UNIVERSITY OF
GOTHENBURG

Delayed-acceptance approximate Bayesian computation Markov chain Monte Carlo: faster simulation using a surrogate model

Master's thesis in Mathematical Statistics

ANDREA KROGDAL

Department of Mathematical Sciences
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019:NN

Delayed-acceptance approximate Bayesian computation Markov chain Monte Carlo: faster simulation using a surrogate model

ANDREA KROGDAL

UNIVERSITY OF
GOTHENBURG

Department of Mathematical Sciences
Division of Mathematical Statistics
University of Gothenburg
Gothenburg, Sweden 2019

Delayed-acceptance approximate Bayesian computation Markov chain Monte Carlo: faster simulation using a surrogate model

ANDREA KROGDAL

Supervisor: Umberto Picchini, Department of Mathematical Sciences
Examiner: Petter Mostad, Department of Mathematical Sciences

Master's Thesis 2019:NN
Department of Mathematical Sciences
Division of Mathematical Statistics

University of Gothenburg
SE-412 96 Gothenburg

Delayed-acceptance approximate Bayesian computation Markov chain Monte Carlo: faster simulation using a surrogate model

ANDREA KROGDAL
Department of Mathematical Sciences
University of Gothenburg

Abstract

The thesis introduces an innovative way of decreasing the computational cost of approximate Bayesian computation (ABC) simulations when implemented via Markov chain Monte Carlo (MCMC). Bayesian inference has enjoyed incredible success since the beginning of 1990's thanks to the re-discovery of MCMC procedures, and the availability of performing personal computers. ABC is today the most famous strategy to perform Bayesian inference when the likelihood function is analytically unavailable. However, ABC procedures can be computationally challenging to run, as they require frequent simulations from the data-generating model. In this thesis we consider learning a so-called "surrogate model", one that is cheaper to simulate from, compared to the assumed data-generating model, and in this manner save computational time. The strategy implemented is known in MCMC literature as "delayed acceptance MCMC", however to the best of our knowledge has not been previously adapted into an ABC framework. Simulation studies consider the approach on two different models, producing Gaussian data and g-and-k distributed data, respectively. For the most challenging example we observed that our approach, consisting in a delayed-acceptance ABC algorithm, led to a 20-folds acceleration in the MCMC sampling, compared to a standard ABC-MCMC algorithm.

Keywords: ABC, MCMC, delayed acceptance, DA, surrogate model.

Acknowledgements

Throughout the work on my master thesis I have received a tremendous amount of support and assistance from my supervisor as well as my friends and family.

First, I would like to express my deepest appreciation to the person who have played by far the most important role during my work, my supervisor professor Umberto Picchini. He has contributed not just with his wide range of knowledge in the area of my thesis, but he has also shown an incredible ability to support me while facing several challenges during this process. I am truly grateful that I was fortunate to have Umberto as my supervisor. Thank you for everything.

I would also like to acknowledge my friend Abraham Deniz for using his precious time to proofread my work, thank you.

Last but not least, my friends and my family. Without you I would not have survived this. Thank you for always being there.

Andrea Krogdal, Gothenburg, December 2019

Contents

1

Introduction

1.1 Background

Approximate Bayesian computation (ABC) today is a big research area due to its increasing popularity. This is because ABC provides using Bayesian inference when the likelihood is intractable. The likelihood is often intractable when the model is complex, which is often the case in real data application. Even if ABC comes around targeting the posterior distribution without the use of the likelihood function, it can be computationally very inefficient. The aim of this thesis is to introduce a more computationally efficient ABC method, called Delayed acceptance approximate Bayesian computation (DA-ABC). The main purpose of using DA-ABC is when ABC methods are computationally heavy.

ABC methods have been used in several application areas e.g. the first works were in population genetics by [1] and [2]. More examples are in astronomy by [3], ecology by [4], systems biology by [5] and finance by [6]. More specifically, this thesis is focusing on Approximate Bayesian computation Markov chain Monte Carlo (ABC-MCMC), which is using the Metropolis-Hastings sampler. Due to the fact of not using the likelihood function in ABC-MCMC methods, it will not be able to target the exact posterior distribution, but will instead provide an approximation of it. Let x_0 be observed data believed to come from the model $p(x|\theta)$, where θ is an unknown parameter vector. Then ABC can provide draws approximately from the posterior distribution, $\pi(\theta|x)$. This is done by proposing a parameter vector θ^* via e.g. a transition kernel, then generate a data-set with this proposed parameter vector, $x^* \sim p(x|\theta^*)$. Continue by calculating the distance between the generated data-set and the observed data-set, $\rho(x^*, x_0)$ for some distance function ρ. If this distance is smaller than a pre-defined threshold ε, θ^* is accepted as a sample of the posterior distribution with probability α. The acceptance rate of the proposed θ^* is low, and is in best case scenarios around 1%. This can result in computationally heavy simulations if the model p is complex. Due to this low acceptance rate, we introduce the Delayed acceptance ABC-MCMC which will instead use a surrogate model, which will at a first step evaluate whether θ^* is a good proposal. Only if θ^* seems as a good proposal, a data-set will be generated from the model p. In this manner, we hope to avoid simulating from the model unnecessarily i.e. when θ^* is likely to be rejected. It is important that the surrogate model is cheaper to evaluate than the real model p.

1.2 Outcome

This thesis is divided into sections. In section 2, the theory behind the first ABC method is described, namely Approximate Bayesian computation rejection, and then followed by the theory behind ABC-MCMC. Further in the section is the theory behind the delayed acceptance approach and how it is merging with ABC-MCMC. In this section, also a pseudo code for DA-ABC-MCMC will be introduced. Moreover, section 3 contains some implementation used in the algorithms for both ABC-MCMC and DA-ABC-MCMC, which will be taken for granted in the next following sections. The section will also give an introduction of diagnostic plots and why the interpretation and analysing of them is important for the aim of this thesis. Section 4 is the implementation of DA-ABC-MCMC for Gaussian distribution, which is considered as a simple case. This is just a first step of interpretation, if DA-ABC-MCMC is a suitable method for computationally efficiency purpose, before moving on to a more complex case. Section 5 contains the implementation of DA-ABC-MCMC for a more complex case, namely of g-and-k distribution. Finally, section 6 will contain a discussion and conclusion around the implementation and interpretation. Also, a discussion about further research is included.

2

Theory

2.1 Approximate Bayesian Computation, ABC

Exact Bayesian makes use of the fact that the posterior distribution is proportional to the prior distribution multiplied with the likelihood function. This means we have the prior function $\pi(\theta)$ for each parameter in the parameter vector $\theta \in \Theta$. Thereafter we have observed data $x_0 \in \mathcal{X}$, believed to come from a model having likelihood $p(x|\theta)$. We update the prior $\pi(\theta)$ via the likelihood function $p(x|\theta)$ and the posterior can be expressed as $\pi(\theta|x) \propto \pi(\theta)p(x|\theta)$. Now, the posterior can be used for Bayesian inference of θ. This method encounter problems if the likelihood function is analytically or computationally intractable, which is often the case, especially if the model is complex. The model is often very complex in real-data applications, due to the high interest of finding a posterior without needing a likelihood function. Approximate Bayesian computation (ABC), also called likelihood-free computation, provides a way to simulate draws from the posterior when the likelihood is intractable.

2.1.1 ABC-Rej Algorithm

What lays ground to the concept of ABC methods is the ABC-rejection algorithm, which was first introduced by [1]. It is also the simplest method and works basically; propose a candidate parameter vector θ^* via a proposal function, usually the prior $\pi(\theta)$, and simulate a synthetic data-set from the model given the proposed parameters $x^* \sim p(x|\theta^*)$. If $x^* \approx x_0$, assume θ^* are parameters which could describe the observed data and θ^* is kept and accepted as a part of the posterior distribution. Reversely, if x^* do not seemed to describe the observed data x_0, θ^* is rejected. The parameter vectors which are accepted due to the observed data can be considered as draws from the approximate posterior distribution. Algorithm 1 shows this simulation in pseudo code.

Algortihm 1: ABC-Rej

1. $\theta^* \sim \pi(\theta)$, propose a parameter vector.
2. $x^* \sim p(x|\theta^*)$, generate a synthetic data-set from the model, given the proposed parameters.
3. If $x^* \approx x_0$ accept θ^* as a part of the posterior distribution.

There is a lot of different methods how to decide whether $x^* \approx x_0$ or not. [1] defined it as: if a distance d between the generated data and the observed data is smaller than a pre-defined threshold ε then accept θ^*. The distance function $\rho(x^*, x_0)$ can for instance be euclidean, but there are other used distances e.g. [7]. Later, [2] introduced a way of calculating the distance d between summery statistics instead, $d = \rho(S(x^*), S(x_0))$, which is now the most used approach, especially since it is more efficient when the observed data have a high sample size or the model is complex. Including this distance, we are not able to find the exact posterior, but we are able to find an approximation of the marginal posterior, defined as

$$\pi(\theta|\rho(S(x^*), S(x_0)) \leq \varepsilon) \propto \pi(\theta) \int_{\mathcal{X}} \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon) p(x|\theta)\, \mathrm{d}x,$$

where $\mathbb{1}$ is the indicator function. Depending on which value ε is set to, the posterior is more or less precise approximated.

2.1.2 ABC-MCMC Algorithm

The ABC-rejection algorithm is simple but ineffective. The acceptance rate is very low and can even be zero. There are several ABC algorithms which have been shown more effective, but they are build from the same principle. One of the most used one is the ABC-MCMC algorithm, which uses Metropolis-Hastings sampler to target the posterior distribution $\pi(\theta|\rho(S(x^*), S(x_0)) \leq \varepsilon)$. MCMC stands for Markov chain Monte Carlo and was first introduced in ABC methods by [8]. ABC-MCMC algorithm starts with sampling a proposal parameter vector θ^* from a proposal function q, where q is acting as a transition kernel. One uses θ^* to generate a data-set x^* from the model, $x^* \sim p(x|\theta^*)$. Retain θ^* as part of the posterior distribution by Metropolis-Hasting approach i.e. in short terms, this means accepting θ^* with probability α where

$$\alpha = \min\left\{ 1\,, \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon) \frac{\pi(\theta^*) q(\theta|\theta^*)}{\pi(\theta) q(\theta^*|\theta)} \right\} \tag{2.1}$$

For details to this conclusion of how to calculate the probability α, see [9]. The pseudocode for ABC-MCMC algorithm is shown in algorithm 2.

Algorithm 2: ABC-MCMC

1. $\theta_1, i = 1$, set starting values.
2. $\theta^* \sim q(\theta|\theta_i)$, propose a new parameter vector via a proposal function.
3. $x^* \sim p(x|\theta^*)$, generate a synthetic data-set from the model given the proposed parameter vector.
4. With probability,

$$\alpha = \min\left\{ 1 \, , \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon) \frac{\pi(\theta^*) q(\theta_i|\theta^*)}{\pi(\theta_i) q(\theta^*|\theta_i)} \right\}$$

   let $\theta_{i+1} = \theta^*$, otherwise let $\theta_{i+1} = \theta_i$.
5. $i = i + 1$, go back to step 1. Stop after desired N iterations.

By looking at both ABC-Rej and ABC-MCMC algorithms at the step where θ^* is proposed, ABC-MCMC compared to ABC-Rej is proposing via a transition kernel q. This is of benefit for ABC-MCMC since the next proposal parameter vector θ^* is based on the last accepted θ, and will be more likely to explore in the higher probability areas of the distribution and less likely to spend time in the low probability areas of the distribution. In this manner, ABC-MCMC saves time compared to ABC-Rej. Altough this can come with difficulties, e.g. it encounter problems when it gets stuck in areas and are not able to capture the whole distribution. For ABC-MCMC, there is in need of setting a starting value θ_1, which can be a sensitive choice for the algorithm.

2.2 Delayed acceptance, Approximate Bayesian computation, DA-ABC

Here we introduce a delayed acceptance approach in ABC-MCMC algorithms. The idea with delayed acceptance is to postpone the evaluation of the computationally expensive model $p(x)$ and obtain faster ABC-MCMC simulation.

2.2.1 Introducing Delayed Acceptance for Metropolis Hastings Algorithm

The Metropolis-Hastings algorithm is a tool used for producing samples where direct sampling from a model $p(x)$ is difficult or impossible and is a common sampling method in MCMC algorithms. The model $p(x)$ is not necessarily the likelihood in Bayesian inference, but we will in this section consider sampling from a generic distribution with density $p(x)$.

The Metropolis-Hastings algorithm works, as you propose a move x^* and then accepting or rejecting the move with probability α. You propose the move x^* via a transition kernel $q(x^*|x)$, given the last accepted move. The accepted moves creates a Markov chain with $p(x)$ as the stationary distribution. In a simulation point of view, imagine we are at iteration i. Then the following steps are done:

- Sample $x^* \sim q(x|x_i)$, from the proposal distribution.
- Calculate the acceptance probability

$$\alpha = \min\left\{ 1\,, \frac{p(x^*)q(x_i|x^*)}{p(x_i)q(x^*|x_i)} \right\} \tag{2.2}$$

- With probability α, set $x_{i+1} = x^*$, otherwise set $x_{i+1} = x_i$.

