Module 11: Lesson 4 Lab

Callum Arnold

21 July, 2021

Contents

Background

In Lecture 4 we were concerned with models for disease transmission which incorporate households. Although we looked at two distinct cases with regards to what sort of data are available, in this lab session we will focus on the case where there is only *final outcome* data available. In other words, the available data consist only of the final number of cases in each household.

The main objective of this lab session is to draw Bayesian inference for the model parameters by implementing a Markov Chain Monte Carlo algorithm. Below, there is a brief description of the model (see the lecture notes for more details).

The model, the data and the likelihood

We assume that a population of N individuals is partitioned into households. Each individual in the population has, independently, a constant "risk'' per unit time of becoming infected from the community. In

addition, within a household, the disease spreads according to the mechanism of an SIR model.

Suppose that the data consist of the set of numbers $n = \{n(j,k)\}$ where $n(j,k)$ = number of households in which j out of k initial susceptibles become infected. Therefore, the likelihood takes the form

$$\pi(n|\alpha, p) = \prod_j \prod_k q(j,k)^{n(j,k)}$$

where $q(j,k)$ is the probability that in a household of size k the number of individuals who ever become infected is j.

In the lecture we have derived an expression for the probability $q(j,k)$ and showed that it depends on two parameters:

- p which is the probability that an individual avoids infection from outside the household and
- α which is the person-to-person infection rate between individuals within the same household.

Preliminaries

Download the file `households.R` from the module's website

The file contains two functions:

- `compute.conditional.prob` which computes the probability that j susceptibles become infected in an SIR model with constant infectious period of length c, a initial infectives and m initial susceptibles.

```r
# this is a function to compute the final size P(T=k|Y=y) = probability
# that k-y susceptibles become infected with y initial infectives
# and n-y susceptibles

# m initial susceptibles
# a initial infectives
# alpha infection rate
# the length of the fixed infectious period
compute.conditional.prob <- function(m, j, a, alpha, c) {
  if (j == 0) {
    res <- (exp(-alpha * m * c)) ^ a
  }
  else {
    part.one <- exp(-alpha * (m - j) * c) ^ (j + a)
    part.two <- choose(m, j)

    sum <- 0

    for (k in 0:(j - 1)) {
      sum <-
        sum + choose(m - k, j - k) * compute.conditional.prob(m, k, a, alpha, c) /
        (exp(-alpha * (m - j) * c) ^ (k + a))
    }
    res <- part.one * (part.two - sum)
  }
  return(res)
}
```

- `compute.marginal.probab` which computes the probability that in a household of size n the number of individuals who ever become infected is k.

```
# We wish to compute P(T = k), for k = 0, ..., n
# "n" is the size of the household
# "alpha" is the person to person infection rate
# "c" is the length of the (fixed) infectious period
compute.marginal.prob <- function(k, n, c, alpha, p) {
  prob <- 0

  for (y in 0:n) {
    if (k >= y) {
      cond.prob <- compute.conditional.prob(n - y, k - y, y, alpha, c)
    }
    else {
      cond.prob <- 0
    }
    prob <- prob + cond.prob * dbinom(y, n, 1 - p)
  }
  prob
}
```

In addition, Table 1 presents some Asian influenza epidemic household data taken from Longini & Koopman (1982). There is a community which consists of 42 households each of them of size 3.

```
cases <- c(0, 1, 2, 3)
households <- c(29, 9, 2, 2)
mat <- matrix(NA, nrow = 4, 2)
mat[, 1] <- cases
mat[, 2] <- households
mat <- as.data.frame(mat)
colnames(mat) <- c("No.of.Cases", "No.of.Households")
mat
```

```
##   No.of.Cases No.of.Households
## 1           0               29
## 2           1                9
## 3           2                2
## 4           3                2
```

```
##   No.of.Cases No.of.Households
## 1           0               29
## 2           1                9
## 3           2                2
## 4           3                2
```

Exercises

Exercise 1

Go through each of the functions in the file `households.R` and make sure that you understand what is going on.

Exercise 2

Question

Write a function in R which computes the log-likelihood of the data given the parameters p and α assuming that the infectious period is constant and of length one unit.

Answer

$$\pi(n|\alpha, p) = \prod_j \prod_k q(j,k)^{n(j,k)}$$

$$q(j,k) = P(T = j)$$

```
log_likelihood <- function(alpha, p){
  ll <- 29 * log(compute.marginal.prob(
    k = 0, n = 3, c = 1, alpha = alpha, p = p
    )) +
    9 * log(compute.marginal.prob(
    k = 1, n = 3, c = 1, alpha = alpha, p = p
    )) +
    2 * log(compute.marginal.prob(
    k = 2, n = 3, c = 1, alpha = alpha, p = p
    )) +
    2 * log(compute.marginal.prob(
    k = 3, n = 3, c = 1, alpha = alpha, p = p
    ))

  return(ll)
}
```

Exercise 3

Question

Suppose that we are interested in drawing Bayesian inference for the model parameters and therefore, we could employ an MCMC algorithm to draw samples from the posterior distribution:

$$\pi(p, \alpha|n) \propto \pi(n|\alpha, p) \times \pi(p) \times \pi(\alpha)$$

Assume that *a priori* $p \sim \text{Beta}(\mu_p, \nu_p)$ and $\alpha \sim \text{Gamma}(\mu_\alpha, \nu_\alpha)$.

