MCMC application

Anton Cronet

The data was provided by Uppsala University as an example dataset to practice Bayesian Statistics.

Based on data showing which genes are present in 100 randomly selected individuals, and the theoretical frequencies these should appear, the goal is to estimate the "inbreeding factor". This factor is part of the theoretical equations determining the frequencies of the possible genotypes.

The individual log likelihoods of the AA, Aa, and aa genotypes are given by:

$$nAA * log(f * p + (1 - f) * p^2)$$

$$nAa * log((1 - f) * 2 * p * (1 - p))$$

$$naa * log(f * (1 - p) + (1 - f) * (1 - p)^2)$$

Hence the log likelihood of p and f given the observed data is:

$$loglik < -nAA*log(f*p+(1-f)*p^2)+nAa*log((1-f)*2*p*(1-p))+naa*log(f*(1-p)+(1-f)*(1-p)^2)$$

```r
genes <- read.csv("gener.csv")

nAA <- sum(genes == "AA")
nAa <- sum(genes == "Aa")
naa <- sum(genes == "aa")

loglikelihood <- function(params, data) {
  p <- params[1]
  f <- params[2]

  if (p >= 0 & p <= 1 & f >= 0 & f <= 1) {
    lik_AA <- nAA * log(f * p + (1 - f) * p^2)
    lik_Aa <- nAa * log((1 - f) * 2 * p * (1 - p))
    lik_aa <- naa * log(f * (1 - p) + (1 - f) * (1 - p)^2)

    return(lik_AA + lik_Aa + lik_aa)
  } else {
    return(-Inf)
  }
}
```

The prior distributions for p and f both are beta distributions since these are limited between 0 and 1

```r
logprior <- function(params) {
  p <- params[1]
  f <- params[2]

  prior <- dbeta(p, 2, 2, log = TRUE) + dbeta(f, 2, 2, log = TRUE)
  return(prior)
}
```

The density of the posterior distribution is computed by multiplying the prior with the likelihood, since we are working with logs, these are added together:

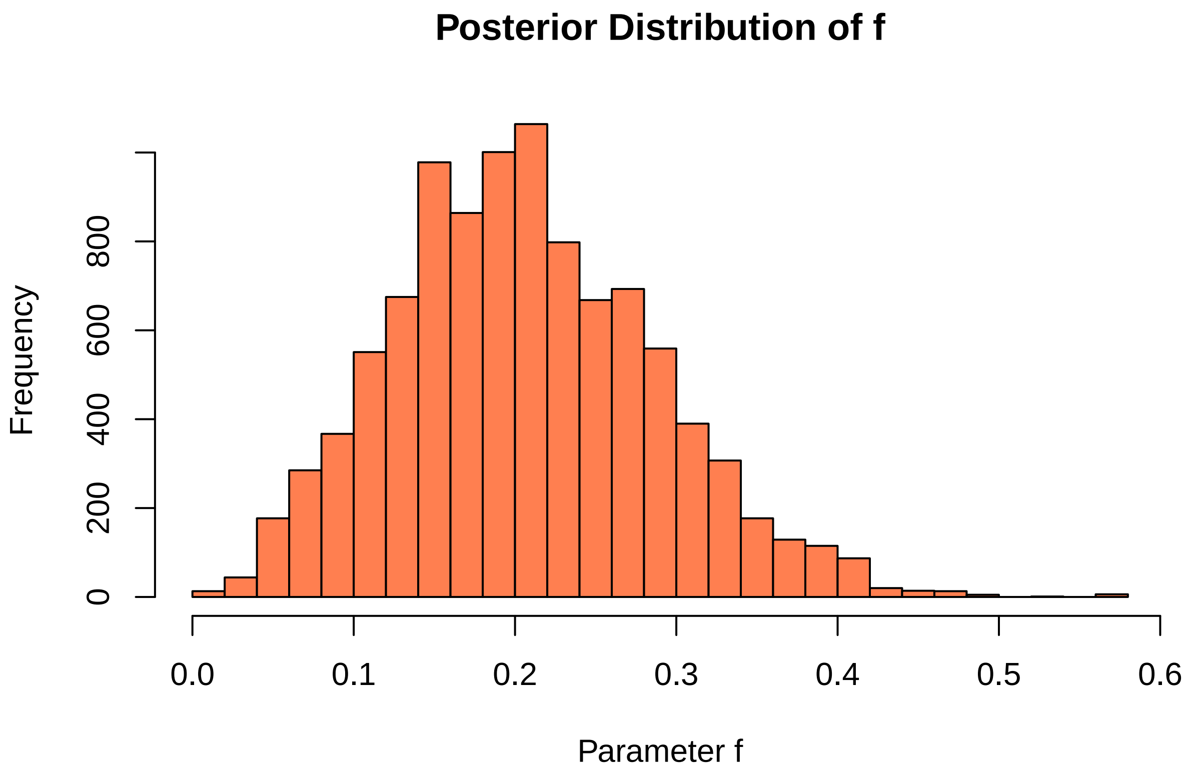
```
logposterior <- function(params, data) {
  loglik <- loglikelihood(params, data)
  prior <- logprior(params)

  return(loglik + prior)
}
```

First MCMC is run for 1000 iterations with the initial parameters 0.5 and 0.5, and then use the results as better initial guesses to run 10000 iterations. The initial burn-in period leads to the new initial values being 0.496 and 0.199. The second run of the MCMC algorithm saves all the results of p and f 's distributions respectively.

```
source("mcmciter.R")
# Set initial values for parameters
initial_params <- c(0.5, 0.5)
sigma <- 0.1
n.iter <- 1000

result_first_run <- mcmc.iter(initial_params, logposterior, sigma, n.iter)

new_params <- tail(result_first_run$sample, 1)
n.iter <- 10000

result <- mcmc.iter(new_params, logposterior, sigma, n.iter)
```

Parameter f, representing the inbreeding factor has the following histogram:

```
hist(result$sample[, 2], breaks = 30, col = "coral", main = paste("Posterior Distribution of f"), xlab=
```



```
# Extract the posterior samples for parameter f
posterior_f <- result$sample[, 2]


# Compute the 95% credible interval
credible_interval <- quantile(posterior_f, c(0.025, 0.5, 0.975))

cat("Point estimate for f:", credible_interval[2], "(50% quantile)")
```

```
## Point estimate for f: 0.2008129 (50% quantile)
```

```
cat("95%  Interval for f: [", credible_interval[1], ",", credible_interval[3], "]\n")
```

```
## 95%  Interval for f: [ 0.06106699 , 0.3824192 ]
```