Assignment 5

Yufei Yin

Exercise 1 (5 points):

In the Bayesian Computing (Part 3) lecture slides, we considered a target distribution (i.e., posterior distribution) that was a bivariate normal distribution:

$$p(\theta|y) = N_2(0, \Sigma_{2\times 2}),$$

where $\Sigma_{2\times 2}$ is the variance-covariance matrix with variances equal to 1 and correlation equal to 0.8.

Construct a Metropolis algorithm to sample from the above target distribution. You may use a starting value of (0, 0) and the jumping distribution from the lecture slides, or you may use another (but still symmetric) jumping distribution. (Of course you are encouraged to try different starting values and jumping distributions to learn about the algorithm, but you only need to turn in one set to be marked.) Obtain 5500 draws from $p(\theta|y)$ using your Metropolis algorithm, and discard the first 500 draws as burn-in. The end result is a sample of 5000 draws from $p(\theta|y)$. Use those draws to produce the figures requested below as necessary.

For this exercise, please turn in the following:

1.

A description of your Metropolis algorithm in words, specific to this problem. That is, you should clearly state how proposed values for $\theta = (\theta_1, \theta_2)$ are drawn, what the acceptance ratio is, etc., not just restate the general form of the algorithm from the lecture notes. Or, to put it a third way, a knowledgeable reader should be able to translate your description into code without needing additional information.

Our target distribution is a bivariate normal distribuion: $p(\theta|y) = N_2(0, \Sigma_{2\times 2})$

$\Sigma_{2\times 2}$ is

$$\begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Metropolis algorithm

We set our starting value as $\theta = (0, 0)$, then start interating. For each iteration we sample θ^* from jump distribution. In this problem, the choice of our jump distribuion is $N_2(\theta^{t-1}, k\Sigma_{2\times 2})$ where $\Sigma_{2\times 2}$ is the same as target distribuion and k is 0.3. After sampling θ^*, we calculate the ratio of the densities. the ratio is between θ^* in target distribution and θ^{t-1} in target distribution. if the jump increases the density, set $\theta^t = \theta^*$, if the jump decreases the density, set $\theta^t = \theta^*$ with probability equal to the density ratio, r, and set $\theta^t = \theta^{t-1}$ otherwise. Finally we will get $\theta = (\theta_1, \theta_2)$ after the iteration.

Steps

1. We set our starting value as $\theta = (0, 0)$
2. Then, for t = 1,2,. . . ;

(a) Sample θ^* from $J_t(\theta^*|\theta^{t-1})$, with J_t symmetric

The choice of our jump distribution J_t in this problem is $N_2(\theta^{t-1}, k\Sigma_{2\times 2})$, and we choose k as 0.3.