What values should we choose for the hyper-parameters μ_p, ν_p, μ_α and ν_α such that we end up with fairly uninformative priors for the model parameters p and α?

Answer

Setting $\mu_\alpha = 1$ results in an exponential distribution with rate ν_α. If we pick a low value for it's rate, it will be relatively flat over its initial values, so it will be fairly uninformative

$$\mu_p = 1$$
$$\nu_p = 1$$
$$\mu_\alpha = 1$$
$$\nu_\alpha = 0.001$$

Exercise 4

Question

Write a function in R which draws samples from $\pi(p, \alpha|n)$ using MCMC and by adopting uninformative priors for the parameters of interest p and α.

Hint: In the lecture we discussed two different ways to update the parameter p. Either using an independence sampler or a random walk Metropolis. For the purposes of this exercise, update p by an independence sampler.

Answer

```
mcmc.ind.sampler <- function(iter, sigma) {

  # values for the prior hyper-parameters
  mu.p <- 1.0
  nu.p <- 1.0

  mu.alpha <- 1.0
  nu.alpha <- 10^(-3)


  # initial values for the Markov Chain
  alpha.cur <- 0.5;
  p.cur <- 0.8;

  # create a matrix to store the output.
  res <- matrix(NA, nrow = iter, ncol=2);
  res[1,] <- c(alpha.cur, p.cur);


  for (i in 2:iter) {


    ###############
    # update alpha
    ###############

    # propose new value
    alpha.can <- rnorm(1, alpha.cur, sigma)

    # alpha is strictly positive, therefore any proposed value which is
    # negative is automatically rejected.
    if (alpha.can > 0.0) {

      # otherwise, if it is positive the accept it with some probability:

      # compute the log-densities
      log.pi.cur <- log_likelihood(alpha = alpha.cur, p = p.cur) +
        log(dgamma(alpha.cur, mu.alpha, nu.alpha))

      log.pi.can <- log_likelihood(alpha = alpha.can, p = p.cur) +
        log(dgamma(alpha.can, mu.alpha, nu.alpha))

      # log-qratio is zero since we are doing a random Walk Metropolis
      log.q.ratio <- 0;

      u <- runif(1)
      if (log(u) < log.pi.can - log.pi.cur) {
        alpha.cur <- alpha.can
```

```
      }
    }

    ############
    # propose p
    ############
    p.can <- runif(1, 0, 1)


    # compute the log-densities
    log.pi.cur <- log_likelihood(alpha = alpha.cur, p = p.cur)
    log.pi.can <- log_likelihood(alpha = alpha.cur, p = p.can)

    # log-qratio is zero
    log.q.ratio <- 0
    u <- runif(1)
    if (log(u) < log.pi.can - log.pi.cur) {
      p.cur <- p.can
    }

    # store the output
    res[i,] <- c(alpha.cur, p.cur)

  }

  res

}
```

Exercise 5

Question

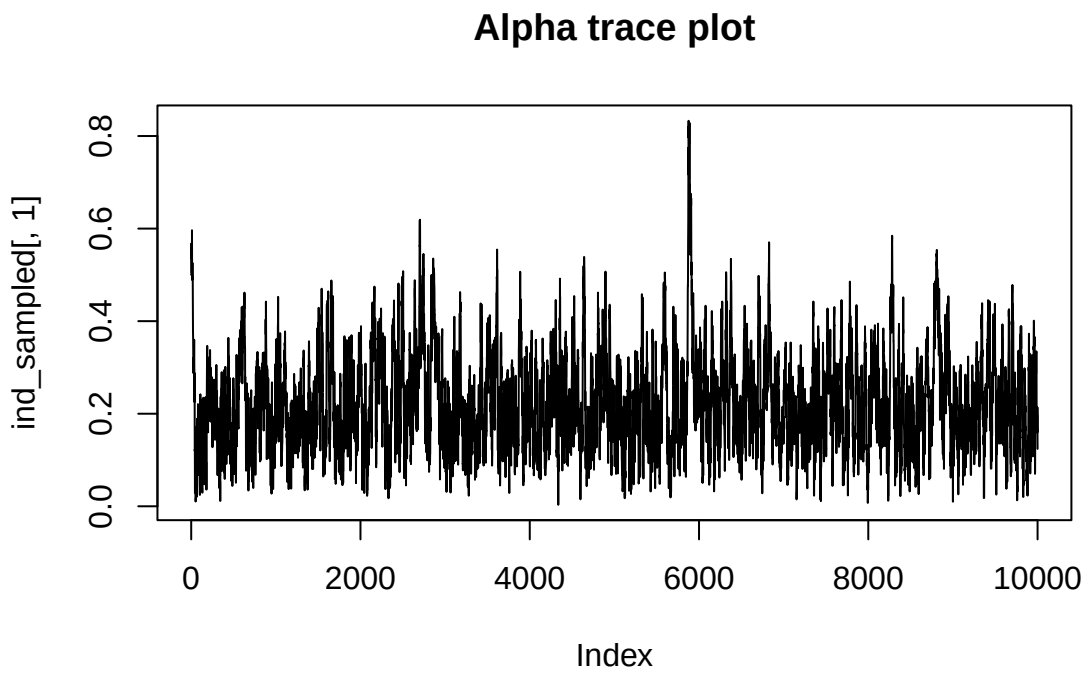
Investigate how correlated the posterior samples of α and p are.

