Grenoble INP : Ensimag

Projet de Spécialité

Simulation-Based Inference for Microbial Evolution

Authors :

Sana Ailla
Akram Elbouanani
Morad Laglil
Hajar Rouchdi

Academic Year 2022/2023

Contents

Absract

Unknown models are commonly encountered in science, but traditional analytical methods for inferring these models often rely on oversimplified assumptions that can lead to inaccurate results. However, recent advances in computational power have enabled the development of simulation-based techniques, such as Approximate Bayesian Computation (ABC), which can overcome the limitations of analytical methods. ABC techniques rely on comparing summary statistics from simulations to observed data to assess the fit between the model and the data, enabling more accurate inferences of unknown models. In this study, we used a microbial evolution simulator to test a range of ABC procedures for accurately inferring mutation rates and fitness parameters under increasingly stringent hypotheses. Our results show that ABC techniques allow for precise inference of these parameters, with an average mean error of around 3%. This new approach to microbial evolution provides more reliable conclusions compared to traditional methods, making it a valuable tool for researchers in this field.

Key words : Microbial Evolution, Approximate Bayesian Computation, Mutation Rate, Simulation-Based Inference, Kolmogorov-smirnov.

Introduction

Unknown models are common in various fields of science, including ecology and epidemiology. Analytical solutions, which rely on simplifying assumptions, have traditionally been used to infer unknown models [1]. However, when the model deviates from classical hypotheses, these methods can lead to inaccurate inferences. The need for analytical solutions was due to the computational limitations that prevented performing many calculations [2]. However, with the emergence of more powerful computational resources, new methods for inferring unknown models are emerging [3].

Approximate Bayesian Computation (ABC) techniques are a family of simulation-based approaches that allow us to infer unknown models without explicitly solving the likelihood function [4]. ABC methods rely on comparing summary statistics from simulations to observed data to assess the fit between the model and the data. This approach allows us to avoid some of the computational challenges associated with traditional likelihood-based methods, such as the requirement of evaluating complex likelihood functions [5].

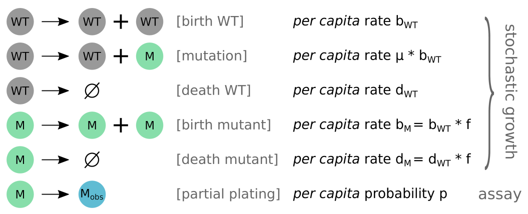
ABC methods have been applied successfully in various fields, including ecology, epidemiology, and genetics. In the field of microbial evolution, ABC techniques have shown promise in accurately inferring mutation rates and making robust inferences about the evolutionary history of microbial species [6].

The emergence of more powerful computational resources has enabled the development of new methods for inferring unknown models, such as ABC techniques. These simulation-based approaches offer a valuable tool for accurately inferring parameters in complex models with high dimensionality, which are often challenging for traditional analytical methods [6]. By focusing on ABC techniques, we can overcome the limitations of traditional analytical methods and obtain more accurate inferences of unknown models.

Using a given simulator for microbial evolution, we investigated a range of procedures based on Approximate Bayesian Computation (ABC) to gather, visualize, interpret, and deduce mutation rates or both mutation rates and fitness parameters under increasingly stringent hypotheses in a biological context. The simulator models biological conditions and generates data that closely resemble real-life scenarios. It simulates population growth with birth, death, and the accumulation of mutations, and provides the number of mutants, mutational events, wild type individuals, and the total population size at the end of the simulation. By gathering data from simulations that match the observed data, we can examine the posterior distribution of parameters and deduce the parameters that underlie the observed data. Our simulation framework incorporates Approximate Bayesian Computation (ABC) techniques, which allow us to precisely infer mutation rate or both mutation rates and fitness parameters. This capability enables us to draw more reliable conclusions about microbial evolution compared to current methods.

Model Description and Approximate Bayesian Computation techniques

1.1 Model Description



WT → WT + WT [birth WT] *per capita* rate b_WT
WT → WT + M [mutation] *per capita* rate μ * b_WT
WT → ∅ [death WT] *per capita* rate d_WT
M → M + M [birth mutant] *per capita* rate b_M = b_WT * f
M → ∅ [death mutant] *per capita* rate d_M = d_WT * f
M → M_obs [partial plating] *per capita* probability p
stochastic growth
assay

Figure 1.1: **Forward model.** WT and M represent wild-type and mutant individuals (a single class of mutants is considered). M_{obs} represent the fraction of mutants which will be observed when assaying the population by selective plating, and is different from M in case of partial plating (only a sample p of the population is plated). μ is the rate of mutation from wt to m per genome per division. Birth and death rates of the mutant and the wild-type are linked by $b_m = b_wt \times f$ and $d_m = d_wt \times f$ where f is the fitness effect of the mutation.

The Birth-Death-Mutation-Selection-Sampling model helps us understand how genetic mutations occur in populations. In this model, there are two types of individuals: wild-type and mutant. Mutations from the wild-type to the mutant are represented by the symbol μ. The fitness effect factor, denoted by f, connects the birth and death rates of the wild-type and mutant individuals. When scientists observe a population of bacteria, they may use a technique called selective plating to count the number of mutant individuals. However, the fraction of mutants observed through selective plating M_{obs} may differ from the actual fraction of mutants in the population M. The Birth-Death-Mutation-Selection-Sampling model assumes certain values for its parameters, such as $p = 1$ and $f = 1$, to simplify the model. Finally, the fitness effect term is used to affect both the birth and death rates of the mutant. This is based on experimental observations that show that the growth and death rates of bacteria are often strongly linked.

1.2 Approximate Bayesian Computation techniques

1.2.1 ABC-Rejection Algorithm

ABC techniques use simulation-based methods to approximate the posterior distribution of a model's parameters given observed data. One approach to ABC is the rejection algorithm, which involves simulating data sets based on a proposed model and accepting simulated data sets that are within a certain distance of the observed data in terms of summary statistics. This involves tuning a distance metric to achieve an appropriate trade-off between acceptance rate and accuracy of the posterior distribution approximation.

To compare two samples and determine if they come from the same distribution, we used two types of non-parametric Statistical Tests :

- The Mann-Whitney U test

The Mann-Whitney U test involves ranking the combined data from both samples and calculating the sum of the ranks for each sample. The test statistic is then calculated as the smaller of the two sums, and the p-value is determined by comparing the test statistic to the expected distribution under the null hypothesis that the two samples come from the same distribution.