(b) Calculate the ratio of the densities,

$$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$

(c) Set

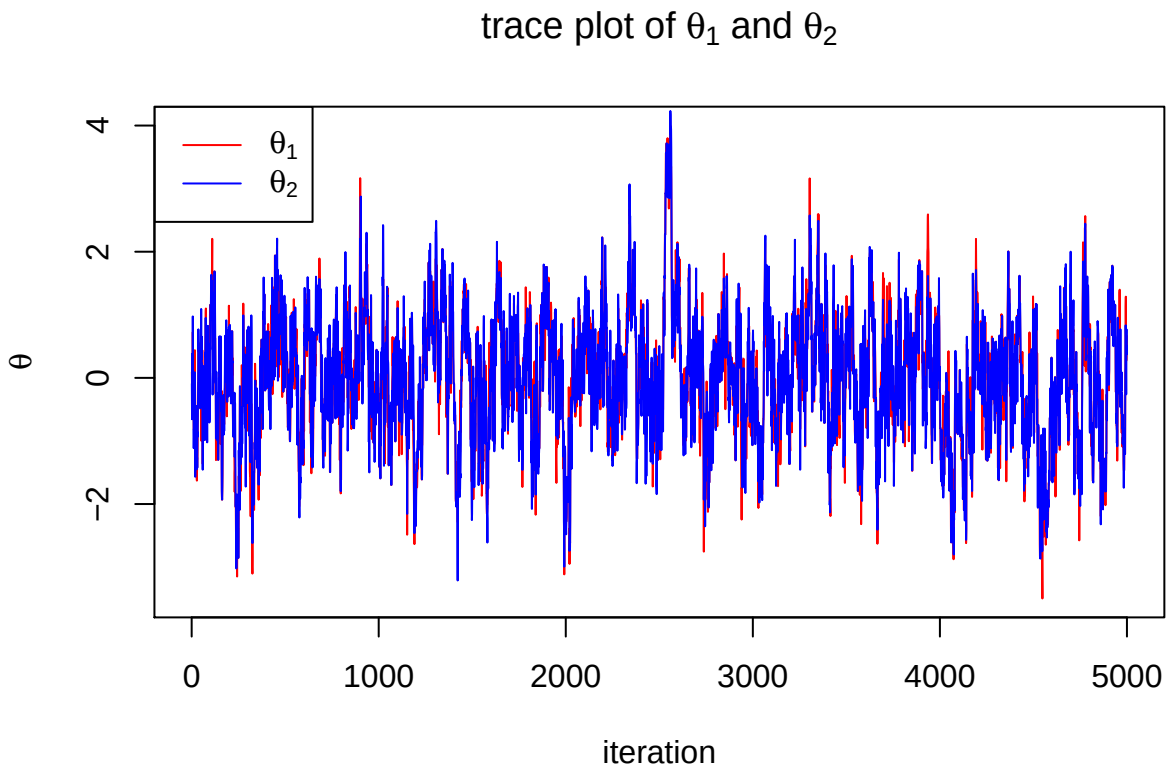
$$\theta^{(t)} = \begin{cases} \theta^* & \text{with probability min (r, 1)} \\ \theta^{(t-1)} & \text{otherwise} \end{cases}$$

2.

Trace plots of the draws for θ_1 and θ_2. Also comment on the trace plots. Does the Markov chain appear to have converged? Is the chain mixing well, or is there (visual) evidence of high autocorrelation? (You do not need to make the autocorrelation function plot, just to provide a visual qualitative judgement.)

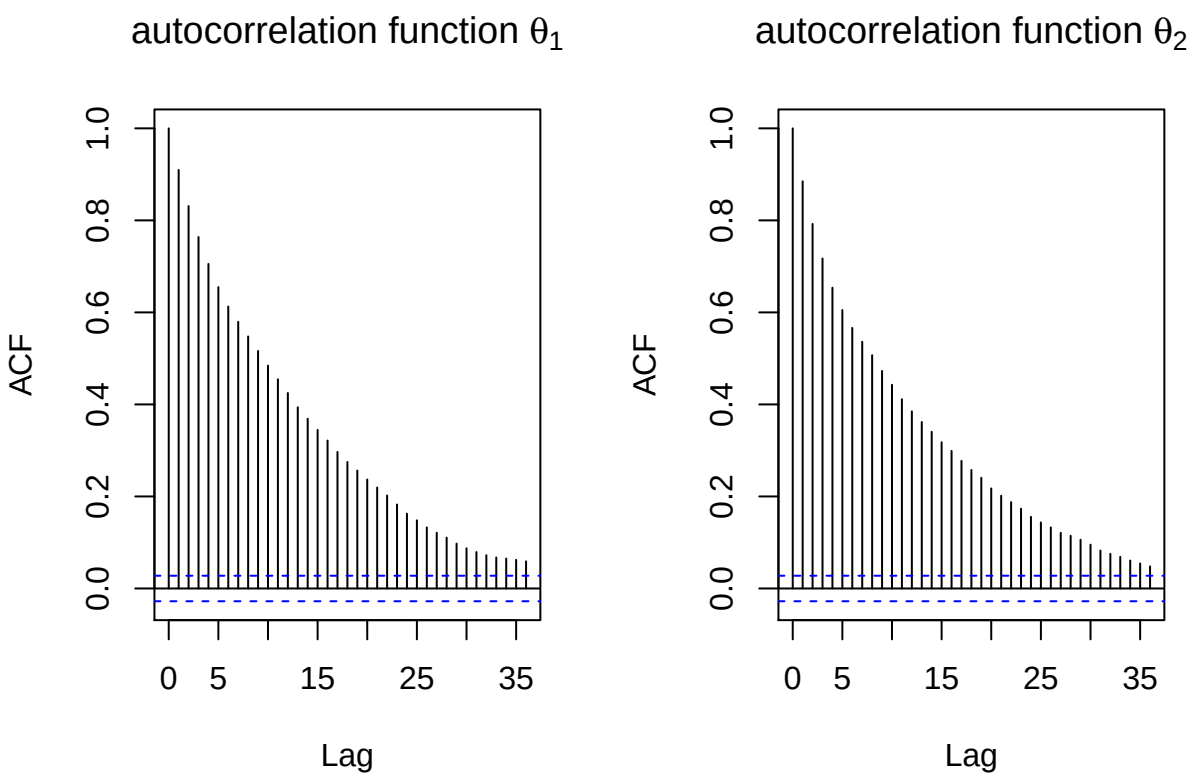
```
source("Metropolis Algorithm.R")
```

```
plot(c(1:5000), theta_samples[,1],
     type="l",
     col = "red",
     xlab="iteration",
     ylab=expression(theta),
     main = expression(paste("trace plot of ",theta[1]," and ",theta[2])))
lines(theta_samples[,2], col = "blue")
legend("topleft",
       col = c("red", "blue"),
       lwd = c(1,1),
       legend = c(expression(theta[1]),expression(theta[2])))
```



