Approximate Bayesian Computation (ABC)

Carando Fabio (5206221), Dal Ben Simone (5201062), Martina Dotti (5206705)

1. Introduction

Approximate Bayesian Computation (ABC) is a statistical approach used to address Bayesian inference problems in complex situations where the exact computation of the posterior distribution is difficult or impossible.

$$P(\theta|Y) = \frac{P(Y|\theta)P(\theta)}{P(Y)} = \frac{P(Y|\theta)P(\theta)}{\int P(Y|\theta)P(\theta)\ d\theta}$$

In ABC framework (indeed also called *likelihood - free methods*) the likelihood is not directly computed, but instead an approximation of the likelihood is employed to assess the discrepancy between observed data and data generated by the model.

In the context of ABC, the inability to compute the exact likelihood often arises from computational complexity or mathematical models that render direct computation impractical. Therefore, ABC provides an alternative that circumvents the direct computation of the likelihood.

Instead of the likelihood, in ABC, simulated data is generated using values sampled from the model's parameters according to a prior distribution. These simulated data are generated repeatedly until they match a pre-defined similarity measure, such as the distance between the summary statistic of the observed data and that of the simulated data.

If the generated simulated data is similar to the observed data in terms of summary statistics, then the parameter values that produced such simulated data are considered as approximate samples from the posterior distribution.

The basic idea is to generate samples from a prior distribution of parameters, generate simulated data using those parameters, and then compare a summary statistic of the simulated data with that from the observed data.

The ABC is based on a fundamental concept: instead of directly comparing the observed data with those generated by the model, the summary statistics are compared. Summary statistics are measures such as mean, median, or standard deviation. The idea is that if the generative model were correct, the summary statistics computed from the observed and model-generated data should be similar.

The inference process in ABC takes place through the following steps:

1. Sampling from a prior distribution of the parameters;
2. Generating simulated data from the generative model using the sampled parameters;
3. Computing summary statistics of the simulated data;
4. Comparing the summary statistics of the observed data with those of the simulated data;
5. If the summary statistics are sufficiently similar, the associated parameters are accepted as approximate samples from the posterior distribution;
6. Repeating the previous steps until enough approximate samples is obtained.

Rubin (1984) makes a visionary statement that *"Bayesian statistics and Monte Carlo methods are particularly well-suited for the task of comparing multiple models on a single dataset"*. Furthermore, in that article, he presents the first description of the Approximate Bayesian Computation (ABC) algorithm. In the so called 'prehistoric' ABC, the goal was to explain to the frequentists the Bayesian concept of posterior distribution.

In this initial description of ABC, the aim is to find exact equivalence between the simulated and observed data. In the later first ABC version, which we will see, the distance between the summary statistics < epsilon is used.

The following is the "prehistoric" version of the ABC Algorithm.

Algorithm 1

```
for $i = 1$ to $N$ do
repeat
    (a) Generate $\theta'$ from the prior distribution $\pi()$
    (b) Generate $z$ from the likelihood $f(\ |\theta')$
until $\mathbf{z} = \mathbf{y}$        set $\theta_i = \theta'$
end for
```

It is straightforward to show that the outcome $\theta_1, \theta_2, ..., \theta_N$ resulting from this algorithm is an iid sample from the posterior distribution since

$$f(\theta_i) \propto \sum_{\mathbf{z}} \pi(\theta_i) f(\mathbf{z}|\theta_i) 1_{\mathbf{y}}(\mathbf{z}) = \pi(\theta_i) f(\mathbf{y}|\theta_i) \propto \pi(\theta_i|\mathbf{y})$$

Notice that the equality between z = y is incorrect in probabilistic terms because the probability at one point of a continuous function is 0 by definition.

2. The first ABC

Algorithm 2

```
for $i = 1$ to $N$ do
repeat
    (a) Generate $\theta'$ from the prior distribution $\pi()$
    (b) Generate $z$ from the likelihood $f(\ |\theta')$
until $\rho(\eta(\mathbf{z}), \eta(\mathbf{y})) \leq \varepsilon$        set $\theta_i = \theta'$
end for
```

where the parameters of the algorithm are:

- η, a function defining a statistic which most often is not sufficient;
- $\rho > 0$, a distance measure;
- ε, a tolerance level.