- The Kolmogorov-Smirnov test

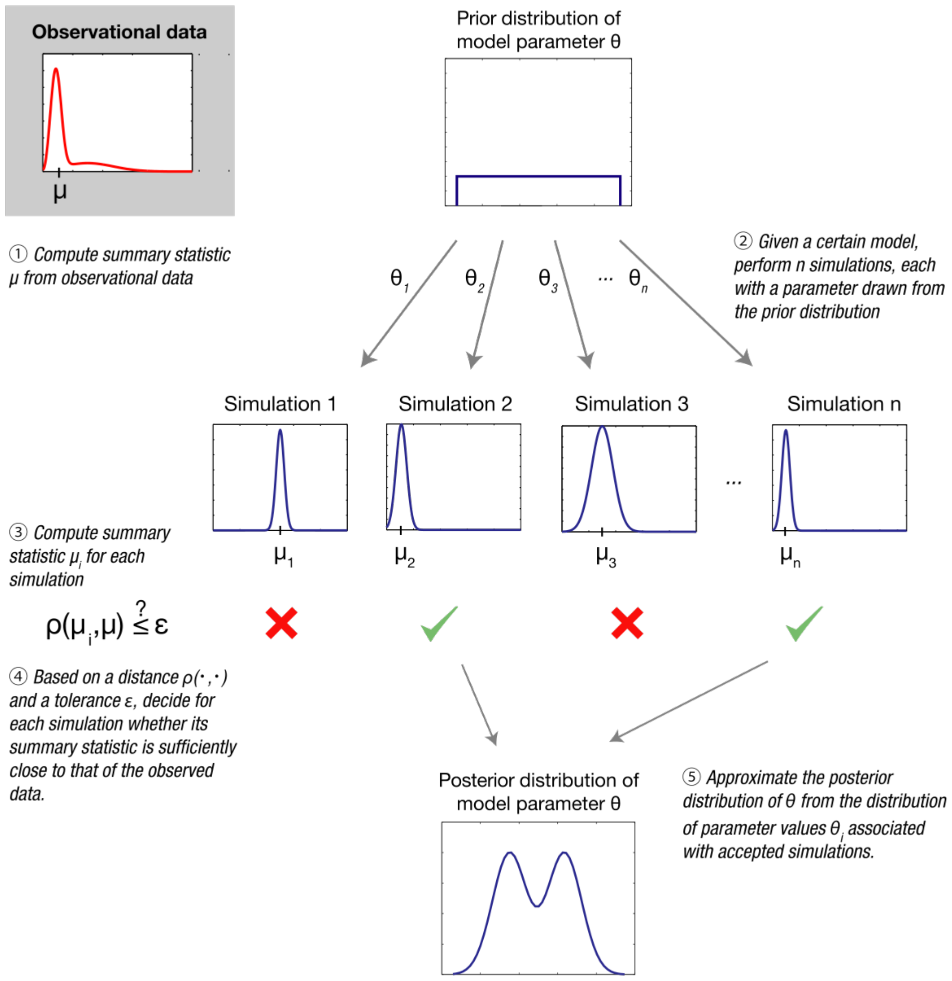

Observational data
μ
Prior distribution of model parameter θ
① Compute summary statistic μ from observational data
θ1
θ2
θ3
... θn
② Given a certain model, perform n simulations, each with a parameter drawn from the prior distribution
Simulation 1
Simulation 2
Simulation 3
Simulation n
③ Compute summary statistic μi for each simulation
μ1
μ2
μ3
μn
ρ(μi,μ) ≤? ε
④ Based on a distance ρ(·,·) and a tolerance ε, decide for each simulation whether its summary statistic is sufficiently close to that of the observed data.
Posterior distribution of model parameter θ
⑤ Approximate the posterior distribution of θ from the distribution of parameter values θi associated with accepted simulations.

Figure 1.2: **Schematic diagram of the ABC process to estimate the posterior distribution of a model parameter.** Here we start with an uninformed uniform prior distribution (top middle). We sample n times from the prior distribution and produce simulation data using our model (middle row). Prior samples that produce simulation data similar to the observed data (top left) are keep as samples that make up the posterior distribution of the model parameter (middle bottom). This is known as ABC rejection sampling. Image from Wikipedia.

The Kolmogorov-Smirnov test involves calculating the empirical distribution functions for each sample and comparing them to determine the largest discrepancy between the two sets of data. The test statistic is then calculated as the maximum difference between the CDF of the two samples, and the p-value is determined by comparing the test statistic to the expected distribution under the null hypothesis that the two samples come from the same distribution.

1.2.2 Markov chain Monte Carlo methods (MCMC)

ABC-MCMC is another approach to Approximate Bayesian Computation that involves generating a sequence of simulated data sets that gradually improve their match to the observed data in terms of summary statistics. The resulting posterior distribution is approximated based on the simulated data sets that are closest to the observed data in terms of summary statistics. MCMC methods are often used in ABC-MCMC, which generate a sequence of samples from a probability distribution that converges to the posterior distribution.

1.2.2.1 Metropolis Hastings algorithm

We opted to use the Metropolis Hastings algorithm, a widely used MCMC method. This algorithm can be applied to estimate the mutation rate that closely approximates experimental data or the mutation rate and fitness parameter by iteratively generating candidate values from a proposal distribution and accepting or rejecting them based on a probabilistic criterion.

Solutions

2.1 Algorithms and methods used

2.1.1 Inference one parameter μ

2.1.1.1 Mann-Whitney test

Let's take the Mann-Whitney Test. Assuming μ as the unknown mutation rate, The simulation is run n times with an initial population of N_0 and a final one of N_{final}. Let m be the distribution of the mutants' number at the end of the simulation. Each simulation yields an independent and identically distributed realization of M denoted as m_i, which follows a distribution m_s.

Our goal is to compute the posterior distribution of μ given the observed data $\{m_1, m_2 ...., m_n\}$ and prior knowledge about μ.

