gamma}{\lambda + \gamma} = \frac{0.50}{1.90 + 0.50} \approx 0.20.$$

In such cases, the likelihood surface becomes degenerate, and the MLE does not exist. Consequently, our bootstrap analysis is conditioned on realizations where the MLE exists, and the resulting frequentist confidence intervals are conditional on MLE existence.

Across all three approaches, the estimates are broadly consistent. The posterior means under both Bayesian models lie close to the MLEs, and the associated credible intervals

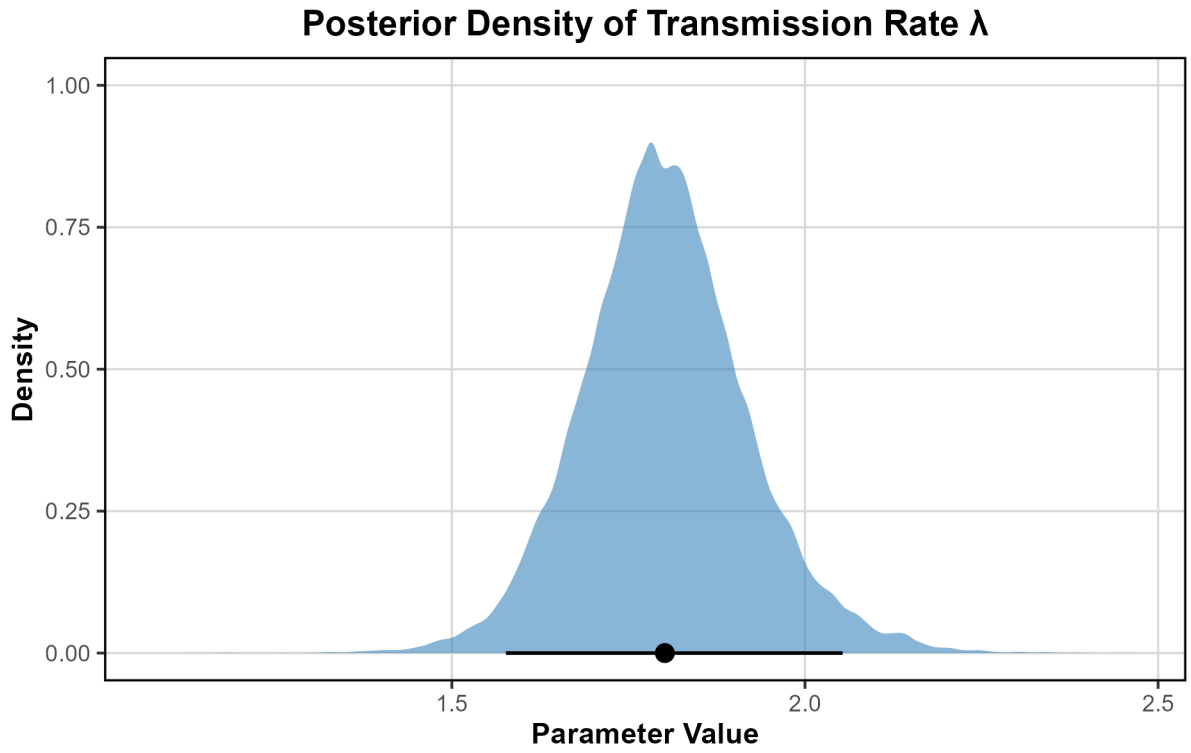

Figure 7.4: Posterior density of λ under the Bayesian model with informative priors, based on the 1978 influenza outbreak at a boarding school in England. The black dot indicates the posterior mean, and the horizontal line shows the 95% credible interval.

(or confidence intervals, in the frequentist case) show considerable overlap. This indicates that the data are sufficiently informative to yield stable estimates regardless of prior specifications.

Figure 7.4 shows the posterior density for λ under the informative prior model. Densities for the remaining parameters (γ, R_0, and mean recovery time) are presented in Appendix D (Figures D.3-D.5).

This data has been analyzed previously, for instance in Grinsztajn et al. [2021]. They employed a Bayesian approach and modeled it as an ODE, as described in Section 6.4. They got a posterior mean of λ of 1.73 and a posterior mean of γ as 0.54, roughly similar to our results.

In Figure 7.5, we see a plot of the median estimated number of infected individuals for the informative priors model, along with a 95% credible interval. At several time points, the observed number of infected lies outside this credible interval, suggesting a non-negligible amount of observation noise.

7.4 Conclusion

In this chapter, we have shown that a fully Bayesian treatment of a stochastic SIR model with a negative-binomial observation process provides a probabilistic reconstruction of the 1978 influenza outbreak at the boarding school. By modeling the true number of infected as a latent process and allowing for over-dispersion in the observations, we obtained credible intervals for the unobserved epidemic trajectory (Figure 7.5) that reveal the

