Project

MCMC Methods

C. Leempoels - A. Ndiaye - E. Songo

Contents

Introduction

In the following report, we will be using various MCMC algorithms to explain and predict the haptoglobine gene.

The gene has 3 possible configurations, we have been given a table of incidence for each one from a sample of $n = 500$ individuals. The associated distribution is also known, and is parametrised by $\theta \in [0, 1]$ and $p \in [0, 1]$.

Since the gene frequencies follow the Hardy Weinberg Equilibrium, θ represents the frequency of the allele a. Furthermore, p is the probability that the allele types of the parents are identical by descent, *i.e.* an inbreeding coefficient.

We cannot get any information about whether or not the parents of the individuals are related just from x^{obs}, but we can infer it through the Bayesian paradigm.

Model

The haptoglobine has 3 different possible configurations AA, aa, aA. If the population is randomly mating, genes frequencies are following Hardy Weinberg Equilibrium. In this exercise, we assume that the environment favors certain genes associations (e.g., geographically structured population). We describe this using an inbreeding model parametrised by $\theta \in [0, 1]$ and $p \in [0, 1]$.

$$\mathbb{P}(X = AA) = p(1-\theta) + (1-p)(1-\theta)^2$$

$$\mathbb{P}(X = aa) = p\theta + (1-p)\theta^2$$

$$\mathbb{P}(X = aA) = 2(1-p)\theta(1-\theta)$$

p is referred to as the inbreeding coefficient and represents the probability that the allele types from parents are identical by descent.

We observe the following data.

```
x_obs = rep(c("AA",'aa', 'aA'), times = c(302, 125, 73))
```

Alleles configurations	Count
aa	125
aA	73
AA	302

1 EM algorithm

1.1 Write a latent variable representation of the model using a latent Bernoulli random variable of parameter p, denoted Z.

In this part, we represente numerically the differente allele configurations as follow:

- When the configuration is AA , x=0
- When the configuration is Aa , x=1
- When the configuration is aa , x=2.

The given allele probability distribution can be represented with a latent variable Z. The experiment can be described as following.

- Z is latent vector such that $(Z_i) \underset{iid}{\sim} \mathcal{B}er(p)$
- If $Z_i = 1$, the haptoglobine allele follows a Bernoulli distribution of parameter θ. In this case, the allele type from parents are identical by descent. The only two configurations possible are AA and aa.
- If $Z_i = 0$, the haptoglobine allele follows a binomial distribution of probability parameter θ and a number of trials equal to 2. The 3 configurations of the allele are possible.

Hence (Z_i, X_i) has the following density function:

$$p(z_i, x_i) = p\left[\theta^{\frac{x_i}{2}}(1-\theta)^{1-\frac{x_i}{2}}\right]^{z_i}\delta_{xi\neq 1} \times (1-p)\left[\binom{2}{x_i}\theta^{x_i}(1-\theta)^{2-x_i}\right]^{1-z_i}$$

1.2 Give the objective function of the EM algorithm associated to this latent representation and compute the updates for estimating p and θ.

We set n the number of observations. From the density function, we can derive the likelihood associated:

$$\mathcal{L}(p,\theta|x,Z) = \prod_{i=1}^{n} p(z_i,x_i) = \prod_{i=1}^{n} p\left[\theta^{\frac{x_i}{2}}(1-\theta)^{1-\frac{x_i}{2}}\right]^{z_i} \times (1-p)\left[\binom{2}{x_i}\theta^{x_i}(1-\theta)^{2-x_i}\right]^{1-z_i}$$

And the log-likelihood:

$$\log(\mathcal{L}(p,\theta|x,Z)) = \sum_{i=1}^{n} \log(p) + z_i\left[\frac{x_i}{2}\log(\theta) + (1-\frac{x_i}{2}) + \log(1-\theta)\right]$$
$$+ log(1-p) + (1-z_i)\left[\log(\binom{2}{x_i} + x_i\log(\theta) + (2-x_i)\log(1-\theta)\right]$$

The objective function of the EM algorithm associated to the latent representation of the question 1 is:

$$Q(\psi,\psi^{(t)}) = \mathbb{E}\left[\log(\mathcal{L}(p,\theta|x,Z))|\, x\right]$$

With $\psi = (p,\theta)$

As x is observed, we have:

$$Q(\psi,\psi^{(t)}) = \sum_{i=1}^{n} \log(p) + \mathbf{E}[z_i|x]\left[\frac{x_i}{2}\log(\theta) + (1-\frac{x_i}{2}) + \log(1-\theta)\right]$$
$$+ log(1-p) + (1-\mathbf{E}[z_i|x])\left[\log(\binom{2}{x_i} + x_i\log(\theta) + (2-x_i)\log(1-\theta)\right]$$

Let's compute $\mathbb{E}[z_i|x]$:

$$\begin{aligned}\mathbb{E}[z_i|x] &= \frac{\mathbb{P}(x_i|z_i=1)\mathbb{P}(z_i=1)}{\mathbb{P}(x_i|z_i=1)\mathbb{P}(z_i=1)+p(x_i|Z_i=0)\shortmid(z_i=0)} \\ &= \frac{\mathbb{P}(x_i|z_i=1)p}{\mathbb{P}(x_i|z_i=1)p+\mathbb{P}(x_i|Z_i=0)(1-p)}\end{aligned}$$

With:

$$\mathbb{P}(x_i|z_i=1) = \theta^{\frac{x_i}{2}}(1-\theta)^{1-\frac{x_i}{2}}$$
$$\mathbb{P}(x_i|z_i=0) = \binom{2}{x_i}\theta^{x_i}(1-\theta)^{1-x_i}$$

To simplify the calculus, we are going to use the updating formulas given in the practical session number 1 for mixture models.