The algorithm steps can be summarized as follows:

1. Generate θ from the prior distribution $\pi()$;
2. Generate a set of simulated data $\mathbf{z}$ using the θ generated in 1;
3. Calculate the summary statistics of the simulated data;

4. Compute the discrepancy between the summary statistic of the observed data $\eta(y)$ and that of the simulated data $\eta(z)$. If the discrepancy is below a predefined threshold, accept θ.
5. Repeat.

The output of these algorithms is a set of thetas on which it is possible to subsequently infer also without being interested in calculating the posterior distribution.

3. MCMC - ABC

In practice, using simulations from the prior distribution $\pi(\cdot)$ is inefficient because this does not account for the data at the proposal stage and thus leads to proposed values located in low posterior probability regions. So, we introduce an MCMC-ABC algorithm targeting the approximate posterior distribution π_ε. The heart of the Monte Carlo methods in Bayesian terms is the ability to simulate from the posterior distribution if the latter is not available.

Algorithm 3

```
Use Algorithm 2 to get $(\theta^{(0)}, \mathbf{z^{(0)}})$ from the target $\pi_\epsilon(\theta, \mathbf{z}|\mathbf{y})$.
for $i = 1$ to $N$ do
    (a) Generate $\theta'$ from the Markov kernel $q(\ |\theta^{(t-1)})$
    (b) Generate $z$ from the likelihood $f(\ |\theta')$
    (c) Generate $u \sim Unif(0,1)$
if $u \leq \frac{\pi(\theta')q(\theta^{(t-1)}|\theta')}{\pi(\theta^{(t-1)})q(\theta'|\theta^{(t-1)})}$ and $\rho(\eta(\mathbf{z}'), \eta(\mathbf{y})) \leq \epsilon$ then set $(\theta^{(t)}, \mathbf{z^{(t)}}) = (\theta', \mathbf{z}')$
else $(\theta^{(t)}, \mathbf{z^{(t)}}) = (\theta^{(t-1)}, \mathbf{z^{(t-1)}})$
end if
end for
```

The acceptance probability used in the Algorithm above does not involve the calculation of the likelihood and it thus satisfies ABC requirements. It also produces an MCMC algorithm which exactly targets $\pi_\epsilon(\theta, \mathbf{z}|\mathbf{y})$ as its stationary distribution.

$$\frac{\pi_\epsilon(\theta', \mathbf{z}'|\mathbf{y})}{\pi_\epsilon(\theta^{(t-1)}, \mathbf{z^{t-1}}|\mathbf{y})} \times \frac{q(\theta^{(t-1)}|\theta')f(\mathbf{z^{t-1}}|\theta^{(t-1)})}{q(\theta'|\theta^{(t-1)})f(\mathbf{z}'|\theta')} =$$
$$= \frac{\pi(\theta')f(\mathbf{z}'|\theta')1_{A_{\varepsilon,y}}(\mathbf{z}')}{\pi(\theta^{(t-1)})f(\mathbf{z^{t-1}}|\theta^{(t-1)})1_{A_{\varepsilon,y}}(\mathbf{z^{t-1}})} \times \frac{q(\theta^{(t-1)}|\theta')f(\mathbf{z^{t-1}}|\theta^{(t-1)})}{q(\theta'|\theta^{(t-1)})f(\mathbf{z}'|\theta')} =$$
$$= \frac{\pi(\theta')q(\theta^{(t-1)}|\theta')}{\pi(\theta^{(t-1)})q(\theta'|\theta^{(t-1)})}1_{A_{\epsilon,y}}(\mathbf{z}')$$

The initialization of the MCMC sampler with the rejection sampler (Algorithm 2) can be bypassed since the Markov chain forgets its initial state. The computational cost of the initialization is then reduced. But then we have to run the MCMC longer to achieve convergence. As noted above, the ABC approximation depends on tuning parameters (the summary statistic η, the tolerance ε, and the distance ρ) that have to be chosen prior to running the algorithm and the calibration of which is discussed below. The tolerance ε is somewhat the easiest aspect of this calibration issue in that, when ε goes to zero, the ABC algorithm becomes exact.