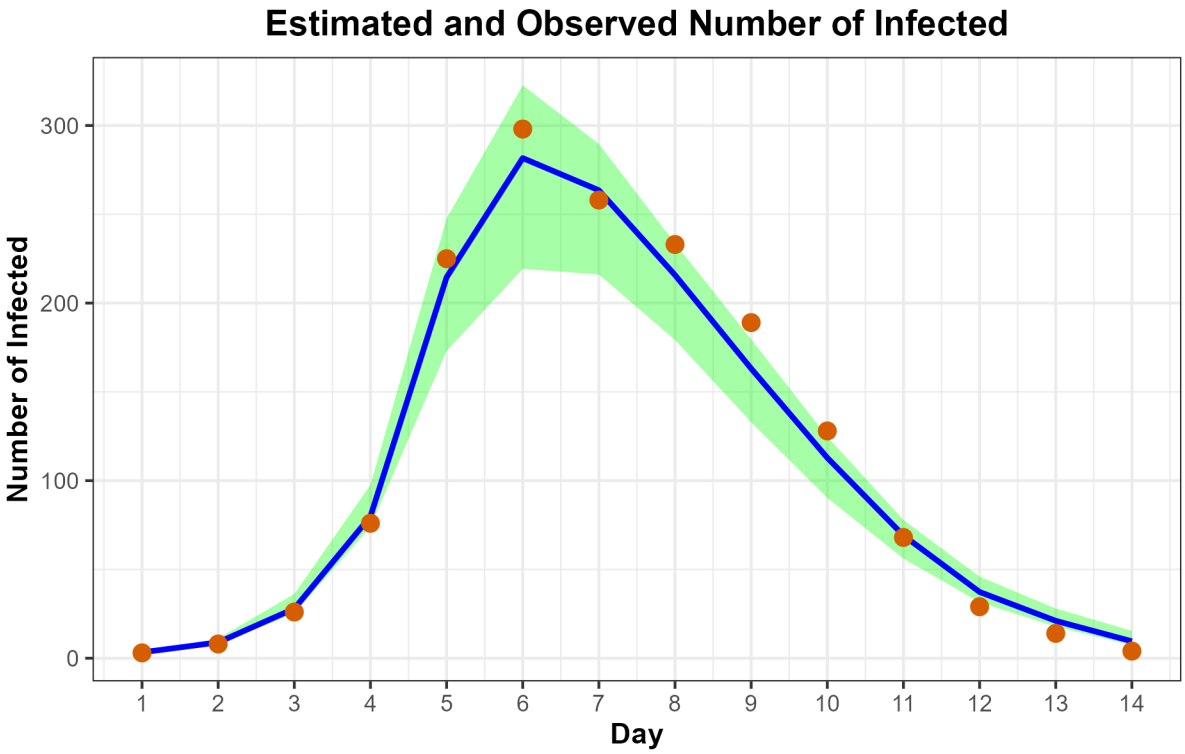

Figure 7.5: The dark orange points are the observed students in bed, the blue line is the median estimate of the number of infected, and the green ribbon is the 95% credible interval for the number of infected.

extent of observation noise, something a simple approach without latent states would miss.

Our results were fairly robust to prior specification, with both informative and vague priors yielding posterior mean estimates that closely align with those obtained using maximum likelihood methods.

Moreover, our stochastic formulation captures random fluctuations inherent in small, closed populations more naturally than deterministic ODE models, helping explain slight differences when compared to previous ODE-based Bayesian analyses such as in Grinsztajn et al. [2021]. Overall, this application demonstrates the usefulness of stochastic SIR models and illustrates how they can be effectively used and interpreted within a Bayesian framework.

8 Conclusion

This thesis has presented a comprehensive treatment of Particle Markov chain Monte Carlo (PMCMC) methods for Bayesian inference in state-space models (SSMs). Beginning with a review of fundamental Monte Carlo techniques, including importance sampling and sequential importance sampling, we highlighted key challenges such as particle degeneracy and explored resampling strategies to address them. Building upon this foundation, the formal development and theoretical properties of PMCMC algorithms were detailed, illustrating how they enable exact Bayesian inference despite relying on unbiased likelihood estimates obtained via particle filtering.

To bridge theory and practice, we introduced `bayesSSM`, an `R` package designed to perform Bayesian inference in SSMs using Particle Marginal Metropolis-Hastings (PMMH). This package includes features such as adaptive tuning of proposals, automatic selection of the number of particles, and diagnostic tools, thereby facilitating the application of PMCMC methods by practitioners. The practical utility of these methods was demonstrated through a case study applying PMCMC to a stochastic Susceptible-Infectious-Recovered (SIR) model fitted to historical influenza outbreak data from a British boarding school. This example illustrated the importance of accounting for stochasticity in small-population epidemic models and showcased the potential of PMCMC for rigorous Bayesian analysis in epidemiological research.

While the results confirm the viability and flexibility of PMCMC methods, several limitations remain. The computational demands of PMCMC can be substantial, particularly for high-dimensional models or long time series. Future research directions include extending `bayesSSM` to incorporate alternative methods for Bayesian inference beyond PMMH, as well as exploring other variations of particle filters. Further development of adaptive strategies and parallelization techniques could also enhance the scalability and efficiency of these methods.

In conclusion, this thesis contributes both theoretical insights and practical tools to advance Bayesian inference for complex dynamic systems modeled via state-space frameworks. By combining methodological rigor with accessible software, it aims to empower researchers to incorporate uncertainty and stochastic dynamics more fully in their analyses, fostering improved understanding and decision-making in applied fields.

All code and supplementary materials used throughout this thesis are openly available at: `https://github.com/BjarkeHautop/master-thesis`.

Bibliography

H. Andersson and T. Britton. *Stochastic Epidemic Models and Their Statistical Analysis*, volume 151 of *Lecture Notes in Statistics*. Springer, New York, 2000. ISBN 978-0-387-98994-5. doi: 10.1007/978-1-4612-1158-7. URL https://doi.org/10.1007/978-1-4612-1158-7.

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, 05 2010. doi: 10.1111/j.1467-9868.2009.00736.x. URL https://doi.org/10.1111/j.1467-9868.2009.00736.x.

M. A. Beaumont. Approximate bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41(Volume 41, 2010):379–406, 2010. ISSN 1545-2069. doi: https://doi.org/10.1146/annurev-ecolsys-102209-144621. URL https://www.annualreviews.org/content/journals/10.1146/annurev-ecolsys-102209-144621.

R. Bhattacharya and E. C. Waymire. *Stationary Processes and Discrete Parameter Markov Processes*. Probability and Its Applications. Springer, 2009. ISBN 978-0-387-76895-3. doi: 10.1007/978-0-387-76896-0.

Y. BRESLER. Two-filter formulae for discrete-time non-linear bayesian smoothing. *International Journal of Control*, 43(2):629–641, 1986. doi: 10.1080/00207178608933489. URL https://doi.org/10.1080/00207178608933489.

J. M. Carcione, J. E. Santos, C. Bagaini, and J. Ba. A simulation of a COVID-19 epidemic based on a deterministic SEIR model. *Front. Public Health*, 8:230, May 2020.