The process of using the Mann-Whitney test for ABC will be as follows:

- **Step 1:** Generate M values of μ, denoted as μ_i for $i = 1, 2, ..., M$, from a uniform distribution between 10^{-8} and 10^{-6}.
- **Step 2:** For each value of μ_i, simulate the microbial population N times to obtain N realizations of m, denoted as $m_{i,j}$ for $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$.
- **Step 3:** For each value of μ_i, compute a summary statistic T_i compute the Mann-Whitney test statistic U, which captures the difference between the distribution of $m_{i,j}$ and the given distribution of mutants at the end of the experiment m_s.
- **Step 4:** Define a criterion for accepting or rejecting μ_i based on the difference between T_i and a threshold value . Let R_i be a binary random variable that indicates whether $_i$ is accepted ($R_i = 1$) or rejected ($R_i = 0$) based on this criterion.
- **Step 5:** Use Bayes' theorem to update the posterior distribution of μ based on the accepted values of μ_i and their corresponding Mann-Whitney test statistics T_i:
- **Step 6:** Use the posterior distribution of mu to estimate the mutation rate and its uncertainty, such as the mean, median, credible intervals, or highest posterior density intervals.

2.1.1.2 Metropolis Hasting

The Metropolis-Hastings algorithm is a sampling algorithm that allows us to generate a Markov chain such that its initial distribution is the target distribution of interest. It is particularly useful for sampling from a distribution $\pi(\mu)$ when the normalization constant of this distribution is intractable. In a Bayesian context, this algorithm is often used to sample from the posterior distribution.

The algorithm we use is built on a rejection algorithm. For a given state μ^t, the algorithm specifies, based on an instrumental transition kernel $q(\mu^t, \mu^*)$, how to generate the next state μ^{t+1}, either by accepting the proposed candidate point, in which case $\mu^{t+1} = \mu^*$, or by rejecting it, in which case $\mu^{t+1} = \mu^t$.

The algorithm can be summarized as follows:

- **Step 1:** Initialize μ^t and y^t.
- **Step 2:** Propose $\mu^* \sim q(\mu^*|\mu^t)$.
- **Step 3:**. Generate y^* conditionally on μ^* and compute $I_y(y^*)$.
- **Step 4:** If $I_y(y^*) = 1$, go to step 5, otherwise stay at μ^t and return to step 2.

- **Step 5:** Calculate acceptance probability $\alpha = \min\left(1, \frac{\pi(\mu^*)q(\mu^t|\mu^*)}{\pi(\mu^t)q(\mu^*|\mu^t)}\right)$.
- **Step 6:** Generate $u \sim U(0,1)$. If $u < \alpha$, set $\mu^t = \mu^*$, otherwise stay at μ^t.
- **Step 7:** Return to step 2.

The variable $I_y(y^*)$ is a Boolean expression that evaluates whether a generated distribution meets an acceptability criterion. Specifically, it determines whether the generated distribution is similar enough to the initial generated distribution and the distribution generated in the previous iteration.

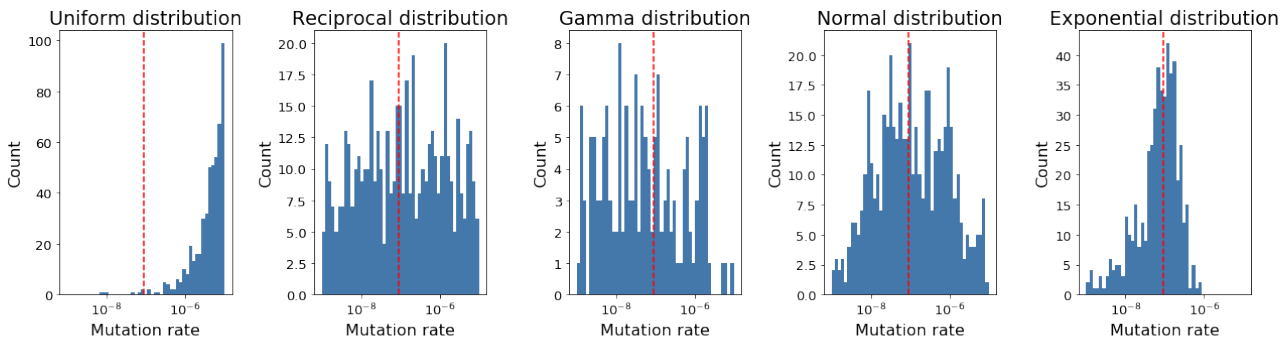
Choice of Parameters

In order to investigate the impact of prior choice on the accuracy of our predictions, we first considered the range of mutation rates observed in bacteria, which have been shown to typically fall within the interval of $[10^{-9}, 10^{-5}]$. This range was selected based on empirical observations and has been widely reported in the literature[7][8]. While this assumption is not critical to the validity of our inference, it is important for computational efficiency, as some simulators may struggle to accurately model extremely high or low mutation rates. By restricting the prior range to biologically realistic values, we can optimize the performance of our simulator and make more accurate predictions.

We settle for the following priors : A uniform prior and a log-uniform prior as they are non-informative, a Gamma, a Normal and an Exponential distribution.

Next, we parameterize our priors such that they're able to overcome any bias introduced by the observed data, which means incorporating statistics of the observed data that are resistant to outliers in our parameters. For the Gamma prior, we use both the median and interquantile range, while for our Log-Normal distribution, we use the median and the geometric mean of our observed data and we use the median for the exponential distribution.

Using those different priors, for a random value of μ, we generate an observed data that we then give to our priors. In the figure below are shown the distributions of the different priors and in red the value mu. We are furthermore able to verify that our priors are able to recover values that are too small or too big as well, where the mean or the median of the observed data divided by the number of the samples is too different from the actual value of μ, showing that our priors are resistant to outliers and bias.



Those same models are reused to provide transition kernels, centering the generated values around the last generated value.

Lastly, we use two different acceptance criteria for the newly generated data. The first one evaluates, using a KS-test and a given error value, the closeness of the generated data to the initial observed data. Second, we restrict this further by imposing that the generated data must also be better than the previously generated one, by using a KS-test to compare both to the initial observed data and only picking the newly generated data if it is, with a given threshold, better than our previous one.

2.1.2 Inference couple of parameters

2.1.2.1 ABC-Rejection

This approach aims to infer the mutation rate (μ) and fitness parameter (f) of a population given an observed dataset and other fixed parameters. The approach works by repeating the following process for N iterations:

- Generate a random couple (μ, f) from a log-uniform prior distribution.