4. Calibration of ABC

4.1. Summary statistics

Several authors have considered the fundamental difficulty associated with the choice of the summary statistic, $\eta(\mathbf{y})$, which one would like to consider as a quasi-sufficient statistic. First, for most real problems, it is impossible to find non-trivial sufficient statistics which would eliminate the need of a choice of statistics. Second, the summary statistics of interest are usually determined by the problem at hand and chosen by the experimenters in the field.

4.2. Tolerance threshold and ABC approximation error

The choice of the tolerance level ϵ is mostly a matter of computational power: smaller ε's are associated with higher computational costs and the standard practice is to select ε as a small percentile of the simulated distances $\rho\{\eta(\mathbf{z}), \eta(\mathbf{y})\}$.

Another way to choose the optimal value for ε is to use the tolerance as an additional parameter of the model, simulating from a joint posterior distribution, namely

$$f(\theta, \varepsilon|\mathbf{y}) \propto \xi(\varepsilon|\mathbf{y}, \theta)\pi_\theta(\theta)\pi_\varepsilon(\varepsilon)$$

where $\xi(\varepsilon|\mathbf{y}, \theta)$ plays the role of the likelihood, π_θ and π_ε are the corresponding priors on θ and ε.

5. Sequential improvements: ABC - SMC (Toni et al., 2009)

Sequential techniques can enhance the efficiency of the ABC algorithm by learning about the target distribution.

As in ABC rejection, we define a prior distribution $P(\theta)$ and we would like to approximate a posterior distribution $P(\theta|D_0)$.

In ABC - SMC we do this sequentially by constructing intermediate distributions, which converge to the posterior distribution. We define a tolerance schedule $\varepsilon_1 > \varepsilon_2 >$ ... $\varepsilon_R \geq 0$, where R is the number of rounds.

We sample from a prior distribution until N particles i.e. sampled parameters have been accepted (have reached the distance smaller than ε_1). We call the sample of all accepted particles "Population 1".

We then sample a particle θ^* (in our Example 2 case, we sample both the previously sampled mean and variance) from Population 1 and perturb it to obtain a perturbed particle $\theta^{**} \sim K(\theta|\theta^*)$, where K is a perturbation kernel (e.g. a Gaussian random walk).

In Example 2, in practice, we have (re)sampled each time a mean and a variance value (θ^*) from the sample of accepted means and variances obtained in round 1 (20% of the total) and then, after having simply perturbed them by adding an error measure, we have put those values (θ^{**}) in the rnorm() function creating a new temporary sample of the same size of the original one.

This new simulated dataset $D^* \sim f(D|\theta^{**})$ is used to accept the particle θ^{**} if $d(D_0, D^{**}) \leq \varepsilon_2$ (in Example 2 the Euclidean distance has been used).

We repeat this until we have accepted N particles in Population 2. and finally we repeat the same procedure for the following populations, until we have accepted N particles of the last Population R, where Population R is a sample of particles that approximates the posterior distribution.

ABC - SMC is computationally much more efficient than ABC rejection.

EXAMPLE 1. ABC algorithm

As we have discussed, the ABC main steps are to simulate data from prior distributions for the parameters of interest and then compare this simulated data to the observed data to determine how similar they are. Parameters values from the prior that produce data that are similar to the observed data can then be saved and they will form an approximated posterior distribution for the parameters.

In this paragraph we reproduce an example similar to the one by Paul Blisachk in his *ABC Inference for Normal mean and variance* paper, but instead of using a dataset containing 30 observations from a Normal distribution with unknown mean and variance, we have created a random vector of 30 observation from a Normal with mean = -2 and variance = 5. We will refer to this vector as our 'observed data' and to infer its distribution parameters, pretending not to know them, we will first use the standard ABC.

We start sorting the data and setting the number of simulations equal to 50000 and the number of parameters that will be saved to 100 (i.e. we will keep the parameters for the 100 simulated data sets that produce most similar data w.r.t. our observed data).

