CHALMERS

GÖTEBORGS UNIVERSITET

Approximate Bayesian computation with applications to time series data and spread of covid-19

Approximate Bayesian computation med tillämpningar på tidsseriedata och spridning av covid-19

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Henrik Häggström
Jesper Jäghagen
Jibbril Ndaw Berbres
Samuel Wagner

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Approximate Bayesian computation with applications to time series data and spread of covid-19

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers

Henrik Häggström Jesper Jäghagen Jibbril Ndaw Berbres Samuel Wagner

Handledare: Umberto Picchini

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Preface

This report is a bachelor's thesis produced at the department of Mathematical Sciences at Chalmers University of Technology. The authors are four students at the engineering mathematics programme. A group diary as well as individual diaries have been kept during the course of the project. We would like to thank our supervisor Umberto Picchini Ph.D for invaluable feedback and guidance throughout the project.

Contribution report

We began the project by a series of four case studies, all of which were conducted in groups of two, where new groups were formed each week and both groups worked on the same project. The case studies spanned approximately one week each, at the end of which we discussed the results as a group, such as any differences in our implementations and any questions or problems that we might have encountered in the process. The first two case studies were to implement the ABC rejection algorithm and the SMC-ABC algorithm, respectively. In these two case studies, the inference was conducted on the auto-regressive (AR) model as seen in Wiqvist *et al.* [1], which is discussed in this thesis, and a moving average (MA) model as seen in Marin *et al.* [2]. The third case study was to conduct inference on the Lotka-Volterra model using the two ABC algorithms, following the example set by Toni *et al.* [3]. The fourth and last case study was to implement semi-automatic summary statistics as an extension of the case study the previous week.

In the weeks following the initial four case studies, we decided to include the AR model and the Lotka-Volterra model in the final report. In addition, we wanted to conduct inference in a model with apparent real-world applications and decided to investigate the Susceptible-Infected-Recovered (SIR) model with application to the ongoing covid-19 pandemic. As it was known by group member that the Public Health Agency of Sweden published data of covid-19 cases each week, we decided to restrict ourselves to the spread of the virus in Sweden.We decided on the following "work packages". The first was expansion of the SIR model, which was entrusted to Samuel. A few different models were produced, the most applicable of which we called the "SEIRDS" (Susceptible-Exposed-Infected-Recovered-Deceased-Susceptible) model. The SEIRDS model took e.g. exposure before ability to infect others, mortality rate and loss of immunity after recovery from infection into account, but was unfortunately disregarded at a later stage in the project due to lack of data for the different groups. Another work package was to analyse data from the Public Health Agency of Sweden and make it possible to use this data with the existing code, which was entrusted to Henrik and Jibbril. A natural progression of this work then resulted in Henrik and Jibbril formulating prior distributions for the parameters of the SIR model and writing the section on the SIR model in the thesis as well. Jesper was entrusted to investigate the concept of "posterior predictive trajectories" and also implement the Metropolis-Hastings algorithm in the different models, which included formulating likelihood functions in all models.

Seeing as we had already studied the theory of Bayesian inference, the two ABC algorithms and the summary statistics at this stage, we started writing the theory section of the report, the subsections of which was distributed equally amongst the group members. Jesper wrote about Bayesian inference, Henrik about the ABC rejection algorithm and Jibbril about the SMC-ABC algorithm. In the initial theory section, theory involving the SIR model and its extensions was included and written by Samuel, but was later reduced and moved to section 5.1 and appendix F.2.

After coming to an end with the previously mentioned work packages, the structure of the report was decided in a group effort. At this point, we also decided to merge the otherwise two distinct sections of "results" and "discussion", as this seemed like the most natural way to present our work. Responsibility for writing the AR model section was given to Jesper and responsibility of writing the Lotka-Volterra model section was given to Samuel. Extra responsibility of proofreading and structure of the report was given to Jibbril and Samuel. Other areas of responsibility and a detailed view of the division of labour in the final thesis is seen in table 1.

As both Henrik and Jibbril wrote parts of "results and discussion" in the SIR modelling section

the individual responsibilities will be clarified. Henrik was responsible for producing results from runs in the form of figure 12, 13 and 14, as well as table 4, and wrote up until the paragraph beginning with ”One curious thing”. Jibbril generated figure 15 and 16, and wrote the remaining part of ”results and discussion”.

It should be noted that all the algorithms, i.e. the two ABC algorithms and the Metropolis-Hastings algorithm, were implemented in Python from scratch and there was no reliance on preexisting packages. This implies that quite a large portion of time was spent on implementing all the algorithms to begin with, but also in modifying the algorithms for the different models. However, it has also contributed to the group's deep understanding of the algorithms and hopefully to the quality of the final work.

Section	**Written by**	**Miscellaneous**
Preface	Jibbril	
Contribution report	Samuel	
Popular science presentation	Jesper	
Abstract	Jibbril	
1. Introduction	Samuel	
1.1 Purpose	Samuel	
1.2 Limitations	Jibbril	
2.1 Bayesian inference	Jesper	
2.2 Approximate Bayesian computation	Henrik	
2.3 The ABC rejection algorithm	Henrik	
2.4 The sequential Monte Carlo ABC algorithm	Jibbril	
2.5 Semi-automatic summary statistics	Jibbril	
3. An introductory example: the autoregressive model	Jesper	
3.1 Theory	Jesper	
3.2 Implementation	Jesper	
3.3 Results and discussion	Jesper	
4. The Lotka-Volterra model for population dynamics	Samuel	
4.1 Theory	Samuel	
4.2 Implementation	Samuel	
4.3 Results and discussion	Samuel	
5. SIR modelling of covid-19 data in Sweden	Henrik	
5.1 Theory	Samuel	
5.2 Implementation	Henrik	
5.3 Results and discussion	Henrik and Jibbril	H up to the paragraph ”One curious thing...”, J after that
6. Final discussion and conclusions	Jibbril	Samuel wrote a few sentences regarding the Lotka-Volterra model
A. The stochastic Ricker model	Henrik	
B. The sequential Monte Carlo ABC algorithm	Jibbril	
C. The Metropolis-Hastings algorithm	Jesper	
D. Theoretical derivations in the autoregressive model (all)	Jesper	
E. The Lotka-Volterra model (all)	Samuel	
F.1 Markov chains	Henrik	
F.2 Expansions of the SIR model	Samuel	

Table 1: Author of every section in the thesis.

Popular science presentation

Mathematical models are used in almost all pioneering research and in more advanced research areas are the scenarios we want to model quite complex. One challenge when building a complex model is to fit the model to observed data. This is where the approximate Bayesian computation, or ABC, toolbox comes in handy. The ABC toolbox is a process which can estimate the unknown parameters in a model by computer simulation in a way that other statistical processes cannot. How you might ask? The short answer is that you don't require any direct probabilistic relation between the data and your parameters when performing an ABC estimation. What the ABC process does instead is to generate data and compare it to the observed data. In this way we do consider the probabilistic data-parameter relation, also called the "likelihood function", without knowing what it looks like!

Are there any drawbacks with the ABC method? Unfortunately yes, the greatest strength of ABC is also its greatest weakness, that we do not know anything about the likelihood function. This makes the ABC processes more simulation heavy and therefore much slower than other methods which directly uses the data-parameter relation. One can compare that the performance of the ABC method and a likelihood based method to a race between a dirt bike and a sports car. Under the condition that we are driving on a race track, the sports car will win over the dirt bike every time. However, if we drive off-road, the sports car cannot even manage one lap because the environment is not right for the car's design. In other words are likelihood based methods generally faster and more exact than the ABC method when we have a race track in form of the likelihood function. However, without the data-parameter relation will the ABC method be the only available method that can fit the model to the data.

Figure 1: Which is faster, the dirt biker or the sports car? Well it depends on the circumstances. The same goes for parameter estimation with the ABC method versus a likelihood based method. Pictures retrieved from [4] and [5].

An easy way of understanding the ABC method is through examples. We have therefore applied the ABC method to some time series models where the goal is to estimate their parameters. We have chosen to work with the autoregressive, Lotka-Volterra and SIR models. The autoregressive time series model is used in fields such as economics and biology and is fairly simple. This makes it a good introduction to the ABC method. The Lotka-Volterra or predator-prey model describes population dynamics in biological systems and is a bit trickier than the autoregressive model. The Lotka-Volterra model serves as a bridge to the SIR model as it introduces how models which are based on ordinary differential equations can acquire parameter estimations with the ABC methodology. The SIR model is an epidemiological model that we will estimate parameters for using real data on the covid-19 spread in Sweden gathered from the public health agency of Sweden. No conclusions will be drawn about the covid-19 pandemic, but this rather shows the usefulness of the ABC methodology.

Abstract

Bayesian statistical methods are one of the most popular tools for parameter inference. However, due to their dependence on a well defined and analytically tractable likelihood function, they are not always applicable to models of arbitrary complexity. Approximate Bayesian computation (ABC) methods counteract this by introducing a Bayesian framework utilising model simulations to bypass the need for a closed-form likelihood function. In this thesis we explore two ABC methods, the ABC rejection algorithm and the sequential Monte Carlo ABC (SMC-ABC) algorithm. We use them to infer parameters for the autoregressive AR(2) model, the Lotka-Volterra model for predators and prey, and the SIR-model for modeling the spread of Covid19 in Sweden. Subsequently, we compare the generated parameter distributions to the exact posteriors produced by the more well-known Metropolis-Hastings algorithm. The results indicate that the ABC methods produce posterior distributions comparable to those generated by the Metropolis-Hastings algorithm, that the SMC-ABC algorithm is more computationally efficient than the ABC rejection algorithm, and that the resulting posterior distributions are quite independent of the choice of prior.

Sammanfattning

Bayesianska metoder är ett av de populäraste verktygen vid parameterinferens. Dessa metoder kan dock inte alltid användas då de är beroende av en väldefinierad likelihoodfunktion. Approximate Bayesian Computation (ABC) är en samling metoder som löser detta problem genom att presentera ett Bayesianskt ramverk utan beroende på en likelihoodfunktion på sluten form. I denna rapport utforskar vi två ABC metoder: ABC rejection algoritmen och Sequential Monte Carlo ABC (SMC-ABC) algoritmen. Vi använder sedan dessa till att generera parametrar för en autoregressiv modell kallad AR(2), Lotka-Volterra modellen för rovdjur och bytesdjur samt SIR-modellen för spridningen av covid-19 i Sverige. Vi jämför sedan dessa med exakta posteriors för parametrar genererade av den mer välkända Metropolis-Hastings algoritmen. Våra resultat indikerar att ABC metoderna genererar posterior fördelningar som är jämförbara med de som genereras av Metropolis-Hastings algoritmen, att SMC-ABC är beräkningsmässigt effektivare än ABC algoritmen, samt att valet av a-priori-fördelning inte har någon större påverkan på de resulterande a-posteriori-fördelningarna.

Contents

1 Introduction

In the late 1800s, Sir Francis Galton invented the Galton board, also known as the bean machine, which can be seen in figure 2. Let us say that we were to drop a single pellet. What is the probability of this pellet landing in a specific partition? To calculate this, we would need to know the probability of the pellet going right or left in the first level of obstacles, right or left in the second level, and so on. Intuitively, one could argue the pellet has equal opportunity of going in either direction, but we might not know this with absolute certainty. To find out, we could run a "simulation" by dropping a large number of pellets and obtain a pattern much like the one seen on the right-hand side in figure 2. From this pattern, it is plain to see that the pellets follow a binomial distribution, but what are the parameter values? The binomial distribution $B(n, p)$ is defined by the number of trials n, in this case 10 as there are ten levels of obstacles, and the probability of success p, which can be set to the probability of a pellet going one step to the right in each level. In figure 2, we can see that the middle partition has the most pellets, which when starting from zero is partition number five. Knowing that the mean value of a binomial distribution is given by np, we can see that $p = 0.5$. Although dropping a large number of pellets tends to give a similar pattern every time, the partition with the most pellets might differ for a number of reasons. If we proceed to do multiple simulations, each time noting the partition with the most pellets and calculating p, we could put together the results to form a probability distribution for the possible values of p. Thus, we have used simulation to draw conclusions regarding the parameter p, also known as parameter inference.

Figure 2: A Galton board before and after it is turned, retrieved from [6]. On the left, we see the pellets gathered at the bottom. When the Galton board is turned, the pellets fall into the pattern seen on the right, having gone through ten levels of cylindrical obstacles.

Similar principles can be applied when conducting parameter inference in various models. One such simulation based method is approximate Bayesian computation (ABC), institutionalised by Tavaré *et al.* [7] and Pritchard *et al.* [8]. ABC is, however, conducted in a reversed fashion to the previous case of the Galton board in that a set of parameters $\boldsymbol{\theta}$ are submitted to a model that subsequently produces an output. When a model possesses this characteristic, it is known as "generative", which is a requirement for ABC to be employed. For example, models defined by systems of ordinary differential equations (ODEs) are generative models in the sense that input parameters $\boldsymbol{\theta}$ produce an output given by the exact or numerical solution to the system. To determine the accuracy of the output, or how "good" the result is, the output of the model is compared to preexisting observed data, assumed to have been generated by the very same model, and is assessed based on a tolerance level ϵ. Parameters corresponding to outputs that are sufficiently close are then saved, or "accepted", and collectively constitute an approximate "posterior distribution", which is a probability distribution of possible parameter values given knowledge of observed data. Using the accepted parameter values, we can also produce so called "posterior trajectories", which is the corresponding set of outputs from the generative model. The posterior trajectories can then be used to e.g. estimate and predict future events in time series models.

The ABC methods require no formulation of "likelihood functions", which are otherwise used to describe the probability of observed data conditional to parameter values. In models involving non-deterministic dynamics, such as stochastic differential equations or in stochastic biomathematical models, these likelihood functions cannot be solved analytically, thus making the ABC methodology essential to conduct inference. However, for the models considered in this thesis, the likelihood function can be formulated, giving us the opportunity to not only show how the ABC methodology

can be applied, but also to what extent it deviates from samples of the true posterior distribution, as provided by the Metropolis-Hastings algorithm.

1.1 Purpose

The main purpose of this thesis is to investigate the ABC methodology. We explore two ABC algorithms, the ABC rejection and sequential Monte Carlo ABC (SMC-ABC) algorithms. We explain how these work and how they can be implemented. In doing so, we show applications of ABC parameter inference first in an autoregressive AR(2) model, as an introductory example. We then proceed to the Lotka-Volterra model, describing the dynamics of predator and prey populations, using artificially observed data. Finally, we conduct inference on the Susceptible-Infected-Recovered (SIR) model, widely used in modelling of epidemics, using real data of confirmed covid-19 cases in Sweden. The previously mentioned likelihood function can be formulated in all models considered in this thesis, which in reality eliminates the necessity for ABC methodology. However, the models have been chosen so that sampling from the true posterior distribution by the Metropolis-Hastings algorithm can be used as a benchmark for the performance of the ABC algorithms, allowing for more explicit analysis of obtained results.