Answer

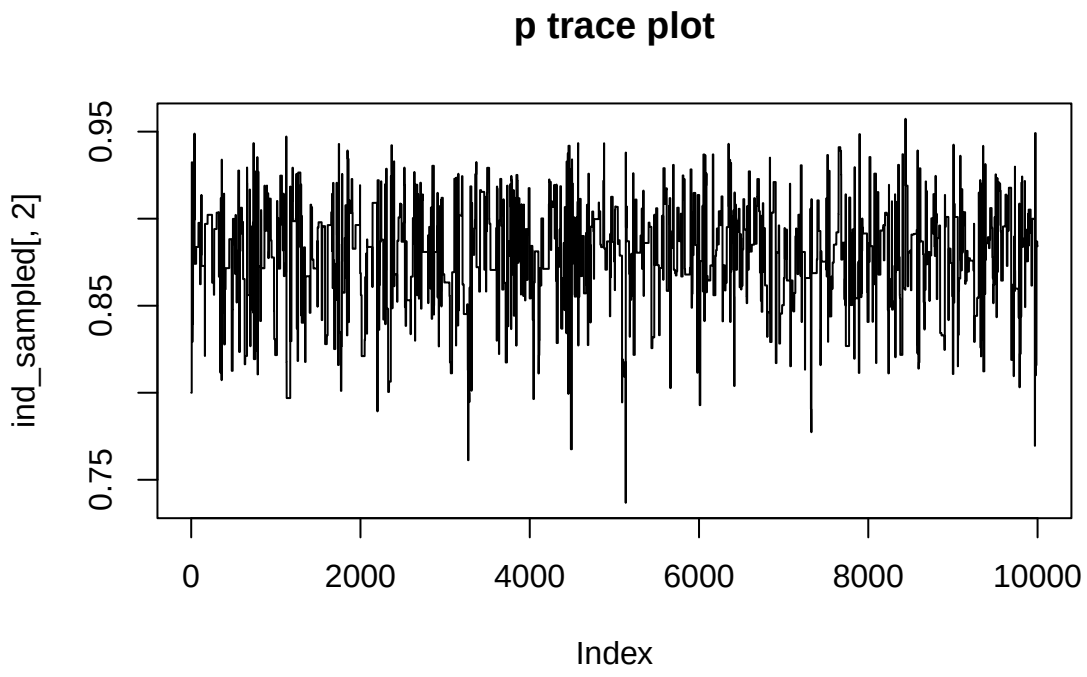
```
ind_sampled <- mcmc.ind.sampler(iter = 10^4, sigma = 0.05)
```

Let's first look at the trace plots of the parameters to make sure it all went OK

```
plot(ind_sampled[, 1], type = "l", main = "Alpha trace plot")
```

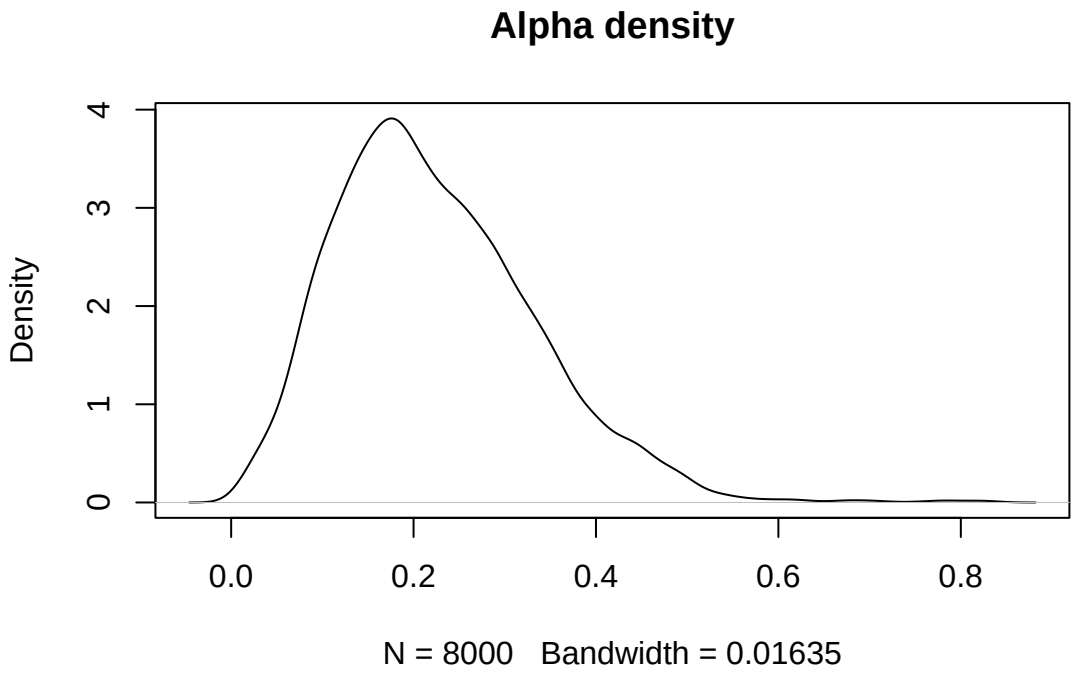



```
plot(ind_sampled[, 2], type = "l", main = "p trace plot")
```