Centers for Disease Control and Prevention. How flu spreads, 2025. URL https://www.cdc.gov/flu/spread/index.html. Accessed: 2025-04-03.

Y.-C. Chen, P.-E. Lu, C.-S. Chang, and T.-H. Liu. A time-dependent sir model for covid-19 with undetectable infected persons. *IEEE Transactions on Network Science and Engineering*, 7(4):3279–3294, 2020. doi: 10.1109/TNSE.2020.3024723.

Communicable Disease Surveillance Centre and Communicable Diseases (Scotland) Unit. Influenza in a boarding school. *British Medical Journal*, 1:578, 1978. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1603269/pdf/brmedj00115-0064.pdf.

J. Dahlin and T. B. Schön. Getting started with particle metropolis-hastings for inference in nonlinear dynamical models. *Journal of Statistical Software, Code Snippets*, 88(2):1–41, 2019. doi: 10.18637/jss.v088.c02. URL https://www.jstatsoft.org/index.php/jss/article/view/v088c02.

P. de Valpine, D. Turek, C. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, 26: 403–417, 2017. doi: 10.1080/10618600.2016.1172487.

R. Douc, O. Cappé, and E. Moulines. Comparison of resampling schemes for particle filtering, 2005. URL https://arxiv.org/abs/cs/0507025.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12, 01 2009.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, Jul 2000. ISSN 1573-1375. doi: 10.1023/A:1008935410038. URL https://doi.org/10.1023/A:1008935410038.

J. Dureau, K. Kalogeropoulos, and M. Baguelin. Capturing the time-varying drivers of an epidemic using stochastic dynamical systems. *Biostatistics*, 14(3):541–555, 01 2013. ISSN 1465-4644. doi: 10.1093/biostatistics/kxs052. URL https://doi.org/10.1093/biostatistics/kxs052.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1st edition, 1994. doi: 10.1201/9780429246593. URL https://doi.org/10.1201/9780429246593.

A. Endo, E. van Leeuwen, and M. Baguelin. Introduction to particle markov-chain monte carlo for disease dynamics modellers. *Epidemics*, 29:100363, 2019. ISSN 1755-4365. doi: https://doi.org/10.1016/j.epidem.2019.100363. URL https://www.sciencedirect.com/science/article/pii/S1755436519300301.

A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457 – 472, 1992. doi: 10.1214/ss/1177011136. URL https://doi.org/10.1214/ss/1177011136.

A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN 9781439840955. URL https://books.google.dk/books?id=ZXL6AQAAQBAJ.

C. J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473 – 483, 1992. doi: 10.1214/ss/1177011137. URL https://doi.org/10.1214/ss/1177011137.

J. O. Giraldo and D. H. Palacio. Deterministic sir (susceptible-infected-removed) models applied to varicella outbreaks. *Epidemiology and infection*, 136(5):679–687, May 2008. ISSN 0950-2688. doi: 10.1017/S0950268807009260. URL https://www.ncbi.nlm.nih.gov/pubmed/17655783. PMC2870859.

L. Grinsztajn, E. Semenova, C. C. Margossian, and J. Riou. Bayesian workflow for disease transmission modeling in stan. *Statistics in Medicine*, 40(27):6209–6234, 2021. doi: https://doi.org/10.1002/sim.9164. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.9164.

B. Hautop. *bayesSSM: Bayesian Methods for State Space Models*, 2025. URL https://github.com/BjarkeHautop/bayesSSM. R package version 0.5.0.9004, commit 358f3b72918f2e6c15067dd23f14e5249d328fb5.

H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000. doi: 10.1137/S0036144500371907. URL https://doi.org/10.1137/S0036144500371907.

E. L. Ionides, D. Nguyen, Y. Atchadé, S. Stoev, and A. A. King. Inference for dynamic and latent variable models via iterated, perturbed bayes maps. *Proceedings of the National Academy of Sciences*, 112(3):719–724, 2015. doi: 10.1073/pnas.1410597112. URL https://www.pnas.org/doi/abs/10.1073/pnas.1410597112.

T. Jombart, S. Frost, P. Nouvellet, F. Campbell, B. Sudre, S. W. Park, J. R. Pulliam, J. Schumacher, and E. Brown. outbreaks: Dataset of infectious disease outbreaks, 2022. URL https://www.reconverse.org/outbreaks/. R package version 1.9.0.

N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin. On Particle Methods for Parameter Estimation in State-Space Models. *Statistical Science*, 30(3):328 – 351, 2015. doi: 10.1214/14-STS511. URL https://doi.org/10.1214/14-STS511.

M. Kiderlen. Survey sampling and stereology lecture notes, 2022.

A. A. King, D. Nguyen, and E. L. Ionides. Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, 69(12):1–43, 2016. doi: 10.18637/jss.v069.i12.

D. Kroese, T. Taimre, and Z. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics. Wiley, 2013. ISBN 9781118014950. URL https://books.google.dk/books?id=Trj9HQ7G8TUC.

J. S. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, 1995. doi: 10.1080/01621459.1995.10476549. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476549.

Q. Liu, D. Jiang, T. Hayat, A. Alsaedi, and B. Ahmad. A stochastic sirs epidemic model with logistic growth and general nonlinear incidence rate. *Physica A: Statistical Mechanics and its Applications*, 551:124152, 2020. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2020.124152. URL https://www.sciencedirect.com/science/article/pii/S037843712030011X.

J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, Nov 2012. ISSN 1573-1375. doi: 10.1007/s11222-011-9288-2. URL https://doi.org/10.1007/s11222-011-9288-2.

P. Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer New York, 2004. ISBN 9780387202686. URL https://books.google.dk/books?id=8LypfuG8ZLYC.

M. K. Pitt, R. d. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012. doi: \url{https://doi.org/10.1016/j.jeconom.2012.06.004}.

L. F. Price, C. C. Drovandi, A. Lee, and D. J. N. and. Bayesian synthetic likelihood. *Journal of Computational and Graphical Statistics*, 27(1):1–11, 2018. doi: 10.1080/10618600.2017.1302882. URL https://doi.org/10.1080/10618600.2017.1302882.