1.2 Limitations

ABC methods are a broad subject of study, thus we have decided to limit our investigation in the following ways: We will mainly focus on the implementation of the algorithms, not the theory behind them. Some theory is necessary to understand the algorithms from a practical standpoint, but proofs of convergence etc. will not be presented. We also limit ourselves to only considering two ABC algorithms, the ABC rejection algorithm and the SMC-ABC algorithm. Lastly, although we are working with real data for the SIR-model, we are not trying to make any sort of statement with regards to the real values of the parameters. We are not epidemiologists, so any results are only to show that inference is possible.

2 Prerequisites

2.1 Bayesian inference

The fundamental idea of Bayesian inference is to improve parameter estimations based on observed data. When applying Bayesian inference to infer parameters $\boldsymbol{\theta}$ in a specific model there are three steps to follow. The first step is to formulate a prior belief about possible parameter values, which is acquired by previous experiences or by other studies. This prior belief can be encoded as a probability distribution that is by convention called the prior distribution $\pi(\boldsymbol{\theta})$. The second step is to see how this prior belief $\pi(\boldsymbol{\theta})$ will be shifted by making statistical inference on the parameters with observed data $\mathbf{y}$. The observed data $\mathbf{y}$ given our parameters $\boldsymbol{\theta}$ can also be statistically interpreted as a distribution $\pi(\mathbf{y}|\boldsymbol{\theta})$, the so called likelihood function. The final step is now to apply Bayes' theorem in order to get the distribution of the inference, the so called posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{y})$. Bayes' theorem states that

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

where the denominator, $\pi(\mathbf{y})$, is the marginal likelihood distribution. The marginal likelihood can be interpreted as a scaling constant. For exact inference, the marginal likelihood is required, but if we only intend to take samples from the posterior distribution it is not. In the latter case we only consider what our posterior is proportional to with respect to the parameters $\boldsymbol{\theta}$, namely

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

In other words, the necessary information to draw samples from the posterior $\pi(\boldsymbol{\theta}|\mathbf{y})$ is only the prior distribution $\pi(\boldsymbol{\theta})$ as well as the likelihood function $\pi(\mathbf{y}|\boldsymbol{\theta})$. As previously mentioned, the likelihood function must be theoretically derived, while the prior is constructed out of a qualified

guess when performing the Bayesian inference. Now that we know how to calculate a posterior distribution, we can explore how this can be used in sampling algorithms.

A useful and widely used algorithm within Bayesian statistics is the Metropolis-Hastings algorithm. This algorithm is used to generate samples with a Markov chain from a posterior distribution without explicitly calculating the posterior. After a sufficient number of iterations, the algorithm will begin to draw samples from the true posterior of the inferred parameters $\boldsymbol{\theta}$. The longer the Metropolis-Hastings algorithm runs, the more the sampled distribution converges towards the true posterior distribution. For further explanation of the Metropolis-Hastings algorithm, see appendix C. The Metropolis-Hastings algorithm can favourably be used as a benchmark for other sampling algorithms, because we get samples from the true posterior.

2.2 Approximate Bayesian computation

The Metropolis-Hastings is dependent on the likelihood function in order to sample from the posterior distribution. When dealing with complex models, the associated likelihood function may be analytically intractable, for example it may require computation of high-dimensional integrals. An example of this is the stochastic Ricker model that can be found in appendix A. Approximate Bayesian computation (ABC) circumvents this problem by estimating the posterior distribution of model parameters without explicitly evaluating the likelihood function. To give a simple example of a generative model and how this works, let us assume that we do not know how to sample from a generic normal distribution $\mathcal{N}(\mu, \sigma^2)$, but we do know how to sample from a standard normal distribution $u \sim \mathcal{N}(0, 1)$. The generative model

$$\mathcal{M}(\boldsymbol{\theta}) : z = \mu + u \times \sigma \tag{1}$$

allows for generation of samples $z \sim \mathcal{N}(\mu, \sigma^2)$, $\boldsymbol{\theta} = (\mu, \sigma)$. Hence (1) is the generative model $\mathcal{M}(\boldsymbol{\theta})$ for the generic normal distribution. Note that simulating z from the generative model $\mathcal{M}(\boldsymbol{\theta})$ is equivalent to drawing samples from the likelihood $\pi(z|\boldsymbol{\theta})$ even if the likelihood is analytically unavailable.

2.3 The ABC rejection algorithm

The ABC rejection algorithm given in algorithm 1 will be presented in a similar way to the useful introduction provided by Marin *et al.* [2]. The observed data $\mathbf{y} \in D \subseteq \mathbb{R}^n$ is assumed to come from the generative model of study. The algorithm draws a set of parameters $\boldsymbol{\theta}' \sim \pi(\boldsymbol{\theta})$ and generate data $\mathbf{z} \sim f(\cdot|\boldsymbol{\theta}')$ from the generative model. The proposed parameters $\boldsymbol{\theta}'$ are then accepted or rejected based on the proximity of the generated data $\mathbf{z} \in D$ to the previously observed data $\mathbf{y}$. To determine the proximity of the generated data $\mathbf{z}$ to the observed data the Euclidian distance

$$\rho(\mathbf{z}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (z_i - y_i)^2} \tag{2}$$

is used throughout this report unless stated otherwise. The tolerance level of acceptance associated with this distance is denoted as $\epsilon > 0$. Marin *et al.* [2] states that algorithm 1 generates N draws from the marginal in $\mathbf{z}$ of the approximated joint posterior distribution

$$\pi_\epsilon(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y}) = \frac{\pi(\boldsymbol{\theta}) f(\mathbf{z}|\boldsymbol{\theta}) \mathbb{1}_{A_{\epsilon,\mathbf{y}}}(\mathbf{z})}{\iint_{A_{\epsilon,\mathbf{y}} \times \boldsymbol{\theta}} \pi(\boldsymbol{\theta}) f(\mathbf{z}|\boldsymbol{\theta}) \, dz \, d\theta},$$

where $\mathbb{1}_C$ is the indicator function on the set C and $A_{\epsilon,\mathbf{y}} = \{\mathbf{z} \in D | \rho(\mathbf{z}, \mathbf{y}) < \epsilon\}$. The theoretical idea behind the ABC rejection algorithm is that when the tolerance ϵ is small

$$\pi_\epsilon(\boldsymbol{\theta}|\mathbf{y}) = \int \pi_\epsilon(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y}) \, d\mathbf{z} \approx \pi(\boldsymbol{\theta}|\mathbf{y}).$$

Algorithm 1: ABC rejection

for *i=1 to N* **do**
  repeat
    Generate $\boldsymbol{\theta}'$ from the prior distribution $\pi(\cdot)$
    Simulate $\mathbf{z}$ from the generative model $f(\cdot|\boldsymbol{\theta}')$
  until $\rho(\mathbf{z}, \mathbf{y}) < \epsilon$
  set $\boldsymbol{\theta}_i = \boldsymbol{\theta}'$
end

When $\epsilon = 0$ the ABC rejection algorithm will according to Marin *et al.* [2] provide independent and identically distributed samples from the exact posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{y})$.

2.4 The sequential Monte Carlo ABC algorithm

One problem facing the ABC rejection algorithm is the time required to calculate an approximate posterior distribution, particularly in the case where the posterior is very different from the prior. For example, consider a situation where one samples from a uniform prior over a square $\{(x, y) : x, y \in [-10, 10]\}$, with an ϵ accepting values inside a smaller square $\{(x, y) : x, y \in [-0.1, 0.1]\}$. With this, only one in ten thousand samples would be accepted. Generating a posterior distribution of one hundred points would thus require an average of one million iterations. If the generative model is complex this will quickly become computationally expensive. Given this conundrum, a number of variations have been proposed to improve upon the general ideas of section 2.3. Beaumont et al. [9] present one such variation called population Monte Carlo approximate Bayesian computation. This process is also known as sequential Monte Carlo approximate Bayesian computation (SMC-ABC). SMC-ABC combines the ideas of ABC rejection with a sequential approach to learn about the posterior while continually improving the sampler.

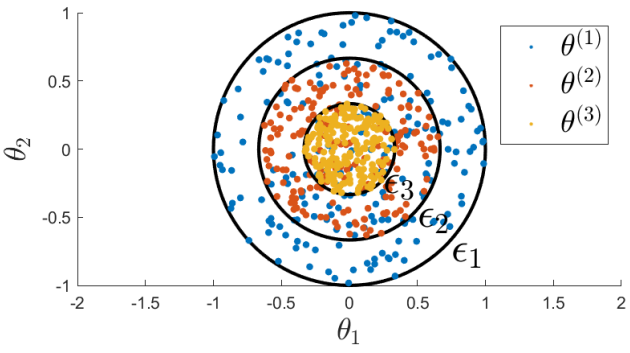


Figure 3: Simulated example of the process by which the SMC-ABC algorithm refines the posterior for $N = 200$ points. The first generation (blue) of parameters $\theta^{(1)}$ are bounded by ϵ_1. For the second generation (red) we have drawn θ_i^* with replacement from $\theta^{(1)}$ and perturbed the particles using a normal distribution. If the perturbed particle falls inside the stricter ϵ_2 then it is accepted and added to $\theta^{(2)}$. To further refine the result, an even stricter epsilon ϵ_3 is introduced and the process is repeated.

In figure 3 we see a visual representation of the SMC-ABC algorithm, and in algorithm 2 we see the full implementation. To help the reader's understanding, we will attempt to explain it in a somewhat simplified way. The interested reader can consult appendix B for a more detailed explanation. First, let $\boldsymbol{\theta}^{(1)}$ be the population of parameters generated by running the ABC rejection algorithm. The SMC-ABC algorithm now draws a new set of parameters $\boldsymbol{\theta}^*$ from $\boldsymbol{\theta}^{(1)}$ with replacement. This draw is made in keeping with a set of weights associated to each particle in $\boldsymbol{\theta}^{(1)}$. Consider one of these draws $\boldsymbol{\theta}_i^*$. Given that $\boldsymbol{\theta}^{(1)}$ is the result of the ABC rejection algorithm, the distance between the data $\mathbf{z} \sim f(\mathbf{z}|\boldsymbol{\theta}^{(1)})$ and the observed data $\mathbf{y}$ will be less than some $\epsilon = \epsilon_1$. Now consider a stricter acceptance criterion $\epsilon_2 < \epsilon_1$. The distance $\rho(\mathbf{z}, \mathbf{y})$ is certainly smaller than ϵ_1, but it may be larger than ϵ_2. As such, the algorithm perturbs $\boldsymbol{\theta}_i^*$ by using it as the mean of a draw from a normal distribution $\boldsymbol{\theta}_i^{(2)} \sim \mathcal{N}(\boldsymbol{\theta}_i^*, \Sigma_1)$ where Σ_1 is two times the covariance matrix of $\boldsymbol{\theta}^{(1)}$. Since $\boldsymbol{\theta}_i^{(2)}$ is now perturbed, if it was previously outside the stricter acceptance criterion ϵ_2, it may now have moved inside it. If it has, $\boldsymbol{\theta}_i^{(2)}$ is accepted, otherwise a new $\boldsymbol{\theta}_i^*$ is

drawn and perturbed. This process continues until N points have been accepted into $\boldsymbol{\theta}^{(2)}$.

Algorithm 2: SMC-ABC

At iteration $t = 1$.
for *i = 1 to N* **do**
  repeat
    Simulate $\boldsymbol{\theta}_i^{(1)} \sim \pi(\boldsymbol{\theta})$ and $\mathbf{z} \sim f(\mathbf{z} \mid \boldsymbol{\theta}_i^{(1)})$
  until $\rho(\mathbf{z}, \mathbf{y}) < \epsilon_1$
  Set $\omega_i^{(1)} = 1/N$
end
Take ϵ_2 as $\min(\alpha_t, \epsilon_1)$
Take Σ_1 as two times the covariance matrix of $\boldsymbol{\theta}^{(1)}$
for *t = 2 to T* **do**
  for *i = 1 to N* **do**
    repeat
      Pick $\boldsymbol{\theta}_i^*$ from the $\boldsymbol{\theta}_j^{(t-1)}$'s with probabilities ω_j^{t-1}
      Generate $\boldsymbol{\theta}_i^{(t)} \sim \mathcal{N}(\boldsymbol{\theta}_i^*, \Sigma_{t-1})$ and $\mathbf{z} \sim f(\mathbf{z} \mid \boldsymbol{\theta}_i^{(t)})$
    until $\rho(\mathbf{z}, \mathbf{y}) < \epsilon_t$
    Set $\omega_i^{(t)} \propto \pi(\theta_i^{(t)}) / \sum_{j=1}^{N} \omega_j^{(t-1)} \varphi(\boldsymbol{\theta}_i^{(t)}; \boldsymbol{\theta}_j^{(t-1)}, \Sigma_{t-1})$
  end
  Take Σ_t as two times the covariance matrix of $\theta^{(t)}$.
  Take ϵ_{t+1} as $\min(\alpha_t, \epsilon_t)$
end

By this point, the SMC-ABC algorithm has produced a tighter posterior for the inferred parameters. Throughout this thesis we will refer to this as the algorithm having completed a new generation of particles. However, there is no reason to stop here. The exact same process can be carried out again for a stricter $\epsilon_3 < \epsilon_2$ yielding an even tighter posterior. In our implementation, we choose ϵ_n as the 15:th percentile of the distances produced by ρ in the previous generation. We denote this value by α_t. In this way, for each passing generation, the posterior distribution approaches the true posterior $\pi(\boldsymbol{\theta}|\mathbf{y})$.

2.5 Summary statistics

One important topic when considering ABC methods is the concept of summary statistics. A summary statistic is simply a way of summarising a data set while retaining as much information as possible. Some examples could be the mean, variance or autocorrelation of a data set. Having one or more suitable summary statistics becomes important when using ABC methods with a large generative model. As the number of raw data points created by a generative model grows, accepting a suggested set of parameters becomes increasingly rare due to the limitation set by the distance $\rho(\mathbf{z}, \mathbf{y}) < \epsilon$. As a result, one must increase the size of ϵ which may degrade the inference. If instead one uses a summary statistic $S(\cdot)$, producing a larger data set does not create the need to increase ϵ, since $\rho(S(\mathbf{z}), S(\mathbf{y}))$ will be similar regardless of the size of the data sets.

One drawback of using summary statistics is that if the statistic is poorly selected there will be a loss of information. For example, if we were to consider only the mean, the data sets $\{99, 100, 101\}$ and $\{1, 100, 199\}$ would be equal despite being very different in character. So how does one choose a correct set of summary statistics? Fernhead and Prangle [10] propose a semi-automated way of constructing summary statistics using linear regression.