We have chosen to use uninformative priors (uniform distributions) for both the mean and variance. In particular $\mu \sim Unif(-20, 20)$ and $\sigma \sim Unif(0, 50)$ trying to be less informative as possible.

Once we have samples from the priors, we can then loop through the prior values, simulating each time data from a normal distribution with the drawn mean and variances.

We finally compare the simulated data to our observed dataset using the Euclidean distance.

```r
set.seed(123)
normal_start_time <- proc.time()

dat <- rnorm(30, mean = -2, sd = 5)
dat <- as.vector(dat)
dat <- sort(dat)

reps <- 50000
save <- 100
dist_vals <- rep(NA,reps)

prior_mean <- runif(reps, -20, 20)
prior_var <- runif(reps, 0, 50)

for(i in 1:reps){
  tmp_dat <- rnorm(length(dat), prior_mean[i], sqrt(prior_var[i]))
  tmp_dat <- sort(tmp_dat)
  dist_vals[i] <- dist(rbind(dat,tmp_dat))
}

indexes <- sort(dist_vals, index.return=T)
saved_indexes <- indexes$ix[1:save]

saved_means <- prior_mean[saved_indexes]
saved_variances <- prior_var[saved_indexes]

normal_total_time <- proc.time() - normal_start_time
```

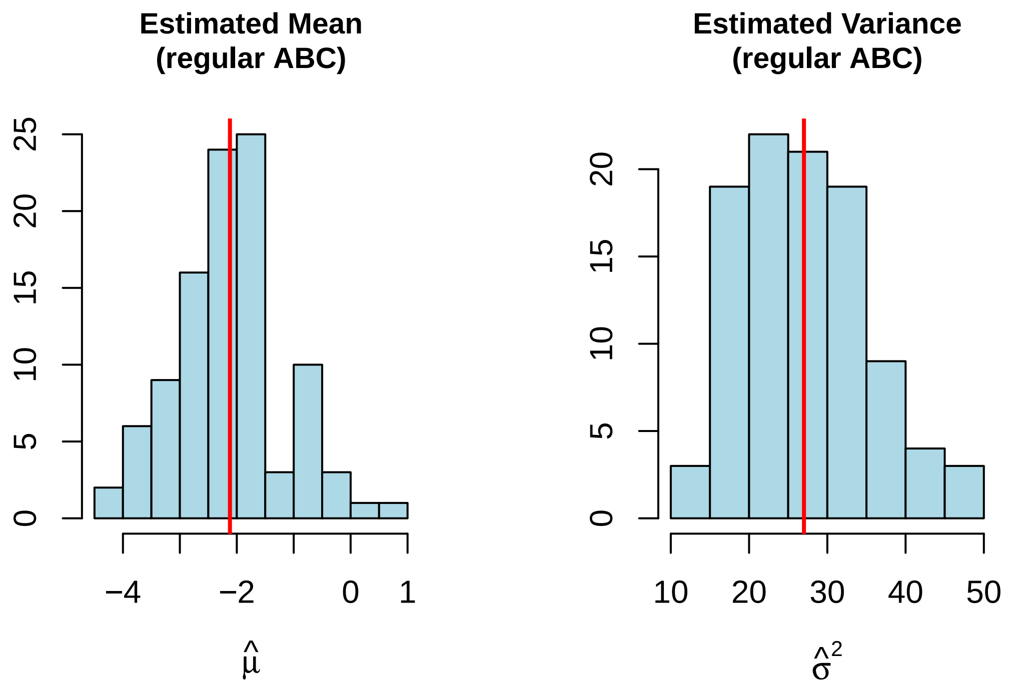
We can now summarize the resulting sample of 100 saved means and variances (where each of these can be seen as draw from the parameters' approximate posterior distributions) and finally plot the estimated approximate posterior distributions found.

```r
res <- data.frame(Parameter=c("Normal Mean","Normal Variance"),
                  Mean=c(mean(saved_means), mean(saved_variances)),
                  SD=c(sd(saved_means),sd(saved_variances)))
knitr::kable(res, digits=4, caption = "Regular ABC summary")
```