As a recall, we proved that:

$$Q(\psi,\psi^{(t)}) = \sum_{i=1}^{n}\sum_{k=1}^{2}\tau_{ik}^{(t)}\log\left[p_k f(x_i|\theta_k)\right]$$

Where $\tau_{ik}^{(t)} = \frac{p_k^{(t)} f\left(x_i|\theta_k^{(t)}\right)}{\sum\limits_{l=1}^{2} p_l^{(t)} f\left(x_i|\theta_l^{(t)}\right)} = \mathbb{E}[z_i|x]$ The update for estimating the mixing probabilities is given by:

$$p = \frac{\sum\limits_{i=1}^{n}\tau_{i,1}^{(t)}}{n}$$

The update of theta is the solution of:

$$\sum_{i=1}^{n}\sum_{k=1}^{2}\tau_{ik}^{(t)}\frac{\partial}{\partial\theta}\log\left[f(x_i|\theta_k)\right] = 0$$

By solving this equation we end up with:

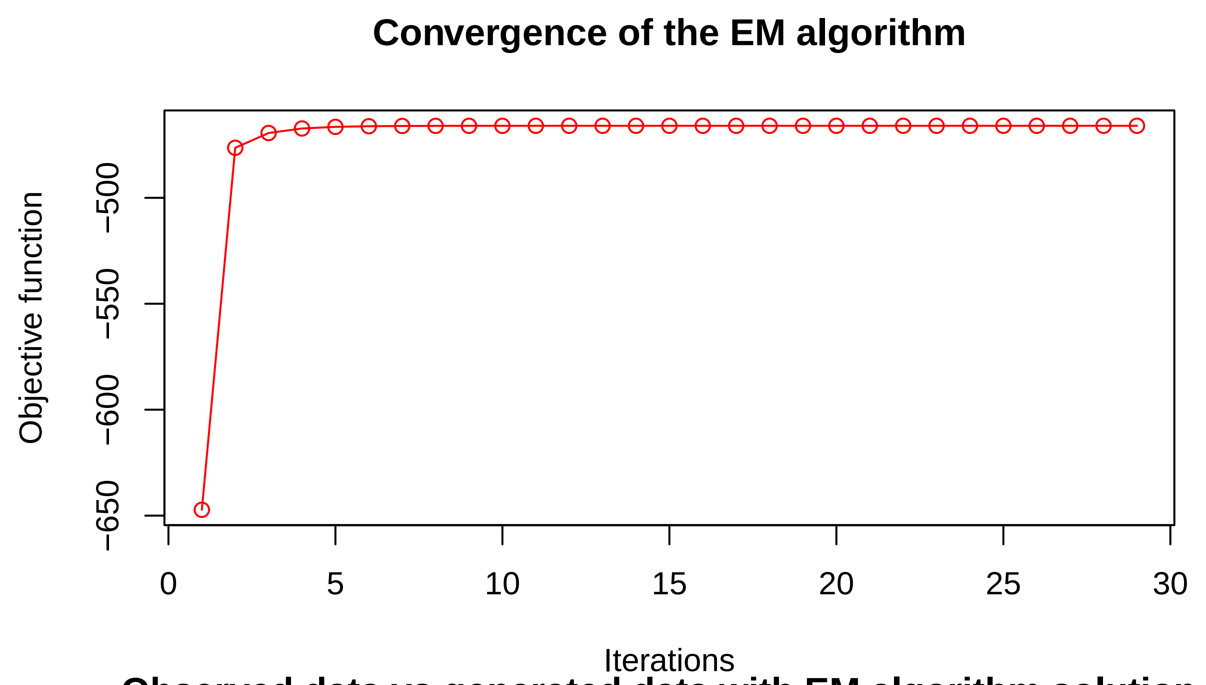
$$\theta = \frac{\sum\limits_{i=1}^{n}\tau_{i,1}\frac{x_i}{2} + \sum\limits_{i=1}^{n}\tau_{i,2}x_i}{\sum\limits_{i=1}^{n}\tau_{i,1} + 2\tau_{i,2}}$$

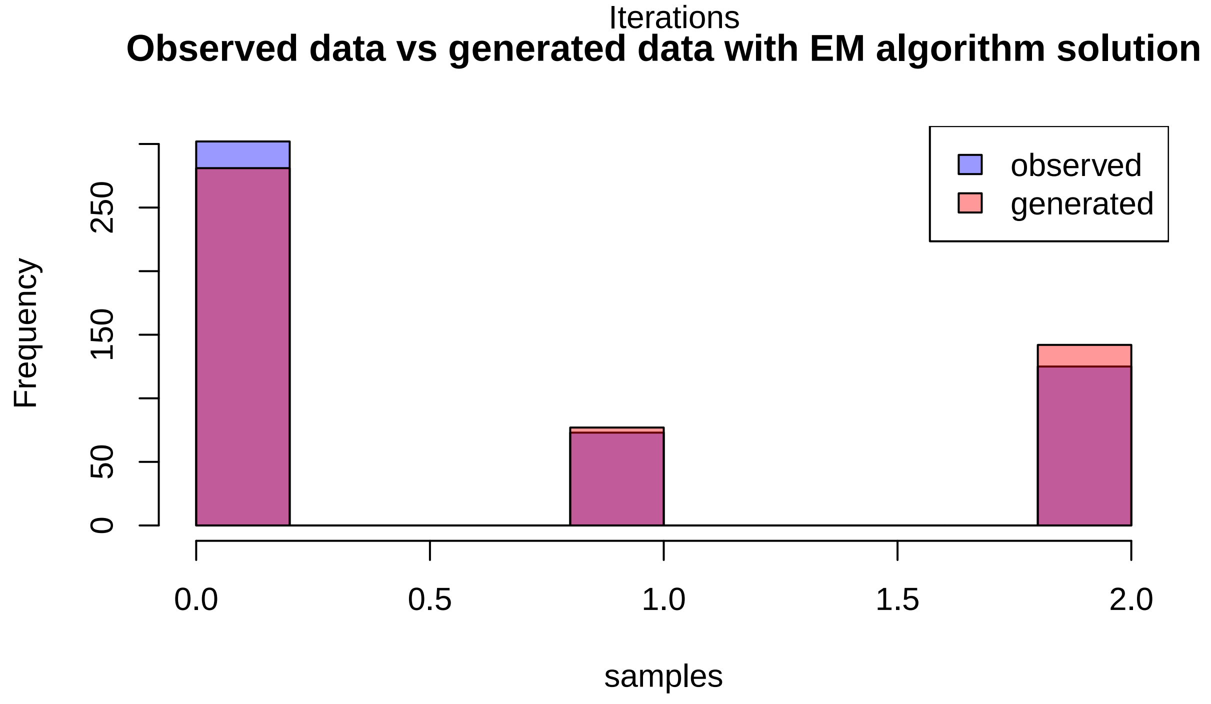
1.3 Implement the corresponding EM algorithm and run it on the data x^{obs} explaining your setting (e.g., initialization, stopping criterion). Check graphically the convergence of your algorithm and the fit between the estimated model and the empirical distribution of the data.