Consider a data set $\mathbf{y}$ and a vector-valued function $f(\mathbf{y})$ consisting of possibly non-linear transformations of the data, for instance $f(\mathbf{y}) = (\mathbf{y}, \mathbf{y}^2, \mathbf{y}^3, \mathbf{y}^4)$. For a simulated parameter $\boldsymbol{\theta}_i$ one can fit the linear regression model

$$\boldsymbol{\theta}_i = E(\boldsymbol{\theta}_i|\mathbf{y}) + \xi_i = \beta_0^{(i)} + \beta^{(i)} f(\mathbf{y}) + \xi_i$$

where ξ_i is some zero-mean noise. The fitted function

$$S(\mathbf{y}) = \hat{\beta}_0^{(i)} + \hat{\beta}^{(i)} f(\mathbf{y}) \tag{3}$$

then estimates $E(\boldsymbol{\theta}_i|\mathbf{y})$. Thus we have acquired a method of generating summary statistics that is less arbitrary than simply choosing them. In addition, the vector-valued function f can be constructed to return a large number of values, thus increasing the chances of some of them being informative with regards to the parameter of interest.

3 An introductory example: the autoregressive model

To get a sense of how the ABC rejection and SMC-ABC algorithms operate, we will first implement them on an autoregressive time series model. The autoregressive time series model of order p, denoted AR(p), is a relatively simple model to implement the ABC algorithms on because the data set is one dimensional and computationally inexpensive to generate. The order of the autoregressive time series was set to $p = 2$ in accordance to Wiqvist *et al.* [1].

3.1 Theory

A general autoregressive time series model AR(p) describes a stochastic process defined by

$$y_j = \xi_j + \sum_{i=1}^{p} \theta_i y_{j-i}, \quad \xi_j \sim \mathcal{N}(0,1),$$

where the time series element y_j depends on the p previous elements in the process and a random standard Gaussian noise ξ_j. By the definition above the autoregressive model of order $p = 2$ is defined by

$$y_j = \theta_1 y_{j-1} + \theta_2 y_{j-2} + \xi_j, \quad \xi_j \sim \mathcal{N}(0,1), \tag{4}$$

where $y_0 = \xi_0$ and $y_1 = \theta_1 y_0 + \xi_1$.

We have to know which $\boldsymbol{\theta}$ values are reasonable for the AR(2) model in order to construct a prior $\pi(\boldsymbol{\theta})$. As derived in appendix D, the triangular area

$$A = \{\theta_1, \theta_2 \in \mathbb{R} \quad | \quad \theta_2 < 1 + \theta_1, \quad \theta_2 < 1 - \theta_1, \quad \theta_2 > -1\} \tag{5}$$

is the range of possible $\boldsymbol{\theta}$ values which gives a stable time series. This means that $\boldsymbol{\theta} \in A$ will produce a time series with values y_j varying around a fixed mean with a constant variance and therefore make parameter estimation easier.

In Box *et al.* [11] is a statistical measure presented which quantifies how observations at different time points are correlated to each other in a time series. This measure is called the autocovariance and is defined as

$$K_k = Cov(y_j, y_{j+k}) = E[(y_j - \mu)(y_{j+k} - \mu)],$$

where $k \in \mathbb{N}$ is a fixed lag constant and $\mu = E[y_j]$ is the expected value of y_j. The autocovariance K_0 is the same as the variance of the time series.

The likelihood function is also derived in appendix D and is given by

$$\pi(\mathbf{y}|\boldsymbol{\theta}) = \mathcal{N}(y_0; 0, 1) \cdot \mathcal{N}(y_1 - \theta_1 y_0; 0, 1) \cdot \prod_{j=2}^{M} \mathcal{N}(y_j - \theta_1 y_{j-1} - \theta_2 y_{j-2}; 0, 1), \tag{6}$$

where M is the length of the time series.

3.2 Implementation

The observed 50 data points that was used in all algorithms were generated from (4) with the parameter values $\boldsymbol{\theta}^{\text{true}} = [0.2, -0.13]$. The same model was also used as the generative model $f(\cdot)$. The prior distribution was set to $\pi(\boldsymbol{\theta}) \sim \mathcal{U}(A)$, where A is the triangular area in (5). As previously stated, this prior granted us parameters which generates a stationary time series. Autocovariances of five different lags were implemented as summary statistics, namely $S(\cdot) = [K_0(\cdot), K_1(\cdot), K_2(\cdot), K_3(\cdot), K_4(\cdot)]$.

Common for both the ABC rejection and SMC-ABC algorithms were that the same amount of posterior particles $N = 1000$ was used. For the SMC-ABC algorithm, the number of generations was set to $T = 5$ and the starting tolerance to $\epsilon_1 = 14$. For the ABC rejection algorithm was the tolerance $\epsilon = \epsilon_T$ fixed, where ϵ_T was the tolerance for the final generation $t = T = 5$ in the SMC-ABC implementation. The ABC rejection and SMC-ABC algorithms were then implemented in Python according to the algorithms described in section 2.3 respectively 2.4. The number of proposed parameters, the CPU-time and the mean absolute error (MAE) was measured for each algorithm. The mean absolute error of the sampled parameters $\boldsymbol{\theta}$ was calculated according to the definition

$$MAE(\boldsymbol{\theta}_j) = \frac{\sum_{i=1}^{N} |\theta_{ij} - \theta_j^{\text{true}}|}{N},$$

where θ_j^{true} is the true parameter value, N is the number of simulations and $\boldsymbol{\theta}_j = [\theta_{1j}, ..., \theta_{Nj}]$ is the vector of corresponding sampled parameters.

The Metropolis-Hastings algorithm was also implemented using the same prior $\pi(\boldsymbol{\theta})$ as the ABC algorithms and the likelihood function $\pi(\mathbf{y}|\boldsymbol{\theta})$ shown in (6). The covariance matrix for the proposal distribution

$$\Sigma = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.1 \end{bmatrix}$$

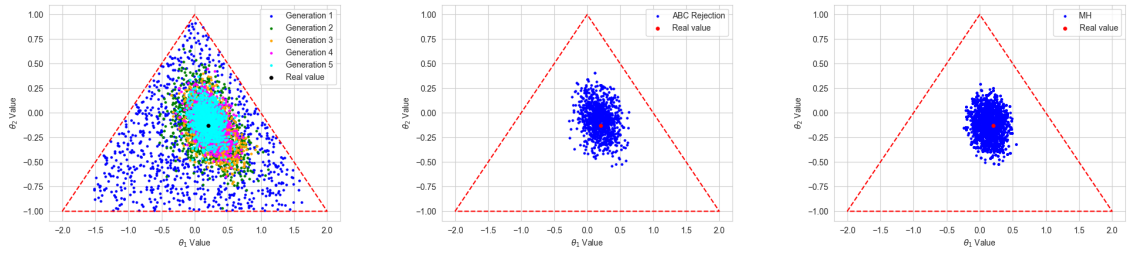
was fixed after a number of pilot runs with different values. The algorithm was then set to run for $N = 10\,000$ iterations. The first 2000 sampled $\boldsymbol{\theta}$ values were removed after the run was completed, in order to exclude the burn-in period. The algorithm in appendix C was implemented in Python with the mentioned choices of parameters. The same performance measurements were taken as for the ABC algorithms.

3.3 Results and discussion

The resulting performance measurements are shown in table 2 for the three algorithms. The 1000 posterior particles for the SMC-ABC and ABC rejection algorithms are shown in figure 4 as well as the sampled 8000 particles from the true posterior via the Metropolis-Hastings algorithm. A comparison of the resulting ABC posterior densities for the parameters θ_1 and θ_2 are visualised in figure 5 as well as the sampled true density. The true parameter values $\boldsymbol{\theta}^{\text{true}} = [0.2, -0.13]$ are also shown as dotted lines in figure 5. All the five generations $t = 1, ..., 5$ are plotted in figure 4(a) where the last generation, coloured in cyan, should be considered as the SMC-ABC algorithms end result. Therefore, the last generation of posterior particles are used in the distribution comparison plot in figure 5. The starting tolerance level $\epsilon_1 = 14$ in the SMC-ABC simulation was reduced to $\epsilon_5 = 0.216$ in the final generation.

Algorithm	Number of proposals	CPU-time	MAE(θ_1)	MAE(θ_2)
SMC-ABC	542 652	601.4s	0.110	0.116
ABC rejection	2 350 161	907.2s	0.124	0.135
Metropolis-Hastings	10 000	89.9s	0.105	0.099

Table 2: Performance measurements for the three different algorithms applied on the AR model.



(a) SMC-ABC (b) ABC rejection (c) Metropolis-Hastings

Figure 4: Scatter plots for the SMC-ABC, ABC rejection and Metropolis-Hastings algorithm inside the triangular prior. The SMC-ABC algorithm produced $N = 1000$ posterior particles per generation t, while the ABC rejection algorithm produced a total of $N = 1000$ posterior particles. The Metropolis-Hastings algorithm produced $N = 8000$ particles from the true posterior.

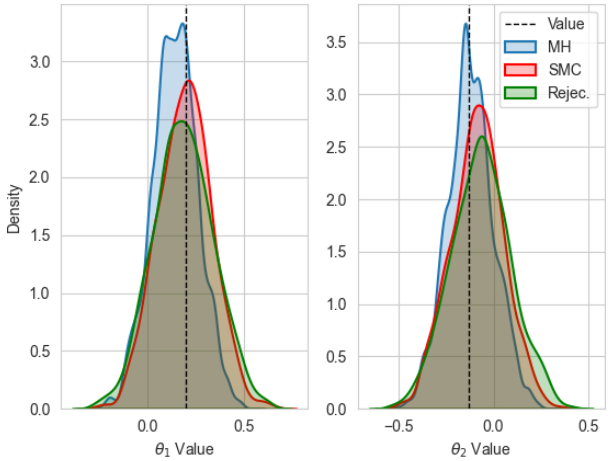


Figure 5: Comparison of the sampled kernel densities between the three algorithms for the θ_1 and θ_2 parameters. MH stands for Metropolis-Hastings in the plot legend.

As seen in figure 4, 5 and in the absolute mean error in table 2 are the resulting posterior distributions produced from the ABC algorithms quite close to the true posterior. Comparing the SMC-ABC with the ABC rejection algorithm, we see that SMC-ABC outperforms the ABC rejection algorithm. This is made obvious by comparing each of the performance measures in table 2 between the two algorithms. The SMC-ABC algorithm takes less parameter proposals and CPU-time to run and also gives a narrower posterior which is more centred around the true values of θ_1 and θ_2, as shown in figure 5. The performance difference is a consequence of that the SMC-ABC algorithm reduces the tolerance ϵ_t in each generation t and perturbs the already accepted posterior particles from the previous generation $t-1$. It is also important to note that it is more likely for the algorithm to perturb more promising particles thanks to the weighting $\omega_i^{(t)}$, which speeds up the convergence process. The ABC rejection algorithm does not have these features and are therefore slower and more computationally heavy.

One more observation is that the Metropolis-Hastings algorithm is faster than the ABC algorithms, as seen in table 2. The Metropolis-Hastings algorithm can output 8000 posterior samples in 10 to 15 % of the CPU-time that the ABC algorithms need to output 1000 posterior samples. This result is not surprising as we can sample directly from the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{y})$ thanks to

the tractable likelihood function $\pi(\mathbf{y}|\boldsymbol{\theta})$, while the ABC algorithms has to sample a new data set, summarise the data with the summary statistic $S(\cdot)$ and calculate the euclidean distance for each simulation.

4 The Lotka-Volterra model for population dynamics

The second model considered is the "Lotka-Volterra" model, defined by a system of ODEs describing dynamics in prey and predator populations. The generative property required from the model to run the ABC algorithms is obtained by solving the system of ODEs exactly or numerically, given parameters $\boldsymbol{\theta}$. To determine the proximity of simulated data to observed data, summary statistics can be used, as previously discussed in section 2.5. As suggested by Toni *et al.* [3] it is possible to simulate data points corresponding only to the same points in time as the observed data, and then compare the two directly by Euclidean distance, which implies choosing the identity function as summary statistics, i.e. $S(\mathbf{y}) = \mathbf{y}$. In the following model, we will however employ semi-automatic summary statistics to determine the accuracy of simulated data.

4.1 Theory

The Lotka-Volterra model is a system of ODEs used to describe the dynamics of a biological system consisting of two interacting species, one predator and one prey. The model is given by

$$\begin{aligned} \frac{dx}{dt} &= ax - xy \\ \frac{dy}{dt} &= bxy - y \end{aligned} \tag{7}$$

where x is the prey population, y is the predator population, a is a parameter affecting the prosperity of the prey population in relation to the predator population and b is a parameter affecting the prosperity of the predator population in relation to the prey population. Firstly, we note that $x \geq 0$ and $y \geq 0$ for all $t \geq 0$. We will also assume that $x > 0$ and $y > 0$ at initial time $t = 0$ is implied, as the model is otherwise redundant. Secondly, we see that $a \in \mathbb{R}$ and $b \leq 0$ gives $dy/dt \leq 0$ where equality is fulfilled if and only if $y = 0$. Thus, $b \leq 0$ results in $y \to 0$ as $t \to \infty$, i.e. an imminent extinction of predators in the system. Additionally, if $a > 0$ and $b \leq 0$ an unrestricted increase in prey population ensues. Furthermore, for $a < 0$ and $b \in \mathbb{R}$ both species will eventually go extinct, as $dx/dt \leq 0$, where equality is fulfilled if and only if $x = 0$, results in $x \to 0$ and subsequently $y \to 0$ as $t \to \infty$.

4.2 Implementation

As previously mentioned, the ABC algorithms rely on generative properties of the investigated model. In our implementation of the ABC algorithms to the Lotka-Volterra model given by (7), input parameters $\boldsymbol{\theta} = [a, b, \sigma]$ produce an output $y^*(t) = y(t) + \xi_t$ for time t, where $y(t)$ is the numerical solution to (7) given $a, b \in \boldsymbol{\theta}$ and ξ_t is a noise term for $t > 0$ such that $\xi_t \sim \mathcal{N}(0, \sigma^2)$, where $\sigma \in \boldsymbol{\theta}$. As seen, the standard deviation of the noise term, σ, was regarded an unknown parameter, and thus a part of the parameter inference and subject to a prior distribution. Seeing as the ABC algorithms have been implemented in Python, the function `odeint()` from the `SciPy` library has been used for the purpose of finding the numerical solutions $y(t)$.

For the parameter inference on the Lotka-Volterra model, the numerical solutions to the system of ODEs were restricted to 16 data points equidistant in time. As presented in section 2.5, semi-automatic summary statistics were then used to compare the simulated and observed data, where preexisting functions from the `sklearn` library in Python were used to perform the required linear regression. The vector-valued function $f(\mathbf{y})$ used in the summary statistics, as seen in (3), was set to $f(\mathbf{y}) = (\mathbf{y}, \mathbf{y}^2)$. Furthermore, the distance between two data sets $\rho(S(\mathbf{z}), S(\mathbf{y}))$ was measured as the square of the Euclidean distance, following the example set by Toni *et al.* [3].