Table 1: Regular ABC summary

Parameter	Mean	SD
Normal Mean	-2.1205	0.9556
Normal Variance	27.0153	7.9612

```r
par(mfrow=c(1,2))
hist(saved_means, main="Estimated Mean\n(regular ABC)", xlab=expression(hat(mu)),
     ylab="", col = "lightblue", cex.main = 0.9)
abline(v=mean(saved_means),col="red",lwd=2)
hist(saved_variances, main="Estimated Variance\n(regular ABC)", xlab=expression(hat(sigma)^2),
     ylab="", col = "lightblue",  cex.main = 0.9)
abline(v=mean(saved_variances),col="red",lwd=2)
```



EXAMPLE 2. ABC with Sequential MC

We now do the same as above, but using a sequential Monte Carlo algorithm with 5 rounds of sampling.

This procedure guarantees a gain in power (i.e. in practice, a faster implementation) being equal the number of saved observation, but it gives us also the possibility of saving more observations from the prior, still remaining faster as the regular ABC.

In this second example, at the end of the 5-th round we will indeed have a total of 500 observations saved for both mean and variance (instead of the 100 resulting in exercise 1) and still taking circa 1/3 the time for the implementation w.r.t regular ABC.

We start setting 5 as the number of rounds and 10000 as the number of iteration for each round. We then set the fraction (namely, tau) of samples to save at each round. In particular, 20% of the data will be saved in the first round and 25% in all the others.

We need also to draw the initial values for mean and variance from the prior distributions (same as before).

```r
sequential_start_time <- proc.time()
rounds <- 5
N <- 10000
tau <- c(0.2,rep(0.25,rounds-1))
Q <- N*tau[1] # NB Q is the number of values to save from the first round

epsilon <- rep(NA,rounds)
dist_vals <- rep(NA,N)

prior_mean <- runif(N, -20, 20)
prior_var <- runif(N, 0, 50)

# 1st ROUND

for(i in 1:N){
  tmp_dat <- rnorm(length(dat), prior_mean[i], sqrt(prior_var[i]))
  tmp_dat <- sort(tmp_dat)
  dist_vals[i] <- dist(rbind(dat,tmp_dat))
}

dist_indexes <- sort(dist_vals, index.return=T)
save_indexes <- dist_indexes$ix[1:Q]
epsilon[1] <- dist_indexes$x[Q] # NB the first epsilon is the max distance value
saved_means <- prior_mean[save_indexes]
saved_variances <- prior_var[save_indexes]

# 2nd - 5th ROUNDS

curr_num_saved <- 0
dist_vals <- rep(NA,Q)
tmp_saved_means <- rep(NA,Q)
tmp_saved_variances <- rep(NA,Q)

for(r in 2:rounds){

  while(curr_num_saved < Q){
    curr_mean <- sample(saved_means, 1)
    curr_mean <- curr_mean + runif(1, -0.05, 0.05)
    curr_variance <- sample(saved_variances, 1)
    curr_variance <- curr_variance + runif(1, 0, 0.05)
    curr_dat <- rnorm(30, curr_mean, sqrt(curr_variance))
    curr_dat <- sort(curr_dat)
    curr_dist <- dist(rbind(dat, curr_dat))

    if(curr_dist < epsilon[r-1]){

      curr_num_saved <- curr_num_saved + 1
      dist_vals[curr_num_saved] <- curr_dist
      tmp_saved_means[curr_num_saved] <- curr_mean
      tmp_saved_variances[curr_num_saved] <- curr_variance

    }
  }

  dist_indexes <- sort(dist_vals, index.return = T)
  save_indexes <- dist_indexes$ix[1:(tau[r] * Q)]
  epsilon[r] <- dist_indexes$x[(tau[r] * Q)]
  saved_means <- tmp_saved_means[save_indexes]
  saved_variances <- tmp_saved_variances[save_indexes]

}

sequential_total_time <- proc.time() - sequential_start_time
```

We can now visualize the results summary as before and plotting the estimated approximate posterior distributions for both mean and variance.

```
res <- data.frame(Parameter=c("Normal Mean","Normal Variance"),
                  Mean=c(mean(saved_means), mean(saved_variances)),
                  SD=c(sd(saved_means),sd(saved_variances)))
knitr::kable(res, digits=4, caption = "SMC - ABC summary")
```