- Simulate a new sample using the generated couple (μ, f) and the other fixed parameters.
- Compare the new sample with the observed data using the Kolmogorov-Smirnov (KS) test and store the results of all N couples.
- If the distance between the simulated and observed data is below a certain threshold (ks_{seuil}), the couple is accepted and added to the posterior distribution.
- The best couple (μ, f) is then selected based on the minimum KS distance, which means the closest distribution to the given data.

2.1.2.2 Grid Sampling Algorithm

This approach involves sampling a grid of possible values for the mutation rate and fitness parameter, and then computing the KS statistic for each combination of values. For a given data generated with an unknown mutation rate and fitness parameter and other known parameters, the algorithm works as follows:

- Define the ranges and steps for the mutation rate and fitness parameter to create a grid of possible values.
- Simulate a new sample for each combination of mutation rate and fitness parameter in the grid, using the other fixed parameters.
- Compare each simulated sample with the observed data using the Kolmogorov-Smirnov (KS) test, and store the results of all combinations.
- Select the combination of mutation rate and fitness parameter that yields the minimum KS distance, which means the closest distribution to the given data.

In this approach, the main idea behind the grid sampling algorithm is the visualization of the results to detect the problem of unidentifiability. By plotting the KS statistics for different combinations of mutation rate and fitness parameter, we can identify regions of the parameter space where the data cannot distinguish between different values of the parameters.

2.1.2.3 ABC-MCMC algorithm with a threshold value on the couple (f,μ)

The goal of this section is to develop a generalized ABC-MCMC method (with a threshold value) for inferring two independent parameters, mutation rate μ and fitness f, from experimental data. In our algorithm, we employ an exponential transition kernel with a mean set to the current parameter value, as this allows us to explore the entire interval around the current value. In the absence of prior information, we use a log-uniform distribution to propose new values for μ and f. Our ABC-MCMC method uses an acceptance criteria based on a threshold value for the ratio of the distance to the exprimental data between the current value and proposed values (for exemple distance based on kolomogrovsemrinov test). Specifically, the proposed value can be accepted if:

$$\frac{kss_{\text{current}}}{kss_{\text{proposed}}} > seuil$$

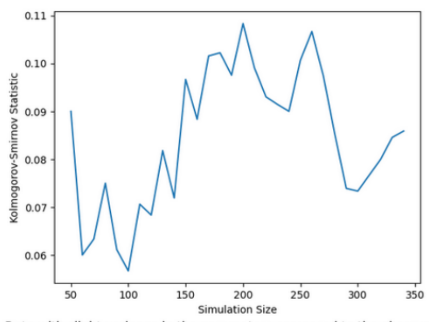
Where seuil is the pre-defined threshold value and $kss_{current}$ and $kss_{proposed}$ are the distances between the generated data using the current and proposed parameter values, respectively, and the experimental data. If this criteria is met then we calculate the acceptance probability of the proposed value:

$$\alpha(x, x') = \min\left(1, \frac{p(x')q(x'|x)kss_{current}}{p(x)q(x|x')kss_{proposed}}\right)$$

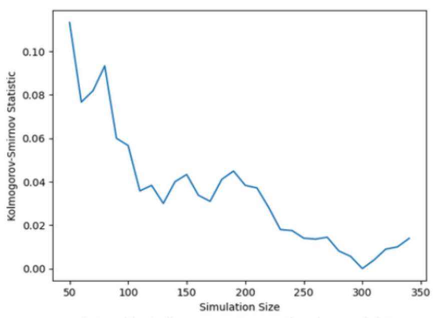
Here $x = (\mu, f)$, $p(x)$ and $p(x')$ denote the marginal probabilities of the current and proposed parameter values, respectively. $q(x|x')$ and $q(x'|x)$ represent the proposal distributions used to generate the current and proposed values, respectively. By using this acceptance probability, we can ensure that the proposed value is accepted if it is sufficiently close (comparing to the current value) to the experimental data and if it satisfies the predefined threshold for the ratio of distances.
Tuning the threshold value for the acceptance criteria can be challenging. If the threshold is set too high, the algorithm can become stuck in an undesired region, leading to a low acceptance rate and large estimation error. On the other hand, if the threshold is set too low, the algorithm may fail to converge to the true value and oscillate around it. After testing our algorithm on various cases, we found that a threshold of 0.95 strikes a good balance between these trade-offs.

2.1.3 The impact of the simulation size on the Kolmogorov-Smirnov test

In all methods that we use in this project, we need to run the simulator. Running the simulator with specific parameters and generating a sufficient number of samples can be time-consuming, especially when the mutation rate μ is small and the fitness is large, among other reasons. To reduce the computational time, we decided to restrict the fitness interval to [0.1, 2]. However, the runtime of our algorithms is still dependent on the number of simulations conducted. To address this issue, we need to find a way to reduce the simulation size without compromising the accuracy of the Kolmogorov-Smirnov test used for comparing distributions.
One potential concern with using small sample sizes in the K-S test is that it may have low power and may not detect differences between distributions even when they exist. To address this concern, we will investigate the impact of simulation size on the K-S test and determine the smallest simulation size that aligns well with our observed data (in this case, 300 samples, data observed parameters are generated randomly).



Data with slight variance in the parameters compared to the observed data



Data with similar parameters as the observed data

We observe that the simulation with 300 samples yields a KS test result close to other simulations with a smaller number of samples, particularly when the size is greater than 200. Thus, we conclude that the KS test is not highly sensitive to the size of the compared data when considering a size larger than 200. To generalize this study, we will reduce the sample size to 60% of the original size in case we need to speed up the computation.