We randomly initialize p_0 and θ_0 in the EM algorithm and we use the objective function to check the convergence. When the objective function reach its maximum, it can't increase more. So the difference between the values of the objective function is almost the same from an iteration to the next one. We control this difference with the parameter eps which is set to 10^{-9} by default.

```r
algo_EM <- function(X, p_0 =runif(1), theta_0 = runif(1), max_iter = 10000,
                    eps = 10^{-9}) {
  n <- length(X)
  # We initialize the tau_i,k vectors
  tau <- matrix(data = rep(0, 2 * n), nrow = n, ncol = 2)
  p <- c(p_0, 1-p_0)
  theta <- theta_0
  objective <- numeric(max_iter) # We initialize a vector in which we will
  # save the value af the objective function  at each iteration
  for (iter in 1:max_iter) {
    tau_1 <- p[1] * (theta^(X / 2)) * ((1 - theta)^(1 - X / 2)) * (X != 1) # updating tau
    tau_2 <- p[2] * choose(2, X) * (theta^(X)) * ((1 - theta)^(2 - X))
    tau[, 1] <- tau_1 / (tau_1 + tau_2)
    tau[, 2] <- tau_2 / (tau_1 + tau_2)
    objective[iter] <- sum(log(tau_1 + tau_2)) # Calculating the objective function

    p <- colMeans(tau) # Updating p
    # Updating theta
    theta <- sum(tau[, 1] * X / 2 + tau[, 2] * X) / (sum(tau[, 1]) + 2 * sum(tau[, 2]))
    if ((iter > 2) && (abs(objective[iter] - objective[iter - 1]) < eps)) {
      print(paste("Number of iterations before convergence:", as.character(iter)))
      return(c(p, theta, objective[1:iter]))
    }
  }

  return(c(p, theta, objective))
}

x_obs_EM <- c(rep(0, 302), rep(2, 125), rep(1, 73))
Out_EM <- algo_EM(x_obs_EM)
```

```
## [1] "Number of iterations before convergence: 29"
```


Convergence of the EM algorithm
Objective function
−650
−600
−550
−500
0
5
10
15
20
25
30
Iterations


Observed data vs generated data with EM algorithm solution
Frequency
0
50
150
250
0.0
0.5
1.0
1.5
2.0
samples
observed
generated

2 Metropolis Hastings

2.1 Implement a Metropolis Hastings scheme using a Gaussian distribution as proposal density to sample from the posterior distribution of (p, θ).

Denote the observations by x^{obs} with sampling distribution $f(.|p, \theta)$ and log likelihood $\mathscr{L}(p, \theta|x^{obs})$.

```
obs_distribution = function(x, p, theta){
  AA = p*(1-theta) + (1-p)*(1-theta)^2
  aa = p*theta + (1-p)*theta^2
  aA = 2*(1-p)*theta*(1-theta)
  return( (x=="AA")*AA + (x=="aa")*aa + (x=="aA")*aA )
}

log_likelihood_obs = function(p, theta){
  error_out = -1000000
  out = tryCatch( sum(log(obs_distribution(x_obs, p, theta)) ),
    error = function(cond) return(error_out),
    warning = function(cond) return(error_out)
  )
  return(out)
}
```

Let $\pi(.)$ be the prior on (p, θ), with independent distribution on p and θ.

```
# prior distribution using a uniform distribution
log_prior_unif = function(p, theta)
  return(log(dunif(p) * dunif(theta)))

# prior distribution using a beta distribution
log_prior_beta = function(p, theta, a1, b1, a2, b2)
  return(log(dbeta(p, a1, b1) * dbeta(theta, a2, b2)))
```

The posterior distribution we are going to sample from with the Metropolis Hastings algorithm is :

$$\pi(p, \theta|x^{obs}) \propto \mathscr{L}(p, \theta|x^{obs})\pi(p, \theta)$$

```
mh = function(n, burn_in, init, log_lik_obs, log_prior, sd_proposal, ...) {
  iter = n + burn_in
  samples = matrix(rep(0,2*iter), ncol = 2)
  samples[1, ] = init
  h_current = do.call(log_lik_obs, as.list(init)) + do.call(log_prior, c(as.list(init),...))
  for (i in 2:iter) {
    prop = rnorm(2, samples[i - 1, ], sd_proposal)
    h_new = do.call(log_lik_obs, as.list(prop)) + do.call(log_prior, c(as.list(prop),...))
    if (log(runif(1)) < h_new - h_current) {
      samples[i, ] = prop
      h_current = h_new
    } else samples[i, ] = samples[i - 1, ]
  }
  samples = as.data.frame(samples[(burn_in + 1):iter, ])
  colnames(samples) = c("p","theta")
  rate = round(dim(unique(samples))[1]/dim(samples)[1], 4)
  return(list(samples = samples, rate = rate))
}
```