Table 2: SMC - ABC summary

Parameter	Mean	SD
Normal Mean	-2.2690	1.2062
Normal Variance	26.0615	9.7492

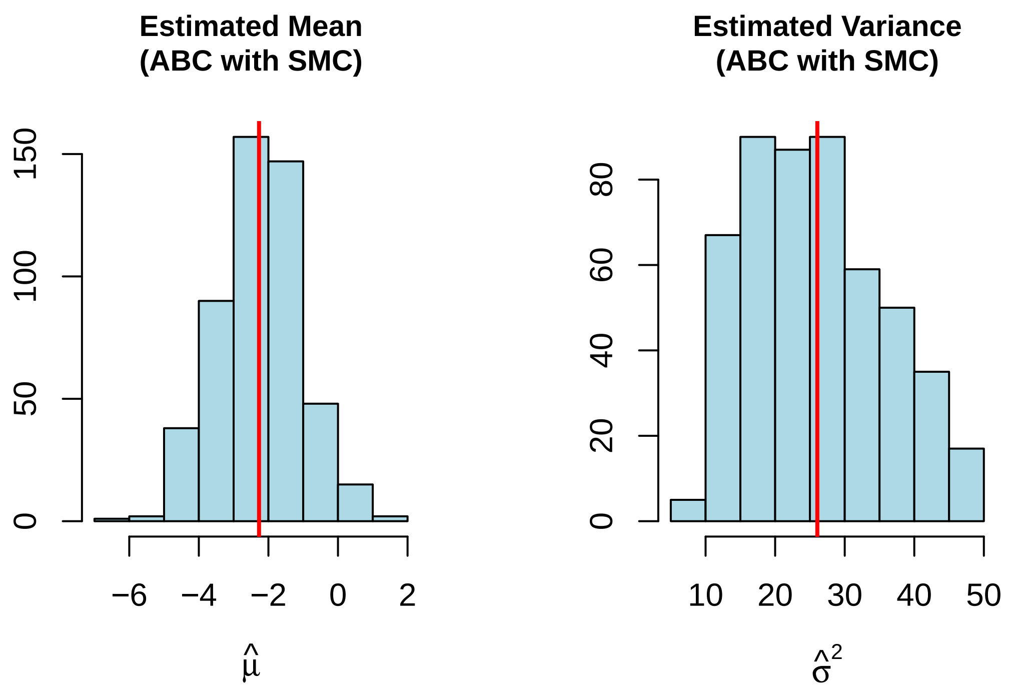
Comparing these last results with the ones obtained previously in exercise 1, we can see that the SMC - ABC algorithm does a slightly worse job in estimating the mean of the distribution, but it is more precise in the variance estimation, on average.

Also, the standard deviations associated to the parameters summaries get slightly widen in the SMC - ABC case.

However, in the plots below we can clearly see that the estimated approximate posterior distribution for the mean looks better w.r.t. the regular ABC case, despite the fact that in the SMC - ABC the point estimate for the mean is slightly bigger/farther from the truth.

Finally, the timing summary confirms the power gained by using the SMC - ABC w.r.t. the regular ABC, which we have proved to be much slower.

```
par(mfrow=c(1,2))
hist(saved_means, main="Estimated Mean\n(ABC with SMC)", xlab=expression(hat(mu)),
     ylab="", col = "lightblue",  cex.main = 0.9)
abline(v=mean(saved_means),col="red",lwd=2)
hist(saved_variances, main="Estimated Variance\n(ABC with SMC)",
     xlab=expression(hat(sigma)^2), ylab="", col = "lightblue",  cex.main = 0.9)
abline(v=mean(saved_variances),col="red",lwd=2)
```



```
timing <- data.frame(Algorithm=c("ABC","ABC with SMC"),
                     Time=c(normal_total_time[3],sequential_total_time[3]),
                     Improvement=c(normal_total_time[3]/normal_total_time[3],
                                   normal_total_time[3]/sequential_total_time[3]))
knitr::kable(timing, digits=4, caption = "Time improvement from regular ABC to SMC - ABC")
```