2.2 Quantification of Results

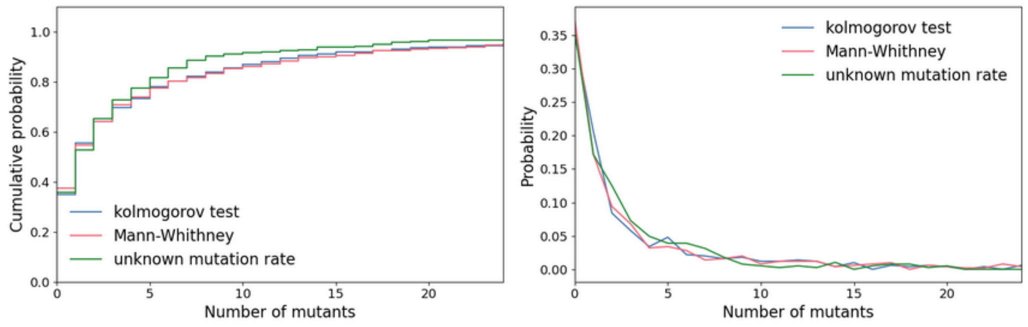
To qualify the best solution/modeling we used :

- Histograms and Cumulative Distribution Function CDF/ Probability Density Function PDF plots are used to visually compare observed and simulated data distributions and assess model fit.
- By comparing the CDF and PDF plots of the observed data and the simulated data, we can determine how well the model fits the data.
- A plot is used to evaluate the Metropolis Hastings algorithm's performance in estimating the mutation rate μ and the fitness parameter f. The plot includes a blue line for accepted μ values, a red line for the actual μ value, and a green line for the average of accepted values. Comparing the blue and red lines indicates accuracy, while the green line provides an overall indication of the average accepted value. The same approach is applied to estimate the fitness parameter f.
- Boxplots were used to display the errors for each kernel used in the Metropolis Hastings algorithm. The errors were calculated as the absolute difference between the estimated and actual values of μ and f, divided by the actual value. Comparing the errors for each kernel allowed us to determine which kernel was the most effective in estimating μ and f
- A hitmap is used to visually represent the relationship between parameters and resulting data, indicating the distance between simulated and observed data. It helps identify parameter values that are likely to be true and highlight any issues with unidentifiability of parameters.

Results and Interpretations

3.1 Mutation rate μ

3.1.0.1 Mann–Whitney U and Kolmogorov–Smirnov tests



Using the kolmogorov-Smirnov test and Mann–Whitney U test we can conclude that a value of estimated mutation rate is $\mu = 8,8 \times 10^{-8}$.

3.1.0.2 Methropolis Hastings

The accuracy of the inference is first evaluated via a given observed data to which we know the value of mu.

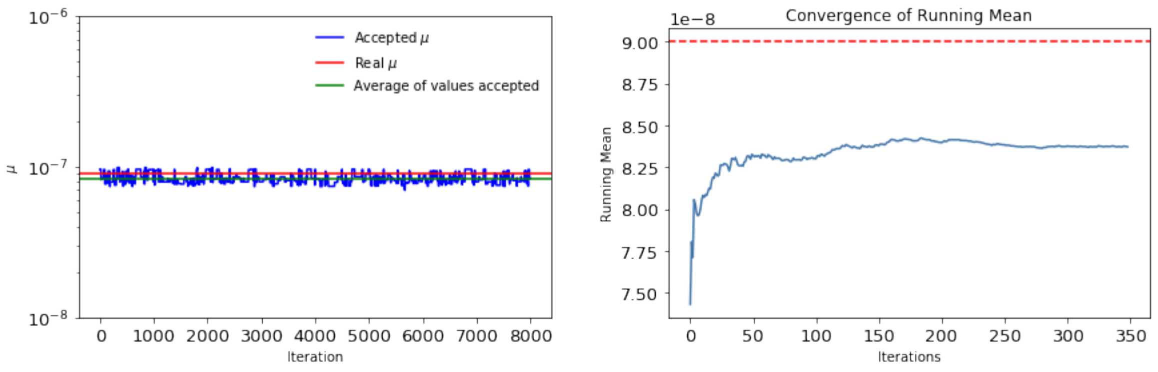


Figure 3.1: Mean value of mutation rate values accepted by gamma prior is $\mu = 8,379275512972397e - 8$ with 10000 iterations from which 2000 are for burn in. Real value $\mu = 9e - 8$

We observe the convergence of our algorithm, albeit not to the exact same value, but with a relative error of 0.7%, in a relatively small amount of iterations.

Using the comparison between the current generated data and the previous one as an acceptance criterium, and an exponential prior, we generate an observed data for 200 random values μ and compare the inferred mutation rate to the real value. For 1000 iterations of the Metropolis-Hastings algorithm from which 100 are for burn-in, we get the following relative errors for each kernel used.

All kernels relatively perform the same when it comes to accuracy, although some may provide faster converge. All in all, we are able to confidently conclude that out Metropolis-Hastings algorithm works and infers the mutation rate with a 3% relative error on average.

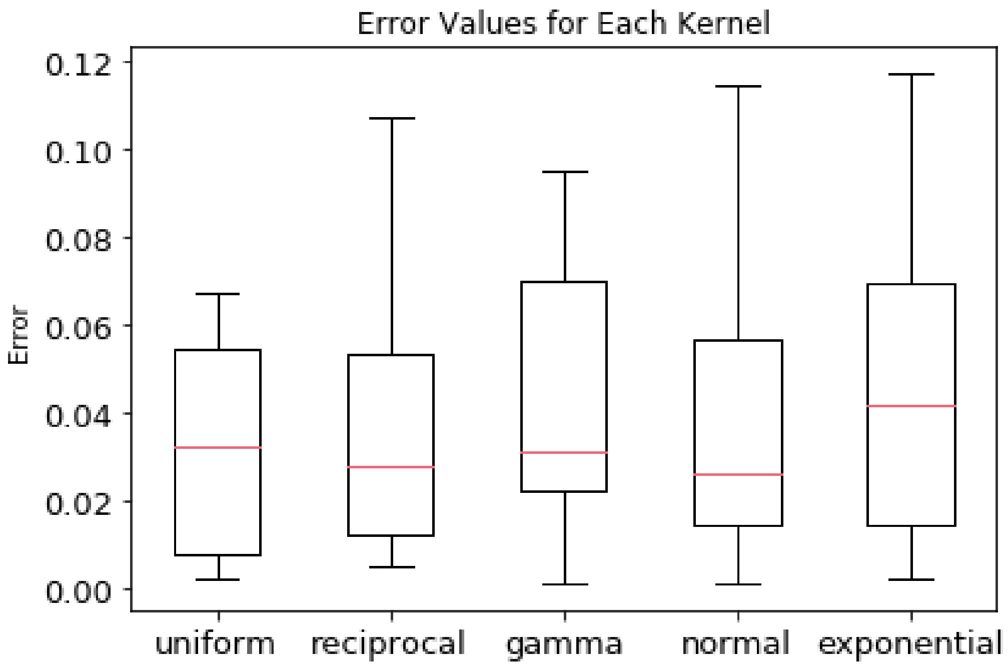


Figure 3.2: Relative error for different kernels. In red : median.