C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2004.

Shimao. How is this minimum variance worked out for this importance sampling estimator? Cross Validated, https://stats.stackexchange.com/q/324834, 2018. (Version: 2018-01-24).

Stan Development Team. Stan modeling language users guide and reference manual, version 2.36, 2024. URL https://mc-stan.org.

Stan Development Team. RStan: the R interface to Stan, 2025a. URL https://mc-stan.org/. R package version 2.32.7.

Stan Development Team. Prior choice recommendations, 2025b. URL https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations. Accessed: 2025-02-25.

M. Sunnåker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. Approximate bayesian computation. *PLOS Computational Biology*, 9(1):1–10, 01 2013. doi: 10.1371/journal.pcbi.1002803. URL https://doi.org/10.1371/journal.pcbi.1002803.

D. Vats and C. Knudson. Revisiting the Gelman–Rubin Diagnostic. *Statistical Science*, 36(4):518 – 529, 2021. doi: 10.1214/20-STS812. URL https://doi.org/10.1214/20-STS812.

A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667 – 718, 2021. doi: 10.1214/20-BA1221. URL https://doi.org/10.1214/20-BA1221.

A. Vehtari, D. Simpson, A. Gelman, Y. Yao, and J. Gabry. Pareto smoothed importance sampling, 2024. URL https://arxiv.org/abs/1507.02646.

H. Wickham and J. Bryan. R packages, 2025. URL https://r-pkgs.org/. Accessed: 2025-05-08.

S. N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, Aug 2010. ISSN 1476-4687. doi: 10.1038/nature09319. URL https://doi.org/10.1038/nature09319.

A MCMC

Markov chain Monte Carlo (MCMC) methods are used to generate samples from complex probability distributions when direct sampling is infeasible. Let $\pi(x)$ be a density that we want to sample from. An MCMC algorithm constructs a Markov chain $\{X_t\}_{t\geq 1}$ with transition kernel $P(x, A)$ which has $\pi(x)$ as stationary distribution.

One widely used MCMC method is the Metropolis-Hastings algorithm. It generates a sequence of samples from $\pi(x)$ by proposing a candidate x' from a proposal distribution $q(x'|x_t)$ based on the current state x_t. The candidate is accepted with probability:

$$\alpha = \min\left\{1,\ \frac{\pi(x')q(x_t|x')}{\pi(x_t)q(x'|x_t)}\right\}$$

If the candidate is accepted, the next state is set to $x_{t+1} = x'$; otherwise, $x_{t+1} = x_t$.

A key property ensuring that the chain has the desired stationary distribution is that the algorithm satisfies the detailed balance condition with respect to $\pi(x)$, that is

$$\pi(x_t)q(x' \mid x_t)\alpha(x_t, x') = \pi(x')q(x_t \mid x')\alpha(x', x_t),$$

for all states x_t and x'. Provided that the Markov Chain satisfies some required regularity conditions on the proposal distribution it has the correct stationary distribution. A sufficient condition is that

$$\pi(x) > 0 \implies q(x \mid x') > 0 \quad \text{for any } x, x',$$

ensuring irreducibility, and that the chain is aperiodic [Robert and Casella, 2004].

In practice, when using MCMC methods for inference the first part of the chain is discarded to allow the chain to converge to the stationary distribution. This is referred to as *burn-in*.

MCMC diagnostics

Since convergence of MCMC is only guaranteed asymptotically, we must rely on diagnostic methods to assess convergence when working with a finite number of samples. Assume that we do MCMC to do inference about a parameter θ, which, for ease of notation, we suppose is a scalar. For models involving multiple parameters, the diagnostic methods described below are applied to each parameter individually.

Potential Scale Reduction

A method to evaluate the convergence of a MCMC chain is to run several independent chains and compare the behavior between them. This is the idea of the *potential scale*

reduction statistic $\widehat{R}$ first introduced in Gelman and Rubin [1992]. The potential scale reduction statistic compares the between-chain variance to the within-chain variance. If all chains are at equilibrium, these will be the same, and $\widehat{R}$ will be 1. If they haven't converged to a common distribution, the $\widehat{R}$ statistic will be greater than 1.

Suppose we have a set of K Markov chains θ_k which each has M samples $\theta_k^{(m)}$. The between-chain variance estimate is

$$B = \frac{M}{K-1}\sum_{k=1}^{K}\left(\overline{\theta}_k^{(\cdot)} - \overline{\theta}_\cdot^{(\cdot)}\right)^2,$$

where $\overline{\theta}_k^{(\cdot)}$ is the mean for chain k

$$\overline{\theta}_k^{(\cdot)} = \frac{1}{M}\sum_{m=1}^{M}\theta_k^{(m)},$$

and $\overline{\theta}_\cdot^{(\cdot)}$ is the overall mean of the chains

$$\overline{\theta}_\cdot^{(\cdot)} = \frac{1}{K}\sum_{k=1}^{K}\overline{\theta}_k^{(\cdot)}.$$

The within-chain variance is averaged over the chains,

$$W = \frac{1}{K}\sum_{k=1}^{K}s_k^2,$$

where

$$s_k^2 = \frac{1}{M-1}\sum_{m=1}^{M}\left(\theta_k^{(m)} - \overline{\theta}_k^{(\cdot)}\right)^2.$$

The variance estimator is given by a mixture of the within-chain and cross-chain sample variances,

$$\widehat{\mathrm{Var}}^{+}(\theta) = \frac{M-1}{M}W + \frac{1}{M}B, \tag{A.1}$$

This weighted combination accounts for the uncertainty in both the within-chain and between-chain variances. Finally, we can define the potential scale reduction statistic as

$$\widehat{R} = \sqrt{\frac{\widehat{\mathrm{Var}}^{+}(\theta)}{W}}.$$

Gelman et al. [2013] introduced split-$\widehat{R}$ as a more sensitive diagnostic for convergence in MCMC sampling. The method involves splitting each of the K chains into two halves: the first $M/2$ samples and the last $M/2$ samples, giving $2K$ chains. The standard $\widehat{R}$ statistic is then computed across these $2K$ chains. By comparing the two halves of each chain, split-$\widehat{R}$ can detect issues such as slow mixing or nonstationarity within individual chains that might be overlooked when only comparing different chains.