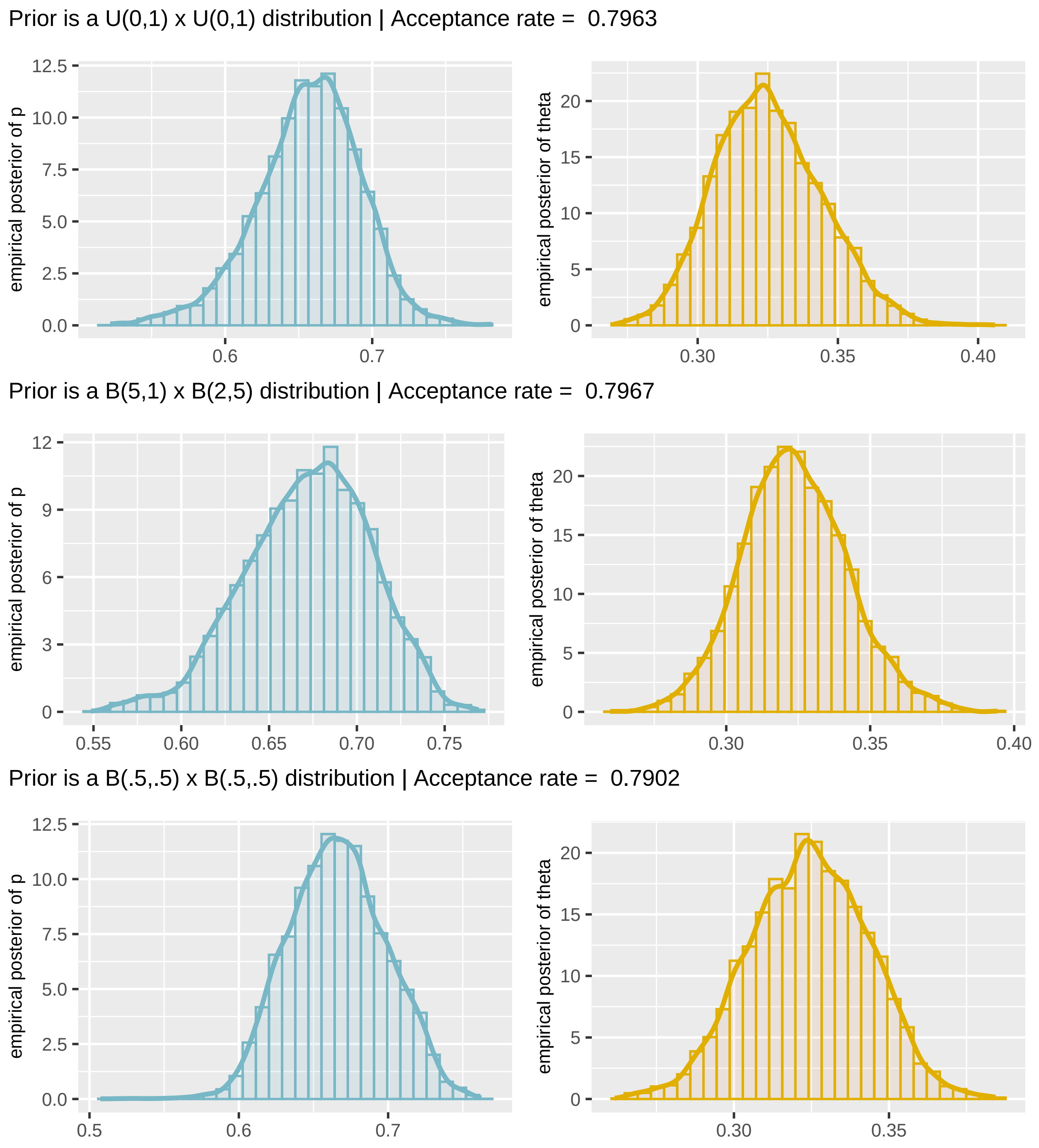


Fig. 2.1 : Sampling with the Metropolis Hastings algorithm ($\sigma_{proposal} = 0.01$)

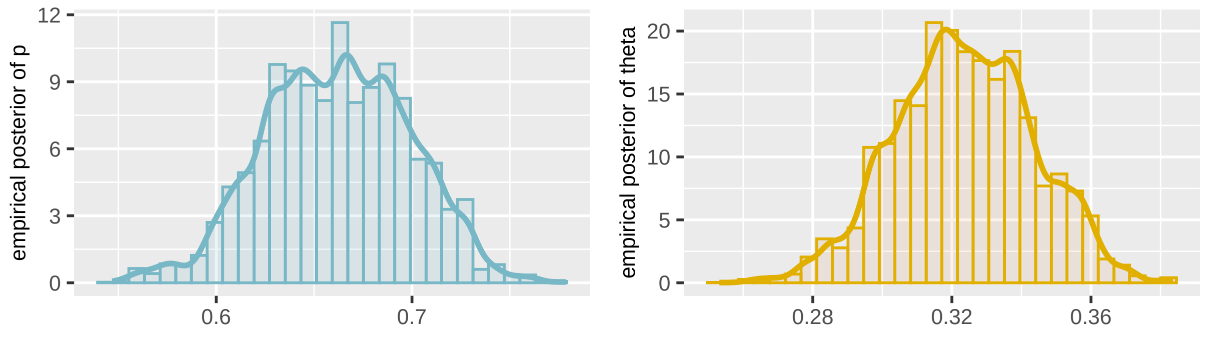
This scheme is not very sensitive to the choice of the prior or its parameters.