3.2 Mutation rate and fitness parameter (μ, f)

3.2.1 ABC-rejection and Grid-Sampling

We generated a `data1` with fixed parameters p = 1 , N_0 = 10, N_f = 1e7 and dt = 0.3, μ = 2e-8 , f = 1.17. The grid sampling algorithm divided the parameter space into a grid of 64 by 64 points, with a total of 4096 possible parameter combinations. For each parameter combination, the simulator was used to generate a sample , and the distance between the generated sample and the true distribution was calculated using the Kolmogorov-Smirnov (KS) test.

The resulting plots above represent a heat map that displays the KS p-value for each combination of parameter values used in the grid sampling algorithm and the posterior of μ and f using the accepted values based on a threshold of ks = 0.3. Each cell in the heat map represents a unique combination of mu and f values, and the color of the cell indicates the KS p-value obtained for that combination.

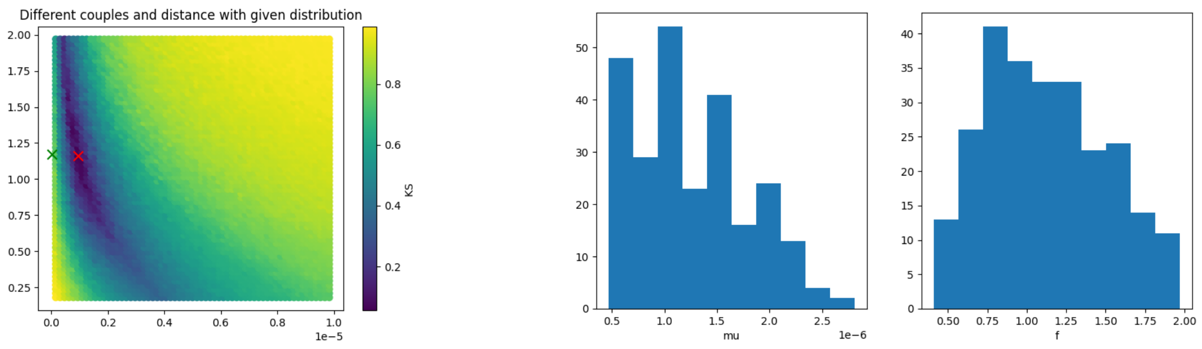


Figure 3.3: Hitmap for data1 and Posteriors for mu and f

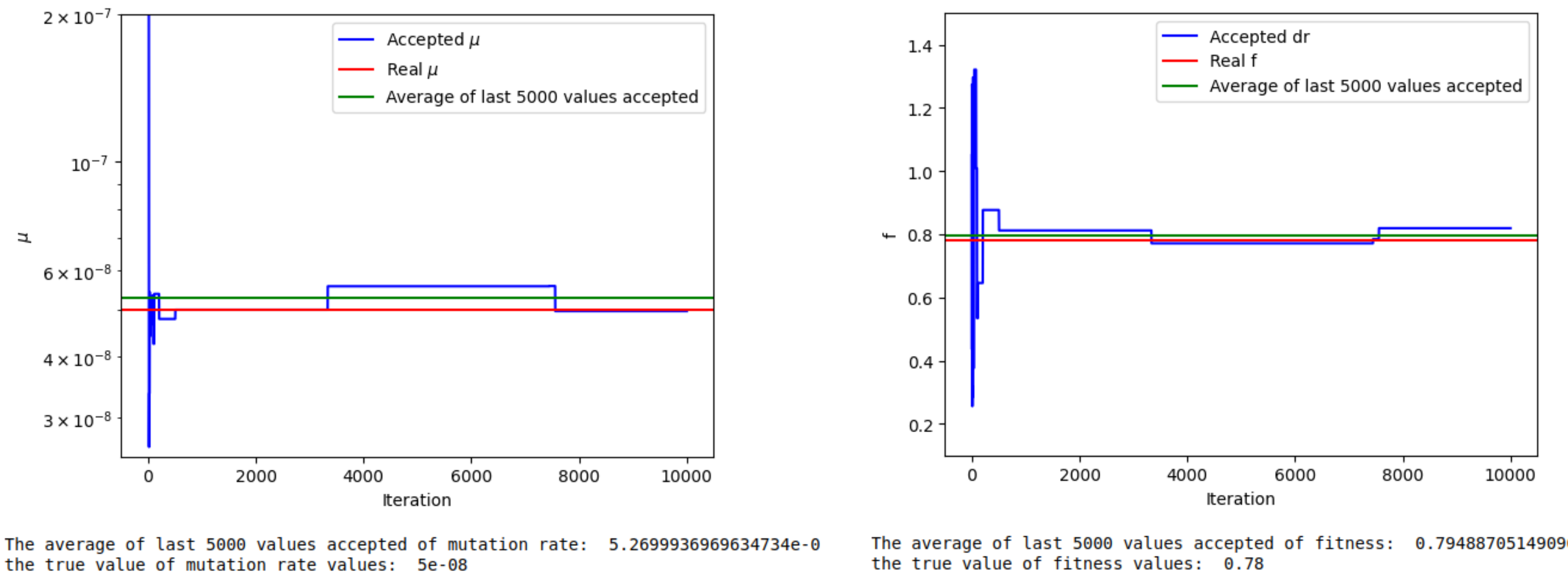
- We can observe that there are certain regions in the heat map where the KS p-value is low, indicating that the generated samples using those parameter values are significantly different from the observed data. However, there is a distinct region where the KS p-value is minimized, indicating that the generated samples using the corresponding parameter values are the closest to the observed data in terms of distributional similarity. This allows us to identify the values of μ and f that are most likely to have generated the observed data.

- The inferred value for the couple is $(1.161171875 \times 10^{-6}, 9.375090625 \times 10^{-7})$ with a 95% confidence interval of $[0.52557813, 1.87404805]$ for f and $[4.68759531 \times 10^{-7}, 2.34375766 \times 10^{-6}]$ for μ.

- Both the algorithm mentioned and the ABC-rejection algorithm correspond to a brute force calculation with a high computational cost, which may not always lead to accurate results. This is why MCMC methods are often used as an alternative, as they can provide more accurate results in a shorter amount of time.