A typical guideline is that $\widehat{R}$ values above 1.01 indicates that further sampling is needed [Stan Development Team, 2024, Vehtari et al., 2021].

Effective Sample Size

An $\widehat{R}$ close to 1 does not guarantee that an MCMC sample is reliable [Vats and Knudson, 2021]. A sufficiently large MCMC Effective Sample Size (MCMC ESS) is also required to obtain stable inferences for the quantities of interest. The MCMC samples will typically be positively autocorrelated within a chain. The MCMC ESS is the number of independent samples with the same estimation power as the M autocorrelated samples. We follow the definitions given in Stan Development Team [2024], Vehtari et al. [2021]. We again let K be the number of chains each consisting of M samples and assume all the chains have reached the stationary distribution $p(\theta)$ with mean μ and variance σ^2. The autocorrelation ρ_t at lag $t \geq 0$ is defined as

$$\begin{aligned}\rho_t &= \frac{1}{\sigma^2}\int (\theta^{(n)} - \mu)(\theta^{(n+t)} - \mu)p(\theta)\, d\theta \\ &= \frac{1}{\sigma^2}\int \theta^{(n)}\theta^{(n+t)}p(\theta)\, d\theta,\end{aligned}$$

where we used that $\theta^{(n)}$ and $\theta^{(n+t)}$ have the same stationary distribution. We then define the MCMC ESS M_{eff} of M samples by

$$M_{\text{eff}} = \frac{M}{1 + 2\sum_{t=1}^{\infty}\rho_t}.$$

For independent draws (i.e., $\rho_t = 0$ for $t \geq 1$), we recover $M_{\text{eff}} = M$. When draws are positively correlated, however, M_{eff} is smaller, reflecting the reduced information content of the chain, while if they were negatively correlated, the effective sample size will exceed the number of iterations. The MCMC ESS is a particularly important measure since the standard error of the estimate of a parameter decreases by $1/\sqrt{M_{\text{eff}}}$ and not $1/\sqrt{M}$.

In practice, the integral of the joint distribution $p(\theta)$ is intractable and thus we need to estimate the effective sample size. We can estimate the autocorrelation ρ_t by

$$\hat{\rho}_t = 1 - \frac{W - \frac{1}{K}\sum_{k=1}^{K} s_k^2\,\hat{\rho}_{t,k}}{\widehat{\text{Var}}^{+}(\theta)},$$

where $\hat{\rho}_{t,k}$ is an estimate of the autocorrelation at lag t for the kth Markov chain, and $\widehat{\text{Var}}^{+}(\theta)$ is defined in Equation (A.1). Because of the increased noise of $\hat{\rho}_t$ as t increases, in practice a truncated sum of $\hat{\rho}_t$ is used. We apply Geyer's initial monotone sequence criterion, as defined in Geyer [1992], to ensure stability. The effective sample size is estimated by

$$\widehat{M}_{\text{eff}} = \frac{KM}{\hat{\tau}},$$

where

$$\hat{\tau} = 1 + 2\sum_{t=1}^{2k+1}\hat{\rho}_t = 1 + 2\sum_{t=0}^{m}\hat{P}_t - \hat{\rho}_0 = -1 + 2\sum_{t=0}^{m}\hat{P}_t,$$

and $\hat{P}_t = \hat{\rho}_{2t} + \hat{\rho}_{2t+1}$. Summing over pairs starting from lag 0 ensures that the sequence $\hat{P}_t$ values is non-negative and non-increasing [Geyer, 1992]. So, if we observe negative estimates of the autocorrelations it is due to finite-sample noise. Thus, we define an initial positive sequence by choosing the largest m such that $\hat{P}_t > 0$ for all $t \in \{1, \ldots, m\}$. We also enforce monotonicity by modifying the sequence $\hat{P}_t$ so that it does not exceed the smallest preceding value, ensuring a non-increasing sequence.

B Bayesian Model Assessment

This chapter gives a quick introduction to some common tools used for assessing a Bayesian model.

Prior predictive check

Before fitting a Bayesian model it is useful to assess whether the prior distributions are reasonable in the context of the model. This can be done using a prior predictive check, where data is simulated using parameters drawn from the prior. If the generated data is implausible, it suggests that the priors may be too weakly or strongly informative.

Posterior predictive check

After fitting a Bayesian model a posterior predictive check is used to evaluate how well the fitted model explains the observed data. Here, data is simulated using parameter values sampled from the posterior distribution. If the replicated data fails to resemble the observed data, it suggests that the model may not capture key aspects of the underlying data-generating process.

Prior Sensitivity Analysis

Prior sensitivity analysis assesses how sensitive a model's inferences are to the choice of prior distributions. It involves re-running the model with different reasonable priors and comparing the resulting posterior distributions. This analysis helps determine the extent to which the prior influences the posterior.

Since re-fitting the same model with various priors can be computationally expensive, an alternative approach is to use importance sampling. Suppose our original prior is $p_0(\theta)$ and we wish to assess the sensitivity of our results to an alternative prior $p_1(\theta)$. For each MCMC sample θ_i, we compute the importance weight

$$w(\theta_i) = \frac{p_1(\theta_i)}{p_0(\theta_i)},$$

and then use these weighted samples to approximate the posterior under the new prior.

To improve the stability of these importance weights, especially when some weights are extremely large, methods such as Pareto smoothed importance sampling (PSIS) have been proposed [Vehtari et al., 2024].

C Numerical Stability Tricks

This is a collection of some of the numerical tricks used in the implementation of the algorithms to avoid underflow and improve efficiency.