This means it is of interest sampling from the distribution $p(x)$ in every move, both accepted and rejected. Assume the evaluation of $p(x)$ is computationally expensive, e.g. because the data-set x is big, then this sampling will be inefficient. The DA approach wants to come around sampling from $p(x)$ in every iteration and only when the proposed x^* is a good candidate and in that way save computational time. DA suggests splitting the acceptance probability stage α into two stages, α_1 and α_2. A proposed move is only accepted if it goes through both acceptance stages in chronological order. In the first stage, we only evaluate a surrogate model $p^s(x)$ which can be deterministic or stochastic and cheaper to evaluate than $p(x)$. Again, assume we are at iteration i. The DA approach together with Metropolis-Hastings would in a step-wise manner be done as:

- Sample $x^* \sim q(x|x_i)$, from the proposal distribution.
- Calculate the acceptance probability at stage 1,

$$\alpha_1 = \min\left\{ 1\,, \frac{p^s(x^*)q(x_i|x^*)}{p^s(x_i)q(x^*|x_i)} \right\} \tag{2.3}$$

- With probability α_1, move to stage 2, otherwise set $x_{i+1} = x_i$ and start over.

$$\alpha_2 = \min\left\{ 1\,, \frac{p(x^*)p^s(x_i)}{p(x_i)p^s(x^*)} \right\} \tag{2.4}$$

- With probability α_2, set $x_{i+1} = x^*$, otherwise set $x_{i+1} = x_i$.

Only if the proposed x^* gets accepted at α_1, it is evaluated on the computational heavy function $p(x)$ and it is first when x^* gets accepted at stage 2, it is really accepted. In this manner, we hope to avoiding to evaluate the possibly expensive $p(x)$, when the proposed x^* is not a good candidate. And since the surrogate model will target the intended distribution $p(x)$ exactly, the Markov Chain will still have $p(x)$ as stationary distribution.

This way of splitting the acceptance probability into two stages has been introduced in [10]. Note that α_1 in (2.3) is exactly as α in (2.2), except for using the surrogate models p^s ratio instead of the model p. Also, since the transition kernels ratio is already used in α_1, there are no need to include it in the second acceptance stage α_2.

2.2.2 Introducing Delayed Acceptance for ABC-MCMC, DA-ABC-MCMC

The theory for the DA approach seems as a good strategy for saving computational time as long as the surrogate model p^s is of good choice. In this thesis, we want to learn a specific surrogate model for each case in ABC inference. Before moving on in detail how this will be accomplished, we first take a look how the DA approach will look as step-wise in ABC-MCMC inference by combining the ABC-MCMC algorithm with the DA approach for Metropolis-Hastings algorithm. Let $p(x)$ in section 2.2.1 be the model in Algorithm 2. By just combining the DA approach in Metropolis Hastings algorithm, explained in the previous section 2.2.1, together with Algorithm 2 and assume we are at iteration i, the goal is to accomplish the following steps:

1. $\theta^* \sim q(\theta|\theta_i)$, propose a new parameter vector via a proposal function.
2. $\hat{x}^* \sim p^s(x|\theta^*)$, generate a synthetic data-set from the surrogate model given the proposed parameter vector.
3. With probability,

$$\alpha_1 = \min\left\{ 1 \, , \mathbb{1}(\rho(S(\hat{x}^*), S(x_0)) \leq \varepsilon) \frac{\pi(\theta^*) q(\theta|\theta^*)}{\pi(\theta) q(\theta^*|\theta)} \right\} \tag{2.5}$$

   go to next step, otherwise set $\theta_{i+1} = \theta_i$ and start over.
4. $x^* \sim p(x|\theta^*)$, generate a synthetic data-set from the true model given the proposed parameter vector.
5. With probability

$$\alpha_2 = \min\left\{ 1 \, , \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon) \right\} \tag{2.6}$$

   let $\theta_{i+1} = \theta^*$, otherwise let $\theta_{i+1} = \theta_i$.

Notice that the synthetic data-set simulated by the surrogate model in step 2 is only used in calculating the distance $d = \rho(S(\hat{x}^*), S(x_0))$. Instead of finding a surrogate model which can simulate a whole data set similar to the real model, this thesis introduce using the surrogate model to predict the distance $d = \rho(S(\hat{x}^*), S(x_0))$ given the proposal parameter vector θ^* instead. Hence, this imply it is important that $p^s(x)$ covers the support of $p(x)$, otherwise this approach would fail. The prediction of the distance can be done by for example using regression analysis, as long as it is cheap to evaluate. In this thesis, let the surrogate model described by a

regression model be denoted as $p^s_\phi(S(x_0)|\theta)$, since it is depending on the originate summery statistics $S(x_0)$ and the parameters of interest are the parameter vector θ. ϕ denotes the corresponding regression model parameters. For example, if the regression model is linear regression, then the distance could be predicted by the following formula:

$$d_i = \rho(S(x_0), S(x_i)) = \beta_0 + \beta_1\theta_{i,1} + ... + \beta_b\theta_{i,b} + \delta_i,$$

$$\text{for} \quad i = 1, ..., M \quad \text{and} \quad \delta_i \sim \mathrm{N}(0, \nu^2).$$

$\phi = (\beta_0, \beta_1, ..., \beta_b)$ are the regression parameters and δ is the error term. Once $\hat{\phi}$ is obtained, we can denote $\hat{d}_i$ as the following:

$$\hat{d}_i = \hat{\beta}_0 + \hat{\beta}_1\theta_{i,1} + ... + \hat{\beta}_b\theta_{i,b}.$$

Using regression analysis requires some training data of size M. Imagine using the regular ABC-MCMC (algorithm 2) to collect training data with response variable $\{d_m\}_{m=1}^M$ and with the corresponding covariates $\{\theta^*_m\}_{m=1}^M$. Recall, θ^*_m consists of the parameters of model p. When the training data $D = (d_m, \theta_m)_{m=1}^M$ is collected, train the surrogate model on D to obtain $\hat{\phi}$. Note that the collected training data is based on both rejected and accepted proposal parameters, θ^*. Once the surrogate model $p^s_{\hat{\phi}}$ is trained, the simulation of the posterior $\pi(\theta|\rho(S(x), S(x_0)) \leq \varepsilon)$,via DA-ABC-MCMC, can begin. Then for each simulation, as usual, propose a new candidate vector θ^* via a proposal function q given the previous accepted parameter vector. Continue with predicting a distance $\hat{d}$ given the proposed θ^* with the trained surrogate model $p^s_{\hat{\phi}}(S(x_0)|\theta^*)$. Then use this distance in calculating the acceptance probability at the first stage α_1 in (2.5) at step 3, i.e. use $\hat{d}$ instead of $\rho(S(\hat{x}^*), S(x_0))$. Then continue from step 4. This results in DA-ABC-MCMC algorithm explained in algorithm 3.

Algorithm 3: DA-ABC-MCMC

1. **Input:** x_0-observed data, $\tilde{\theta}_1$ - initialize parameter vector for the collection of training data, $\tilde{\varepsilon}$ - initialize the threshold when collecting the training data, M- number of data-points for training the surrogate model, N- number of simulations to obtain the posterior distribution, $D = \{\emptyset\}$.
2. **for** $m = 1 : M$
   - 2.1 $\tilde{\theta}^{(m)} \sim q(\tilde{\theta}|\theta_m)$, propose a new parameter vector via a proposal function.
   - 2.2 $x^{(m)} \sim p(x|\tilde{\theta}^{(m)})$, simulate a data set from the model given the proposed parameter vector.
   - 2.3 $d^{(m)} = \rho(S(x^{(m)}), S(x_0))$, calculate the distance and store $D = D \bigcup (\tilde{\theta}^{(m)}, d^{(m)})$
   - 2.4 With probability $\alpha(\tilde{\varepsilon})$ (2.1) let $\tilde{\theta}_{m+1} = \tilde{\theta}^{(m)}$, otherwise let $\tilde{\theta}_{m+1} = \tilde{\theta}_m$.

   end
3. Train surrogate model $p^s_\phi(S(x_0)|\theta)$ on D to obtain $\hat{\phi}$. θ_1 - initialize parameter vector. ε - initialize the threshold.
4. **for** $i = 1 : N$
   - 4:1 $\theta^* \sim q(\theta|\theta_i)$, propose a new parameter vector via a proposal function.
   - 4:2 $\hat{d} = p^s_{\hat{\phi}}(S(x_0)|\theta^*)$, predict the distance given the proposed parameter vector.
   - 4:3 With probability,

     $$\alpha_1 = \min\left\{ 1 \, , \mathbb{1}(\hat{d} \leq \varepsilon) \frac{\pi(\theta^*) q(\theta|\theta^*)}{\pi(\theta) q(\theta^*|\theta)} \right\}$$

     go to next step 4:4, otherwise set $\theta_{i+1} = \theta_i$ and start over from step 4.
   - 4:4 $x^* \sim p(x|\theta^*)$ generate a synthetic data set from the model given the parameter vector.
   - 4:5 With probability

     $$\alpha_2 = \min\left\{ 1 \, , \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon) \right\}$$

     let $\theta_{i+1} = \theta^*$, otherwise let $\theta_{i+1} = \theta_i$.

   end
5. **Output:** N draws from the posterior $\pi(\theta|\rho(S(x^*), S(x_0)) \leq \varepsilon)$.

Recall that the purpose of the thesis is to make ABC-MCMC (algorithm 2) more computationally efficient. If algorithm 3 would work for this purpose, it is important

that $M << N$, since step 2 in the algorithm is basically the steps in algorithm 2 (ABC-MCMC) and we only use this step to collect training data for the surrogate model p^s_ϕ. Since this step is computationally inefficient, assuming the simulation of a model p is computationally heavy, we want M as small as possible. At the same time, we would find a regression model, which predictions of the distance d makes the ratio $\triangle_\alpha = \alpha_2/\alpha_1$ between the acceptance stages as high as possible, where $\triangle_\alpha \in [0, 1]$. This is an indication of how many of the proposed parameter vector θ^* that survives the first acceptance stage α_1, which also survives acceptance stage 2 α_2. (Note that this will not indicate how many of the proposed parameter vector we reject at the first stage which would have survived the second stage).

A common term in MCMC simulations is the so called burn-in period, which is the beginning of a MCMC run that is eliminated and are not included in the output. An example of the burn-in period is shown in the left plot of figure 3.1, where the burn-in period is from iteration 1 to around 25000. The right plot is the corresponding histogram of the trace plot after the burn-in period. Due to using ABC-MCMC method in step 2 in algorithm 3, we want to avoid redo the burn-in period at step 4, where the DA approach is introduced, by using the information obtained of the ABC-MCMC simulation in step 2 when collecting training data, by initializing the starting values in step 3 for θ_1 and ε.

3

Implementation and Interpretation

In the ABC framework there is constantly introducing techniques which makes it more efficient. Here we will go through which teqniques are used in this thesis and how it will be performed in a simulation point view. This also includes diagnostics about how to interpret the result of the approximated posterior distribution.

3.1 Implementation Details

By looking at the presented Algorithms, there are some parts that needs to be specified. Here, we will go through how the unclear steps will be performed in this thesis.

3.1.1 Mahalanobis Distance

First of all, we need to introduce a suitable distance function ρ for the algorithms presented in section 2. Recall, the distance is calculated between the originate summary statistics $S(x_0)$ and the summary statistics of the synthetic data set $S(x^*)$ simulated from the model given the proposed parameter vector θ^*. Number of summary statistics varies depending on which model the data set is believed to come from. Often, the more complex a model is, the more summary statistics it has. For example, consider euclidean distance for $S = (S_1, ..., S_n)$,

$$\rho(S(x_0), S(x)) = \sqrt{(S_1(x_0) - S_1(x))^2 + \cdots + (S_n(x_0) - S_n(x))^2}.$$

The euclidean distance is very sensitive to the possibly different magnitudes of the several summary statistics, where components of the S vector that are highly variable will dominate the components that vary less. And hence the ABC distance will be more dependent on the former than on the latter, which is something we need to mitigate.

The distance has an important role in ABC methods, since the distance d has an high impact of whether θ^* gets accepted or not and since the distance d is the response variable in the trained model in the DA-ABC-MCMC algorithm. Mahalanobis is a distance that will make all variables have the same influence by including their empirical covariance matrix C. The Mahalanobis distance is calculated as

$$\rho(S(x_0), S(x)) = \sqrt{(S(x_0) - S(x))^T C^{-1} (S(x_0) - S(x))}. \tag{3.1}$$

3.1.2 Threshold, ε

Recall section 2.1.1, the posterior distribution $\pi(\theta|\rho(S(x), S(x_0)) \leq \varepsilon)$ is more or less approximated depending on the size of the threshold, ε. It is desired to have a small threshold and by setting a fixed value on ε from the beginning of the simulation can imply difficulties of targeting the posterior distribution and result in zero acceptance rate. This is a common problem in ABC-algorithms. Due to this, several methods have been introduced to come around this problem. Many of these methods starts with a high ε and then let it decrease to the desired threshold. For example, ABC-SMC (Approximate Bayesian Computation-Sequential Monte Carlo) is based on this concept and [11] have also introduced a way.

We want the threshold to decrease in a "smooth" way and in that way avoid high rejection but still targeting the posterior distribution. In this thesis, we use the following method to decrease the threshold ε, see for example [12]. Set a starting value on the threshold $\varepsilon_{i=1}$, then for every h (e.g. $h = 1000$) iterations update ε_i in the following way:

$$\varepsilon_i = \min\{\varepsilon_{i-h}, \text{quantile}_\gamma(d_{i-h+1}, d_{i-h+2}, ..., d_{i-1})\}. \tag{3.2}$$

γ is a chosen probability value for the quantile of the calculated distances d between iteration $i-h$ and i. Keep lowering the threshold until a functional acceptance rate is obtained. For example in this thesis, a functional acceptance rate is at least 5%. This is also of benefit since it finds the optimal threshold, compared to setting one fixed from the beginning. Slightly different way of decreasing the threshold in ABC methods by also using quantiles is seen in these papers: [13] and [14].

3.2 Diagnostic Plots

Diagnostic plots, in this case, are interpreting tools of how simulations is performing and resulting in. For example, how the posterior is converging to the "right" answer, the densities of the posteriors and how other important parameters in the algorithm behaves during the simulation.