3.2.2 ABC-MCMC algorithm on the couple (μ, f)

One way to evaluate our ABC-MCMC algorithm on a couple of parameters is to generate observed data (samples of number of mutation) with a randomly chosen mutation rate and fitness, while keeping the other parameters fixed as mentioned in the previous section. Let's take the mutation rate to be equal to 5e-8 and the fitness to be equal to 0.78, and then attempt to infer these parameters using the ABC-MCMC algorithm.



```
The average of last 5000 values accepted of mutation rate:  5.2699936969634734e-0
the true value of mutation rate values:  5e-08
```

```
The average of last 5000 values accepted of fitness:  0.79488705149090
the true value of fitness values:  0.78
```

Figure 3.4: Convergence Graph for Mutation Rate and Fitness Using ABC-MCMC

When taking the initial parameters at the mean of each interval, we noticed that the algorithm converged to the true values after the first 2000 iterations.

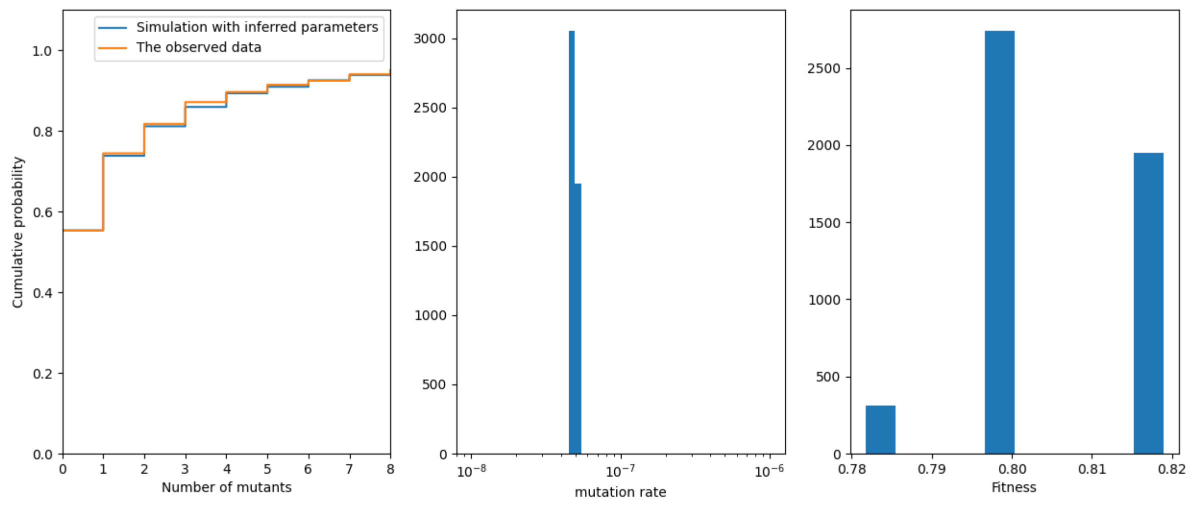
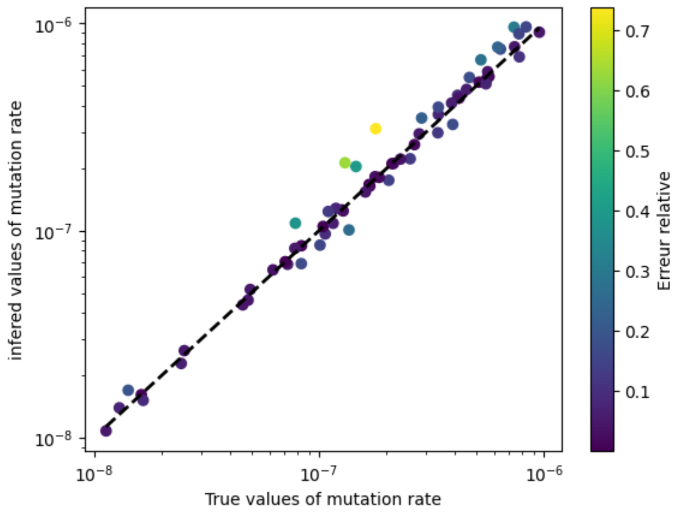


Figure 3.5: Comparison of Cumulative Probability Distributions for Observed Data and Data Generated with Inferred Parameters, and Histograms of Mutation Rate and Fitness Distributions Found with ABC-MCMC by Taking the Last 5000 Accepted Samples

The cumulative probability of the observed data and data generated with the inferred parameters are very close, with a maximum difference of 0.019 (Kolmogorov-Smirnov test of two data). However, the posterior distribution of mutation rate and fitness found with this algorithm has a very small number of samples because the algorithm converges very quickly. As a result, we cannot build a powerful confidence interval with a small number of samples in the posterior.

To validate the performance of our algorithms, it is not sufficient to evaluate them with just one example. Therefore, we ran our algorithms on 100 experimental datasets contained in Challenges 3 to infer both mutation rate and fitness. However, due to the long runtime of the simulator, we had to limit the simulation time. Therefore, if a simulation exceeded 2 seconds, we stopped it and moved on to the next simulation. A summary of the inference error for both mutation rate and fitness is shown in the following graphs where we plot the true values against inferred values with the relative error. The graphs indicate that the error is relatively small, with a mean error of 6 percent for the mutation rate and 11 percent for the fitness.
We have included a user manual that explains how to run our scripts on experimental data, and we have also created Jupyter notebooks to explain our approach during this project.