Log-Sum-Exp Trick

When aggregating the contributions of multiple particles to the log-likelihood, we need to compute a sum of the form

$$L_t = \frac{1}{N} \sum_{i=1}^{N} \exp(\ell_i),$$

where N is the number of particles and ℓ_i is the log-weights. Direct exponentiation of ℓ_i can lead to numerical underflow if ℓ_i is very small. To mitigate this, we use the log-sum-exp trick [Stan Development Team, 2024]. Define

$$M = \max_{1 \le i \le N} \ell_i.$$

Then,

$$\log \left(\sum_{i=1}^{N} \exp(\ell_i) \right) = M + \log \left(\sum_{i=1}^{N} \exp(\ell_i - M) \right).$$

Thus, the incremental log-likelihood is computed as

$$\log L_t = -\log N + M + \log \left(\sum_{i=1}^{N} \exp(\ell_i - M) \right).$$

This approach rescales the log-likelihoods so that the exponential terms do not vanish numerically.

Transformation of Parameters

When parameters are defined on constrained domains, it is often beneficial to transform them into an unconstrained space to facilitate efficient MCMC proposals. Using a standard proposal distribution, such as the normal distribution, directly in the constrained space can lead to proposed values that lie outside the domain, resulting in a likelihood of zero. Suppose we have a vector of parameters

$$\theta = (\theta_1, \theta_2, \ldots, \theta_n),$$

where some components are defined on a constrained domain. We introduce an invertible and differentiable transformation g_i that maps the constrained θ_i into an unconstrained space:

$$\phi_i = g_i(\theta_i), \quad i = 1, 2, \ldots, n.$$

For those components that are already unconstrained, we can simply use the identity mapping, i.e.,

$$g_i(\theta_i) = \theta_i.$$

A common example is proposing values for a standard deviation, which must be positive, and we can then instead propose values in log-space.

If $p(\theta)$ denotes the joint density of θ and $\theta_i = g_i^{-1}(\phi_i)$ is the inverse transformation, the joint density in the ϕ-space is given by the change-of-variables formula:

$$p_\phi(\phi) = p\big(g_1^{-1}(\phi_1), \ldots, g_n^{-1}(\phi_n)\big) \prod_{i=1}^{n} \left| \frac{d}{d\phi_i} g_i^{-1}(\phi_i) \right|.$$

Taking logarithms yields the transformed log density:

$$\log p_\phi(\phi) = \log p\big(g_1^{-1}(\phi_1), \ldots, g_n^{-1}(\phi_n)\big) + \sum_{i=1}^{n} \log \left| \frac{d}{d\phi_i} g_i^{-1}(\phi_i) \right|.$$

D Supplementary Figures

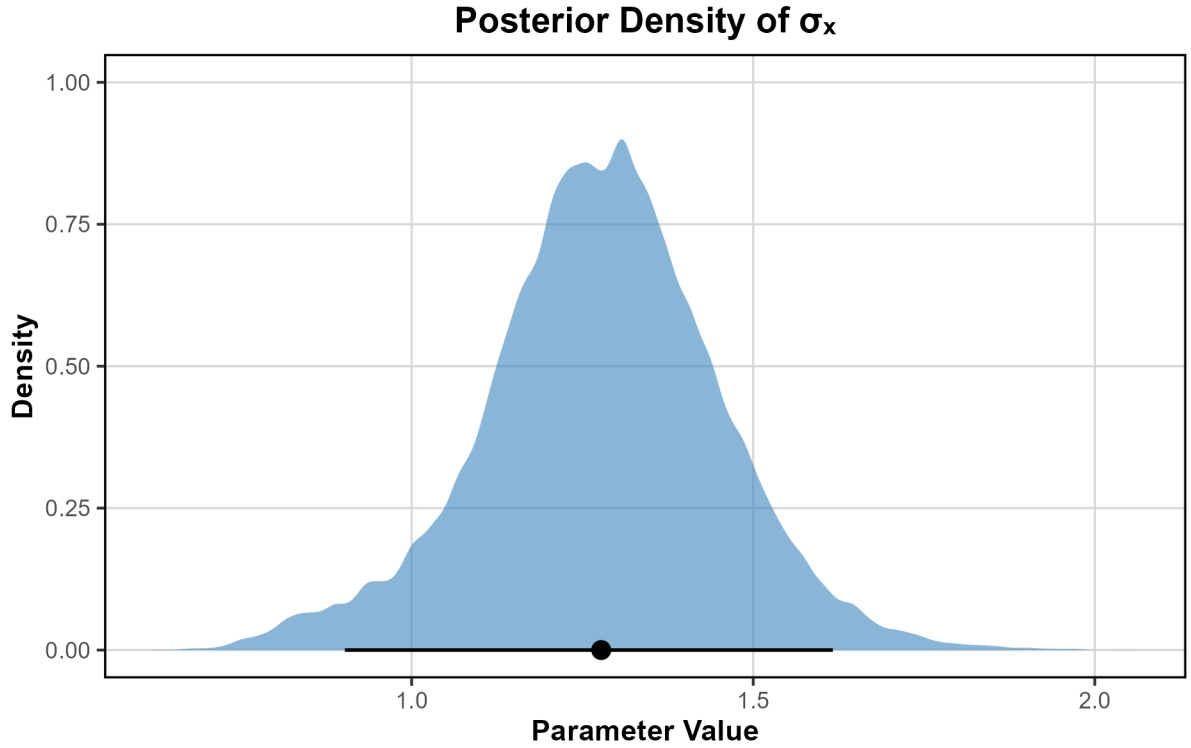

Figure D.1: Posterior density estimate for σ_x in Example 4.2. The black dot indicates the posterior mean, and the horizontal line represents the 95% credible interval.