3.2.1 Trace Plot and Histogram

Trace plots illustrate all values a parameter θ has been assigned and accepted, from its starting value and then its journey of values converging, and hopefully, to the "right" value. This is basically a plot where the x-axis is the timeline of the iterations $i = 1, ..., N$ plotted against the collected parameter values, $\{\theta_i\}_{i=1}^N$. In figure 3.1, the left plot shows a trace plot performing well, i.e. it finds the underlying searched value 1. The right plot is the corresponding histogram of the parameter θ. The histogram is an illustration of the approximate posterior distribution of θ.

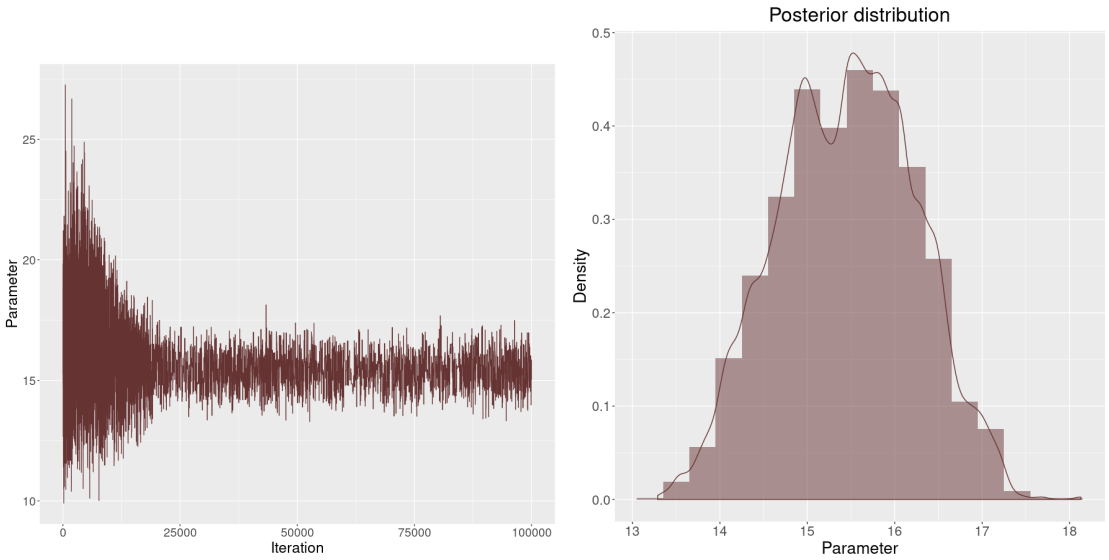

Figure 3.1: Left figure: Example of a trace plot. Right figure: Corresponding histogram without the burn-in period.

Note, the histogram is without the burn-in period i.e. after approximately iteration 25000, shown in the trace plot. It is only the accepted parameters where the acceptance rate is at a desired level which are of interest as a result of the posterior distribution.

3.2.2 Acceptance Rate Plot

The threshold is decreasing during the simulation presented in sec 3.1.2. This is done to avoid zero acceptance rate. This implies that the acceptance rate is going to decrease during the simulation as well, since the higher the threshold ε is, the more parameter θ with a wider value range will be accepted. By keeping track of how the acceptance rate are behaving together with the trace plots and histogram during the simulation, you get an good idea of how "smooth" the converging is. It is also a tool of regulating the decreasing of the tolerance level. When the acceptance rate has decreased to a desired level α^{ar}, we can stop decreasing the threshold. How to decide the current acceptance rate at iteration i, choose a value k, (e.g. $k = 1000$) and set α^{ar} to

$$\alpha_i^{ar} = \frac{\text{Nr. of accepted } \theta^* \text{ between iteration } [i-k, i]}{k}. \tag{3.3}$$

An example of how an acceptance rate plot could look is shown in figure 3.2.

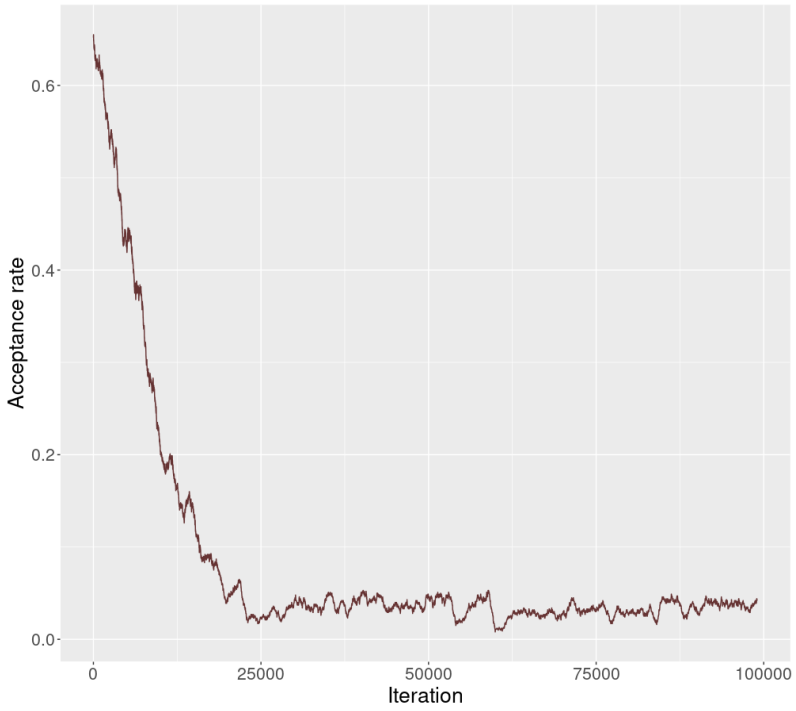

Figure 3.2: Plot of the acceptance rates during a simulation.

Since we are only interested in the obtained parameters after the burn-in period, the acceptance plot is a good tool to see from which iteration i we should start to consider the parameter vector as a sample from the posterior distribution.

3.2.3 Distance Analysis

A key part of this thesis is to predict a distance $\hat{d}$ by a regression model $p^s_\phi(S(x_0)|\theta)$ and then evaluate whether it is worth calculating the real distance $d = \rho(S(x^*), S(x_0)) \leq \varepsilon$) or not, explained in section 2.2.2. In other words, if $\hat{d}$ in Algorithm 3 gets accepted at acceptance stage 1 (step 4:3) then distance d is evaluated. Then there is of interest to compare these two distances. This can be done by simply plotting them against each other and see if their scatter plot looks as a straight line with gradient 1. There is also interesting to analyze if the ratio $\triangle_\alpha = \frac{\alpha_2}{\alpha_1}$ increase when the predicted distances $\hat{d}$ has a better prediction. Also, this comparison can tell how robust the algorithm is to the prediction of $\hat{d}$.

4

Case 1, Gaussian Distribution

The first case is based on a data-set x_0 believed to come from a Gaussian distribution. This example may not be of benefit using the DA-ABC-MCMC approach on. This is an example where we can obtain the exact posterior and we can get an idea of how good DA method works on a simple model before testing it on more complex models.

Assume the data-set x_0 is a sample of a populations IQ levels of sample size n. x_0 is assumed to come from a Gaussian distribution with parameters $\theta = (\mu, \sigma)$. In this case, sample a synthetic data-set $x_0 \sim \mathrm{N}(\mu = 100, \sigma = 15)$. Pretend that we don't know θ and want to find the posterior distribution $\pi(\theta|x)$. Then in this case, let the prior distribution for the parameters be the following:

$$\pi(\theta) \begin{cases} \mu \sim \mathrm{N}(100, 15) \\ \sigma \sim \mathrm{Gamma}(\frac{289}{7}, \frac{17}{7}) \end{cases}$$

where the Normal distribution has parameters mean and standard deviation and the gamma distribution has shape and rate parameters. The likelihood for this model is known and is the following:

$$p(x|\theta = (\mu, \sigma)) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

It is possible to target the posterior distribution with regular Metropolis-Hastings method explained in sec 2.2.1, and will be referred as MCMC simulation. I use the results from MCMC simulation as reference when comparing with ABC-MCMC and DA-ABC-MCMC. For example, if $n = 500$, we want to target the posterior distributions in figure 4.1.

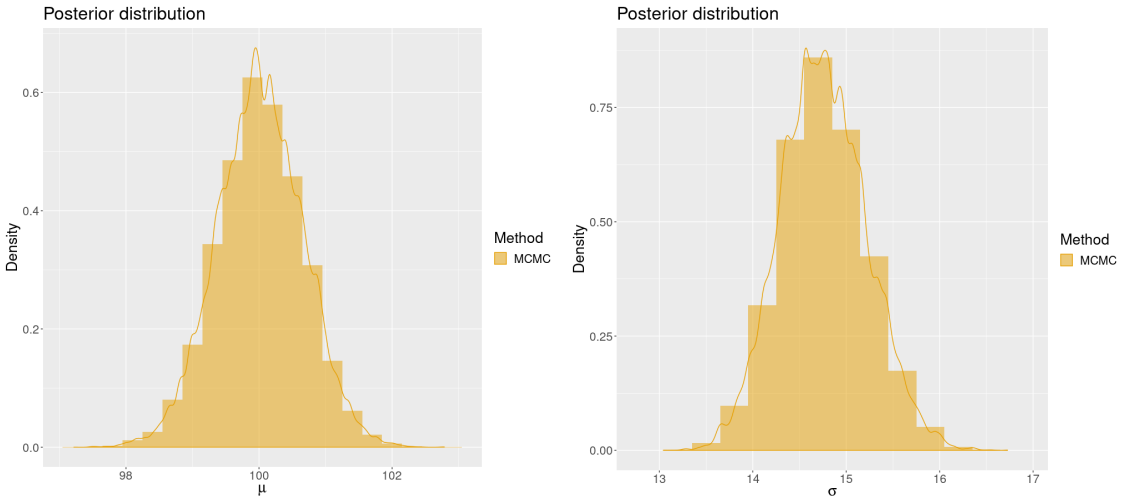

Figure 4.1: Exact posterior distribution for $\theta = (\mu, \sigma)$ via MCMC sampling, with sample size $n = 500$.

4.1 Method

First of all, we need an ABC-MCMC algorithm that works well, since it is a key component in the comparison in time efficiency between ABC-MCMC and DA-ABC-MCMC algorithms, which are one of the main interest in this thesis. Also, the ABC-MCMC is used when collecting training data in the DA-ABC-MCMC algorithm.

4.1.1 ABC-MCMC Algorithm

Recall Algorithm 2 for simulation with ABC-MCMC. Given a data set x_0 of size n, set the starting values to $\theta_1 = (\bar{x_0}, s_{x_0})$, where $\bar{x_0}$ is the sample mean and s_{x_0} is the sample standard deviation. In step 2 in the algorithm, the proposal function $q(\theta^*|\theta)$ is set to the normal density function with fixed standard deviation. In our case, we propose $\theta^* = (\mu^*, \sigma^*)$ in the following way:

$$\begin{aligned} \mu^* &\sim \mathrm{N}(\mu_i, 10), \\ \sigma^* &\sim \mathrm{N}(\sigma_i, 2), \end{aligned}$$

at iteration i. Using normal distribution as proposal function is a common choice in MCMC methods since it allows local and symmetric moves around the last accepted value. It is also a convenience choice when calculating the acceptance probability α in step 4, since the normal density function is symmetric which makes the division $q(\theta_i|\theta^*)/q(\theta^*|\theta_i) = 1$ for any values of θ^* and θ_i.

Moving on to step 4 in Algorithm 2, we need to specify the threshold ε. Start by setting $\varepsilon = \infty$. Then, update ε every 1000 iteration according to (3.2) in sec. 3.1.2 by specifying $h = 1000$ and $\gamma = 0.85$. Continue to update ε until a desirable threshold is obtained, which means until the acceptance rate, α^{ar}, defined in (3.3),

is around 5%. Also, if $\alpha^{ar} < 0.03$ then the previous threshold is tried again.

Further, I used Mahalanobis distance, defined in (3.1), as distance function when calculating the distance between the summery statistics, $\rho(S(x_0), S(x))$. Since the model is Gaussian, it is natural to set the summery statistics to the sample mean and sample standard deviation, $S(x) = (S_1(x) = \bar{x}, S_2(x) = s_x)$.

With these specifications for Algorithm 2 and setting $n = 500$ (sample size of the observed data x_0) and $N = 100000$ (number of MCMC iterations), the following posterior distributions are obtained, shown in figure 4.2.

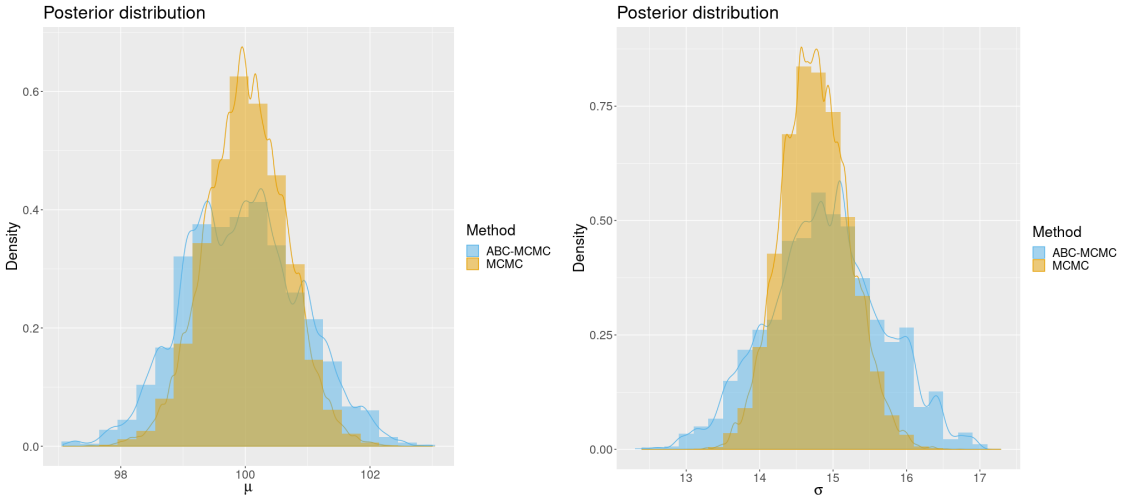

Figure 4.2: The posterior distributions via ABC-MCMC simulation compared with MCMC-simulation, with sample size $n = 500$.