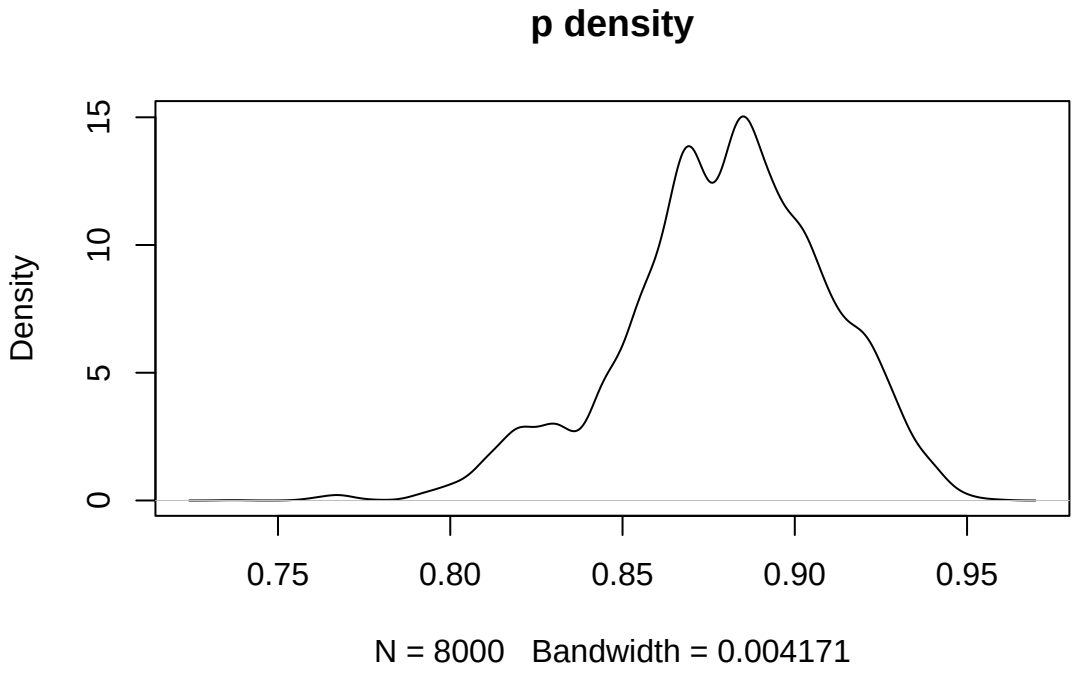


Let's now look at the posterior marginal densities, excluding the first 2000 samples as a burn in period

```
plot(density(ind_sampled[-(1:2000), 1]), main = "Alpha density")
```

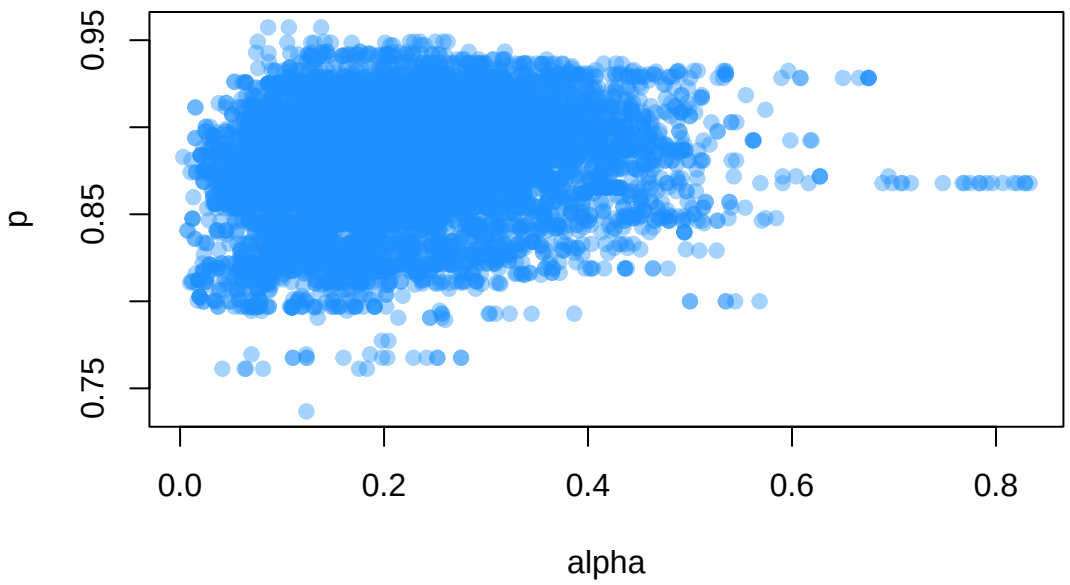



```
plot(density(ind_sampled[-(1:2000), 2]), main = "p density")
```