The trace plots indicate the Markov chin appear to have converged, the chain mixing well, and there is no visual evidence of high autocorrelatoin. trace plots do not look wonderful. but they're not too bad.

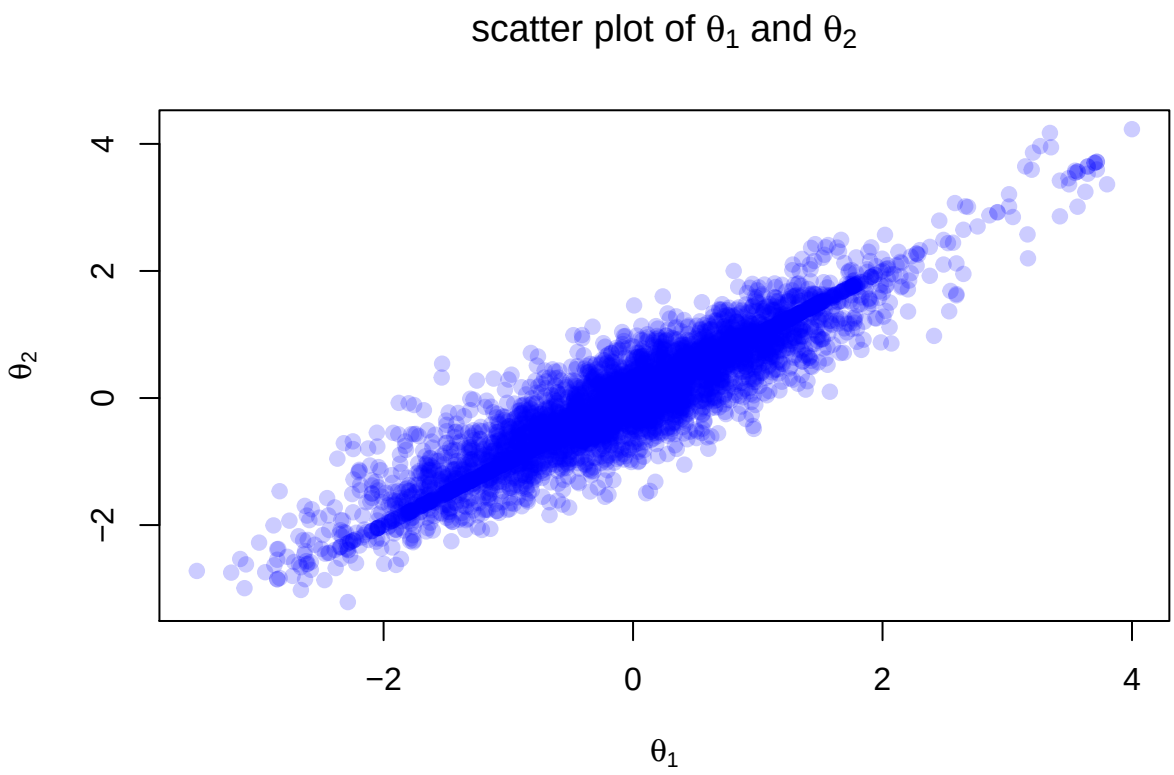
```
par(mfrow = c(1,2))
acf(theta_samples[,1], main = expression(paste("autocorrelation function ",theta[1])))
acf(theta_samples[,2], main = expression(paste("autocorrelation function ",theta[2])))
```