2.2 Test the sensitivity of the scheme to the variance of the Gaussian kernel. Discuss briefly your result.

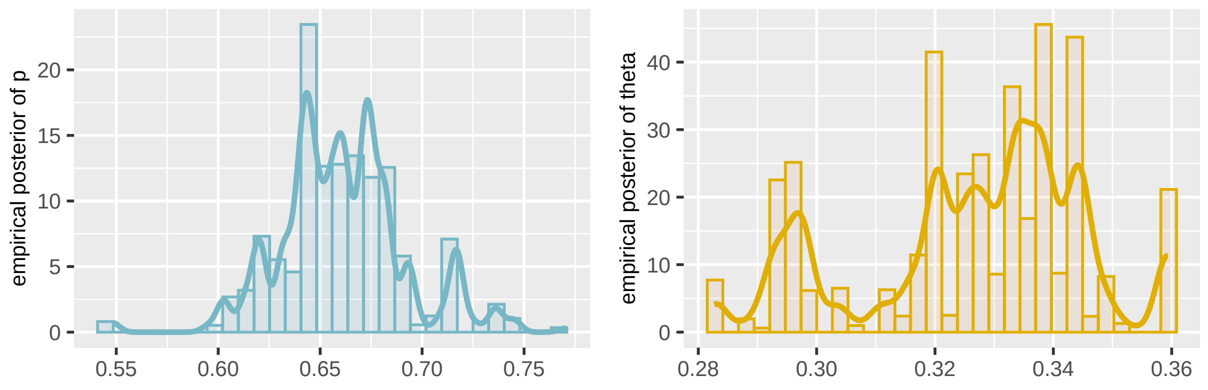
This scheme is very sensitive to the choice of the variance of the gaussian kernel. Moreover, the lower the variance, the higher the acceptance rate. This result is quite intuitive because if the variance is too large then the proposed states for p and θ will not belong to $[0, 1]$.

Fig. 2.2 : Sampling with the Metropolis Hastings algorithm (Uniform prior)

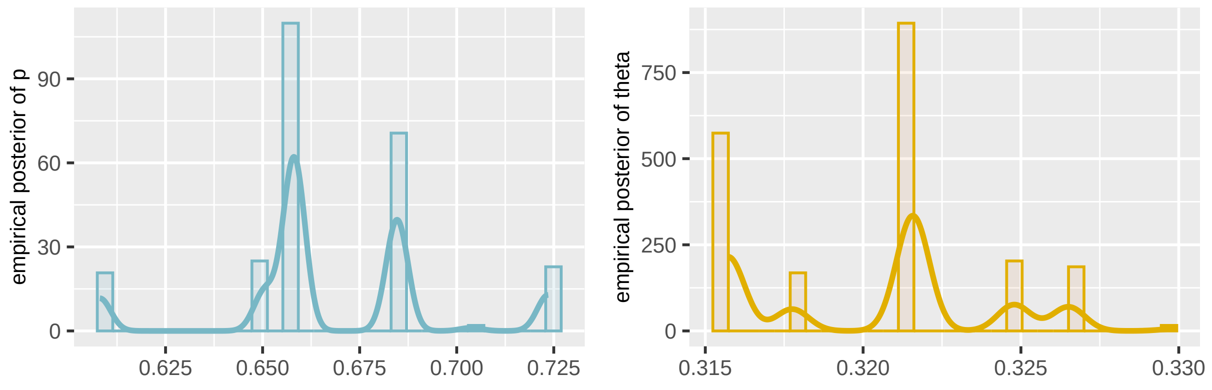
$\sigma_{\text{proposal}} = 0.1$ | Acceptance rate = 0.1096



$\sigma_{\text{proposal}} = 0.5$ | Acceptance rate = 0.0058



$\sigma_{\text{proposal}} = 1$ | Acceptance rate = 6e−04



3 Gibbs sampler

Our aim here is to use the Gibbs Sampler, an MCMC algorithm, to find the distribution of the unknown parameters (z, p, θ).

3.1 Compute the conditional distribution using the latent representation of the model

The likelihood of the latent representation of our model is given for all $z \in \{0,1\}^n$, $p \in [0,1]$ and $\theta \in [0,1]$ by :

$$\mathcal{L}(p,\theta|z,x^{\text{obs}}) = \prod_{i=1}^{n} \pi(x_i, z_i|p,\theta) = \prod_{i=1}^{n} p\left[\theta^{\frac{x_i}{2}}(1-\theta)^{1-\frac{x_i}{2}}\right]^{z_i} \times (1-p)\left[\binom{2}{x_i}\theta^{x_i}(1-\theta)^{2-x_i}\right]^{1-z_i}$$

3.1.1 Conditional distribution $\pi(p,\theta|z,x^{\mathbf{obs}})$

Recall that p and θ have independent prior distributions. Using Bayes' Theorem, we have :

$$\pi(p,\theta|z,x) = \frac{\mathcal{L}(p,\theta|z,x^{\text{obs}})\pi(p,\theta)}{\pi(x,z)} = \frac{\mathcal{L}(p,\theta|z,x^{\text{obs}})\pi(p)\pi(\theta)}{\int\int \pi(x,z|p',\theta')\pi(p')\pi(\theta')dp'd\theta'}$$

As a closed form for the denominator has proven difficult to compute, we estimate it numerically using a classic Monte Carlo method :

$$\hat{\pi}(x,z) = \frac{1}{K}\sum_{k=1}^{K}\pi(x,z|p_k,\theta_k), \quad (p_k,\theta_k)_{k\in[\![1,K]\!]} \overset{i.i.d}{\sim} \pi(p)\otimes\pi(\theta)$$

By definition of the Law of Large Numbers the estimator is strongly consistent. We can choose $\pi(p)$ and $\pi(\theta)$ to be either uniform or gamma distributions.