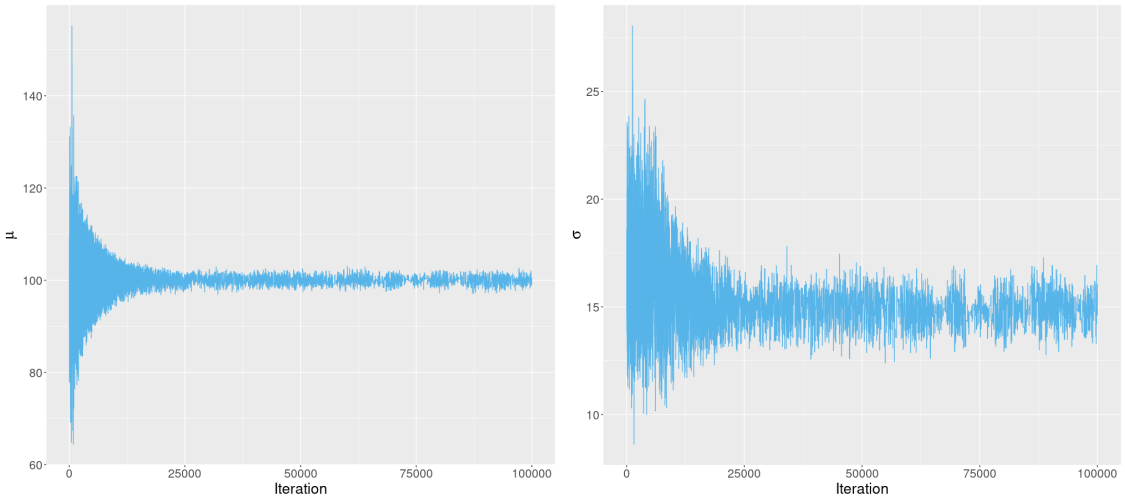

Figure 4.3: Trace plots of the parameters mean and standard deviation from the ABC-MCMC simulation, with sample size $n = 500$.

Figure 4.3 is the corresponding trace plot to the histograms in figure 4.2 of

ABC-MCMC, thus with the burn-in period. The histogram are only based on the chain obtained after the burn-in period i.e. around iteration 25000. By comparing the posterior distribution obtained via ABC-MCMC simulation with MCMC simulation, ABC-MCMC inflates the true variability of the posterior distribution. Notice in particular the heavier tails for the ABC-MCMC case. This is not expected since stopping the decreasing of the threshold ϵ after an acceptance rate around 5% is obtained and remember from section 2.1.1 that the posterior distribution $\pi(\theta|\rho(S(x^*), S(x_0)) \leq \varepsilon)$ is more or less approximated depending on the threshold. When interpreting figure 4.4, there is clearly a connection between the threshold and the acceptance rate. The acceptance rate is defined according to (3.3) with $k = 1000$. Also, the posterior distribution obtained via ABC-MCMC simulation is based on the summary statistics, which is not as explainable as the whole data-set used in the likelihood function (4) used in the MCMC simulation.

Keep in mind that the posterior distribution will have a more informative shape when n increase, since the variance will decrease and will require a smaller threshold to be targeted.

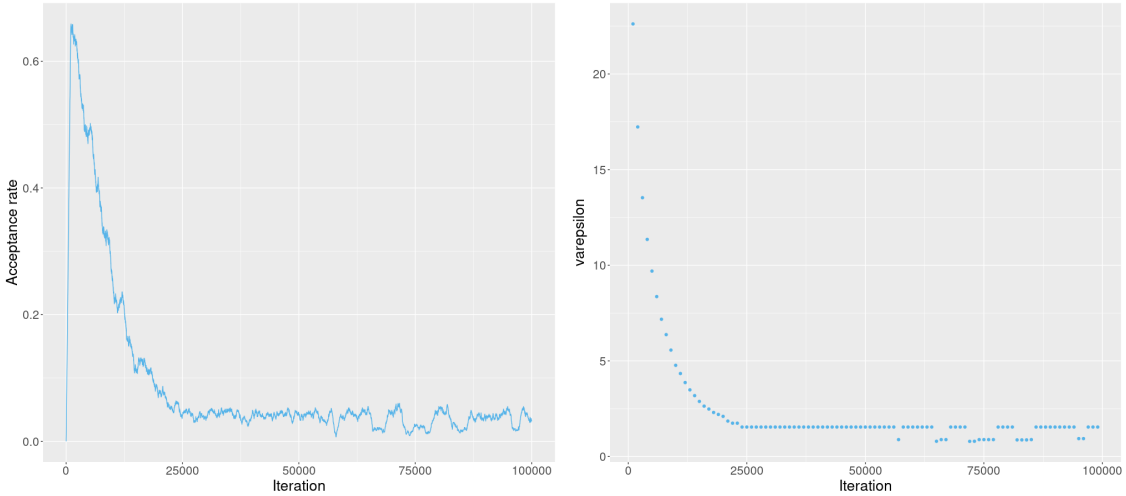

Figure 4.4: Acceptance rate α^{ar} and threshold values ε for ABC-MCMC.

4.1.2 DA-ABC-MCMC Algorithm

Recall algorithm 3 for simulation of DA-ABC-MCMC. In step 2, the collection of training data are simulated, using the ABC-MCMC approach with the same specifications as in the previous subsection 4.1.1. Simulations are performed until R data points are obtained after the burn-in period.

Moving on to step 3, and since the burn-in period already has been simulated when collecting the training data, we want to make use of that information as much as possible to save computational time. This can be done by using the information obtained in step 2 by setting the following starting values at step 3: $\theta_1 = \text{mean}(\tilde{\theta}_{M-R:M})$ and let ε be set to the mean of the thresholds used in iterations $m = M - R : M$. The choice of starting value of ε is based on, to not start with an

unnecessary big value and not a too small.

In step 3, there is in need of defining a suitable surrogate model. Since it is of extra interest of good prediction after the burn-in period, two surrogate models are trained. Let D be the training data collected in step 2 in algorithm 3. The first one is trained on $\{D(d^{(m)} < 3)\}_{m=1}^{M}$ and used when $\varepsilon < 3$. Then, also a "back-up" surrogate model are trained on $D_{1:M}$ in case the threshold $\varepsilon \geq 3$ which is outside the first models range to predict. This means we train $p^s_{\phi_1}(S(x_0)|\theta)$ on $\{D(d^{(m)} < 3)\}_{m=1}^{M}$ to obtain $\hat{\phi}_1$ and train $p^s_{\phi_2}(S(x_0)|\theta)$ on $D_{1:M}$ to obtain $\hat{\phi}_2$. Both surrogate models in this case are defined to the linear regression model in (4.1.2) and are trained on standardized data.

$$\hat{d} = \hat{\beta}_0 + \hat{\beta}_1\mu + \hat{\beta}_2\sigma + \hat{\beta}_3\mu^2 + \hat{\beta}_4\sigma^2 + \hat{\beta}_5\mu^3 + \hat{\beta}_6\sigma^3 + \hat{\beta}_7\mu\sigma.$$

This gives adj-$\mathtt{R}^2 = 0.48$ for $p^s_{\hat{\phi}_1}$ and adj-$\mathtt{R}^2 = 0.8929$ for $p^s_{\hat{\phi}_2}$ and the corresponding residual plots in figure 4.5.

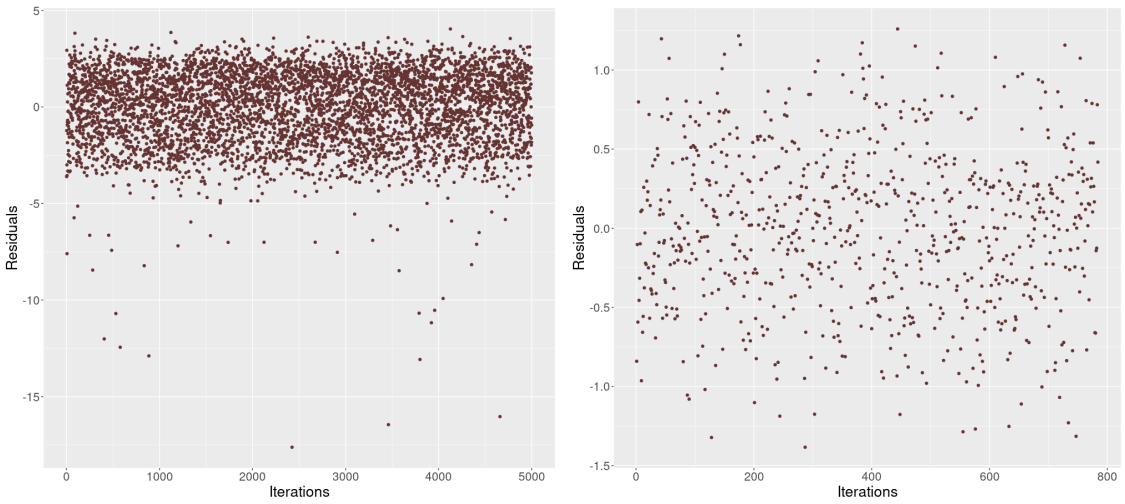

Figure 4.5: Residual plot of $p^s_{\hat{\phi}_2}$ and $p^s_{\hat{\phi}_1}$, respectively .

Although the adj-$\mathtt{R}^2$ is better for $p^s_{\hat{\phi}_2}$ and the residuals are centered around zero (except for some few residuals), we see in the scatter plot in figure 4.6 between the response variable and the covariate (mean parameter) that the covariete is very explainable in the left figure, due to the high adj-$\mathtt{R}^2$ for $p^s_{\hat{\phi}_2}$. Thus the absolute majority of accepted parameters are the ones around value 100 on the x-axis and as long as the distance d is higher than the threshold , it will never be accepted as a sample of the posterior distribution, due to the indicator function $\mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon)$ in α_2.

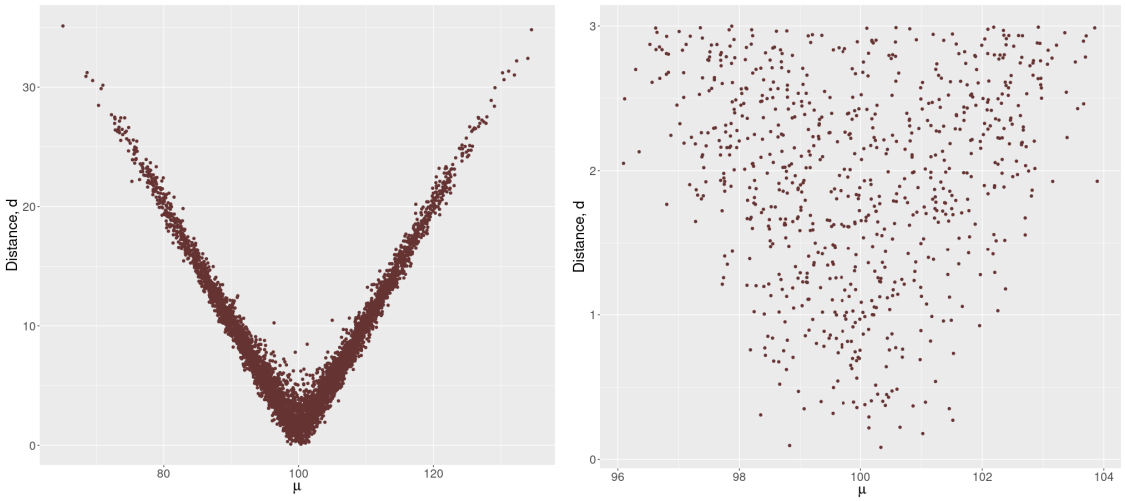

Figure 4.6: Scatter plot of d and μ from the training data for $p^s_{\hat{\phi}_2}$ and $p^s_{\hat{\phi}_1}$, respectively.

Moreover, at step 4 in algorithm 3. The specifications are the same as for ABC-MCMC. When updating the threshold, $\gamma = 0.85$ and $h = 1000$ and the acceptance rate α^{ar} are defined with $k = 1000$. Everything is specified in algorithm 3 and the following posterior distributions are obtained and shown in figure 4.7.

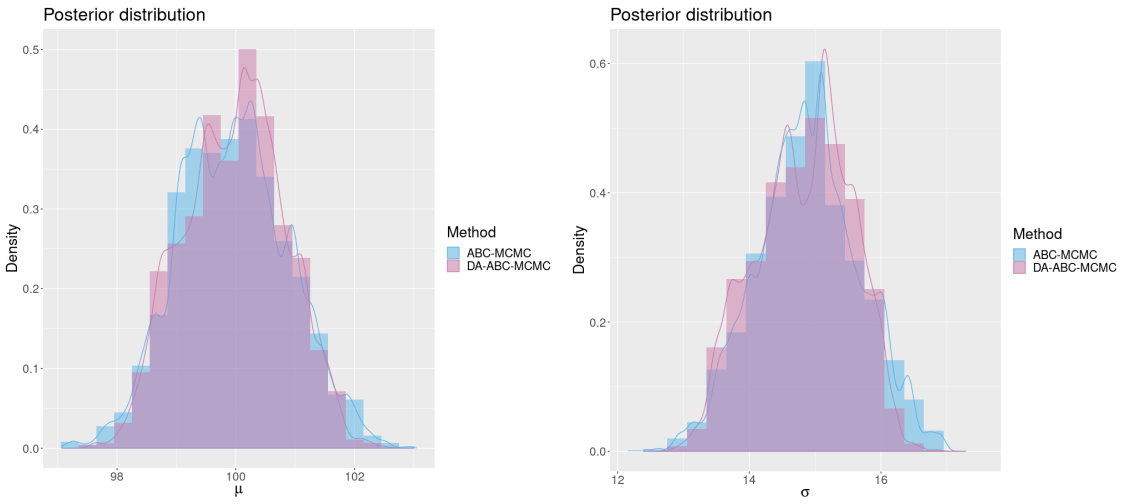

Figure 4.7: Posterior distribution for $\theta = (\mu, \sigma)$ for $n = 500$.