autocorrelation function θ_1
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
15
25
35
Lag
autocorrelation function θ_2
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
15
25
35
Lag

3.

A two-dimensional scatterplot of the draws for θ_1 and θ_2. Comment on whether the draws appear to match the bivariate normal target density. (If they do not, you might consider trying to fix the error, as it is either a bug in your code, or a conceptual misunderstanding of the algorithm!)

```
plot(theta_samples[,1], theta_samples[,2],col=rgb(0,0,1,alpha=0.2),pch=19,
     xlab = expression(theta[1]),
     ylab = expression(theta[2]),
     main = expression(paste("scatter plot of ", theta[1], " and ", theta[2])))
```



According to the scatter plot, the draws apear to match the bivariate normal target density.

4.

Your code, which you may provide in an Appendix if you wish. Your code should be commented and readable, particularly if in a language other than R.

```
# Metropolis Algorithm.R
library(mvtnorm)

# write the Metropolis algorithm
metrop_alg <- function(num_iter){

  # Step 1: set up theta vector and get init. value
  theta <- matrix(rep(NA, 2*num_iter), ncol = 2)
  theta[1,] <- c(0,0)

  # variance-covariance matrix
  S <- diag(2)
  S[1, 2] <- 0.8
  S[2, 1] <- 0.8

  # jump distribuion, variance-covariance matrix scaled by 0.3
  sp <- 0.3

  # Step 2
  for (i in 2:num_iter){

    # Step 2(a): sample theta_star from jumping dist.
    theta_star <- rmvnorm(n = 1, mean = theta[i-1,], sigma = sp*S)

    # Step 2(b): Calculate ratio of densities
    curr <- dmvnorm(x = theta[i-1,],mean = c(0,0), sigma = S)
    prop <- dmvnorm(x = theta_star, mean = c(0,0), sigma = S)
    r <- curr/prop

    # Step 2(c): accept or reject proposed value

    # accept
    if (r > runif(1, min = 0, max = 1)){
      theta[i,] <- theta_star
    }

    # reject
    else{
      theta[i,] <- theta[i-1]
    }
  }

  return(theta)
}

# Obtain 5500 draws
theta_samples <- metrop_alg(5500)

# discard the first 500 draws as burn-in.
theta_samples <- theta_samples[-(1:500),]
```

Exercise 2 (5 points):

Repeat Exercise 1, but replace the Metropolis algorithm with an Independence Sampler as follows:

1.

Use the sample of θ from Exercise 1 to estimate the variance-covariance matrix of the posterior distribution, $\hat{\Sigma}$. (In R this is done easily using the var function.)

```
var(theta_samples)
```

```
##           [,1]      [,2]
## [1,] 1.0130577 0.9141499
## [2,] 0.9141499 0.9685535
```

2.

Approximate the posterior distribution as $N_2(MAP, \hat{\Sigma})$, where MAP is the MAP estimate (i.e., estimate of the posterior mode). Then use this approximation as the jumping distribution in the Independence Sampler.