3.1.1.1 Uniform priors For distributions $\pi(p), \pi(\theta) \sim \mathcal{U}([0,1])$:

$$\pi(p,\theta|z,x) = \frac{\prod_{i=1}^{n}\pi(x_i,z_i|p,\theta)}{\pi(x,z)}$$

3.1.1.2 Beta priors For distributions $\pi(p) \sim Beta(\alpha_p, \beta_p)$ and $\pi(\theta) \sim Beta(\alpha_\theta, \beta_\theta)$:

$$\pi(p,\theta|z,x) = \frac{p^{n+1-\alpha_p}(1-p)^{\beta_p+n-1}\theta^{1-\alpha_\theta}(1-\theta)^{\beta_\theta-1}\prod_{i=1}^{n}\left[\theta^{\frac{x_i}{2}}(1-\theta)^{1-\frac{x_i}{2}}\right]^{z_i}\left[\binom{2}{x_i}\theta^{x_i}(1-\theta)^{2-x_i}\right]^{1-z_i}}{\pi(x,z)\mathbf{B}(\alpha_p,\beta_p)\mathbf{B}(\alpha_\theta,\beta_\theta)}$$

3.1.2 Conditional distribution $\pi(z|p,\theta,x^{\mathbf{obs}})$

Again, with Bayes' Theorem :

$$\pi(z|p,\theta,x) = \frac{\pi(p,\theta|z,x^{\text{obs}})\pi(z,x^{\text{obs}})}{\pi(p,\theta,x^{\text{obs}})} = \frac{\pi(p,\theta|z,x^{\text{obs}})\pi(z,x^{\text{obs}})}{\pi(x^{\text{obs}}|p,\theta)\pi(p)\pi(\theta)}$$

Here $\pi(z,x^{\text{obs}})$ is the marginal distribution and $\pi(x^{\text{obs}}|p,\theta)$ the original distribution of the model evaluated in the observation x^{obs}.

3.2 **Implement a Gibbs sampler to sample from the posterior distribution of (z, p, θ).**

3.3 **Compare the empirical posterior distribution of (p, θ), the marginal distributions of p and θ as well as their posterior means obtained with Metropolis Hastings scheme and Gibbs sampler. Which algorithm do you recommend to use?**

4 ABC algorithm

4.1 Implement a vanilla ABC algorithm (using a L2-standardised distance) to sample from the posterior distribution of (p, θ). Justify the choice of the summary statistics and study the influence of the size of the neighborhood around x^{obs} on the posterior approximation.

Denote the observations by x^{obs} with sampling distribution $f(.|p, \theta)$.

```
# Simulation from the sampling distribution, knowing the parameters
sim_obs = function(n_sim, params){
  p = params[1]
  theta = params[2]
  AA = p*(1-theta) + (1-p)*(1-theta)^2
  aa = p*theta + (1-p)*theta^2
  aA = 2*(1-p)*theta*(1-theta)
  return(sample(c("AA","aa","aA"), n_sim, replace = T, prob = c(AA, aa, aA)))
}
```

Let $\pi(.)$ be the prior on (p, θ), with independent distribution on p and θ.

```
# Simulation from the prior (uniform case)
sim_prior_unif = function()
  return(runif(2))

# Simulation from the prior (beta case)
sim_prior_beta = function(a1, b1, a2, b2)
  return(c(rbeta(1, a1, b1), rbeta(1, a2, b2)))
```

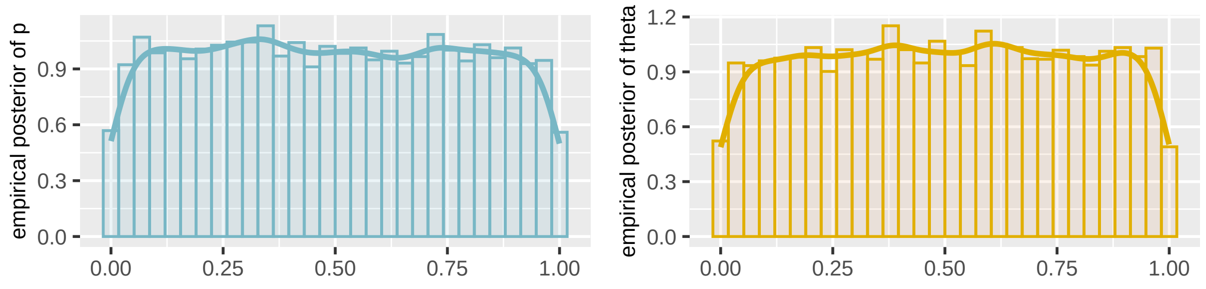
Let S denote the summary statistics we will use in the ABC algorithm. Since our data is not quantitative we will not be able to compute moments but we can use a frequency table to compare the distribution of the observed data and the generated data.

```
S = function(x)
  return(c(sum(x == "AA")/length(x), sum(x == "aa")/length(x), sum(x == "aA")/length(x)))
```

```
abc = function(n_sim, x_obs, sim_obs, sim_prior, S, neighborhood_size, max_iter = 1e8, ...){
  accepted = 0
  refused = 0
  n_obs = length(x_obs)
  samples = matrix(rep(0,2*n_sim), ncol = 2)
  for(i in 1:max_iter){
    params = sim_prior(...)
    x = sim_obs(n_obs, params)
    if (sum((S(x) - S(x_obs))^2) < neighborhood_size){
      accepted = accepted + 1
      samples[accepted, ] = params
    } else refused = refused + 1
    if (accepted == n_sim) break
  }
  samples = as.data.frame(samples)
  colnames(samples) = c("p","theta")
  return(list(samples = samples, rate = round(accepted / (accepted + refused), 4)))
}
```

Fig. 4.1 : Sampling from posterior with the ABC algorithm (U(0,1) x U(0,1) prior)