The parameter inference was carried out using artificially observed data produced by the generative model given parameter values $a = 1$, $b = 1$ and $\sigma = 0.5$. Similar to Toni *et al.* [3], a and b were

given prior distributions $a, b \sim \mathcal{U}(-10, 10)$, which can be considered somewhat uninformative. The standard deviation of the noise was given a prior distribution $\sigma \sim \mathcal{U}(0, 1)$. As the true a, b and σ were known, an additional dimension of analysis of the posterior distributions produced by the ABC algorithms was made possible. Furthermore, the same observed data was used to conduct inference in all algorithms in order to be able to compare their performances. Data of the true prey and predator populations used in this section, i.e. the numerical solution to the Lotka-Volterra model given the true value of parameters a and b, can be found in appendix E.1. The observed data, on which we based all inference, is discretised data from the numerical solution, subject to the previously mentioned noise terms $\xi_t \sim \mathcal{N}(0, 0.5^2)$, which can also be found in appendix E.1. As we wished the data to reflect realistic observations and in order to conduct meaningful inference, all observed data points were conditioned to be non-negative. It could be argued that by seeing that the observed data does not result in extinction of either species, the priors should in fact reflect that $a, b > 0$ is implied. However, solutions corresponding to extinction of either species should be rejected by the ABC algorithms, as this results in $\rho(S(\mathbf{z}), S(\mathbf{y})) > \epsilon$ for sufficiently small ϵ, where $\mathbf{z}$ is generated data showing extinction of either species and $\mathbf{y}$ is observed data.

In our implementation, the SMC-ABC algorithm was restricted to 5 generations. Furthermore, for the Metropolis-Hastings algorithm, the likelihood function was computed as the multivariate normal probability density $\varphi(\mathbf{y}; \mathbf{z}, \sigma^2 \boldsymbol{I_m})$, where $\mathbf{y}$ is the observed data, $\mathbf{z}$ is the generated data, σ is the standard deviation of the noise terms and m is the number of data points in the numerical solution. As previously stated, $m = 16$. The computation of the likelihood is hence based on the essential assumption that the noise is normally distributed, which we know to be true. Naturally, as σ is considered an unknown parameter, the standard deviation of this perturbation is subject to parameter proposals itself. Furthermore, the covariance matrix in the multivariate Gaussian distribution for the proposal of new parameters was tuned from a number of pilot runs and set to $0.01\boldsymbol{I_3}$.

4.3 Results and discussion

The exact Bayesian inference obtained by 10 000 iterations of the the Metropolis-Hastings algorithm with a discarded burn-in period of 2500 iterations, which will be referenced throughout the discussion of the Lotka-Volterra model, can be found in appendix E.2. Corresponding figures illustrating parameter values in the Markov chains and posterior trajectories can be found in appendix E.2 as well.

In figure 6 we see 1000 accepted parameter values of parameters a and b from the ABC rejection algorithms using a tolerance level $\epsilon = 3.336$. Note that accepted parameter values of σ have been omitted from figure 6. A total of 1 521 596 proposed parameters were needed to obtain the result, which gives an acceptance rate of approximately 0.066%. The accepted parameter sets in figure 6 are primarily gathered around the true parameter values. However, there are some scattered particles are in the second and third quadrant, which is caused by the dynamics of the model. As previously mentioned, $a < 0$ results in an extinction of both prey and predator populations at some time $t > 0$. In this case, either ϵ has not been set sufficiently small to reject parameters where $a < 0$, or the noise terms have perturbed the solution towards the observed data, thus accepting the parameter values. Furthermore, we see no particles in the fourth quadrant due to rejection of parameter sets where $a > 0$ and $b < 0$, as these make the growth of the prey population strictly positive and the growth of the predator population non-positive. Most particles accepted by the algorithm are, however, centred around the true value of the parameters, as illustrated by the black dot in figure 6. To closely examine the posterior distributions, including the standard deviation of the noise, σ, we look to figure 9 and the probability density function of the accepted parameter values in the ABC rejection algorithm. As seen, the mode of the posterior distributions of a and b are close to the respective true parameter value. We do however see that the posterior distributions of a and b are considerably wider than the true posterior distribution, such that a large proportion of the distribution tails have been omitted in figure 9. However, more details of the posterior distribution can be found in appendix E.3. Furthermore, despite including the true parameter value, the posterior distribution of σ is imprecise and shows almost uniform properties and the two local maximum points either gravely underestimates or overestimates the value of σ.

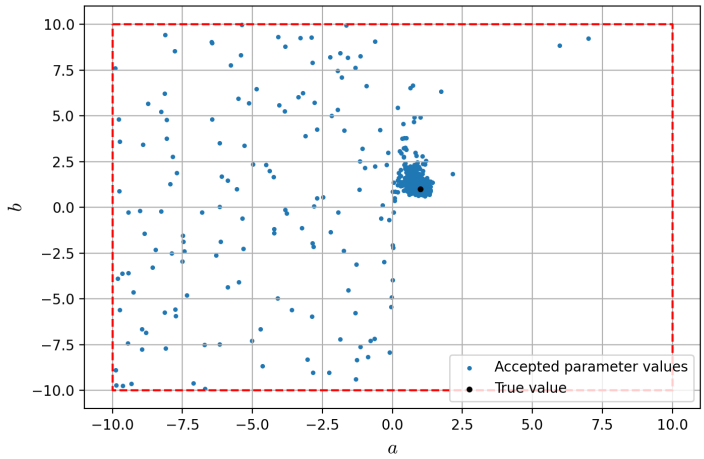


Figure 6: Parameter values of a and b for 1000 accepted parameters from the ABC rejection algorithm with $\epsilon = 3.336$ applied to the Lotka-Volterra model. The red line shows the boundaries of the uniform priors of a and b. The black dot displays the true parameter values.

In figure 7 we see the area enclosed by the posterior trajectories of the accepted parameter values in the ABC rejection algorithm, which are solutions to (7) given accepted parameters a and b without added noise. Note that the 2.5% lowest and the 2.5% highest values have been omitted in each time step, as to avoid solutions corresponding to parameters values outside of the general trend in the posterior distribution. For the avid reader, the posterior trajectories including all accepted parameter sets can be found in figure 23 in appendix E.4. Despite faltering approximations of

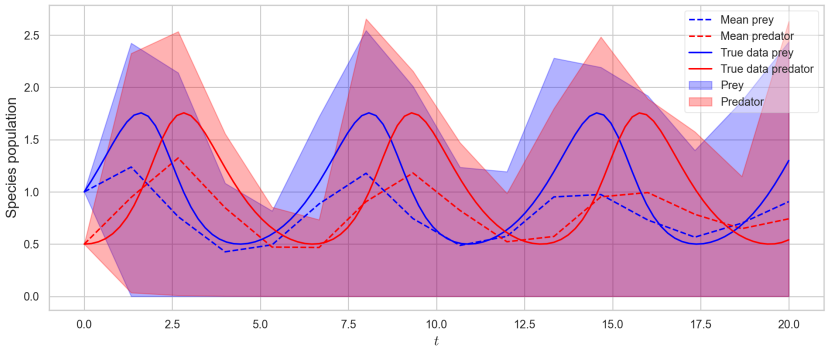


Figure 7: Posterior trajectories of the ABC rejection algorithm applied to the Lotka-Volterra model without added noise. The 2.5% lowest and 2.5% highest values have been omitted in each time step. The curves denoted "true data prey" and "true data predator" are the true prey and predator populations without added noise, whilst the blue and red areas enclose all posterior trajectories of prey and predator populations, respectively. The dashed curves are mean values of all posterior trajectories of a given population in each time step and is not based on a mean of parameter values.

the posterior distribution, we see that the mean of the posterior trajectories capture the true populations quite closely, although they tend to deviate from the crests of the true population curves. In comparison to the posterior trajectories obtained by the Metropolis-Hastings algorithm, as seen in figure 20 in appendix E.2, we see that the area enclosed by the posterior trajectories of the ABC rejection algorithm is larger, allowing for greater uncertainty. As previously seen in figure 6, the ABC rejection algorithm accepts multiple parameter sets where $a < 0$. Thus, as seen in figure 7, there is a possibility that both populations have become extinct by time $t \gtrsim 5$, even though observed data seen in appendix E.1 shows that there have been observations of both

populations for all $t \geq 0$.

Accepted parameter sets of a and b when using the SMC-ABC algorithm can be seen in figure 8. Once again, 1000 accepted parameter values were required. Note that the accepted values of σ have been omitted in figure 8.

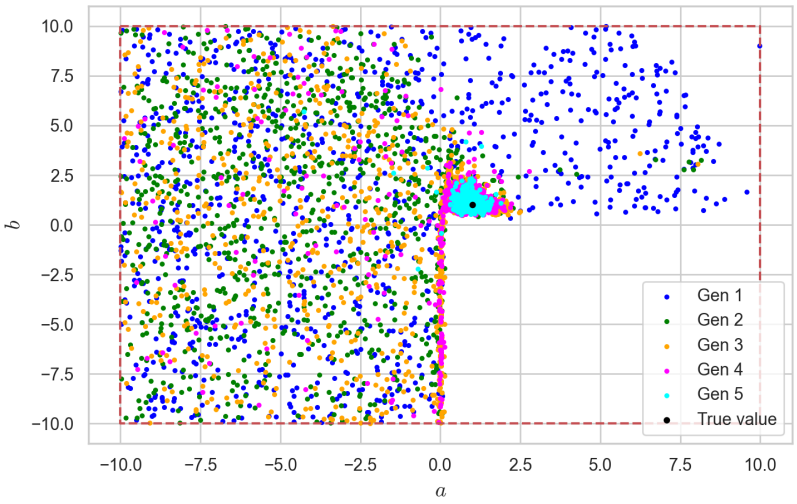


Figure 8: Parameter values of a and b for 1000 accepted parameters from the SMC-ABC algorithm applied to the Lotka-Volterra model with $\epsilon = 1000$ in the first generation. Parameter a is displayed on the x-axis and parameter b is displayed on the y-axis. The red line shows the boundaries of the uniform priors of a and b. The black dot displays the true parameter values.

In the first generation, i.e. in the ABC rejection stage of the SMC-ABC algorithm, the tolerance level was quite generously set to $\epsilon = 1000$, which by the SMC-ABC algorithm was reduced for each generation and in the last generation $\epsilon = 3.336$, which is the same tolerance level as used in the ABC rejection algorithm above. The total number of proposed parameter in the SMC-ABC algorithm was 1 555 799. The value of ϵ, the number of iterations and the acceptance rate in each generation can be seen in table 3. We see that the number of iterations increase and the acceptance rate decreases for each generation as epsilon decreases. We also note that almost two thirds of the total number of iterations were performed in the last generation where we had the lowest value of ϵ.

Generation	ϵ	Number of proposals	Acceptance rate
1	1000	1421	70.37%
2	30.17	24281	4.12%
3	17.67	115124	0.87%
4	9.151	447267	0.22%
5	3.336	967706	0.10%

Table 3: Tolerance level ϵ, number of proposed parameters and acceptance rate in each generation of the SMC-ABC algorithm applied to the Lotka-Volterra model.

As previously seen in the ABC rejection algorithm, the SMC-ABC algorithm accepts some particles in the second and third quadrant, primarily in the younger generations, with only a few discrepancies in the last generation. The reason for the acceptance of parameter pairs where $a < 0$ in younger generations is predominantly due to the relatively large ϵ value. Similarly to the ABC rejection algorithm, the SMC-ABC algorithm rejects almost all parameters in the fourth quadrant due to the nature of the investigated model. In the last generation in figure 8 we do, however, see a clear concentration around the true parameter value with fewer deviations than previously

seen in figure 6. To be able to examine the posterior distribution obtained in the last generation of the SMC-ABC algorithm more closely, we look to figure 9. As seen, the modes of the posterior distribution of a and b are close to their respective true parameter value. We see that the distributions of a and b display shorter tails and are more focused around the true parameter values, in comparison to the ABC rejection algorithm. The approximation of the posterior distribution of a is slightly wider and does not show a quite as well defined peak around the true parameter value as the true posterior distribution, despite being quite close. Furthermore, the approximation of the posterior distribution of b is highly accurate and very similar to the results obtained by the Metropolis-Hastings algorithm, although shifted towards lower values. The posterior distribution of σ spans over the full interval given by the uniform prior distribution, but is tilted towards the lower bound of the interval, in contrast to the result by the ABC rejection algorithm where the spread was greater. The true value of σ is indeed included in the posterior distribution, but the parameter value is underestimated and the distribution itself is quite dissimilar to the true posterior distribution.

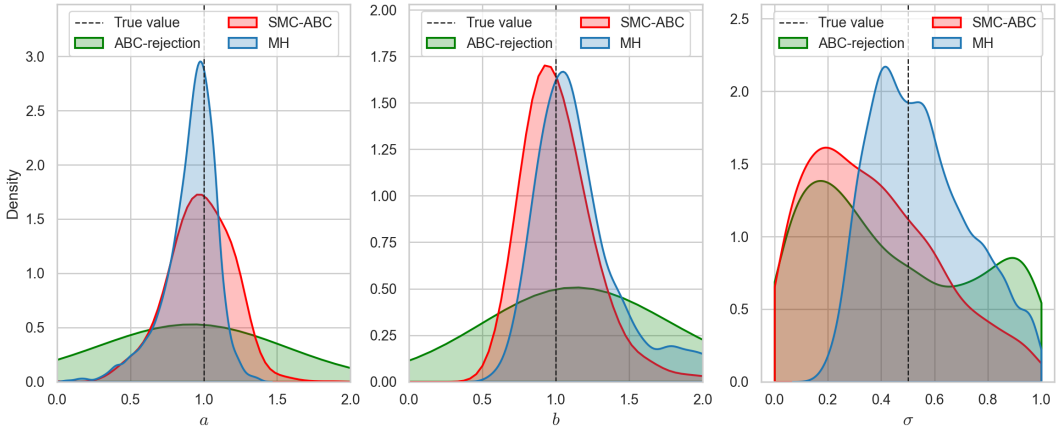


Figure 9: Posterior densities of parameters a, b and σ produced by the different algorithms applied to the Lotka-Volterra model. The posterior distribution curves are kernel density estimates of accepted parameter values in the ABC rejection algorithm and in the last generation of the SMC-ABC algorithm as well as parameter values sampled by the Metropolis-Hastings algorithm. The true value of each parameter is displayed by a dashed line. To display the details of the posterior distributions around the true parameter values, only values $a \in [0, 2]$ and $b \in [0, 2]$ are shown. However, the complementary details of the posterior densities can be found in figures 21 and 22 in appendix E.3.