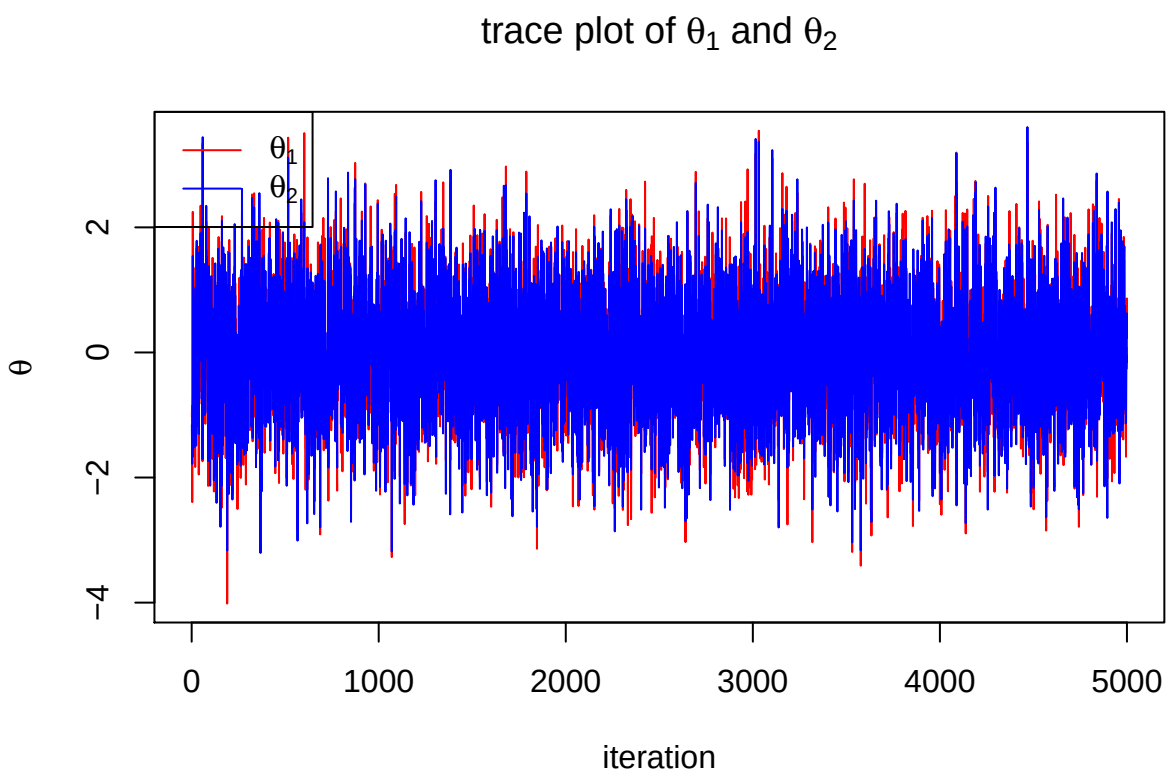
So our new jumping distribuion is $N_2(MAP, \hat{\Sigma})$, where MAP is (0,0), and $\hat{\Sigma}$ is the matrix we obtain from question 1.

update Algorithm

```
metrop_alg <- function(num_iter, jump_size){

  # Step 1: set up theta vector and get init. value
  theta <- matrix(rep(NA, 2*num_iter), ncol = 2)
  theta[1,] <- c(0,0)

  # variance-covariance matrix
  S <- diag(2)
  S[1, 2] <- 0.8
  S[2, 1] <- 0.8

  # Step 2
  for (i in 2:num_iter){

    # Step 2(a): sample theta_star from jumping dist.(updated)
    theta_star <- rmvnorm(n = 1, mean = c(0,0), sigma = jump_size)

    # Step 2(b): Calculate ratio of densities
    curr <- dmvnorm(x = theta[i-1,], mean = c(0,0), sigma = S)
    prop <- dmvnorm(x = theta_star, mean = c(0,0), sigma = S)
    r <- (prop/dmvnorm(theta_star, mean = c(0,0), sigma = S))/
      (curr/dmvnorm(theta[i-1,], mean = c(0,0), sigma = S))

    # Step 2(c): accept or reject proposed value

    # accept
    if (r > runif(1, min = 0, max = 1)){
      theta[i,] <- theta_star
    }

    # reject
    else{
      theta[i,] <- theta[i-1]
    }
  }

  return(theta)
}

# Obtain 5500 draws
theta_samples <- metrop_alg(5500, var(theta_samples))

# discard the first 500 draws as burn-in.
theta_samples <- theta_samples[-(1:500),]
```