Now let's look at the joint distribution of the parameters to check correlations

```
plot(ind_sampled, xlab = "alpha", ylab = "p", col = scales::alpha("dodgerblue", 0.4), pch = 19)
```



Exercise 6

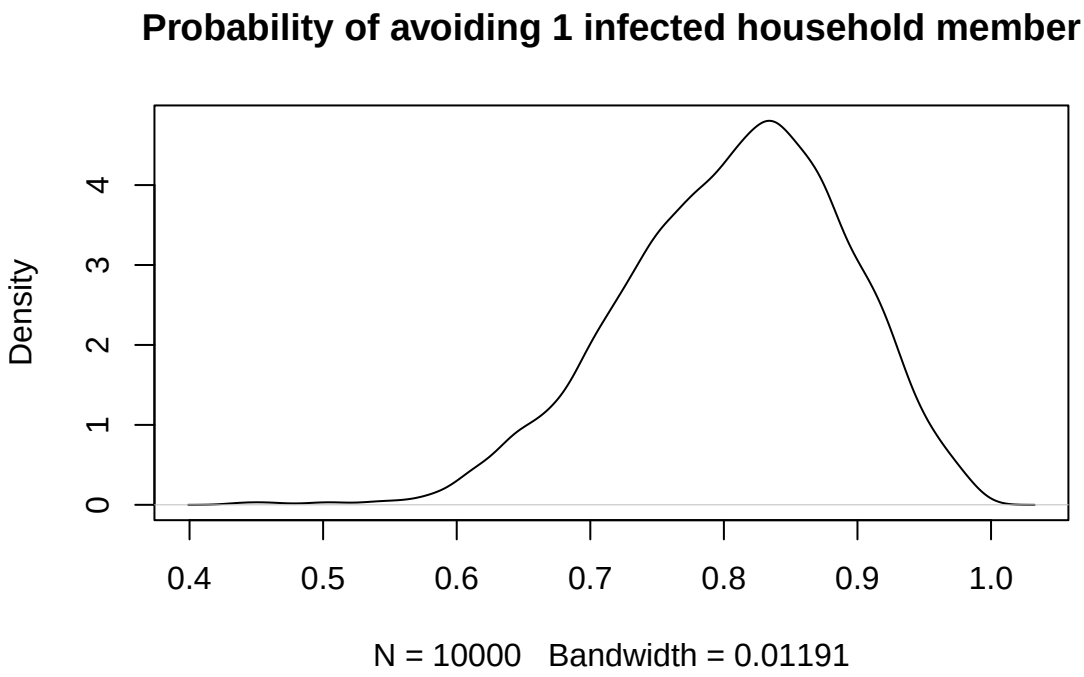
Question

Draw samples from the posterior distribution of the probability that a susceptible individual avoids infection from one infected household member.

Answer

Remember that our posterior distribution for our transmission parameter α is just $q = \exp(-\alpha)$, so we can just sample from that

```
plot(density(exp(-ind_sampled[, 1])), main = "Probability of avoiding 1 infected household member")
```



Exercise 7

Question

In the current version of your MCMC algorithm the parameter p is updated via an independence sampler. Modify your code such that p is now updated using a random walk Metropolis. Does the mixing of your algorithm improve?

Answer

```
mcmc.random.walk <- function(iter, sigma) {

  # values for the prior hyper-parameters
  mu.p <- 1.0
  nu.p <- 1.0

  mu.alpha <- 1.0
  nu.alpha <- 10^(-3)


  # initial values for the Markov Chain
  alpha.cur <- 0.5;
  p.cur <- 0.8;

  # create a matrix to store the output.
  res <- matrix(NA, nrow = iter, ncol=2);
  res[1,] <- c(alpha.cur, p.cur);
```