In figure 10 we see the area enclosed by the posterior trajectories of accepted parameters in the last generation of the SMC-ABC algorithm, once again without added noise. The 2.5% lowest and 2.5% highest values have been omitted in each time step in figure 10, whereas the posterior trajectories including all accepted parameter sets of the last generation can be found in figure 23 in appendix E.4. In comparison to the posterior trajectories produced by the ABC rejection algorithm, the SMC-ABC produces posterior trajectories that are tighter around the true prey and predator population. We note that the posterior trajectories of the prey population in figure 10 do not include values where either population is extinct, which, considering both observed and true data, is a result favourable to that seen in figure 7. We also see that the mean of the posterior trajectories follow the true prey and predator populations more closely than previously seen. In comparison to the result obtained by the Metropolis-Hastings algorithm, as seen in figure 20 in appendix E.2, the enclosure of all posterior trajectories seen in figure 10 is slightly wider, primarily for the predator population.

In summary, we see that the ABC rejection algorithm produced the widest and most imprecise approximation of posterior distributions for all parameters. On the other hand, greater accuracy was displayed by the SMC-ABC algorithm, where the approximation of posterior distribution of b was almost identical to that of the Metropolis-Hastings algorithm around the true parameter

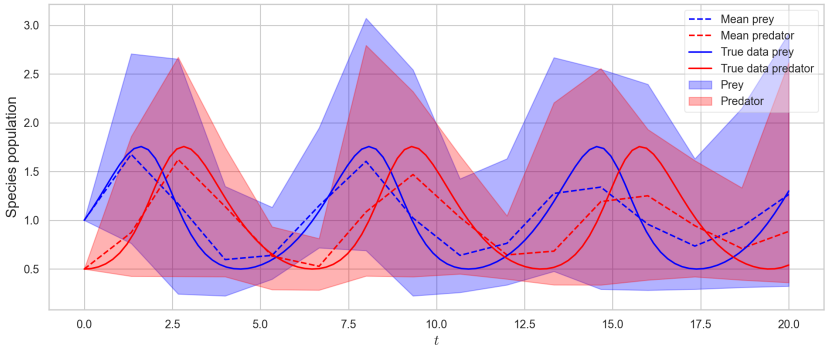


Figure 10: Posterior trajectories of the SMC-ABC algorithm applied to the Lotka-Volterra model without added noise. The 2.5% lowest and 2.5% highest values have been omitted in each time step. The curves denoted "true data prey" and "true data predator" are the true prey and predator populations without added noise. The blue and red areas enclose all posterior trajectories of prey and predator populations, respectively, corresponding to parameter values accepted in the last generation of the SMC-ABC algorithm. The dashed curves are mean values of all posterior trajectories of a given population in each time step and is not based on a mean of parameter values.

value. The posterior distribution of a showed a more accurate and well defined peak around the true parameter value than that of the ABC rejection algorithm, although not quite as detailed as the samples of the true posterior distribution. However, the estimation of σ was imprecise for both ABC algorithms, and even the Metropolis-Hastings algorithm produced a rather wide density of σ values. This can be explained by the nature of the data and the lack of repeated measurements at each time point. If we instead would have, for example, 10 observations of each population in each time point, σ could be estimated based on the relation of these measurements. As this is not the case, the standard deviation must be inferred from single observations. Naturally, this complicates the inference of the σ parameter. Regarding the posterior trajectories, the SMC-ABC algorithm showed the most precise results in relation to the Metropolis-Hastings algorithm. In this sense, out of the two ABC algorithms, it provides parameter inference more suitable for estimations dependent on reliable posterior trajectories, such as the probability of the prey or predator populations being greater than a certain value at a given point in time. In total, it is clear to see that out of the two ABC algorithms, the SMC-ABC algorithm yields the posterior distribution with the best performance in terms of accuracy in relation to samples from the true posterior when applied to the Lotka-Volterra model.

5 SIR modelling of covid-19 data in Sweden

The final model we will consider is the epidemiological Susceptible-Infected-Recovered (SIR) model. Real-world data of the covid-19 epidemic in Sweden will be used as observed data for this model. In this way we can investigate the ABC algorithms parameter inference when the true parameter values are unknown.

5.1 Theory

The SIR model was introduced by Kermack *et al.* [12] in the late 1920s as they used mathematics to determine the longevity of an epidemic. The model has its name from the characteristic division of a population into individuals who are susceptible to the disease (S), infected (I) and recovered (R), the last of which includes both survivors and deceased individuals having previously been

infected. In its simplest form, the SIR model is given by

$$\begin{aligned} \frac{dS}{dt} &= \frac{-rSI}{N_P} \\ \frac{dI}{dt} &= \frac{rSI}{N_P} - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned} \tag{8}$$

where r is the infection rate, γ the recovery rate and N_P the population size. As seen in (8), the model assumes certain characteristics of the disease, such as individuals only being infected once and no delay between the time of infection and the ability to infect others. Furthermore, we see that

$$\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0$$

implying that the model assumes a fixed total population N_P, such that $S + I + R = N_P$ at all times.

5.2 Implementation

In (8) we note that parameters r and γ affect multiple groups of the population simultaneously. Consequently, the linear regression needed to implement semi-automatic summary statistics was rather cumbersome and computationally inefficient, as the interaction between S, I and R needs to be considered as well. Thus, the observed and generated data were compared directly by the Euclidean distance in (2) without any summary statistic. In the same fashion as with Lotka-Volterra, a noise term $\xi_t \sim \mathcal{N}(0, \sigma^2)$ was added to the numerical solution $\mathbf{z}(t) = (S(t), I(t), R(t))$ from the generative model of the ODEs in (8). The standard deviation σ was once again considered an unknown parameter in the generative model and thus its approximated posterior distribution was included in the result. The implementation of the algorithms in Python was carried out in the same way as with the Lotka-Volterra model.

In contrast to the previous models, real data was used for the inference of the SIR model, as compared to generated data for the previous models. Observed data of the covid-19 epidemic in Sweden was obtained from the Public Health Agency of Sweden [13]. More specifically, the number of new confirmed cases of covid-19 each week was considered the number of infected people (I) in the population at that time. As no data for the number of susceptible (S) or recovered (R) individuals was available from this source, these groups were not considered when computing the Euclidean distance between the observed data and the generated data in the ABC algorithms. Thus, the ABC algorithms aim to find parameters that generate data of I similar to the observed data of infected people. It may seem problematic that we do not have full data for the model, but (8) shows that the solution of S will affect the solution of I and that R is a function of the solution of I. This means that parameter values that generate data of I similar to the observed data will also generate reasonable solutions to S and R.

A time resolution of one week was used, even though the original data provides new cases each day. The reason for this is that the number of new reported cases varied heavily depending on the day of the week. Fitting parameters of the SIR model to the entire time span of available data would have been problematic as the rate of infection and the testing rate have varied heavily between different phases of the epidemic. Hence, only the data from week 32 in 2020 to week 3 in 2021 was included, as this roughly correspond to the second "wave" of infection. The number of confirmed new cases of covid-19 from week 32 in 2020 to week 3 in 2021 according to the Public Health Agency of Sweden [13] is shown in figure 11. Furthermore the total population of Sweden N_P=10 415 565 in 2020 according to Statistics Sweden [14].

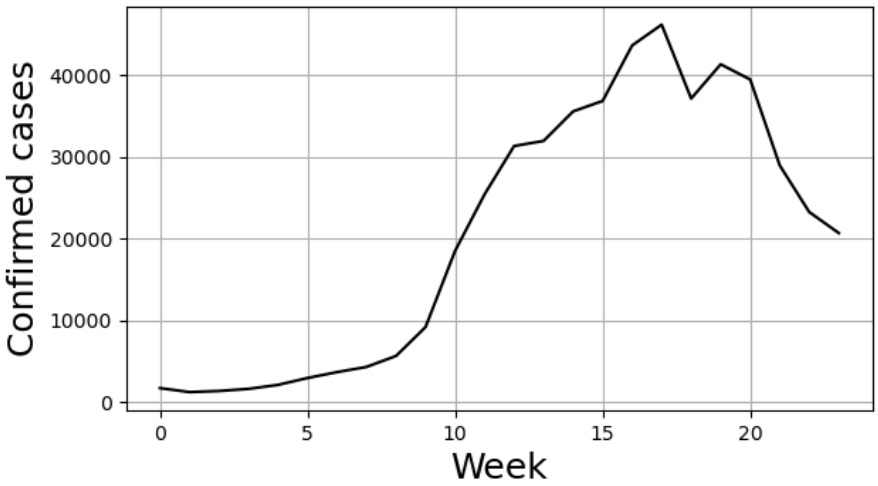


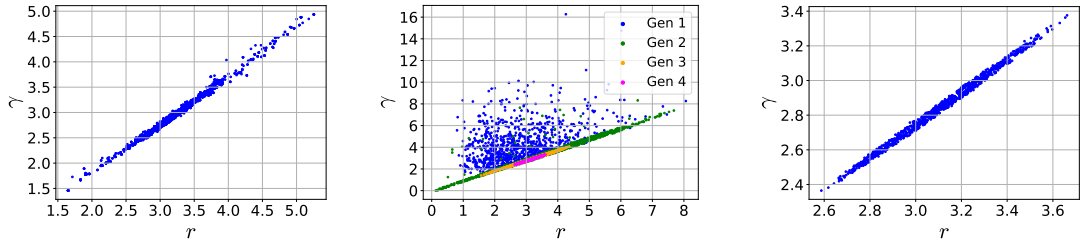
Figure 11: The number of new confirmed covid-19 cases each week from week 32 in 2020 to week 3 in 2021 according to the Public Health Agency of Sweden.

The prior distributions were set to $r, \gamma \sim \text{Lognormal}(1.15, 0.45^2)$ and $\sigma \sim \text{Lognormal}(8, 0.45^2)$. The reasoning behind setting lognormal prior distributions was that negative parameter values of r and γ should not be allowed as this generates unreasonable solutions where the groups S, I and R can become larger than N_P. The standard deviation of the noise σ can obviously not be negative either. Furthermore the lognormal distributions cover all positive values, which is desirable in this case where the true parameter values are unknown. The parameters of the prior distributions were decided and tuned by a number of pilot runs. Note that σ is on a different scale compared to the other parameters because it estimates the noise in the observations, which can be expected to be quite large when $N_P \approx 10^7$.

In the Metropolis-Hastings algorithm, the covariance matrix in the multivariate Gaussian proposal distribution was tuned by computing the sample covariance matrix of the result of a number of pilot runs. Based on the assumption that the noise is normally distributed, the likelihood function was, as for the Lotka-Volterra model, computed as the multivariate normal probability density $\varphi(\mathbf{y}; \mathbf{z}, \sigma^2 \boldsymbol{I_m})$, where $\mathbf{y}$ is the observed data, $\mathbf{z}$ is the generated data and $m = 24$ is the number of time points in the data.

5.3 Results and discussion

In figure 12 we see the posterior distribution of accepted values of r and γ when running the ABC rejection, SMC-ABC and Metropolis-Hastings algorithms applied to the SIR model with covid-19 data. The tolerance level of the ABC rejection run was $\epsilon = 35\,000$. A total number of 358 685 parameters were proposed, which corresponds to an acceptance rate of 0.28%. The last generation of 1000 accepted parameters was considered as the posterior distribution from the SMC-ABC algorithm applied to the SIR model. Table 4 shows the resulting tolerance levels, number of proposed parameters and acceptance rate for each generation with the initial tolerance level set to $\epsilon_0 = 200\,000$. The total number of proposed parameters for all generations was 63 000, which is a reduction by more than a factor of 5 from the number of proposed parameters of the ABC rejection algorithm with the same tolerance level as the last generation of SMC-ABC. The Metropolis-Hastings algorithm ran for 30 000 iterations and a burn-in of 15 000 iterations was removed. Figures of the Markov chains can be found in appendix F.1. Resulting posterior distributions of the parameters r and γ in figure 12 are similar between the three algorithms and clearly shows a linear relationship between the accepted values of r and γ.



(a) Metropolis-Hastings, N=15 000 (b) SMC-ABC, N=1 000 (c) ABC rejection, N=1 000

Figure 12: Posterior parameter values of r and γ for accepted parameters of the ABC rejection, last generation of SMC-ABC and Metropolis-Hastings algorithms applied to the SIR model.

Generation	ϵ	Number of proposals	Acceptance rate
1	130 000	2 293	44%
2	116 000	31 462	3.2%
3	64 000	16 398	6.1%
4	34 600	13 410	7.5%

Table 4: Tolerance level ϵ, number of proposed parameters and acceptance rate in each generation of the SMC-ABC algorithm applied to the SIR model.

Figure 13 shows the posterior densities of r, γ and σ. The posterior densities of r and γ are similar between all three algorithms with approximately the same mode of $r = 3.1$ and $\gamma = 2.8$ respectively. This shows that the SMC-ABC and ABC rejection algorithms manage to produce posteriors distributions that are similar to the ones obtained from Metropolis-Hastings even when using real data. The posterior densities of σ are however not as similar between the algorithms. SMC-ABC and ABC rejection produced posterior distributions with lower modes and very few accepted values above 10 000 compared to Metropolis-Hastings. As previously explained this can be associated with the difficulty in estimating the noise in the data when we only have a single time series of the number of infected. It should also be mentioned that the posterior densities of σ from SMC-ABC and ABC rejection was somewhat dependent on the parameters of its lognormal prior distribution. It suggests that the likelihood was not as informative for σ as it was for the other parameters.