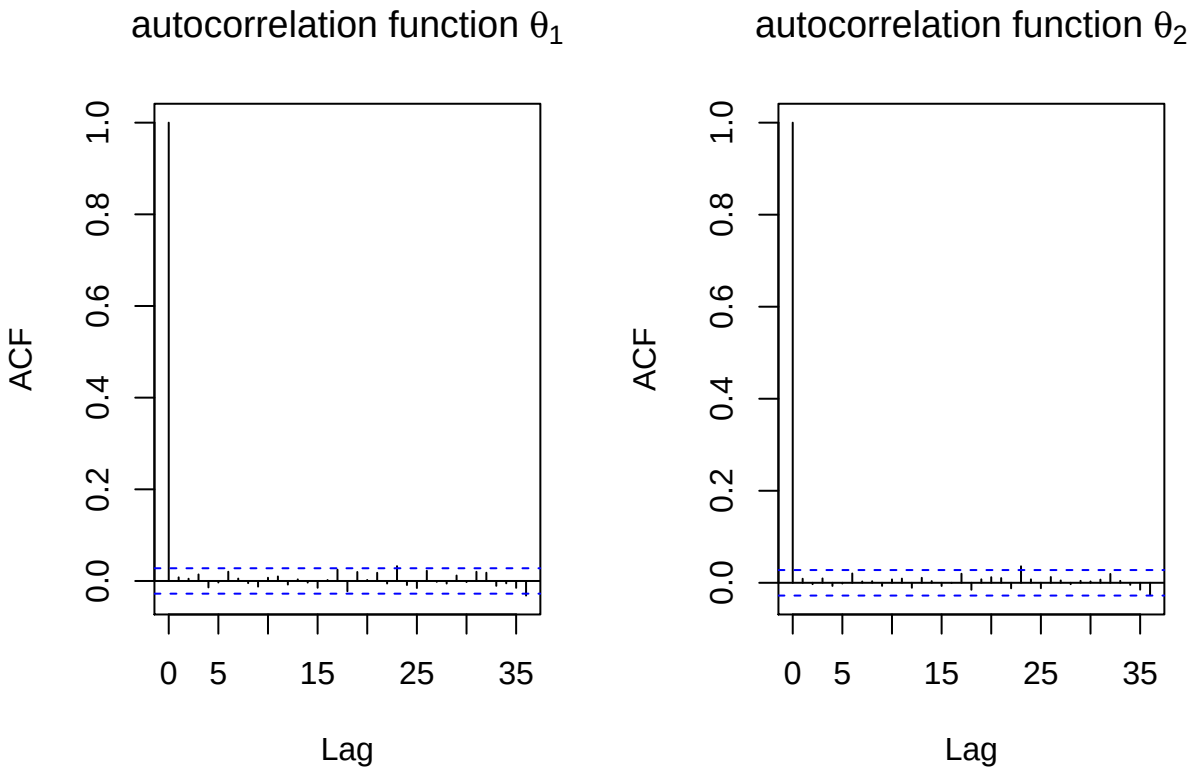
trace plot

```
plot(c(1:5000), theta_samples[,1],
     type="l",
     col = "red",
     xlab="iteration",
     ylab=expression(theta),
     main = expression(paste("trace plot of ",theta[1]," and ",theta[2])))
lines(theta_samples[,2], col = "blue")
legend("topleft",
       col = c("red", "blue"),
       lwd = c(1,1),
       legend = c(expression(theta[1]),expression(theta[2])))
```