Table 3: Time improvement from regular ABC to SMC - ABC

Algorithm	Time	Improvement
ABC	5.33	1.0000
ABC with SMC	1.33	4.0075

6. Post-processing of ABC output: Local linear regression

Improvements to the general ABC scheme have been achieved by viewing the problem as a conditional density estimation and developing techniques to allow for larger ϵ (decreasing the computational costs). This is a post-processing scheme in which the simulation process per se does not change but the analysis of the ABC output does. All simulated summary statistics are indeed included, even those far away from the observed summary statistic, by shrinking the corresponding parameters in a linear manner. More specifically, the simulated θ's are replaced with $\theta^\star = \theta - \{\eta(\mathbf{z}) - \eta(\mathbf{y})\}^T\hat{\beta}$ where $\hat{\beta}$ is obtained by a weighted least squares regression of θ on $\{\eta(\mathbf{z}) - \eta(\mathbf{y})\}$.

7. ABC and model choice

Model choice is one particular aspect of Bayesian analysis that involves computational complexity, if only because several models are considered simultaneously. In addition to the parameters of each model, the inference considers the model index $\mathcal{M}$, which is associated with its own prior distribution $\pi(\mathcal{M} = m)(m = 1, ..., \mathcal{M})$ as well as a prior distribution on the parameters conditional on the value m of the model index, $\pi_m(\theta_m)$, defined on the parameter space Θ_m. The choice between these models is then driven by the posterior distribution of $\mathcal{M}$. Indeed, once $\mathcal{M}$ is incorporated within the parameters, the ABC approximation to the posterior follows from the same principles as regular ABC, as shown by the following pseudo code, where $\eta(\mathbf{z}) = (\eta_1(\mathbf{z}), ..., \eta_{\mathcal{M}}(\mathbf{z}))$ is the concatenation of the summary statistics used for all models (with elimination of duplicates).

Algorithm 4

```
for $i = 1$ to $N$ do
repeat
    (a) Generate $m$ from the prior $\pi(M = m)$
    (b) Generate $\theta_m$ from the prior $\pi_m(\theta_m)$
    (c) Generate $\mathbf{z}$ from the model $f_m(\mathbf{z}|\theta_m)$
until $\rho(\eta(\mathbf{z}), \eta(\mathbf{y})) \leq \varepsilon$        set $m^{(i)} = \theta_m$
end for
```

8. Discussion

Approximate Bayesian Computation allows inference from a large class of models which would otherwise be unavailable. In the present state of the field, its applicability is restricted to a narrow set of models for model selection. However, future advancements must simultaneously broaden the range of models for which ABC can be effectively employed, enhance pre- and post-processing techniques to better control the approximation, and provide a clearer understanding of the limitations of ABC in different scenarios. ABC methods are currently under an intense scrutiny by both statisticians and practitioners, hence the object of an unparalleled development. So, some issues remain unsolved, among which:

• the convergence results achieved so far are unpractical in that they require either the tolerance to go to zero or the sample size to go to infinity. Obtaining exact error bounds for positive tolerances and finite sample sizes would bring a strong improvement in both the implementation of the method and in the assessment of its worth;

• the construction and selection of the summary statistics is so far highly empirical. An automated approach based on the principles of data analysis and approximate sufficiency would be much more attractive and convincing, especially in non-standard and complex settings;

• the method necessarily faces limitations imposed by large datasets or complex models, in which simulating pseudo-data may itself become an impossible task. Dimension-reducing technique that would simulate directly the summary statistics will become necessary.

9. Sources

- https://arxiv.org/pdf/1101.0955.pdf
- https://github.com/pblischak/approximate_bayes
- *Tutorial on ABC rejection and ABC SMC for parameter estimation and model selection*, Tina Toni, Michael P. H. Stumpf, 2009
- https://www.research.unipd.it/retrieve/e14fb26f-fbf3-3de1-e053-1705fe0ac030/erlis_ruli_tesi.pdf
- https://youtu.be/IMMWegIISRg
- https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002803