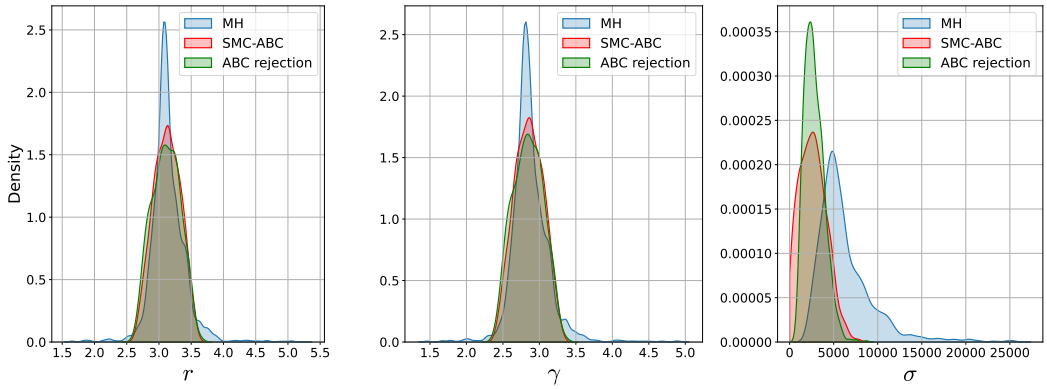
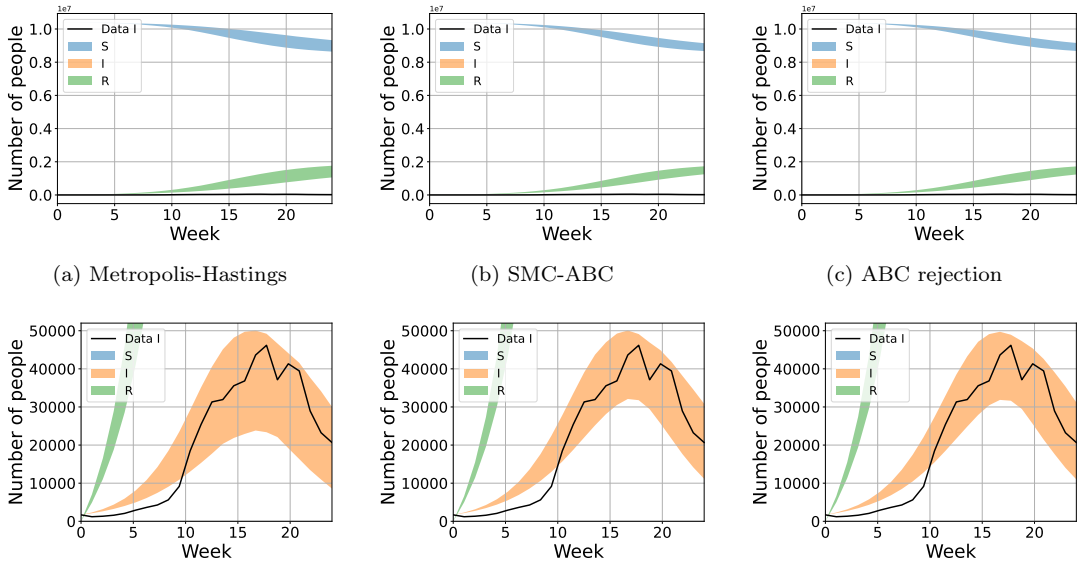


Figure 13: Posterior densities of parameters r, γ and σ from the Metropolis-Hastings (blue) and SMC-ABC (red) algorithms applied to the SIR model. The posterior density curves are kernel density estimates of accepted parameters.



(a) Metropolis-Hastings (b) SMC-ABC (c) ABC rejection

(d) Metropolis-Hastings zoomed in (e) SMC-ABC zoomed in (f) ABC rejection zoomed in

Figure 14: Area enclosed by the posterior trajectories of the groups S, I and R from the Metropolis-Hastings, SMC-ABC and ABC rejection algorithms applied to the SIR model. The 2.5% lowest and 2.5% highest values have been omitted in each time step. The upper figures has a scale of the order 10^7 on the y-axis. The black curve shows the number of new confirmed cases of covid-19 from week 32 in 2020 to week 3 in 2021.

The true posterior sample obtained from Metropolis-Hastings covers a larger interval for all parameters. It might seem counterintuitive that the true posterior is wider than the approximated ones but this can be explained by the difference in number of samples from the posteriors. While the ABC algorithms were set to produce 1 000 accepted parameters because of limitations in computational time, the Metropolis-Hastings algorithm produced 15 000 draws to get a better idea of what the true posterior looks like.

Figure 14 shows the area enclosed by the posterior trajectories of the accepted parameters. The 2.5% lowest and 2.5% highest values of the posterior trajectories have been omitted in each time step to reduce the impact of outliers and show the general trend of posterior trajectories. The areas of S and R are larger, which is natural as we have no data for these groups and therefore greater uncertainty in the resulting inference. When looking at the area enclosed by the posterior trajectories of I and the curve for the observed data, we see that the observed data lies within this area, except for the beginning of the time period. This implies that all algorithms manage to find parameter values that result in posterior trajectories similar to the observed data.

One curious thing is that the particles of the posteriors in figure 12 seem to coincide with a line drawn from approximately $(0.2, 0)$ to $(8, 7.3)$. In fact, looking at the SMC-ABC algorithm result, we see that almost no particles from any generation falls to the right of such a line. This is peculiar since one could imagine that for the initial generations, where ϵ is large, there would be a broader spread. This is related to the mechanics of the SIR model, and is a nice illustration of the SMC-ABC algorithm in action.

In figure 15 we see three solutions to the SIR-model given three different sets of values for r and γ. The left solution corresponds to the mean of the parameters obtained by running the SMC-ABC algorithm on the observed data, $(r, \gamma) = (3.1, 2.8)$. In the centre plot $(r, \gamma) = (4, 1)$, so in figure 12 this would place the particle in the empty bottom right area. Finally, in the right plot, $(r, \gamma) = (1, 4)$ which corresponds to the top left area in figure 12.

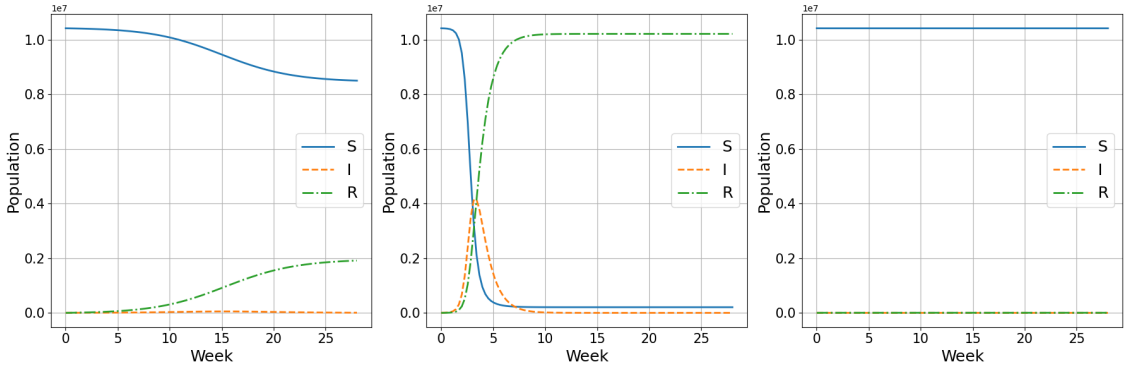
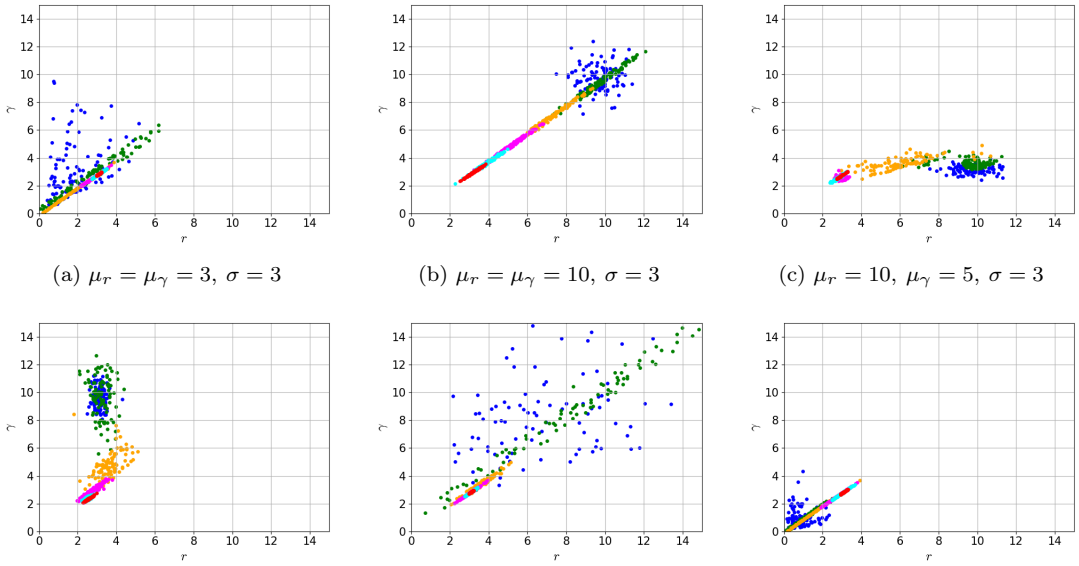


Figure 15: Plots of three solutions to the SIR-model given parameter sets $(r, \gamma) = \{(3.1, 2.8), (4, 1), (1, 4)\}$. The first set of parameters are means of posterior distributions, obtained from running the SMC-ABC algorithm on data provided by the Public Health Agency of Sweden. We see that when the difference between r and γ is large, the curves differ quite a bit from the curves generated by the real data set, especially if $r \gg \gamma$.

As previously mentioned, the SMC-ABC algorithm considers the distance $\rho(S(\mathbf{z}), S(\mathbf{y}))$ when deciding which particles to accept. In this case, that means comparing the distance from the proposed parameter's solution to the value recorded in the data, for each time step. Intuitively this could be seen as comparing the values of the middle and right plots of figure 15 to the values of the left plot. If we consider first the middle plot compared to the left we see that the curves differ greatly. Since $r \gg \gamma$ the epidemic spreads rapidly in the middle plot, resulting in the number of susceptible plummeting and the number of infected skyrocketing. Thus, our distance ρ will be large and rarely accepted. Turning our attention to the right plot we see that the curves are more similar, thus producing smaller values for ρ that in turn are more likely to pass at least the initial values for ϵ.

From this small analysis we can now give an intuitive explanation for the "line" in figure 12. It is simply the border for which the values of r and γ produce solutions to the SIR-model that somewhat resemble the left plot in figure 15. Particles to the left of this line generate solutions resembling the right plot. For earlier generations where ϵ is still somewhat large they are accepted, but as epsilon shrinks the particles are forced to move towards the line. Particles right of the line generate solutions resembling the middle plot. These solutions are so different from the data that even for the initially large ϵ they are not good enough. Thus we realise that the empty bottom corner is actually a result of the SMC-ABC algorithm correctly doing its job. The attentive reader will realise that we have thus learned something important about the model, solely by studying the movement of the SMC-ABC algorithm.

Another thing to consider is the influence of the initial prior distribution on the final posterior for r and γ. In figure 16 we see the results of six realisations for different priors. We see that the final posterior, given a number of generations, always converges to the same location. The one exception to this is if the prior is so wildly inappropriate that the chances of getting even one accepted particle is very low. For example, with a prior multivariate normal distribution with mean $(r = 5, \gamma = 15)$ and standard deviation 1, the algorithm does not get past the first generation. This is due to the fact that getting a draw around the acceptable line is extremely improbable. With this in mind, the results imply that the choice of prior does not influence the resulting r and γ posterior distributions to a large degree.



(a) $\mu_r = \mu_\gamma = 3$, $\sigma = 3$ (b) $\mu_r = \mu_\gamma = 10$, $\sigma = 3$ (c) $\mu_r = 10$, $\mu_\gamma = 5$, $\sigma = 3$

(d) $\mu_r = 5$, $\mu_\gamma = 10$, $\sigma = 3$ (e) $\mu_r = \mu_\gamma = 10$, $\sigma = 8$ (f) $\mu_r = \mu_\gamma = 1$, $\sigma = 2$

Figure 16: Sensitivity analysis ($N = 100$) of the SMC-ABC algorithm working on the FHM dataset. The initial prior (blue) is a multivariate lognormal distribution for r and γ with values given as above. We see that the final posterior (red) after $T = 6$ generations is zeroing in on the same area regardless of whether the initial prior is poorly chosen. This however is only true given an initial ϵ large enough to accept a first generation of particles.

6 Final discussion and conclusions

In conclusion, we would like to consider some of the main points of interest that have been identified in this thesis. The main result is that the ABC methods are well suited to estimating parameters for a wide array of models consisting of both real and simulated data. While the posterior distributions are generally not as tight as the ones generated by the Metropolis-Hastings algorithm, the fact that ABC methods are applicable without a likelihood function make them very useful. The results for the AR and SIR models show that both ABC algorithms can indeed produce posteriors similar to the ones obtained from the Metropolis-Hastings algorithm. The computational time is a concern, but the SMC-ABC extension manages to mitigate this issue quite effectively taking only one fourth and one fifth the number of iterations to get equivalent results for the AR model and SIR model, respectively. In the Lotka-Volterra model, we saw that the SMC-ABC algorithm did not reduce the total number of iterations in relation to the ABC rejection. Instead the SMC-ABC algorithm instead managed to produce considerably more accurate posterior distributions. Regardless, the SMC-ABC algorithm is clearly superior to ABC rejection in all models. Also, it is important to note that all of the models considered in this thesis are fairly simple. For more complex generative models the computational time might still prove bothersome.

A perhaps surprising result is the relative robustness to the prior specification with regards to the parameters of the SIR model. From the sensitivity analysis in figure 16 we see that as long as the prior is good enough to generate a first set of accepted parameters, the sequential nature of the SMC-ABC algorithm and its associated weights will dominate the process and lead the particles towards the same posterior distribution. That being said, it is important to note that for the noise σ, the inference for the SIR model was somewhat dependent on the prior specification. Additionally, the inference for all the parameters were dependent on the initial value of the tolerance ϵ. If ϵ was too small and a poorly calibrated prior distribution was used, the algorithm would not get past the first generation. However, having a larger initial ϵ does not infer any significant negatives. It will quickly be lowered to the selected quantile α_t, as seen in tables 3 and 4.

References

[1] Wiqvist, S., Mattei, P.-A., Picchini, U., and Frellsen, J. "Partially exchangeable networks and architectures for learning summary statistics in approximate Bayesian computation". In: vol. 97. Proceedings of the 36th International Conference on Machine Learning. 2019, pp. 6798–6807.

[2] Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. "Approximate Bayesian computational methods". In: *Statistics and Computing* 22.6 (2012), pp. 1167–1180. URL: https://arxiv.org/abs/1101.0955.

[3] Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, M. P. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *J. R. Soc. Interface* 6 (2009), pp. 187–202. DOI: https://doi.org/10.1098/rsif.2008.0172.

[4] Sundaramoorthy, V. *Man on Sand Dunes Riding Motorcycle*. URL: https://unsplash.com/photos/xtOaajNOe-8. (accessed: 14.05.2021).

[5] Davis, S. *Two Men Driving White Lamborghini Huracan Spyder during Daytime*. URL: https://unsplash.com/photos/2wmZMhJGCr8. (accessed: 14.05.2021).

[6] Argenton, R. T. *Tabuleiros de Galton (antes e depois)*. URL: https://commons.wikimedia.org/wiki/File:Tabuleiros_de_Galton_(antes_e_depois).jpg. (accessed: 20.04.2021).

[7] Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. "Inferring Coalescence Times From DNA Sequence Data". In: *Genetics* 145 (1997), pp. 505–518. DOI: https://doi.org/10.1093/genetics/145.2.505.

[8] Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., and Feldman, M. W. "Population growth of human Y chromosomes: a study of Y chromosome microsatellites". In: *Molecular Biology and Evolution* 16 (1999), pp. 1791–1798. DOI: https://doi.org/10.1093/oxfordjournals.molbev.a026091.