autocorrelation function

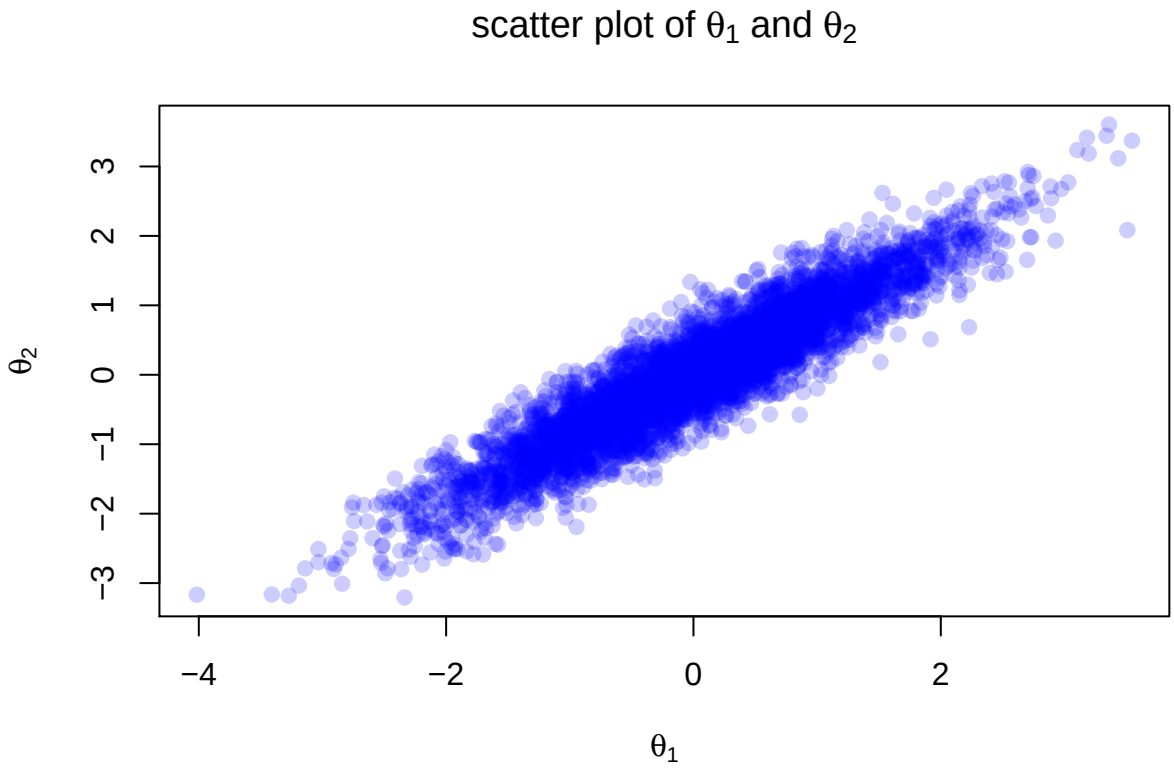
```
par(mfrow = c(1,2))
acf(theta_samples[,1], main = expression(paste("autocorrelation function ",theta[1])))
acf(theta_samples[,2], main = expression(paste("autocorrelation function ",theta[2])))
```



The trace plots indicate the Markov chin appear to have converged, the chain mixing well, and there is no visual evidence of high autocorrelatoin; moreover, autocorrelation is improved compared to exercise 1, because we have good approximation to posterior.

scatter plot

```
plot(theta_samples[,1], theta_samples[,2],col=rgb(0,0,1,alpha=0.2),pch=19,
     xlab = expression(theta[1]),
     ylab = expression(theta[2]),
     main = expression(paste("scatter plot of ", theta[1], " and ", theta[2])))
```



According to the scatter plot, the draws apear to match the bivariate normal target density.

Exercise 3 (5 points):

For this exercise we will reconsider the student heights example that we've used previously. Recall that we measure the heights (in inches) of three students and obtain y1 = 70, y2 = 75, and y3 = 72. Previously we assumed that the heights follow a normal distribution with an unknown mean but known variance. For this exercise, we suppose (more realistically) that both the mean and variance are unknown model parameters for which we specify the following prior distributions:

$$\mu \sim N(\mu_0 = 65, \tau_0^2 = 9)$$
$$\sigma^2 \sim Inv - \chi^2(\upsilon_0 = 175, \sigma_0^2 = 16)$$

Construct a Gibbs sampler and use it to obtain a sample from the resulting posterior distribution. Obtain 5000 draws from $p(\mu, \sigma^2|y)$ after burn-in. This means that you should run your chains for longer than 5000 iterations such that there are 5000 iterations after the Markov chain has appeared to converge; for this you will have to examine the trace plots. Use the sample of 5000 draws to produce the figures requested below as necessary.