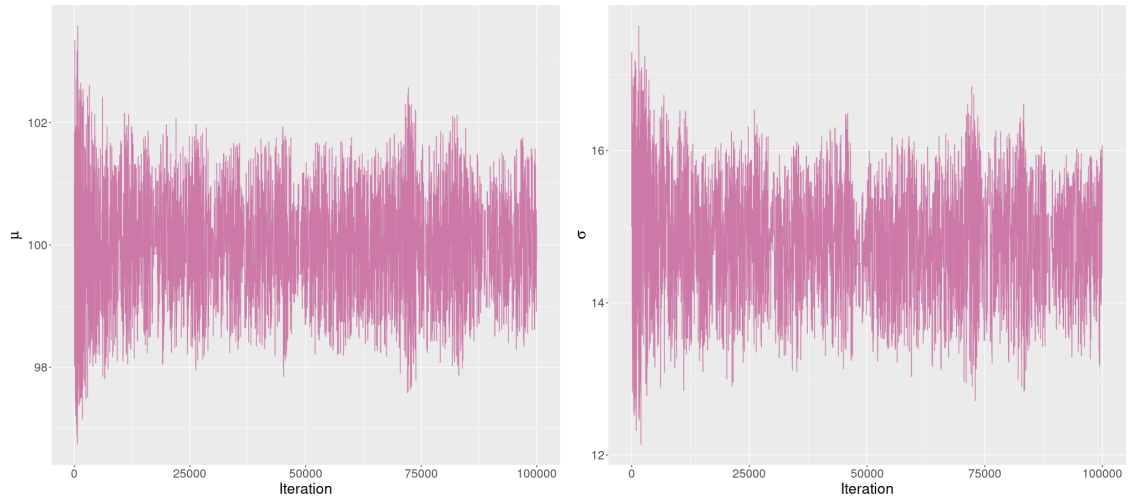

Figure 4.8: Trace plot of $\theta = (\mu, \sigma)$ of the DA-ABC-MCMC simulation for $n = 500$ in step 4 in algorithm 3 for iterations $1 : N$. (Note, the burn-in period for DA-ABC-MCMC is done in the collection of training data D in step 2).

The posterior distributions from algorithm 3 is shown in figure 4.7 and it seems as a good result in targeting the posterior distribution obtained via ABC-MCMC simulation. Further, when interpreting the results from the surrogate model p^s_ϕ, by looking at the distances accepted at the first acceptance probability α_1 compared with the corresponding distances $d = \mathbb{1}(\rho(S(x^*), S(x_0)) \leq \varepsilon)$ seen in a scatter plot in figure 4.9, the surrogate model seems to have a hard time predicting distances close to zero, which we can get a hint from in figure 4.6, the right figure. Even though, DA-ABC-MCMC seems to capture the simulation from ABC-MCMC good and the ratio $\triangle_\alpha = 0.74$.

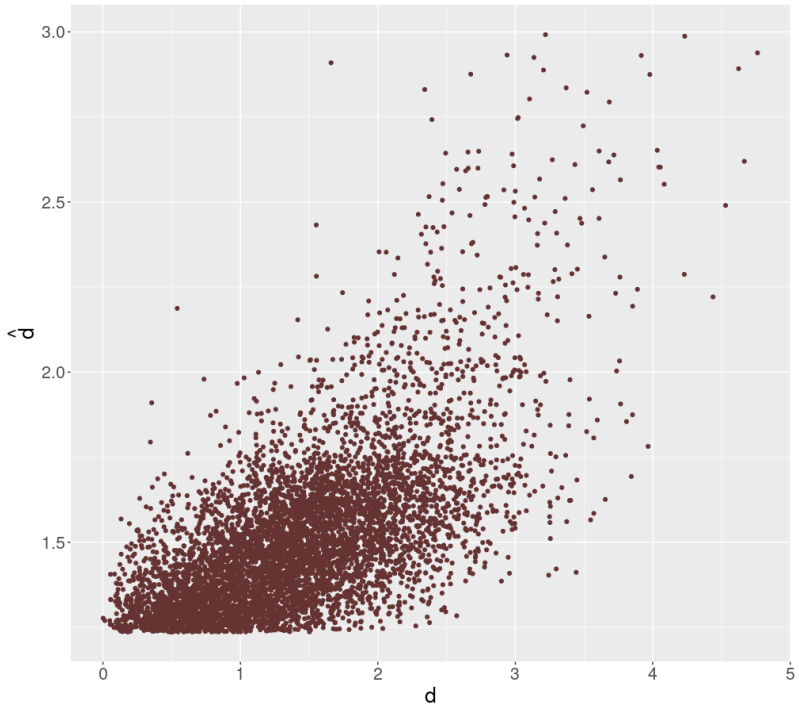

Figure 4.9: Scatter plot of $\hat{d}$ and d.

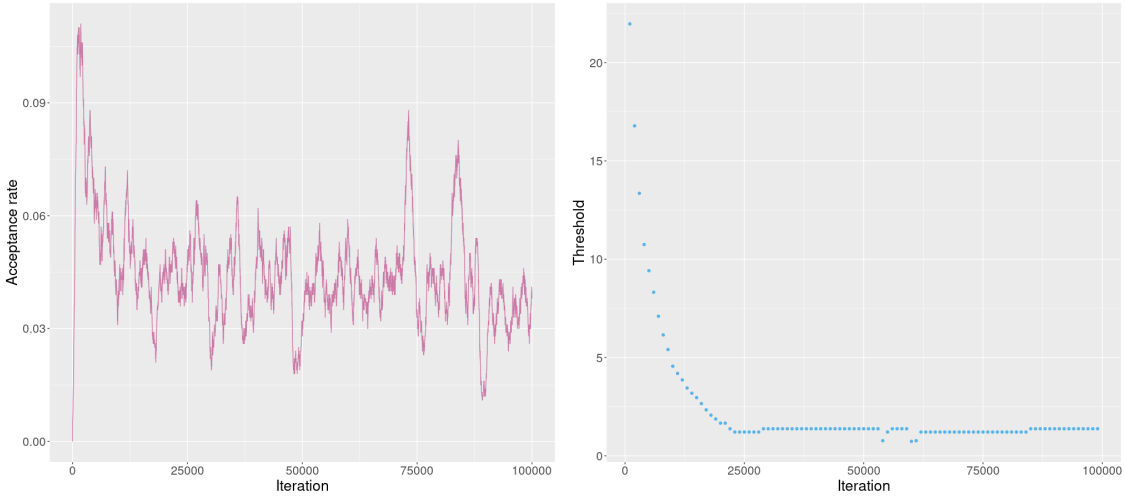

Figure 4.10: Acceptance rate α^{ar} plot and threshold ε from simulation of DA-ABC-MCMC for $n = 500$ and iterations 1:N=100000.

4.2 Results

The DA-ABC-MCMC seems to capture the posterior distribution as the ABC-MCMC generates rather good. In figure 4.11 the results of the computational time for both methods are shown, where the number of iteration is $N = 100000$, respectively. The plot shows the DA-ABC-MCMC method is more and more computational efficient when the sample size increase.

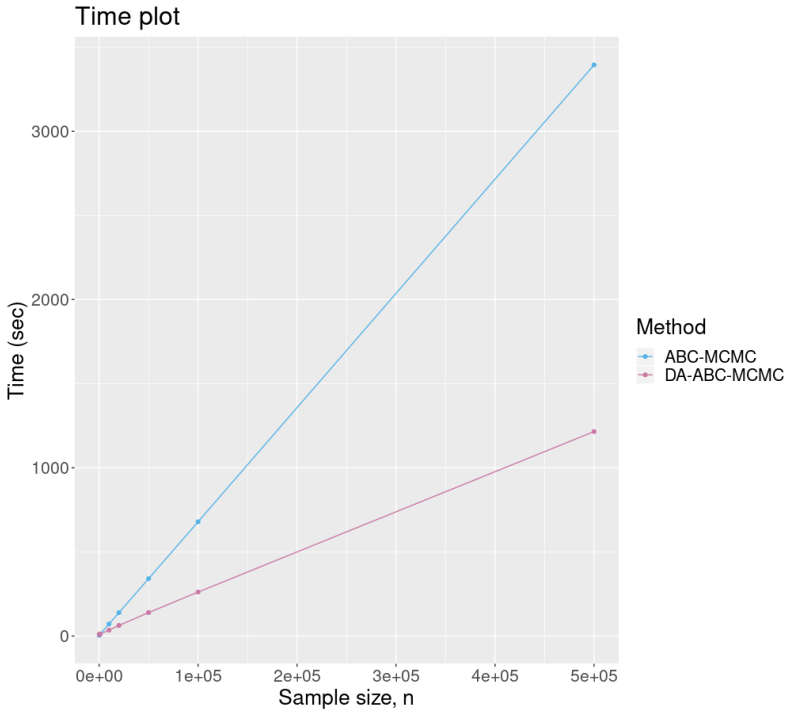

Figure 4.11: The running-time for ABC-MCMC and DA-ABC-MCMC for iteration 1 to N in algorithm 2 and 3, respectively.

Thus, comparing both methods trace plots (figure 4.3 and 4.8), the burn-in period seems shorter for DA-ABC-MCMC than for ABC-MCMC. This is because the burn-in period for DA-ABC-MCMC has already been obtained in step 2 (algorithm 3), when collecting the training data. This means that the approximate posterior distribution for DA-ABC-MCMC can be based on a longer Markov chain, than for ABC-MCMC, when comparing algorithm 2 and 3 for iterations $1 : (N = 100000)$. A common way to measure the "quality" of a MCMC simulation is to calculate the effective sample size (ESS). Due in applications where it is desirable to draw an independent random sample from a probability distribution, in our case from the posterior distribution on parameters. The problem with samples generated from MCMC is that the samples are dependent. Due to the interest of ESS which provides a more fair value of how good an MCMC simulation actually is. Figure 4.12 shows the different ESS for the two parameters respectively.

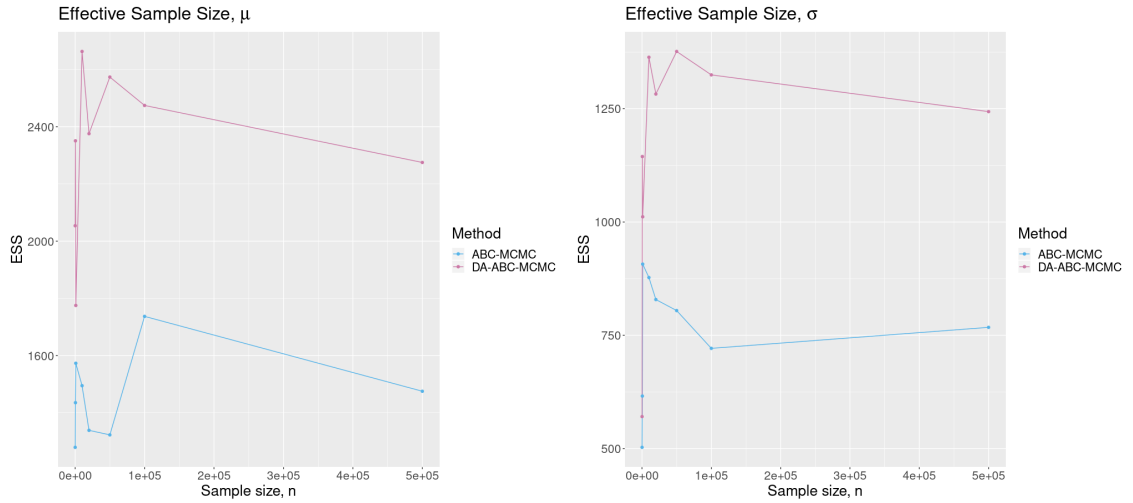

Figure 4.12: Effective sample size (ESS) of the posterior distribution of $\theta = (\mu, \sigma)$.

The ESS is calculated after the burn-in period for both methods and clearly the DA-ABC-MCMC provides a higher ESS. The time difference and ESS should both be taken into account when interpreting the computational efficacy of the both methods. Even if the ESS is a good measure tool for the quality of a Markov chain, it can still be unfair for this specific case due to the threshold. Since the threshold is defined according to (3.2), the value can be different for ABC-MCMC and DA-ABC-MCMC, where a lower threshold is influencing the ESS to lower values. Though the threshold value for the two different methods are similar, where the threshold for ABC-MCMC is slightly higher which can be seen when comparing the different thresholds in the right plots in figure 4.4 and 4.10.

4.3 Analysis and Discussion

The DA-ABC-MCMC seems to perform almost as good as ABC-MCMC, even if the prediction is not perfect. The main concerns are the problems of predicting values close to zero, which can be suspected as a problem when it is desirable of targeting the posterior distribution with a lower threshold.

Again, remember that this model is not complex and a lot of tuning parameters needs to be set e.g. the decreasing of the threshold ε, the decision of a suitable surrogate model. Thus, it seems to work sufficiently good to consider the DA-ABC-MCMC method on more complex models.

5

Case 2, G-and-k Distribution

G-and-k distribution is extraordinary in the way of describing such complex data with only 5 parameters and the form of the distribution used here were first presented by [15]. Due to few parameters, g-and-k has been a common distribution to test the reliability in Approximate Bayesian computation methods e.g. in [16].