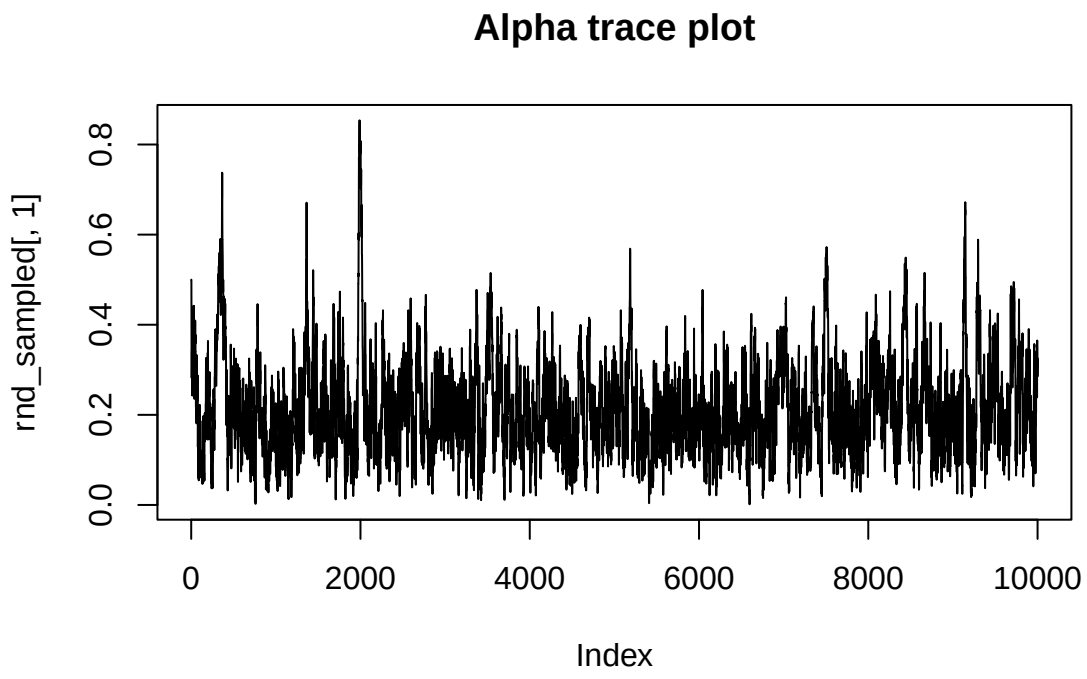
```r
for (i in 2:iter) {


  ###############
  # update alpha
  ###############

  # propose new value
  alpha.can <- rnorm(1, alpha.cur, sigma)

  # alpha is strictly positive, therefore any proposed value which is
  # negative is automatically rejected.
  if (alpha.can > 0.0) {

    # otherwise, if it is positive the accept it with some probability:

    # compute the log-densities
    log.pi.cur <- log_likelihood(alpha = alpha.cur, p = p.cur) +
      log(dgamma(alpha.cur, mu.alpha, nu.alpha))

    log.pi.can <- log_likelihood(alpha = alpha.can, p = p.cur) +
      log(dgamma(alpha.can, mu.alpha, nu.alpha))

    # log-qratio is zero since we are doing a random Walk Metropolis
    log.q.ratio <- 0;

    u <- runif(1)
    if (log(u) < log.pi.can - log.pi.cur) {
      alpha.cur <- alpha.can
    }
  }

  #############
  # propose p
  #############
  p.can <- rnorm(1, p.cur, sigma)

  if (p.can > 0.0 & p.can < 1.0){

    # compute the log-densities
    log.pi.cur <- log_likelihood(alpha = alpha.cur, p = p.cur)
    log.pi.can <- log_likelihood(alpha = alpha.cur, p = p.can)

    # log-qratio is zero
    log.q.ratio <- 0

    u <- runif(1)
    if (log(u) < log.pi.can - log.pi.cur) {
      p.cur <- p.can
    }
  }
```

```
    # store the output
    res[i,] <- c(alpha.cur, p.cur)

  }

  res

}
```

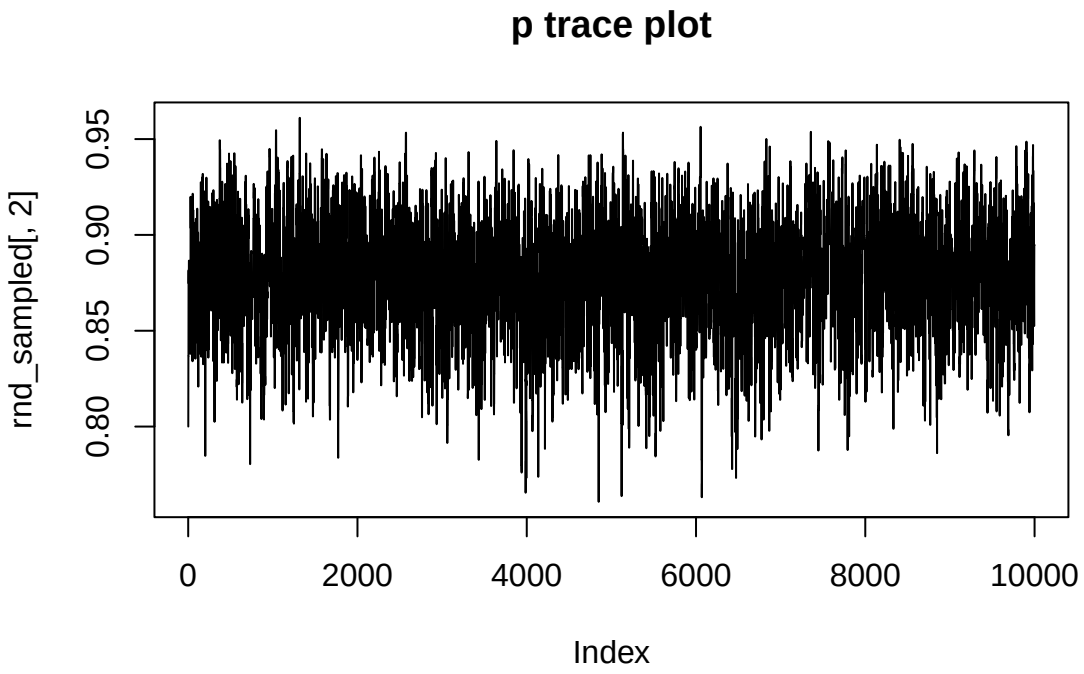
```
rnd_sampled <- mcmc.random.walk(iter = 10^4, sigma = 0.05)
```