Size of the neighborhood = 1.0 | Acceptance rate = 1



Size of the neighborhood = 0.001 | Acceptance rate = 0.004

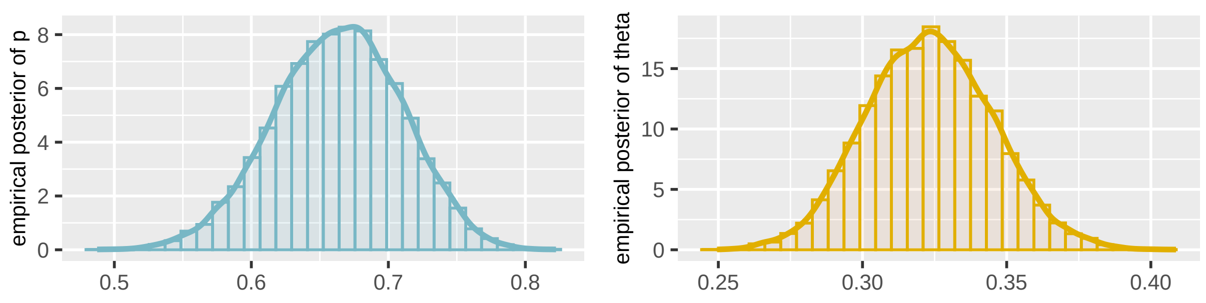
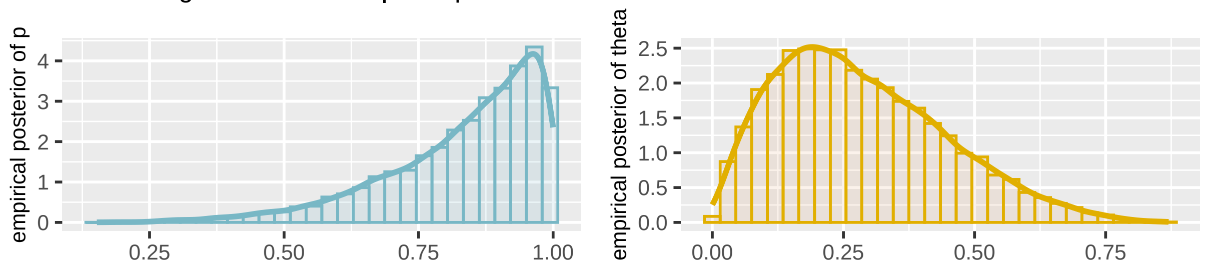
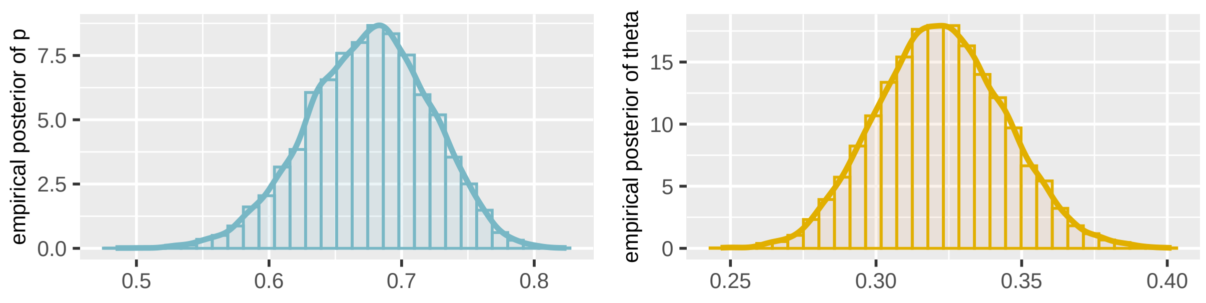


Fig. 4.2 : Sampling from posterior with the ABC algorithm (B(5,1) x B(2,5) prior)

Size of the neighborhood = 1.0 | Acceptance rate = 1



Size of the neighborhood = 0.001 | Acceptance rate = 0.0083



When the neighborhood around x^{obs} is too large, we can see that we sample from the prior distribution instead of the posterior distribution. When the neighborhood is small enough, the mean and standard deviation seem invariant to the choice of the prior.

We are now going to study the impact of the choice of the prior distribution and its parameters on the acceptance rate. We will use as reference the empirical posterior distribution obtained when $\mathscr{U}[0,1] \times \mathscr{U}[0,1]$ is used as prior.

Table 2: Kullback-Leibler distance between the Prior and Reference Posterior distributions of p

Prior of p	Distance to Reference Posterior	ABC acceptance rate	Empirical mean of p
U[0,1]	1.7278	0.004	0.6633
B(5,1)	2.1788	0.0083	0.6752
B(2,5)	2.8695	1e-04	0.6459

Table 3: Kullback-Leibler distance between the Prior and Reference Posterior distributions of theta

Prior of theta	Distance to Reference Posterior	ABC acceptance rate	Empirical mean of theta
U[0,1]	1.752	0.004	0.3233
B(5,1)	2.445	1e-04	0.3282
B(2,5)	1.6754	0.0083	0.3223