If $Z \sim \mathrm{N}(0,1)$ then a random variable X from the g-and-k distribution is described as

$$X = A + BG(Z)H(Z)$$

where $G(z) = 1 + c\tanh(gz/2)$ which adds asymmetry to the distribution and where $H(z) = z(1+z^2)^k$ extends the tails. The g-and-k distribution has 5 parameters: $\theta = (A, B, g, k, c = 0.8)$. The parameter $c = 0.8$ is fixed and we are interested in finding the posterior $\pi(\theta|x)$ for the other parameters $\theta = (A, B, g, k)$. The parameters can be described as; A is the location parameter, B is the scale parameter, g is a shape parameter affecting mainly the skewness and k is also a shape parameter but affects mainly the kurtosis.

As in Case 1, sample a synthetic data set $x_0 \sim \mathrm{gk}(A = 3, B = 1, g = 2, k = 0.5)$ of size n. Then pretending not knowing the parameters θ, one wants to target the posterior distribution $\pi(\theta|x)$, by using the following prior distributions:

$$\pi(\theta) \begin{cases} A \sim \mathrm{Uni}(-10, 10) \\ B \sim \mathrm{Uni}(0, 10) \\ g \sim \mathrm{Uni}(0, 10) \\ k \sim \mathrm{Uni}(0, 10). \end{cases}$$

For the g-and-k distribution, it is not as straight forward to obtain the likelihood and from that target the posterior distribution via MCMC sampling, as it is in Case 1. Thus, there is a package named *gk*, from [17], where there is a built in function **mcmc** which makes it possible to target the posterior distribution via MCMC sampling. The same paper explains the theory behind the built-in functions in the *gk-package*, and includes the details how it is manageable to do MCMC sampling with g-and-k distribution. The **mcmc** function is computationally heavy but it gives a reference to compare the results from ABC-MCMC and DA-ABC-MCMC algorithms. For example, if $n = 5000$ then we want to target the following posterior distributions:

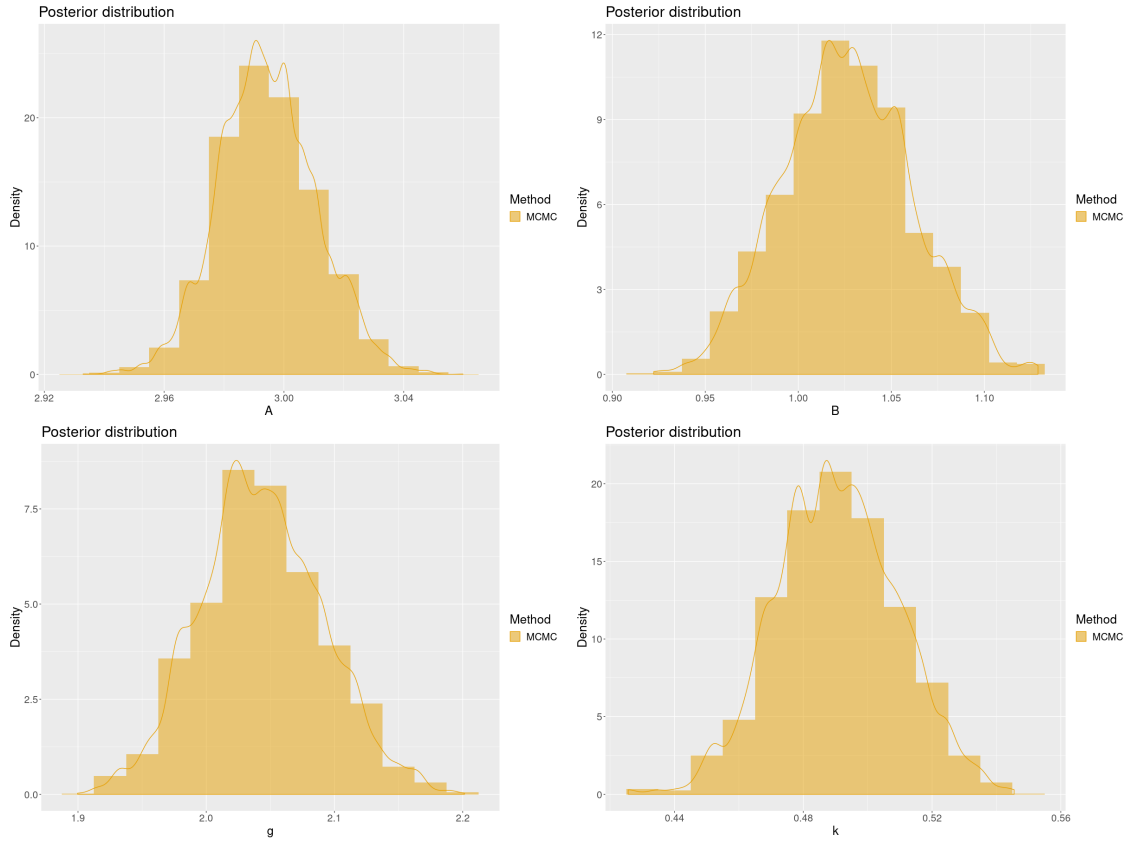

Figure 5.1: Posterior distribution for each parameter in $\theta = (A, B, g, k)$, for sample size $n = 5000$.

5.1 Method

The method procedure for case 2 is very similar to case 1. Thus, in case 1, there were a simple model with less uninformative prior distributions and where a first step to see whether the DA-approach seems workable. Due to this case with a more complex model and with uninformative flat prior distributions, the specifications are more carefully considered.

5.1.1 ABC-MCMC and DA-ABC-MCMC Algorithms

The proposal function used in both algorithms 2 and 3 is again the normal density function with fixed variance, due to the same reasons as in case 1. Assume the simulation is at iteration i. Then the following proposal functions are used to propose $\theta^* = (A^*, B^*, g^*, k^*)$:

$$
\begin{aligned}
A^* &\sim \mathrm{N}(A_i, 0.25) \\
B^* &\sim \mathrm{N}(B_i, 0.1) \\
g^* &\sim \mathrm{N}(g_i, 0.25) \\
k^* &\sim \mathrm{N}(k_i, 0.1).
\end{aligned}
$$

Due to a more complex model, the summary statistics are not as easy to obtain as in case 1. The ones used in this case were presented by [16] and are called ' the robust estimates of the moment based on the octiles'. Let $E_1, E_2, \ldots, E_7$ be the octiles of the data x_0. Then the summary statistics $S(x) = (S_A(x), S_B(x), S_g(x), S_k(x))$ are obtained in the following way:

$$\begin{aligned} S_A &= E_4, & S_B &= E_6 - E_2, \\ S_g &= (E_6 + E_2 - 2E_4)/S_B, & S_k &= (E_7 - E_5 + E_3 - E_1)/S_B. \end{aligned}$$

The acceptance rate α^{ar} are defined with $k = 1000$ for both algorithms. The starting value for the threshold in ABC-MCMC (algorithm 2) is set to $\varepsilon = 10$ and then letting it decrease with $h = 1000$ and $\gamma = 0.85$ until the burn-in period has ended. Then, one uses $\gamma = 0.5$ to try to capture an even smaller threshold. Though if $\alpha^{ar} < 0.01$ then try the previous ε. This is also the settings for step 2 in algorithm 3, when collecting the training data.

Regarding the collected data, a suitable surrogate model p^s_ϕ needs to be defined. Since it is of extra interest of good prediction after the burn-in period, two surrogate models are trained, as in case 1. By looking at the scatter plots of the training data D shown in figure 5.5, the covariates are not as explainable as in case 1. As shown in figure 4.6, there is not as easy to just splitting up the data D as we did in case 1. Instead relying it on the acceptance rate α^{ar}. The first one is trained on $D_{M-R:M}$ and are used after the burn-in period until $R = 5000$ data points are obtained. Then, also a "back-up" surrogate model are trained on $D_{L:M}$ in case the simulation are outside the "5% acceptance area", which is outside the first models range to predict. Let L be the iteration when the acceptance rate reaches $\alpha^{ar} = 0.2$. As can be seen in case 1, the burn-in period does not repeat in the simulation step (step 4, algorithm 3) and are unnecessary to base the prediction model on the whole burn-in period. This means we train $p^s_{\phi_1}(S(x_0)|\theta)$ on $D_{M-R:M}$ to obtain $\hat{\phi_1}$ and train $p^s_{\phi_2}(S(x_0)|\theta)$ on $D_{L:M}$ to obtain $\hat{\phi_2}$. Both surrogate models in this case are defined as the linear regression model

$$\hat{d}_{pred} = \hat{\beta}_0 + \hat{\beta}_1 A + \hat{\beta}_2 B + \hat{\beta}_3 g + \hat{\beta}_4 k + \hat{\beta}_5 A^2 + \hat{\beta}_6 B^2 + \hat{\beta}_7 g^2 + \hat{\beta}_8 k^2 \hat{\beta}_9 Bk + \hat{\beta}_{10} gk \quad (5.1)$$

and are trained on standardized data. As said, the covariates are not as explainable as in case 1. Even for a simple model as in case 1, it was hard to predict the distances close to zero, which will be necessary if a lowered threshold is desired. Since it is of interest to mimic the ABC-MCMC simulation, we instead use to predict a point of the data set of distances. One way of doing this is to use the predicted $\hat{d}_{pred} = \boldsymbol{y}_\theta \hat{\boldsymbol{\beta}}$ and from that point generate a new value,

$$\hat{d} = \boldsymbol{y}_\theta \hat{\boldsymbol{\beta}} + \delta_\theta \sim \mathrm{N}(\boldsymbol{y}_\theta \boldsymbol{\beta}, \nu^2(1 + \boldsymbol{y}_\theta(\boldsymbol{Y'Y})^{-1}\boldsymbol{y}'_\theta)), \quad (5.2)$$

where $\boldsymbol{y}_\theta = (1, \theta^*)$ and use $\hat{\boldsymbol{\beta}}$ as an estimator for $\boldsymbol{\beta} = (\beta_0, ..., \beta_b)$ and let

$$s^2 = \frac{||\boldsymbol{d}_{p^s} - \hat{\boldsymbol{d}}_{p^s}||^2}{n - (b+1)}$$

be an unbiased estimator of ν^2, where ν^2 is the variance of the error term δ for the linear regression model (5.1). $\boldsymbol{Y}$ is the design matrix for the linear regression model (5.1). $\boldsymbol{d}_{p^s}$ are the collected distances from the training data and $\hat{\boldsymbol{d}}_{p^s}$ are the corresponding predicted distances. $b + 1 = 11$ is the number of parameters in the regression model in this case.

Now, one uses $\hat{d}$ as the distance in the indicator function for calculating the acceptance probability α_1 at step 4:3 in algorithm 3. In this manner, it is possible to predict a point from the data set of distances and for a given set of parameter θ^* be able to capture the distances close to zero as well, rather than predicting the expected distance (according to (5.1)) for a given set of parameters.

One uses the training run to set starting values before the simulation in step 4. Set the parameters θ_1 to the mean of the corresponding parameters in $D_{M-R:M}$ and let the threshold ε be the mean of the thresholds used between iteration $M-R:M$. Then, since we really seem to get use of the burn-in period in the training run, let the threshold be updated with $\gamma = 0.5$ and $h = 1000$ for the simulation at step 4.

5.1.2 Comparison of the Algorithms

With the specifications of algorithm 2 and 3 for the g-and-k distribution in the previous section 5.1.1, let $N = 100000$ and $n = 5000$. The following posterior distributions are obtained from the methods respectively for the four parameters A,B,g and k, shown in figure 5.2.

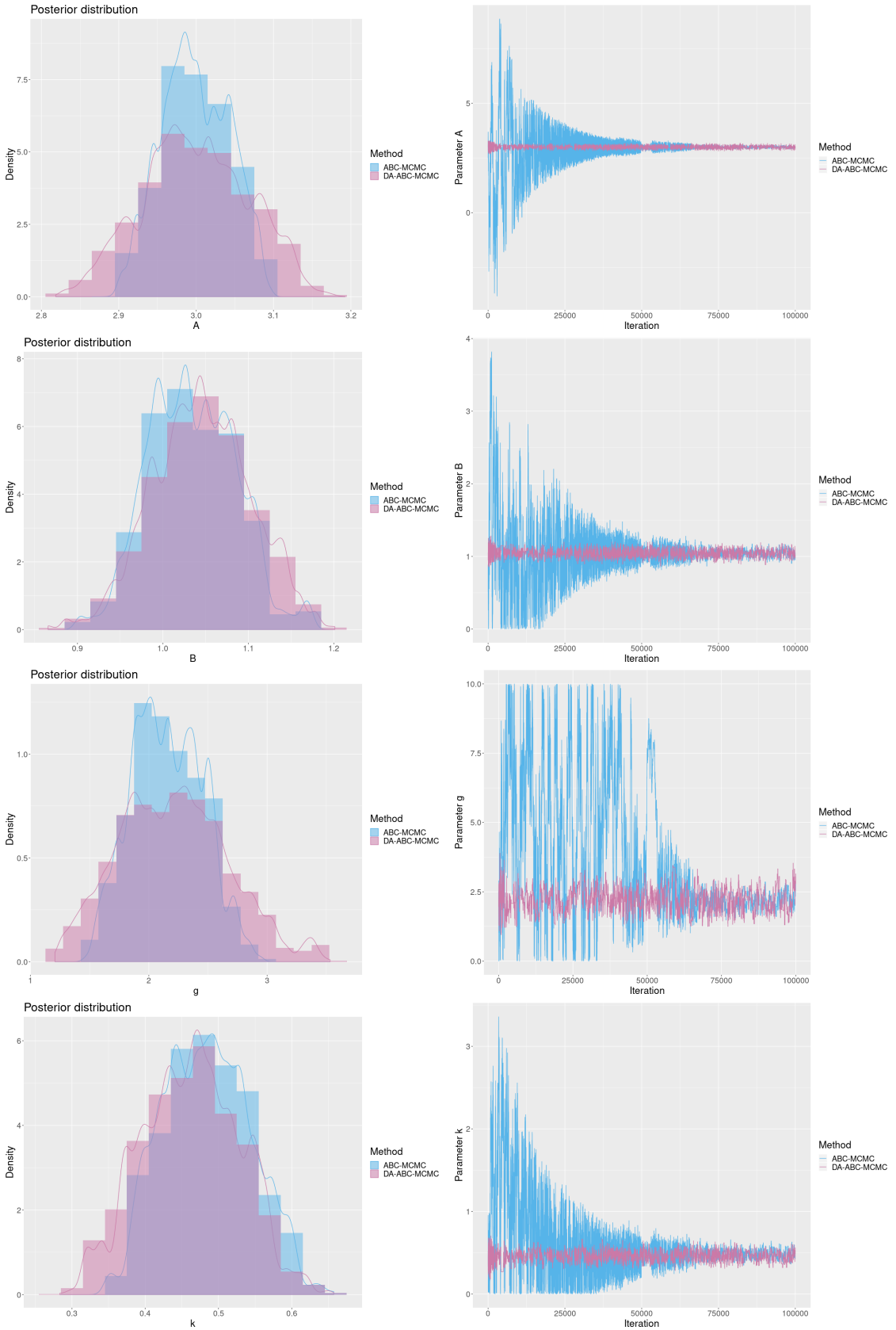

Figure 5.2: Posterior distribution for each parameter in $\theta = (A, B, g, k)$ without the burn-in period. Corresponding trace plots with the burn-in i.e. iteration 1:N in algortihms 2 and 3 for $n = 5000$. (Note, the burn-in period for DA-ABC-MCMC is done in the collection of training data D in step 2) in algorithm 3.