Let's first look at the trace plots of the parameters to make sure it all went OK

```
plot(rnd_sampled[, 1], type = "l", main = "Alpha trace plot")
```

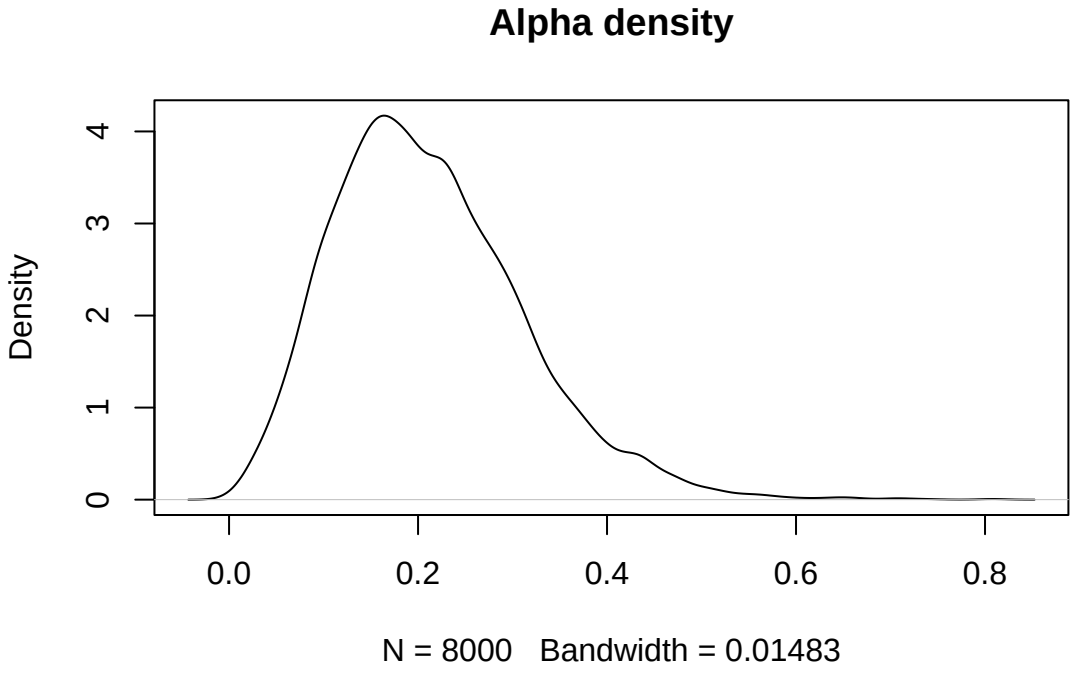



```
plot(rnd_sampled[, 2], type = "l", main = "p trace plot")
```

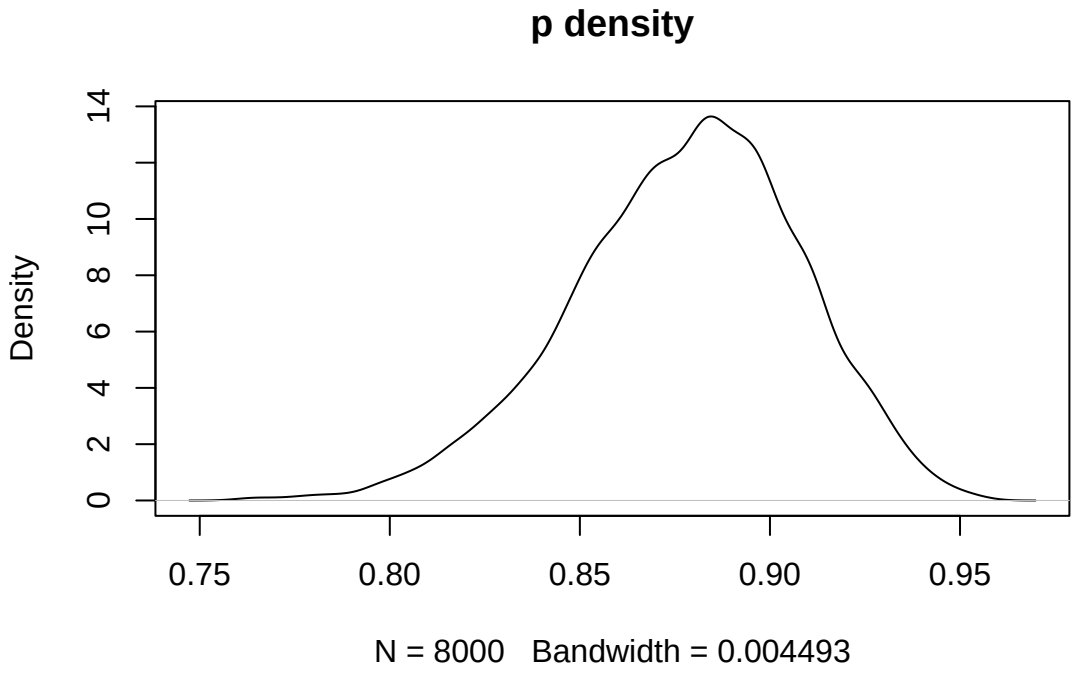


Let's now look at the posterior marginal densities, excluding the first 2000 samples as a burn in period

```
plot(density(rnd_sampled[-(1:2000), 1]), main = "Alpha density")
```