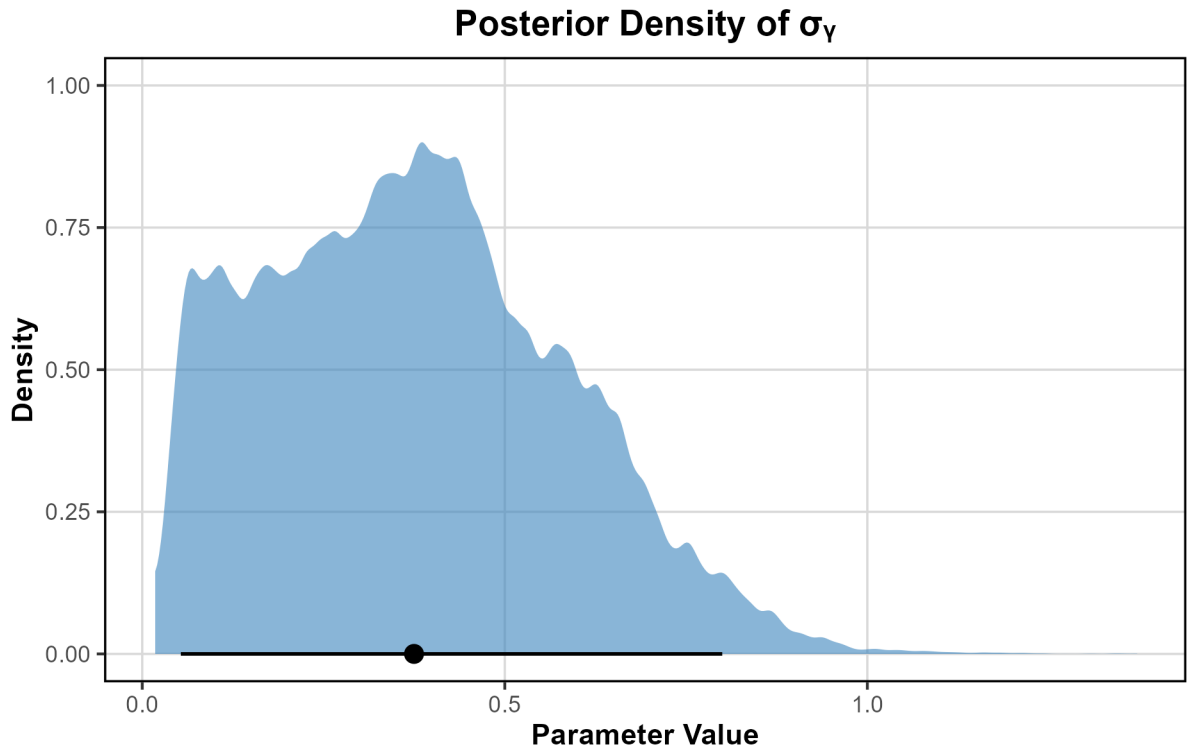

Figure D.2: Posterior density estimate for σ_y in Example 4.2. The black dot indicates the posterior mean, and the horizontal line represents the 95% credible interval.

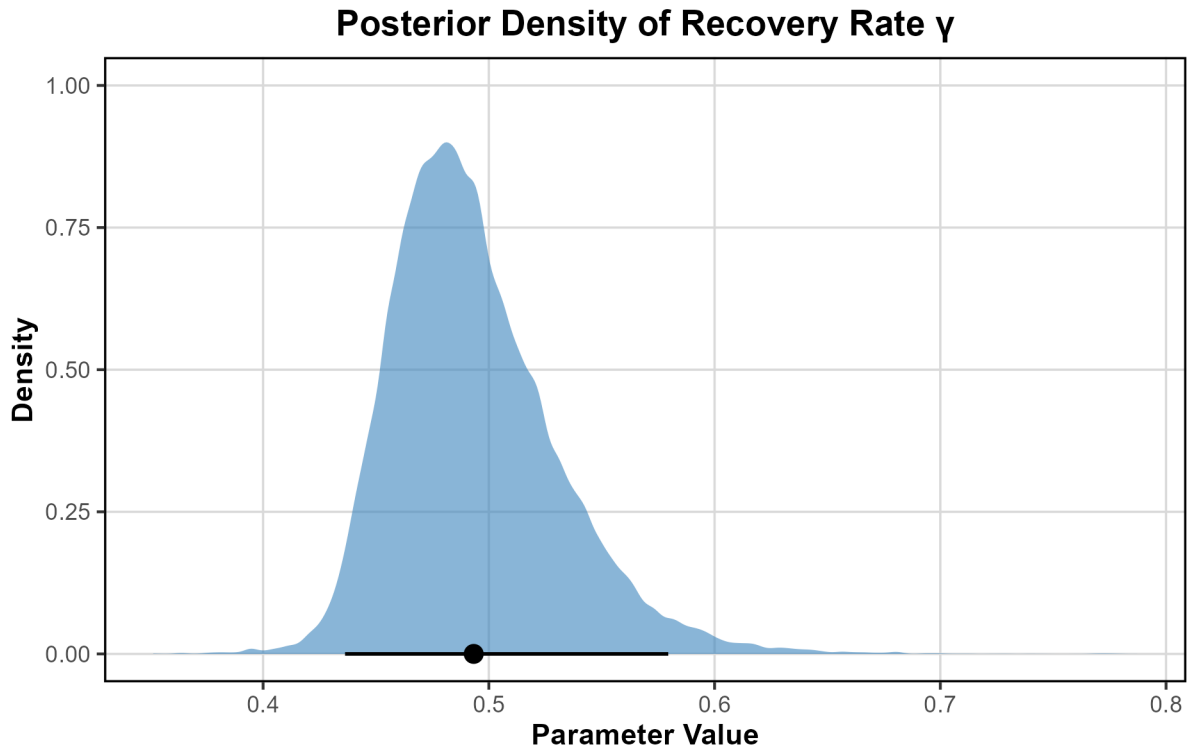

Figure D.3: Posterior density of γ under the Bayesian model with informative priors, based on the 1978 influenza outbreak at a boarding school in England. The black dot indicates the posterior mean, and the horizontal line shows the 95% credible interval.