Note that the histograms in figure 5.2 are only based on after the burn-in period, i.e. after around 75000 iterations for ABC-MCMC and after around 10000 iterations for DA-ABC-MCMC. Recall that why the burn-in period is lower for DA-ABC-MCMC because the burn-in period is simulated when collecting the training data for the surrogate model p_ϕ^s i.e. at step 2 in algorithm 3. The DA-ABC-MCMC algorithm seems to capture the posterior distribution almost as good as ABC-MCMC for parameter B and k, but troubles a little for parameter A and g. Note also that this can due on the threshold, shown in figure 5.3, where ABC-MCMC has a lower threshold and also a little bit lower acceptance rate which can imply reaching the posterior more exact.

Compared to case 1, the burn-in period for ABC-MCMC is much longer in case 2 and there is a big difference between the burn-in period between ABC-MCMC and DA-ABC-MCMC when comparing iteration 1 to N for both algorithms, which can be seen in the trace plots in figure 5.2 and in the acceptance rate figure 5.3.

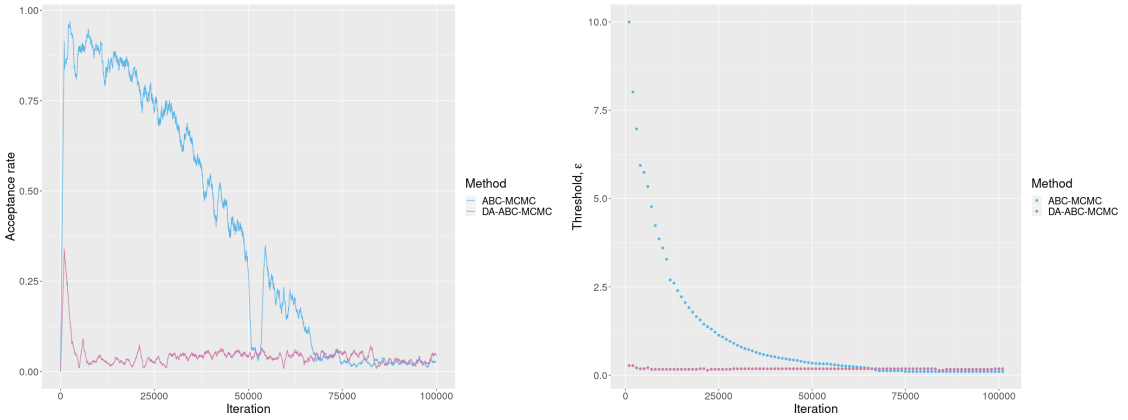

Figure 5.3: Acceptance rate α^{ar} and threshold ε for $n = 5000$ for iteration $1 : N$ in algorithms 2 and 3.

Looking at the residuals (figure 5.4) from the regression model (5.2) used for the two surrogate models, both is centered around zero, though $p^s_{\hat{\phi}_1}$ has less spread. The adj-$\mathtt{R}^2$ for $p^s_{\phi_1}$ and $p^s_{\hat{\phi}_2}$ are adj-$\mathtt{R}^2 = 0.8849$ and adj-$\mathtt{R}^2 = 0.8693$ respectively. Even though the adj-$\mathtt{R}^2$ are high and the residuals looks good, the scatter plots of the covariates (figure 5.5) shows that they are not so much explainable and especially not around zero at the y-axis. The cause of this is due to using $\hat{d}$ (5.2) as predicted distance.

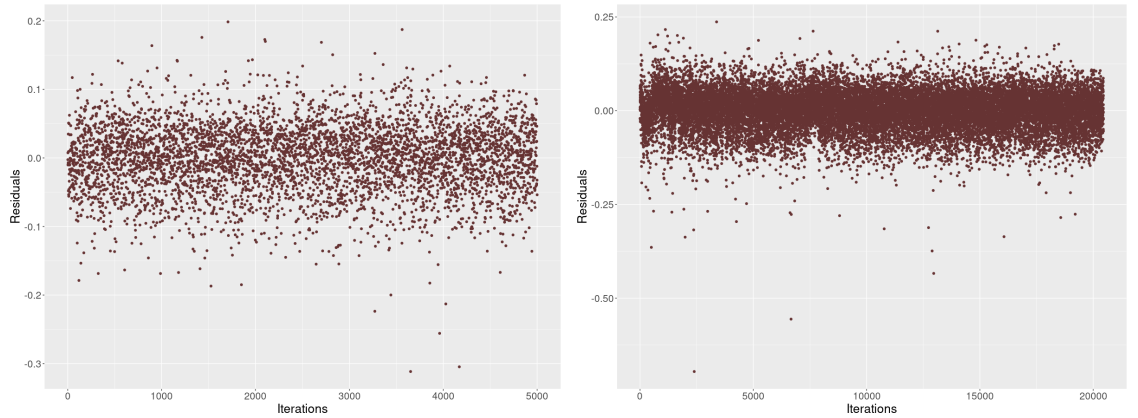

Figure 5.4: Residuals from the regression model for $p^s_{\hat{\phi}_1}$ and $p^s_{\hat{\phi}_2}$, respectively.

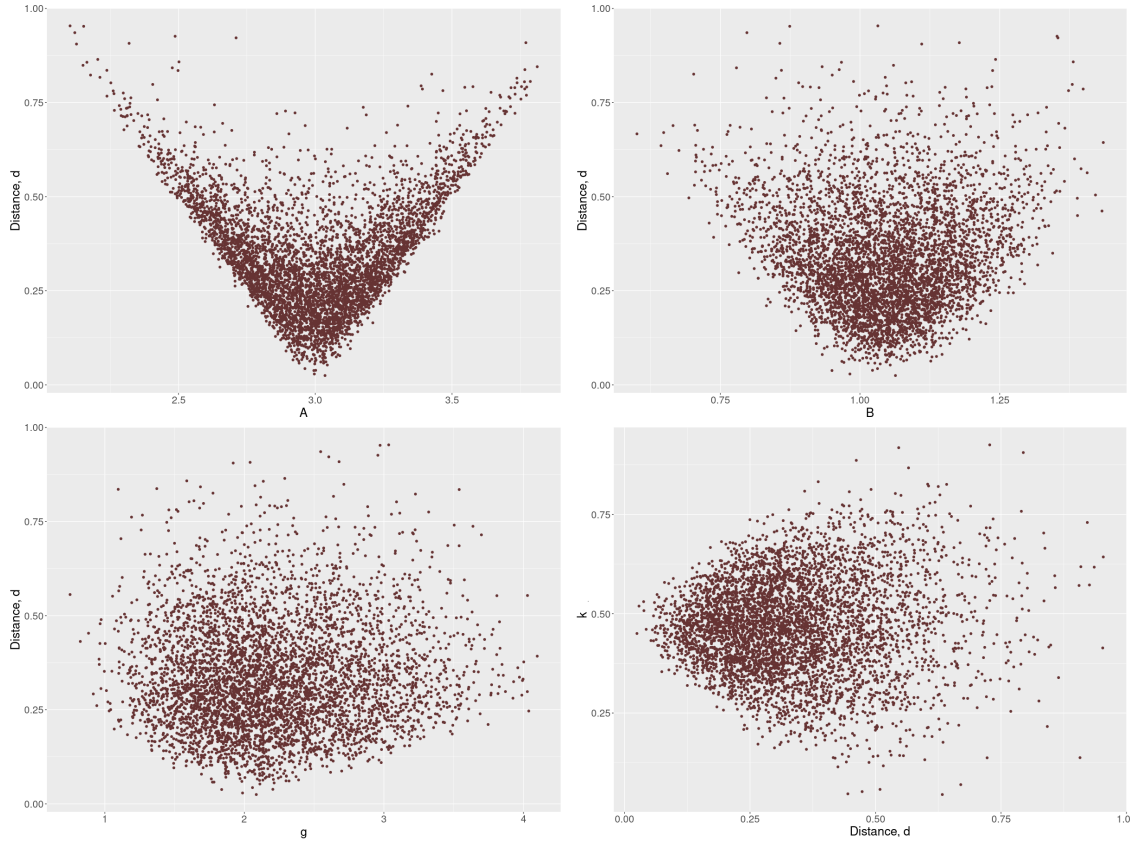

Figure 5.5: Scatter plot of d and for each parameter A, B, g and k from the training data for $p^s_{\hat{\phi}_2}$.

Figure 5.6 is a scatter plot between the predicted distances $\hat{d}$ and the real distances d. This is only the times when $\hat{d}$ have got accepted at α_1. A good correlation is not expected since the predicted distance $\hat{d}_{pred_\theta}$ is stochastic in this case. But what is important is that it predicts distances close to zero.

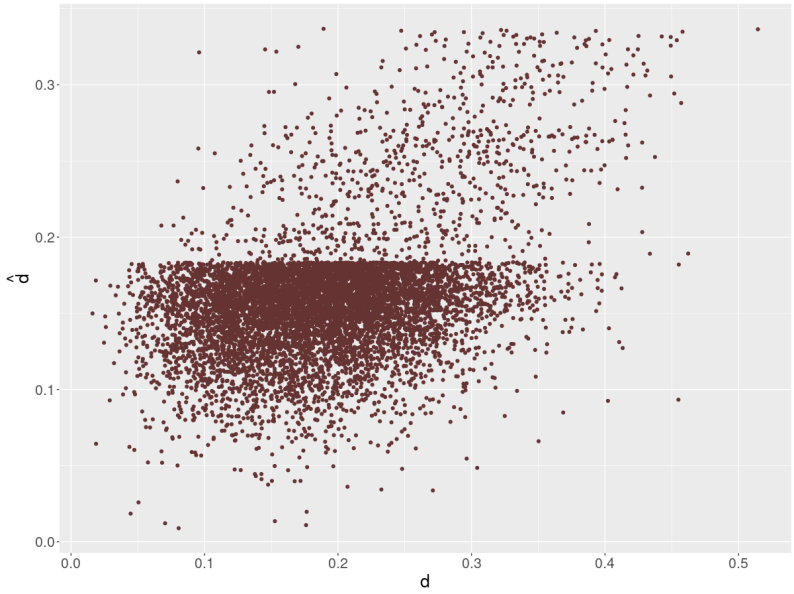

Figure 5.6: Scatter plot between $\hat{d}$ and d (only when $\hat{d}$ survived α_1).

Further, when looking at the ratio $\triangle_\alpha = 0.54$ it does not perform as good as in case 1. Though when looking for different sizes of n in figure 5.7 of $\triangle_\alpha$, it increases to around 0.6 when n increases.

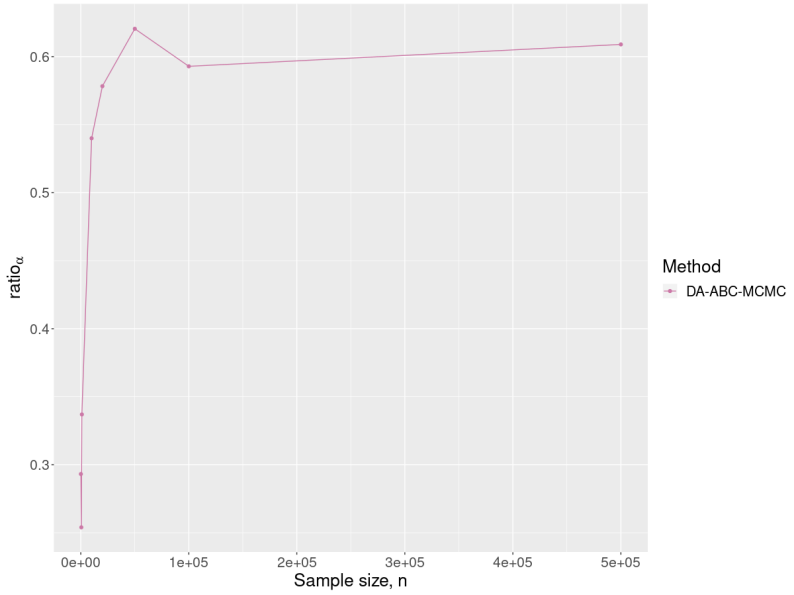

Figure 5.7: $\triangle_\alpha$ for different sample size n.

5.2 Results

The results in terms of capturing the posterior with the DA-ABC-MCMC algorithm may not be perfect but still performs good. When it comes to the results of time efficiency of the two models when $N = 100000$, as shown in figure 5.8, it is similar.