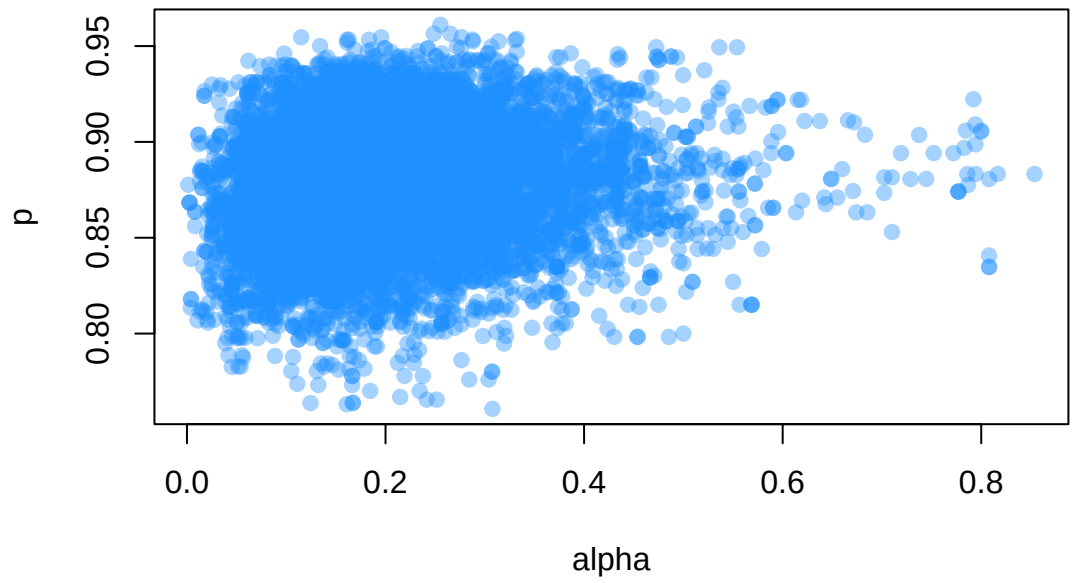


```
plot(density(rnd_sampled[-(1:2000), 2]), main = "p density")
```



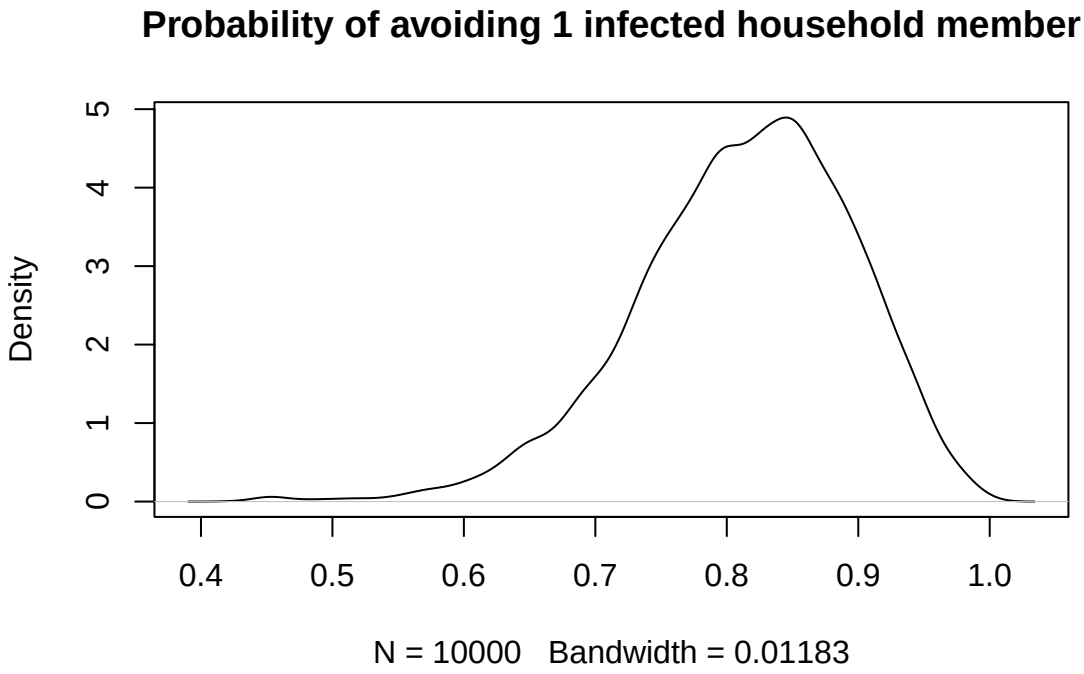
Now let's look at the joint distribution of the parameters to check correlations

```
plot(rnd_sampled, xlab = "alpha", ylab = "p", col = scales::alpha("dodgerblue", 0.4), pch = 19)
```



Remember that our posterior distribution for our transmission parameter α is just $q = \exp(-\alpha)$, so we can just sample from that

```
plot(density(exp(-rnd_sampled[, 1])), main = "Probability of avoiding 1 infected household member")
```



It doesn't appear that it makes much difference to update p via a random walk over the independence sampler.