[9] Beaumont, M. A., Cornuet, J.-M., Marin, J.-M., and Robert, C. P. "Adaptive approximate Bayesian computation". In: *Biometrika* 96.4 (2009), pp. 983–990. DOI: https://doi.org/10.1093/biomet/asp052.

[10] Fearnhead, P. and Prangle, D. "Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation". In: *Journal of the Royal Statistical Society* 74 (2012), pp. 419–474. DOI: https://doi.org/10.1111/j.1467-9868.2011.01010.x.

[11] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics Ser. John Wiley & Sons, Incorporated, 2015. ISBN: 9781118674918.

[12] Kermack, W. and McKendrick, A. "Contributions to the mathematical theory epidemics". In: *Proceedings of the Royal Society A* 115 (1927), pp. 700–721. DOI: https://doi.org/10.1098/rspa.1927.0118.

[13] Folkhälsomyndigheten. *Bekräftade fall i Sverige – daglig uppdatering*. URL: https://fohm.se/smittskydd-beredskap/utbrott/aktuella-utbrott/covid-19/statistik-och-analyser/bekraftade-fall-i-sverige/. (accessed: 30.03.2021).

[14] SCB. *Befolkningsprognos Sverige*. URL: https://www.scb.se/hitta-statistik/sverige-i-siffror/manniskorna-i-sverige/befolkningsprognos-for-sverige/. (accessed: 08.05.2021).

[15] Folkhälsomyndigheten. *Om viruset och sjukdomen*. URL: https://www.fohm.se/smittskydd-beredskap/utbrott/aktuella-utbrott/covid-19/om-sjukdomen-och-smittspridning/om-viruset-och-sjukdomen/#inkubationstid. (accessed: 01.03.2021).

[16] Bjørnstad, O. N., Shea, K., Krzywinski, M., and Altman, N. "The SEIRS model for infectious disease dynamics". In: *Nat Methods* 17 (2020), pp. 557–558. DOI: https://doi.org/10.1038/s41592-020-0856-2.

Appendix

A The stochastic Ricker model

The stochastic Ricker model is defined by

$$\mathcal{M}(\theta) = \begin{cases} y_t \sim Poisson(\phi N_t), & t = 1, ..., T \\ N_t = r\, N_{t-1}\, e^{-N_{t-1}+e_t}, & e_t \sim_{iid} \mathcal{N}(0, \sigma^2). \end{cases}$$

It is a state-space model where y_t are observations and N_t is a unobservable process used to describe the change in time of a population N_t. The models likelihood function

$$\pi(y_{1:T}|r, \phi, \sigma) = \int \prod_{t=1}^{T} \big(\pi(y_t|N_t, r, \sigma)\pi(N_t|N_{t-1})\big)\, dN_1 ... \, dN_t,$$

is a high dimensional integral and hence analytically intractable. Generating $y_t \sim \pi(y|r, \phi, \sigma)$ from the unknown likelihood can however be done by implementing a computer model of $\mathcal{M}(\theta)$ and running it at some $\theta = (r, \phi, \sigma)$. Note that simulating y_t from the generative model $\mathcal{M}(\boldsymbol{\theta})$ is equivalent to drawing samples from the likelihood function.

B The sequential Monte Carlo ABC algorithm

The SMC-ABC algorithm begins by generating a set of particles $\boldsymbol{\theta}_i^{(1)}$, where the superscript denotes the generation of particles under consideration, and $i = 1, 2, \ldots, N$. The generative model f is then used to construct simulated data given the $\boldsymbol{\theta}_i^{(1)}$'s. If the distance function ρ, given the simulated data $\mathbf{z}$ and real data $\mathbf{y}$, falls within our first tolerance ϵ_1, the parameters are accepted. Once the vector $\boldsymbol{\theta}^{(1)} = (\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_N^{(1)})$ is complete, the weight vector $\omega^{(1)} = \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$ is instantiated. The constant α_t is also set to a suitable percentile of all values produced by ρ. At this point the sequential nature of SMC-ABC begins.

For $T-1$ generations, new $\boldsymbol{\theta}_i^*$ are randomly sampled with replacement from the previous generation $\boldsymbol{\theta}_j^{(t-1)}$ with probabilities given by the weights $\omega^{(t-1)}$. These are then used as means for a new generation of particles $\boldsymbol{\theta}^{(t)}$. Each value $\boldsymbol{\theta}_i^{(t)} \in \boldsymbol{\theta}^{(t)}$ can be seen as a perturbed version of some particle in $\boldsymbol{\theta}^{(t-1)}$, whose generated summary statistic falls inside the stricter tolerance ϵ_t. Since ϵ_t is given by taking a suitably high percentile of the shortest distances given by ρ each generation, ϵ_t will monotonically decrease, thus improving the generated particles for each generation.

Once a particle $\boldsymbol{\theta}_i^{(t)}$ generates data that passes the acceptance criterion ϵ_t, its corresponding weight $\omega_i^{(t)}$ is computed. This weight will depend on how good of an approximation the parameter $\boldsymbol{\theta}_i^{(t)}$ is. Thus, $\boldsymbol{\theta}_i^{(t)}$'s with large weights will have a higher probability of being picked for the next generation. This process further increases the rate by which each generation converges towards the posterior distribution. One should note that the function $\varphi(\boldsymbol{\theta}_i^{(t)}; \boldsymbol{\theta}_j^{(t-1)}, \Sigma_{t-1})$ represents a multivariate normal probability density function evaluated at $\boldsymbol{\theta}_i^{(t)}$, given mean $\boldsymbol{\theta}_j^{(t-1)}$ and covariance matrix Σ_{t-1}.

C The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a so called Markov Chain Monte Carlo (MCMC) algorithm, which produces random samples $\boldsymbol{\theta}$ from a probability posterior distribution. The advantage of sampling with the Metropolis-Hastings algorithm is that the posterior distribution does not need

to be theoretically derived, which is useful for multidimensional distributions.

Algorithm 3: Metropolis-Hastings

Set $\boldsymbol{\theta}_1$ to a valid arbitrary value
for *n = 2 to N* **do**
  Generate $\boldsymbol{\theta}^* \sim g(\boldsymbol{\theta}^*|\boldsymbol{\theta}_{n-1})$
  Calculate $A(\boldsymbol{\theta}^*, \boldsymbol{\theta}_{n-1}) = \min\left(1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\mathbf{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}_{n-1})\pi(\mathbf{y}|\boldsymbol{\theta}_{n-1})} \frac{g(\boldsymbol{\theta}_{n-1}|\boldsymbol{\theta}^*)}{g(\boldsymbol{\theta}^*|\boldsymbol{\theta}_{n-1})}\right)$
  Set $u \sim \mathcal{U}(0,1)$
  if $u \leq A$ **then**
    Set $\boldsymbol{\theta}_n = \boldsymbol{\theta}^*$
  else
    Set $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1}$
  end
end

The algorithm constructs a Markov chain where the limiting and stationary distribution is the desired posterior distribution. The draws in $\boldsymbol{\theta}$ can be considered drawn from the posterior distribution after a sufficient amount of iterations. The process of finding parameter values $\boldsymbol{\theta}$ belonging to the posterior distribution is called the burn-in process. The better our initial parameter value $\boldsymbol{\theta}_1$ is, the shorter is the burn-in process. The algorithm is using a proposal distribution $g(\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1})$ to generate new samples of $\boldsymbol{\theta}$. A ratio between the posterior probability of the proposed value $\boldsymbol{\theta}^*$ and previous value $\boldsymbol{\theta}_{n-1}$ is then calculated and stored as $A \in [0,1]$. The newly proposed value $\boldsymbol{\theta}^*$ is accepted if the generated $u \sim \mathcal{U}(0,1)$ is less than or equal to the posterior quota A. Otherwise, the Markov chain will not move and stay at the old value $\boldsymbol{\theta}_{n-1}$. The algorithm then performs a total of N iterations.

Implementing the algorithm for each investigated model, the initial $\boldsymbol{\theta}_1$ value was sampled from the prior distribution $\pi(\boldsymbol{\theta})$. The chosen proposal distribution $g(\boldsymbol{\theta}|\boldsymbol{\theta}_{n-1})$ was a normal distribution $\mathcal{N}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\Sigma})$, which uses the previous parameter value $\boldsymbol{\theta}_{n-1}$ as mean and an fixed variance $\boldsymbol{\Sigma}$.

D Theoretical derivations in the autoregressive model

The range of possible $\boldsymbol{\theta}$ values and the likelihood function of the AR model are derived below in section D.1 respectively D.2.

D.1 Derivation of the prior

To derive the range of possible $\boldsymbol{\theta}$ values are we first required to express the AR(p) model with the so called lag or backshift operator L defined as

$$Ly_j = y_{j-1},$$

where $\{y_j\}_{j\in\mathbb{N}}$ is a time series. We can rewrite (4) with the backshift operator to

$$\xi_j = \underbrace{(1 - \theta_1 L - \theta_2 L^2)}_{\theta(L)} y_j,$$

where the expression in the parentheses is the lag polynomial $\theta(L)$. Setting the lag polynomial $\theta(L) = 0$ gives the series characteristic equation, which can be used to decide if the series is stationary or not. If the norm of both roots of the characteristic equation is larger than 1, then the series is stationary, as stated in Box *et al.* [11]. This grants us time series values y_j which are varying around a fixed mean with a constant variance and therefor make Bayesian inference easier. By solving the equation

$$1 - \theta_1 L - \theta_2 L^2 = 0,$$

we get by the quadratic-formula with roots

$$z_1, z_2 = \frac{-\theta_1 \pm \sqrt{\theta_1^2 + 4\theta_2}}{2\theta_2}.$$

Now we use the stationary condition, which gives the inequality

$$\left| \frac{-\theta_1 \pm \sqrt{\theta_1^2 + 4\theta_2}}{2\theta_2} \right| > 1.$$

Solving for θ_1 and θ_2 will give the following constraints

$$\theta_2 < 1 + \theta_1, \quad \theta_2 < 1 - \theta_1, \quad \theta_2 > -1,$$

which gives us a triangular area of possible parameters.

D.2 Derivation of the likelihood function

To derive the likelihood function $\pi(\mathbf{y}|\boldsymbol{\theta})$ from the AR(2) model, we first look at one single data point. The likelihood given one data point y_j of the AR(2) model can be derived by rewriting equation (4) to

$$\xi_j = y_j - \theta_1 y_{j-1} - \theta_2 y_{j-2}.$$

We know that $\xi_j \sim \mathcal{N}(0, 1)$, which means that

$$y_j - \theta_1 y_{j-1} - \theta_2 y_{j-2} \sim \mathcal{N}(0, 1).$$

In other words, the likelihood of data point y_j is acquired by evaluating $\mathcal{N}(y_j - \theta_1 y_{j-1} - \theta_2 y_{j-2}; 0, 1)$. Because every data point y_j is independent and identically distributed (i.i.d.) we can write the likelihood function as

$$\pi(\mathbf{y}|\boldsymbol{\theta}) = \mathcal{N}(y_0; 0, 1) \cdot \mathcal{N}(y_1 - \theta_1 y_0; 0, 1) \cdot \prod_{j=2}^{M} \mathcal{N}(y_j - \theta_1 y_{j-1} - \theta_2 y_{j-2}; 0, 1),$$

where M is the length of the time series.

E The Lotka-Volterra model

E.1 True and observed data of prey and predator populations

The true prey and predator populations, i.e. the numerical solution to the Lotka-Volterra model given the true value of parameters $a = 1$ and $b = 1$, are represented by the curves seen in figure 17. The observed data, on which all inference is based, is discretised data from these curves that is subject to a noise terms $\xi_t \sim \mathcal{N}(0, 0.5^2)$, as represented by the dots and boxes seen in figure 17. In order for the data to reflect realistic observations all observed data points are conditioned to be non-negative.

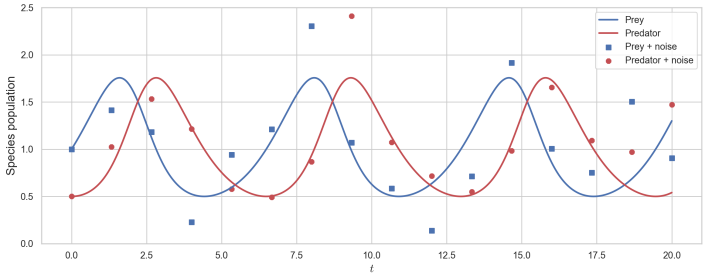


Figure 17: Data used in the Lotka-Volterra model. The true prey and predator populations are illustrated by the blue and red curves, respectively. The observed data used for inference is illustrated by the blue boxes and red dots, corresponding to observed prey and predator populations, respectively.

E.2 Results from the Metropolis-Hastings algorithm

When using the Metropolis-Hastings algorithm with 10 000 iterations and a burn-in period of 2 500 iterations on the Lotka-Volterra model, as discussed in section 4, we obtain a posterior distribution in the a-b plane as seen in figure 18. We see that all visited states are in the first quadrant. Another visualisation of this result is displayed in figure 19 where the Markov chains of the different parameters produced by the Metropolis-Hastings algorithm is shown. As seen, the Markov chains of a and b are varying in a limited interval and display stationary behaviour, whilst σ is more scattered and covers a larger proportion of the prior interval.

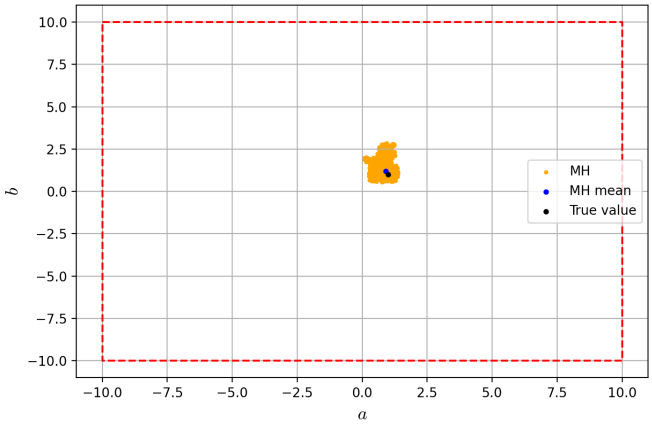


Figure 18: Parameter values of a and b visited by the Metropolis-Hastings algorithm set to 10 000 iterations applied to the Lotka-Volterra model with a burn-in period of 2 500 iterations. Parameter a is displayed on the x-axis and parameter b is displayed on the y-axis. The red line shows the boundaries of the uniform priors of a and b. The blue dot shows the mean value of all visited parameter values and the black dot shows the true parameter values.