Though remember how long burn-in period ABC-MCMC has in this case compared to ABC-MCMC in case 1. Thus, the burn-in period for ABC-MCMC affects the time in DA-ABC-MCMC as well since we simulate from the same procedure when collecting the training data. This means DA-ABC-MCMC are dependent time-wice of the burn-in period of ABC-MCMC and it is the time after that which will be of benefit for DA-ABC-MCMC. To demonstrate this a little bit clearer, table 5.1 shows how long time it takes to simulate 1000 iterations after the burn-in period for ABC-MCMC and DA-ABC-MCMC respectively for different sample sizes n.

Moreover, it is interesting to compare ESS which is shown in figure 5.9, and is based on after the burn-in period for iterations 1 to N for algorithm 2 and 3, where DA-ABC-MCMC is more or less twice as efficient than ABC-MCMC. As in case 1, the result of ESS can differ depending on which value it is on the threshold ε. By looking at the right plot in figure 5.3, the threshold for ABC-MCMC is slightly lower than for DA-ABC-MCMC which should take into consideration when interpreting the result of ESS in figure 5.9.

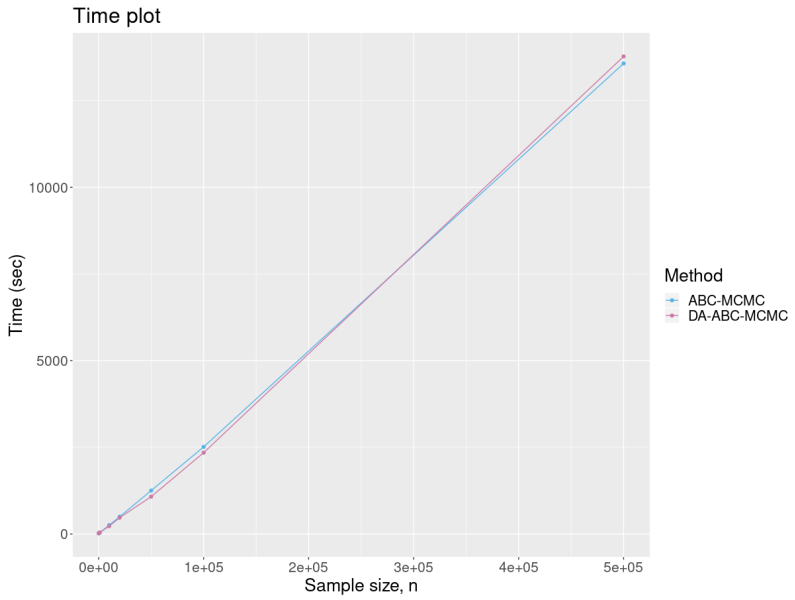

Figure 5.8: The running-time for ABC-MCMC and DA-ABC-MCMC for iteration 1 to N in algorithm 2 and 3, respectively.

n	ABC-MCMC	DA-ABC-MCMC	$\triangle$
100	0.82	0.88	0.93
500	0.92	0.95	0.97
1000	1.09	0.97	1.12
10000	3.78	1.46	2.59
20000	6.88	1.61	4.27
50000	16.25	1.75	9.29
100000	31.42	2.75	11.43
500000	162.49	7.75	20.97

Table 5.1: Table of running time (in sec) for 1000 iteration after the burn-in period for ABC-MCMC and DA-ABC-MCMC. $\triangle$ is the ratio of the running time between ABC-MCMC and DA-ABC-MCMC.

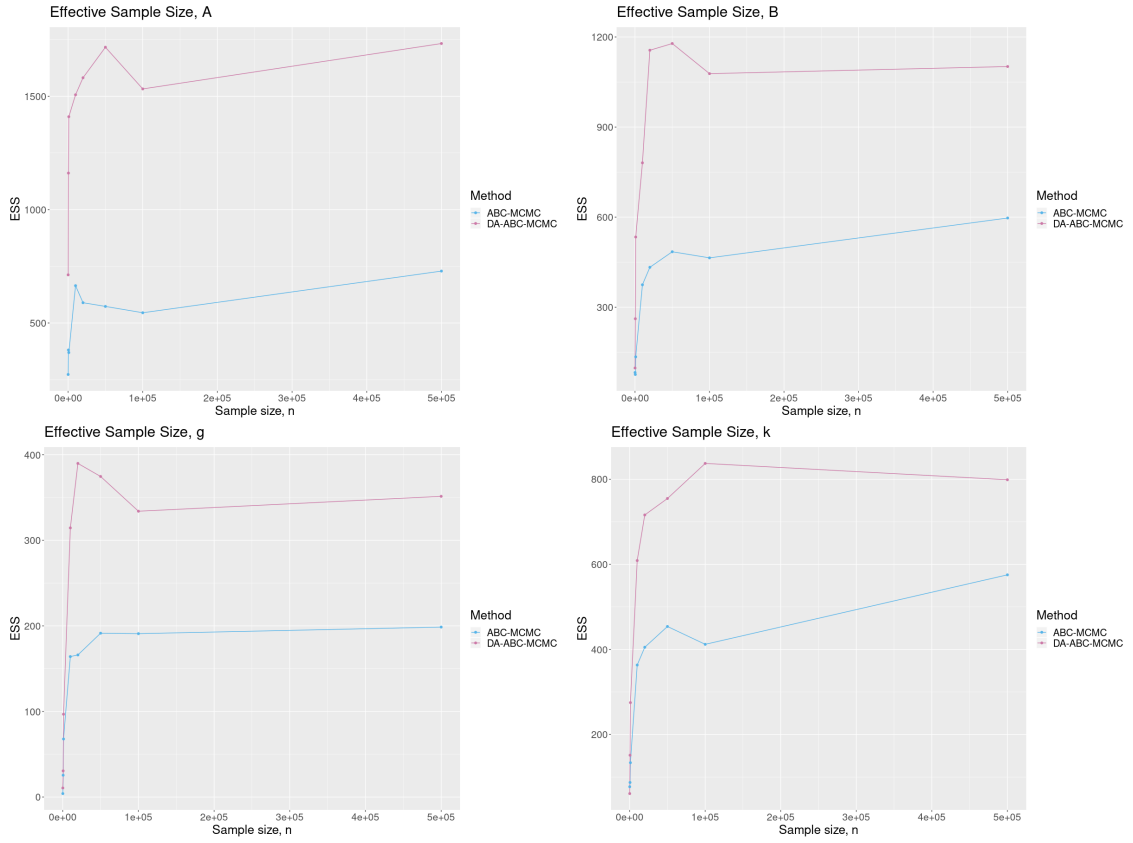

Figure 5.9: Effective sample size (ESS) of $\theta = (A, B, g, k)$ after the burn-in period for iteration 1 to N.

5.3 Analysis and Discussion

First of all, what should be mentioned is that simulation of ABC-MCMC and DA-ABC-MCMC both converges good against the right value, but are not as informative as the exact posterior distribution. This is something ABC-MCMC have trouble

with in general, to capture the posterior when it is dissimilar to the prior. As mentioned in case 1, this is due to using summary statistics instead of the whole data set in the acceptance stage and using a threshold in ABC methods, which will result in less informative result. When n increases, the posterior distribution will get more and more informative, which implies more difficulties for ABC-MCMC to target the exact posterior.

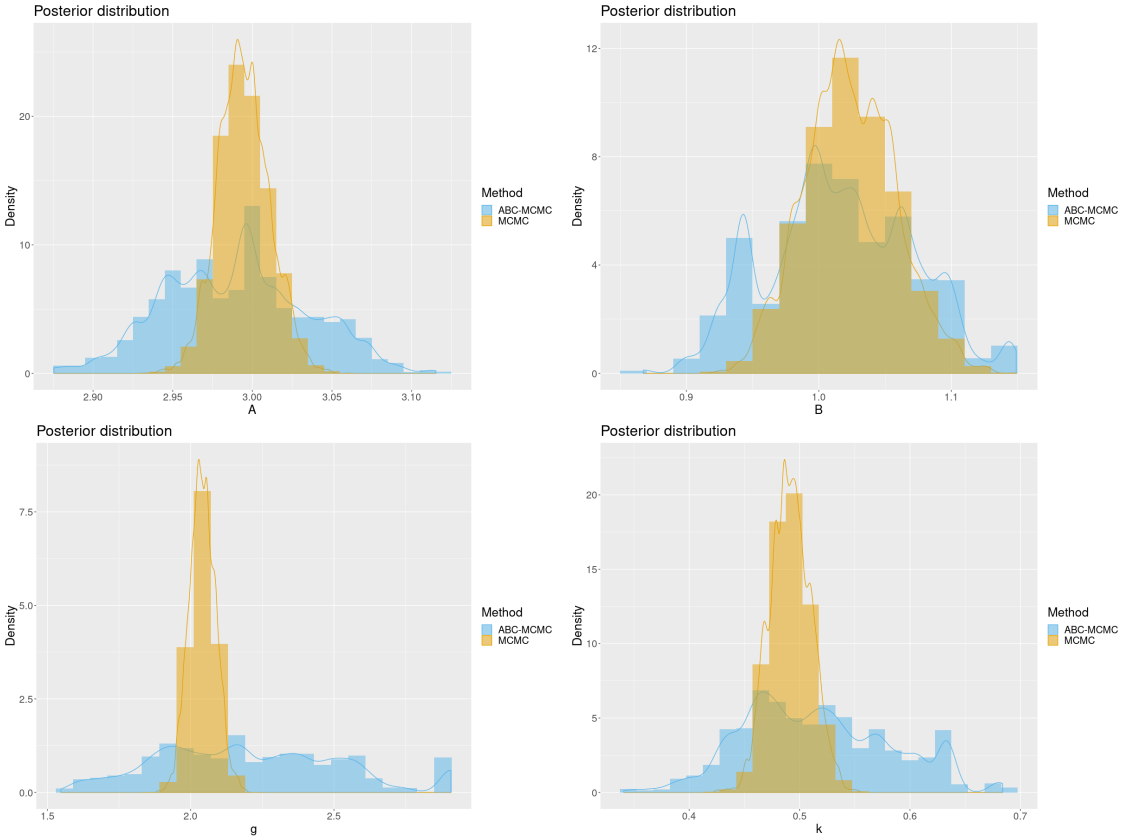

Figure 5.10: The posterior distributions via ABC-MCMC simulation compared with MCMC-simulation, with sample size $n = 5000$.

What made a big improvement in case 2 was adding the step in the prediction of the distance $\hat{d}$ according to (5.2). One problematic thing before was that the simulation of DA-ABC-MCMC algorithm did not converge every time, due to lack of predictions close to zero. The times when the simulation did converge, the ratio $\triangle_\alpha$ where approximately 0.15 or lower, depending on n.

6

Conclusion and Discussion

What still makes the DA-ABC-MCMC algorithm problematic is that it is dependent of the burn-in period from the ABC-MCMC algorithm since the collection of data to the surrogate model is from simulation of ABC-MCMC until desired data points after the burn-in period are simulated. Then it depends on how many draws from the posterior distribution are desired which will decide whether the DA-ABC-MCMC are worth using. Clearly, DA-ABC-MCMC is more computationally efficient than ABC-MCMC for simulation after the burn-in period and when the sample size is high, when interpreting table 5.1.

The DA concept for ABC-MCMC is at best use when the model p is complex and/or the sample size n is big such that the generation of x^* is very computationally inefficient. Notations from these two cases is that the explanatory variables for p^s are more and more explainable when n increase. This will benefit the training of p^s, which is seen in figure 5.7. But on the other hand, it is depending more on a good surrogate model to even be working.

6.1 Further Research

For further research in this field, I would investigate prediction methods and try to find a prediction method which is better on gimmick data. What is special in this case is that we are very interested of good prediction of the distances d close to zero, which will only be accepted if the real d is close to zero depending on the choice of ε. But on the other hand, we will also predict the distances which will not be accepted at the acceptance probability, which is in our cases around 95% which in best case scenario would be closer to 99%.

Bibliography

[1] Simon Tavaré, David J Balding, Robert C Griffiths, and Peter Donnelly. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997.

[2] Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, 16(12):1791–1798, 1999.

[3] Elise Jennings, Rachel Wolf, and Masao Sako. A new approach for obtaining cosmological constraints from type Ia supernovae using approximate Bayesian computation. *arXiv preprint arXiv:1611.03087*, 2016.

[4] Mark A Beaumont. Approximate bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*, 41:379–406, 2010.

[5] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2008.

[6] Laurent E Calvet and Veronika Czellar. Accurate methods for approximate bayesian computation filtering. *Journal of Financial Econometrics*, 13(4):798–838, 2014.

[7] Dennis Prangle et al. Adapting the abc distance function. *Bayesian Analysis*, 12(1):289–309, 2017.

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

[9] Scott A Sisson and Yanan Fan. Likelihood-free markov chain monte carlo. *arXiv preprint arXiv:1001.2058*, 2010.

[10] J Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810, 2005.

[11] Maxime Lenormand, Franck Jabot, and Guillaume Deffuant. Adaptive approximate bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796, 2013.

[12] U. Picchini and R. Everitt. Stratified sampling and resampling for approximate bayesian computation. *arXiv:1905.07976*, 2019.

[13] Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

[14] Daniel Wegmann, Christoph Leuenberger, and Laurent Excoffier. Efficient approximate bayesian computation coupled with markov chain monte carlo without likelihood. *Genetics*, 182(4):1207–1218, 2009.

[15] Michele A Haynes, HL MacGillivray, and KL Mengersen. Robustness of ranking and selection rules using generalised g-and-k distributions. *Journal of Statistical Planning and Inference*, 65(1):45–66, 1997.

[16] Christopher C Drovandi and Anthony N Pettitt. Likelihood-free bayesian estimation of multivariate quantile distributions. *Computational Statistics & Data Analysis*, 55(9):2541–2556, 2011.

[17] Dennis Prangle. gk: An r package for the g-and-k and generalised g-and-h distributions. *arXiv preprint arXiv:1706.06889*, 2017.