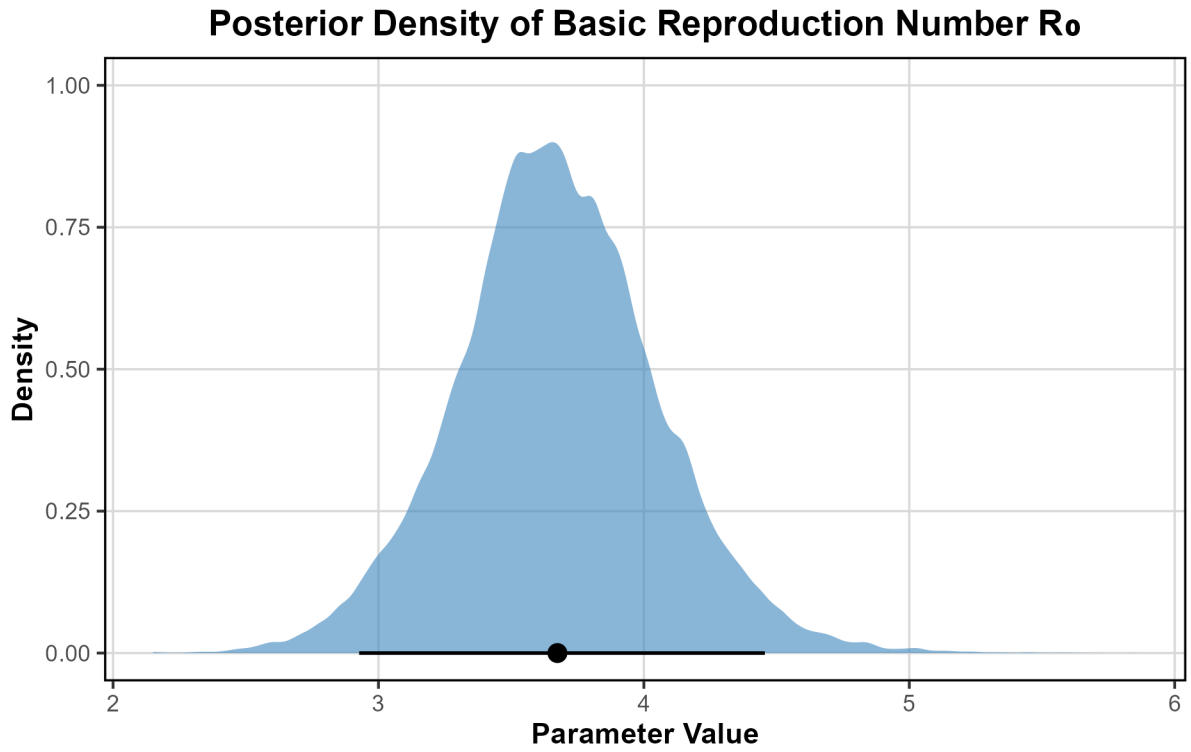

Figure D.4: Posterior density of R_0 under the Bayesian model with informative priors, based on the 1978 influenza outbreak at a boarding school in England. The black dot indicates the posterior mean, and the horizontal line shows the 95% credible interval.

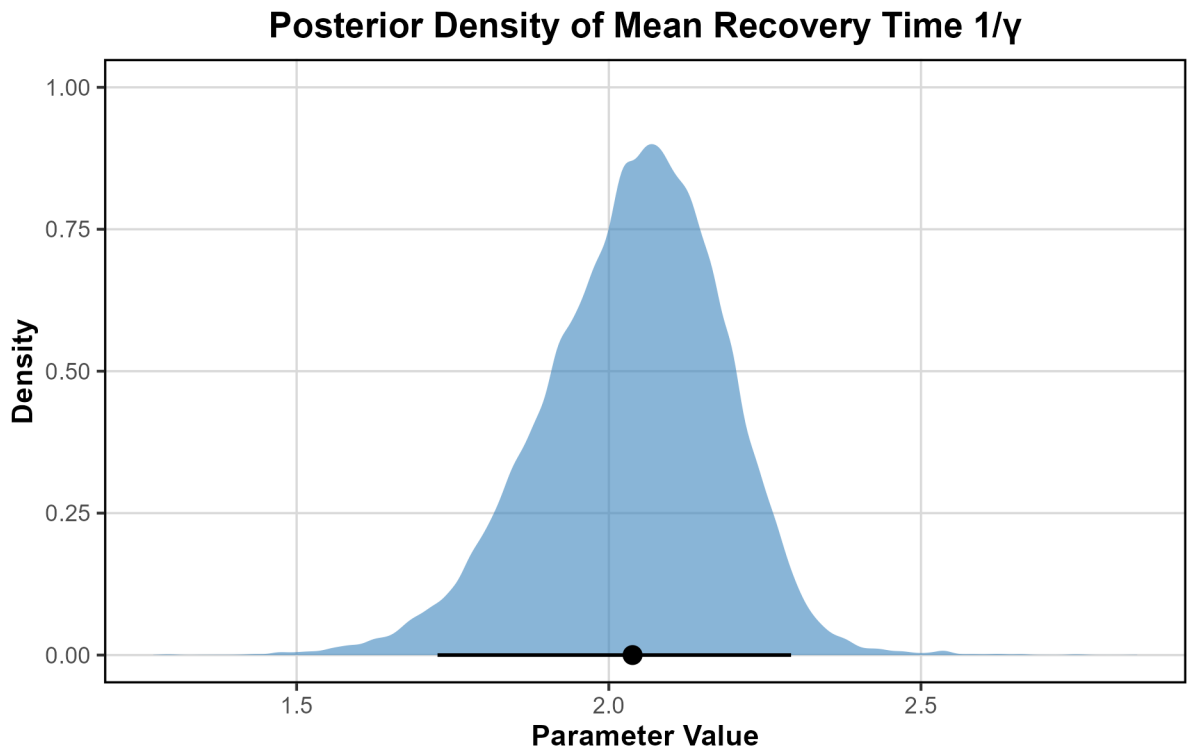

Figure D.5: Posterior density of the mean recovery time under the Bayesian model with informative priors, based on the 1978 influenza outbreak at a boarding school in England. The black dot indicates the posterior mean, and the horizontal line shows the 95% credible interval.