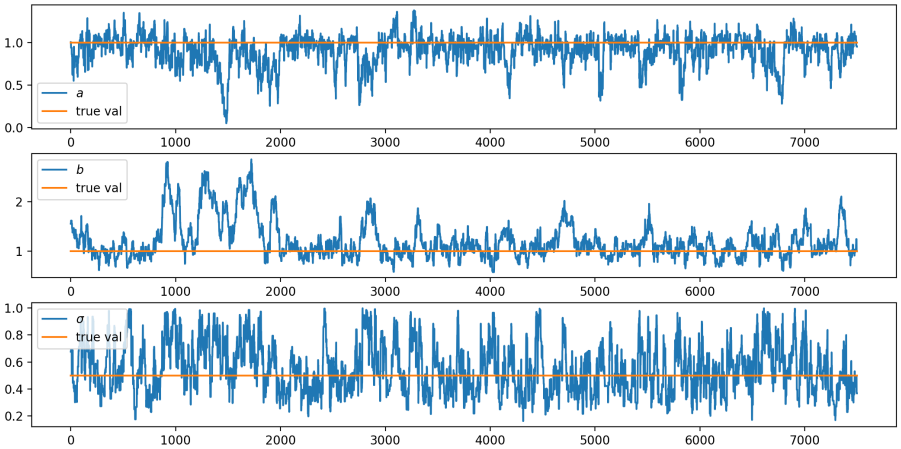


Figure 19: Markov chains of parameters values by the Metropolis-Hastings algorithm set to 10 000 iterations applied to the Lotka-Volterra model with a burn-in period of 2 500 iterations. The orange line shows the true value of the respective parameter.

In figure 20 we see the area enclosed by the posterior trajectories of parameter values obtained by the Metropolis-Hastings algorithm. The 2.5% lowest and 2.5% highest values have been omitted in each time step. We see that the posterior trajectories in figure 20 are quite close to the true population sizes, without showing any signs of extinction of either species.

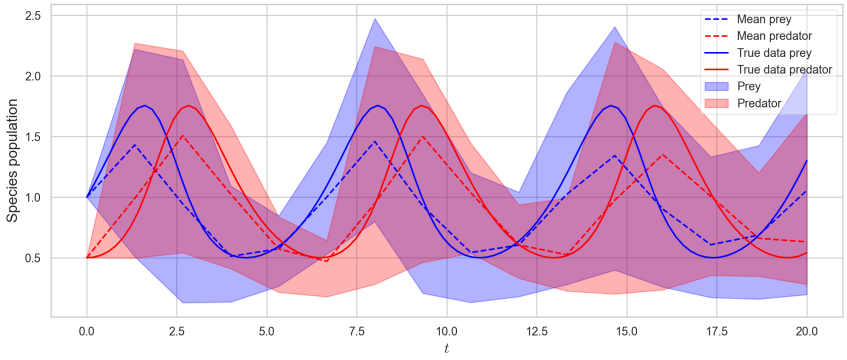


Figure 20: Posterior trajectories of Metropolis-Hastings algorithm applied to the Lotka-Volterra model without added noise. The 2.5% lowest and 2.5% highest values have been omitted in each time step. The curves denoted "true data prey" and "true data predator" is the true data without added noise. The blue and red areas enclose all solutions of prey and predator populations, respectively, to the Lotka-Volterra model given parameter values produced by the algorithm. The dashed curves are mean values of all posterior trajectories of a given population in each time step and is not based on a mean of parameter values.

E.3 Complementary details of posterior distributions

In figures 21 and 22 we see complementary details to figure 9 in section 4.3 of the posterior distributions of the parameters in the Lotka-Volterra model produced by the different algorithms. The posterior probability density curves are kernel density estimates of accepted parameter values generated by the function `kdeplot()` as part of the `seaborn` library in Python.

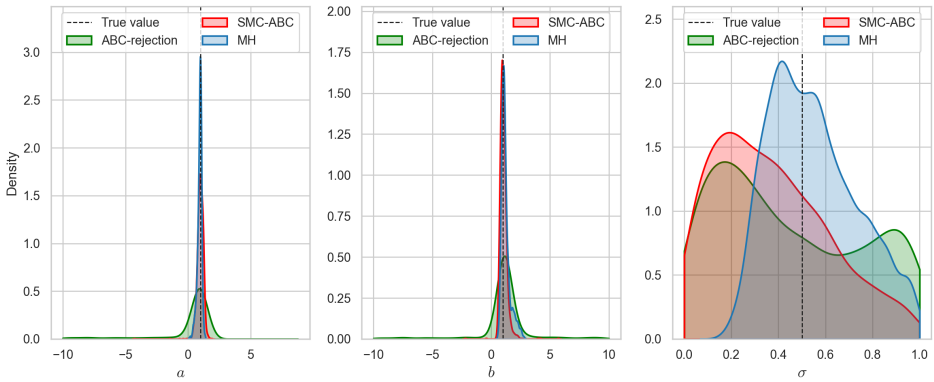


Figure 21: Complete posterior densities of parameters a, b and σ produced by the different algorithms applied to the Lotka-Volterra model. The true value of each parameter is displayed by a dashed line.

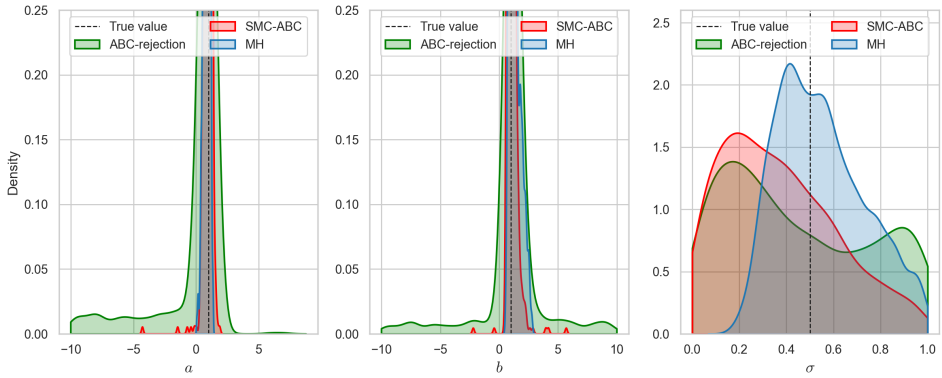


Figure 22: Posterior densities of parameters a, b and σ produced by the different algorithms applied to the Lotka-Volterra model. The curves were generated by a kernel density estimate of accepted parameter values. To display the spread of the posterior density functions, the density axis is scaled to show values close to 0. The true value of each parameter is displayed by a dashed line.

E.4 The complete posterior trajectories

The area enclosed by all posterior trajectories corresponding to accepted parameter sets in the ABC rejection algorithm and the last generation of the SMC-ABC algorithm, as well as all parameter pairs sampled by the Metropolis-Hastings algorithm, and their mean values can be seen in figure 23.

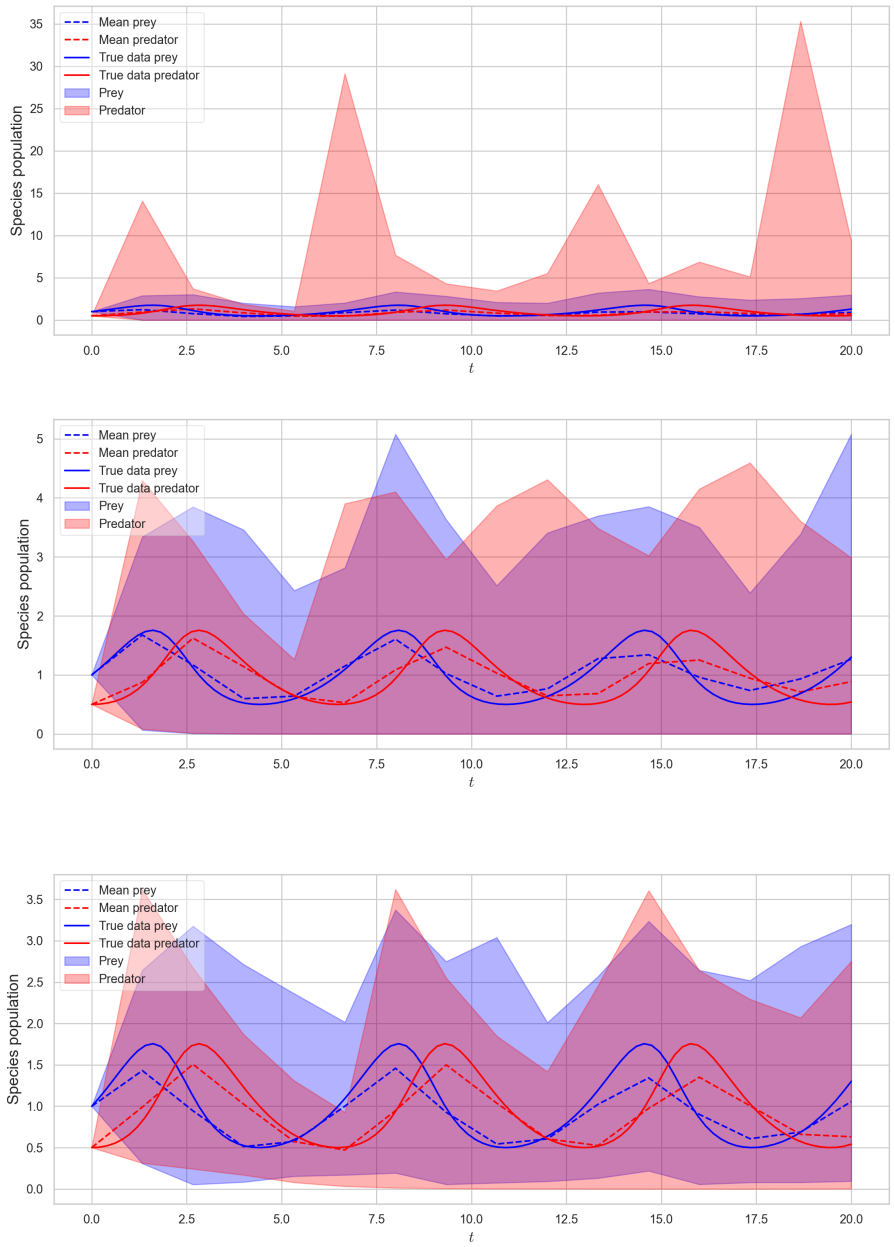


Figure 23: Posterior trajectories of the ABC rejection algorithm (top), the SMC-ABC algorithm (middle) and the Metropolis-Hastings algorithm (bottom) applied to the Lotka-Volterra model without added noise. The curves denoted "true data prey" and "true data predator" are the true prey and predator populations without added noise, whilst the blue and red areas enclose all posterior trajectories of prey and predator populations, respectively. The dashed curves are mean values of all posterior trajectories of a given population in each time step and is not based on a mean of parameter values.

F The SIR model

F.1 Markov chains

Figure 24 shows the Markov chains obtained from the Metropolis-Hastings algorithm applied to the SIR model. The number of iterations was set to 30 000 and a burn-in of 15 000 iterations was removed.

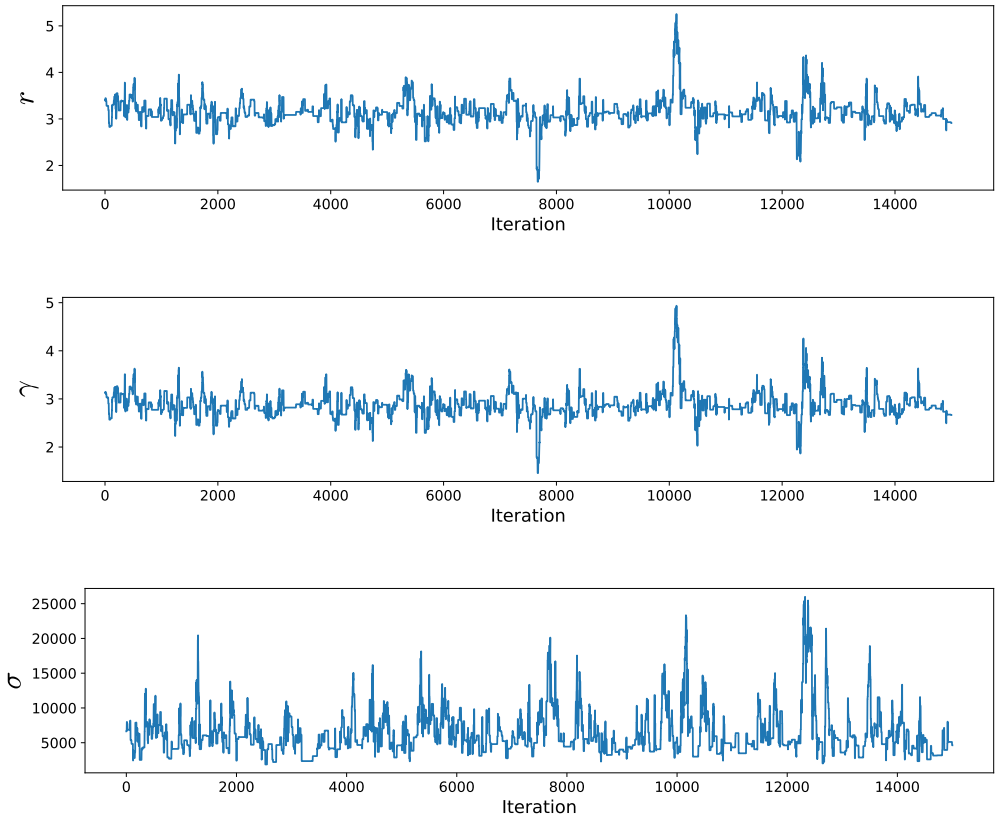


Figure 24: Markov chains of parameter values by the Metropolis-Hasting algorithm set to 30 000 iterations applied to the SIR model. A burn-in of 15 000 iterations was removed.

F.2 Expansions of the SIR model

The basic SIR model can be amended and extended to consider specific characteristics of a disease and society, such as migration, births, deaths of other causes, incubation period, distinction between recovered and deceased, as well as limited immunity after recovery. In the context of the covid-19 epidemic in Sweden and with information available from the Public Health Agency of Sweden [15] in mind, relevant extensions to consider could be incubation period, limited immunity after recovery and distinction between recovered and deceased individuals. To expand the SIR model we consider Bjørnstad *et al.* [16] with an additional set D representing the deceased

individuals in the population and thus obtain

$$\begin{aligned}
\frac{dS}{dt} &= -rSI + \omega R \\
\frac{dE}{dt} &= rSI - \alpha E \\
\frac{dI}{dt} &= \alpha E - \gamma I - \mu I \\
\frac{dR}{dt} &= \gamma I - \omega R \\
\frac{dD}{dt} &= \mu I
\end{aligned}$$

where E is the set of individuals having been exposed to the virus but do not yet have the ability to infect others, α is the incubation rate, μ is the mortality rate and ω is the rate of loss of immunity. Note that

$$\frac{dS}{dt} + \frac{dE}{dt} + \frac{dI}{dt} + \frac{dR}{dt} + \frac{dD}{dt} = 0,$$

i.e. that we are considering a fixed population of size $S + E + I + R + D = N_P$.