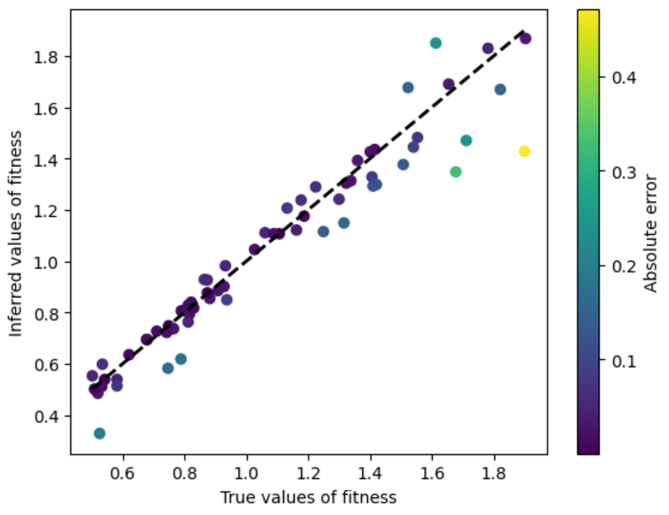
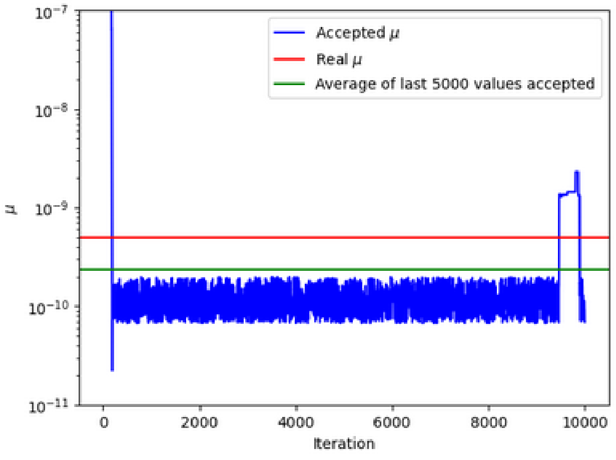


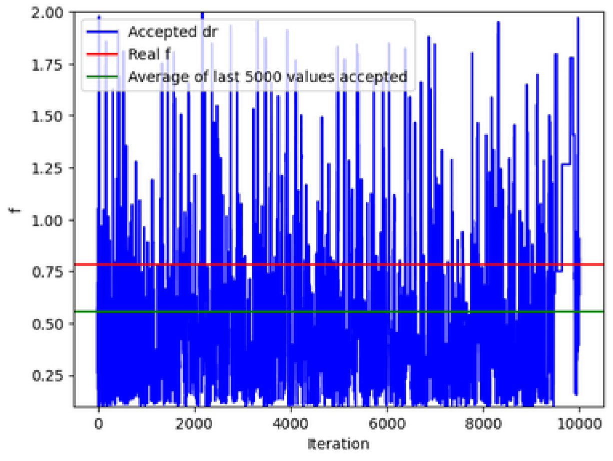
Figure 3.6: Graphs of the true values against inferred values

Discussion

Based on the previous results presented in the report, it appears that the ABC-MCMC algorithm can provide accurate estimates for unknown parameters, as long as the parameter values are not too extreme. However, the algorithm may struggle when the mutation rate is very small (e.g., 1e-11), and when attempting to infer the fitness parameter, which can present challenges in parameter identification.



```
The average of last 5000 values accepted of mutation rate:  2.342829283053579e-10
the true value of mutation rate values:  5e-10
```



```
The average of last 5000 values accepted of fitness:  0.557430287226075
the true value of fitness values:  0.78
```

Figure 4.1: The ABC-MCMC algorithm does not converge to the true value when μ is extremely small. f takes a vast range of possible values, making it difficult to infer the true value due to parameter identification issues.

A way to further advance the study is to explore other avenues, such as developing a method to determine whether a given observed data set allows for the inference of unknown parameters or if there is an issue with parameter identification(for example: detector for identification problems without relying on heatmaps).

One of the most challenging problems in our study is the runtime of the simulator. Although we have reduced the interval of fitness and the number of generated samples, it still takes a long time, especially when inferring multiple observed data (e.g., 100 data in challenges 2 and 3). To mitigate this problem, we decided to set a maximum simulation time of 2 seconds and terminate simulations that exceed this limit. However, this solution can bias the inference process and lead to inaccurate estimations. Therefore, it would be beneficial to investigate under which conditions we should stop the simulation and switch to another one, so that we can reduce the inference time while maintaining the accuracy of our results.

Conclusion

In conclusion, the ABC methods, including rejection algorithms and Metropolis-Hastings, have shown promising results in inferring unknown parameters in the context of genetic evolution. These algorithms were able to accurately estimate parameter values when they were not too extreme. However, there are several assumptions made to improve the efficiency of the inference process while maintaining the accuracy of the results. Overall, ABC algorithms have great potential to be applied in other fields of research that involve simulations and parameter estimation. However, it is important to properly tailor the algorithm to the specific problem and domain to obtain accurate and reliable results (prior information, domain knowledge,...).

User Manual

A set of files is provided in the delivery, containing most of the work done during the project:

- `FirstChallenge.ipynb`: A Jupyter notebook containing the different studies done for the choice of the prior for ABC-rejection sampling, the used statistic for ABC method, choice of transition kernel in ABC-MCMC ...
- `InferenceDeuxParams.ipynb`: A Jupyter notebook containing all the functions and methods used for inference for two parameters, mutation rate and fitness, as well as the evaluation of the final error of the last adopted method.
- `1p_inference.py`: A Python script containing the method adopted for inference for one parameter (mutation rate), taking as input the entire data set and fixed parameters, and performing inference for both parameters. For more details, simply run `./1p_inference.py -h`.
- `2p_inference.py`: A Python script containing the method adopted for inference for two parameters (mutation rate and fitness parameter), taking as input the entire data set and fixed parameters, and performing inference for both parameters. For more details, simply run `./2p_inference.py -h`.
- `Tools.py` : A file containing some functions used throughout the project.

Bibliography

[1] Mark A. Beaumont. Approximate bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41:379–406, 2010.

[2] Mark A. Beaumont. Approximate bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41:379–406, 2010.

[3] Alex R Hall, R Craig MacLean, and Graham Bell. Experimental evolution of antibiotic resistance. *Trends in microbiology*, 25(11):958–967, 2017.

[4] Jean-Michel Marin and Christian P. Robert. *Bayesian Essentials with R*. Springer Science & Business Media, 2012.

[5] Oliver Ratmann and Christophe Fraser. Model fitting and model selection for infectious diseases using Statistical methods. *PLoS ONE*, 4:e5687, 2009.

[6] S.A. Sisson, Y. Fan, and M. Beaumont. Sequential monte carlo for population genetics. *Journal of the American Statistical Association*, 102:1152–1162, 2007.

[7] Simon Tavare, D.J. Balding, R.M. Griffiths, and P. Donnelly. Inferring genetic parameters using monte carlo methods. *Journal of the Royal Statistical Society, Series B*, 53:569–605, 1997.

[8] Michael J Wiser, Noah Ribeck, and Richard E Lenski. Long-term dynamics of adaptation in asexual populations. *Science*, 342(6164):1364–1367, 2013.