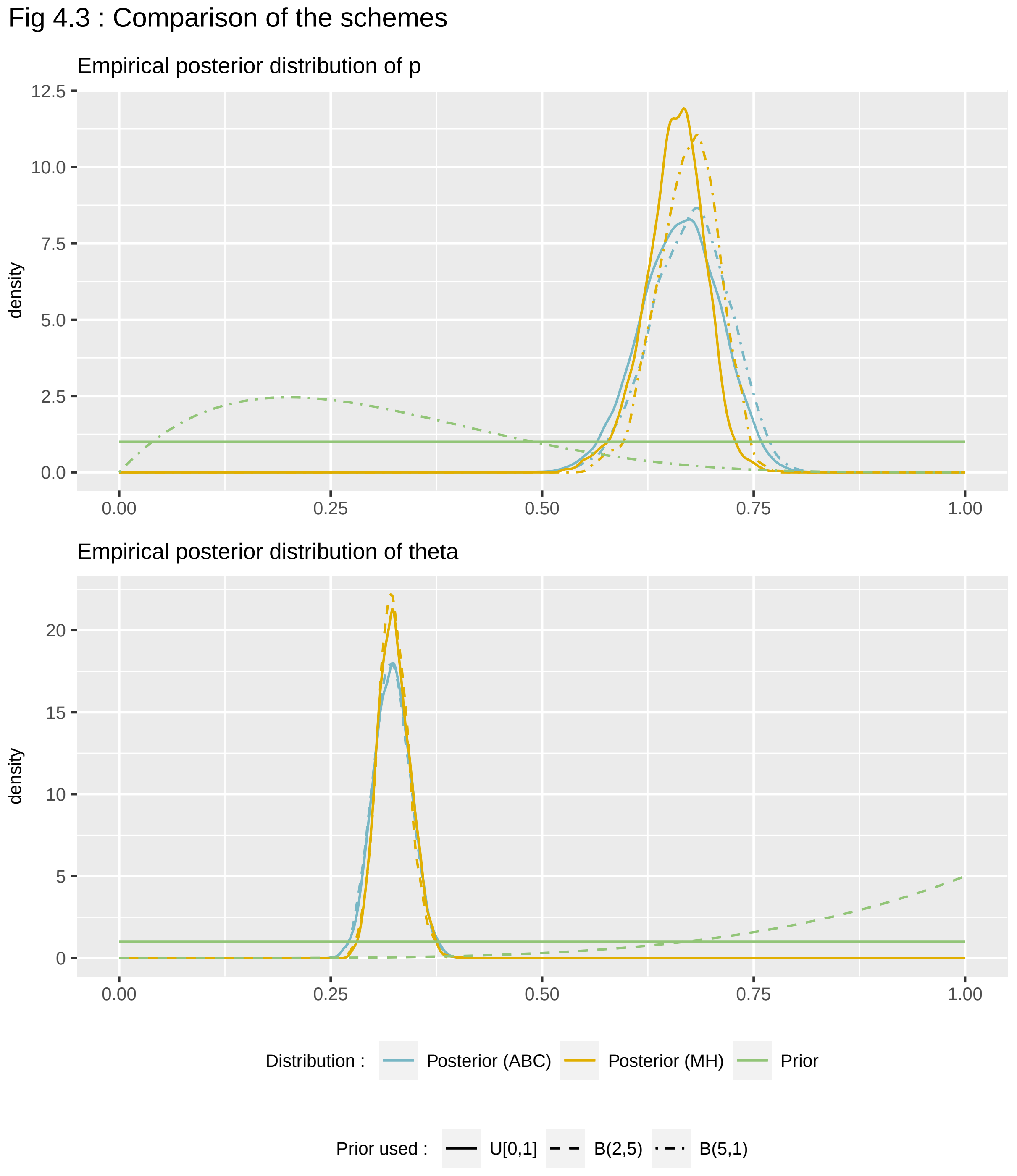
In the ABC algorithm, the choice of the prior (or its parameters) has a huge impact on the acceptance rate and consequently on the simulation cost. It seems better to use a uniform prior than to risk having a prior with a distribution tail where the posterior has most of its mass.

4.2 Compare the empirical posterior distribution of (p, θ), the marginal distributions of p and θ as well as their posterior means to the ones obtained with the previous MCMC schemes.

In order to compare the different schemes, we will use for each of them the best parameters found in the previous sections.

Table 4: Comparison of the schemes

	Empirical mean of p	Empirical mean of theta
MH (U[0,1] x U[0,1] prior)	0.6568332	0.3253200
MH (B(5,1) x B(2,5) prior)	0.6723040	0.3239130
ABC (U[0,1] x U[0,1] prior)	0.6632757	0.3232832
ABC (B(5,1) x B(2,5) prior)	0.6751527	0.3222774



4.3 Hardy Weinberg Equilibrium (model $m = 1$) is embedded in the inbreeding model we used (model $m = 2$). We would like to study if it was reasonable to use a more complex model for our data. Using an ABC routine, compute estimates of the posterior probabilities of models 1 and 2, i.e. $\pi(m|x^{obs})$, under the assumption of a uniform prior on model index m. What is your conclusion?

We sample m with the posterio distribution using the ABC routine. We choose an acceptance rate of 0.05. We end up with a probability close to 1 in favor of the second model. It was reasonable to use a more complex model.

```r
ABC_model_choice<-function(sample_size,obs){
  n<-length(obs)
  m_sample<-numeric(sample_size)
  theta_sample<-numeric(sample_size)
  p_sample<-numeric(sample_size)
  for ( t in 1:sample_size){
    m<-rbinom(1,1,prob=0.5) #we choose a model
    p<-runif(1) # we choose a uniform prior for p
    theta<-runif(1) #uniform prior for theta
    z<-rbinom(n,1,prob=p) #generate z
    x<-(m==0)*rbinom(n,2,theta)+(m==1)*(z*2*rbinom(n,1,theta)+(1-z)*rbinom(1,2,theta)) #generate
    distance<-((length(x[x==1])/n-length(obs[obs==1])/length(obs))^2+(length(x[x==0])/n-length(obs[obs=

    while(distance>0.05){
      m<-rbinom(1,1,prob=0.5) #we choose a model
      p<-runif(1) # we choose a uniform prior for p
      theta<-runif(1) #uniform prior for theta
      z<-rbinom(n,1,prob=p) #generate z
      x<-(m==0)*rbinom(n,2,theta)+(m==1)*(z*2*rbinom(n,1,theta)+(1-z)*rbinom(1,2,theta)) #generate
      distance<-((length(x[x==1])/n-length(obs[obs==1])/n)^2+(length(x[x==0])/n-length(obs[obs==0])/n)^
    }
    m_sample[t]<-m
    theta_sample[t]<-theta
    p_sample[t]<-p

  }
  return (data.frame(m=m_sample,theta=theta_sample,p=p_sample))

}

resultat<-ABC_model_choice(10000,x_obs_EM)

mean(resultat$m)
```

```
## [1] 0.9999
```