For this exercise, please turn in the following:

1.

A description of your Gibbs sampler in words, specific to this problem. For example, you should clearly state the the conditional distributions and how they are used to update μ and σ^2 at each iteration. As before, a knowledgeable reader should be able to translate your description into code without needing additional information.

steps

1. we set our starting value as $\mu^{(0)} = 65$, $\sigma^{2(0)} = 16$
2. For t = 2, ..., N_{iter}, draw from conditional distributions:

$$\mu|\sigma^2, y \sim N(\frac{\frac{\mu_0}{\tau_0^2} + \frac{n\bar{y}}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}, \frac{1}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}})$$
$$\sigma^2|\mu, y \sim Inv - \chi^2(\upsilon_0 + n, \frac{\upsilon_0\sigma_0^2 + n\upsilon}{\upsilon_0 + n}), \quad \upsilon = \frac{1}{n}\sum(y_i - \mu^{(t)})^2$$

So for each iteration, we draw:

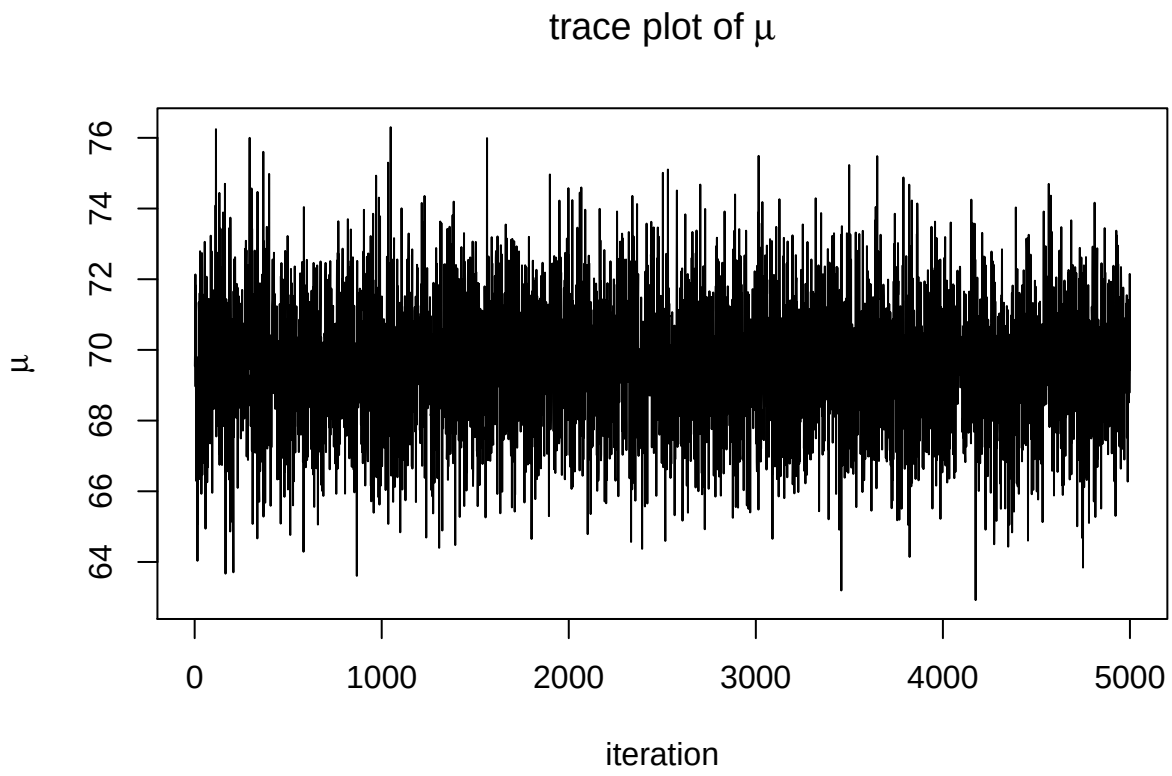
1. $\mu^{(t)} \sim \mu|\sigma^{2(t-1)}, y$
2. $\sigma^{2(t)} \sim \sigma^2|\mu^{(t)}, y$

2.

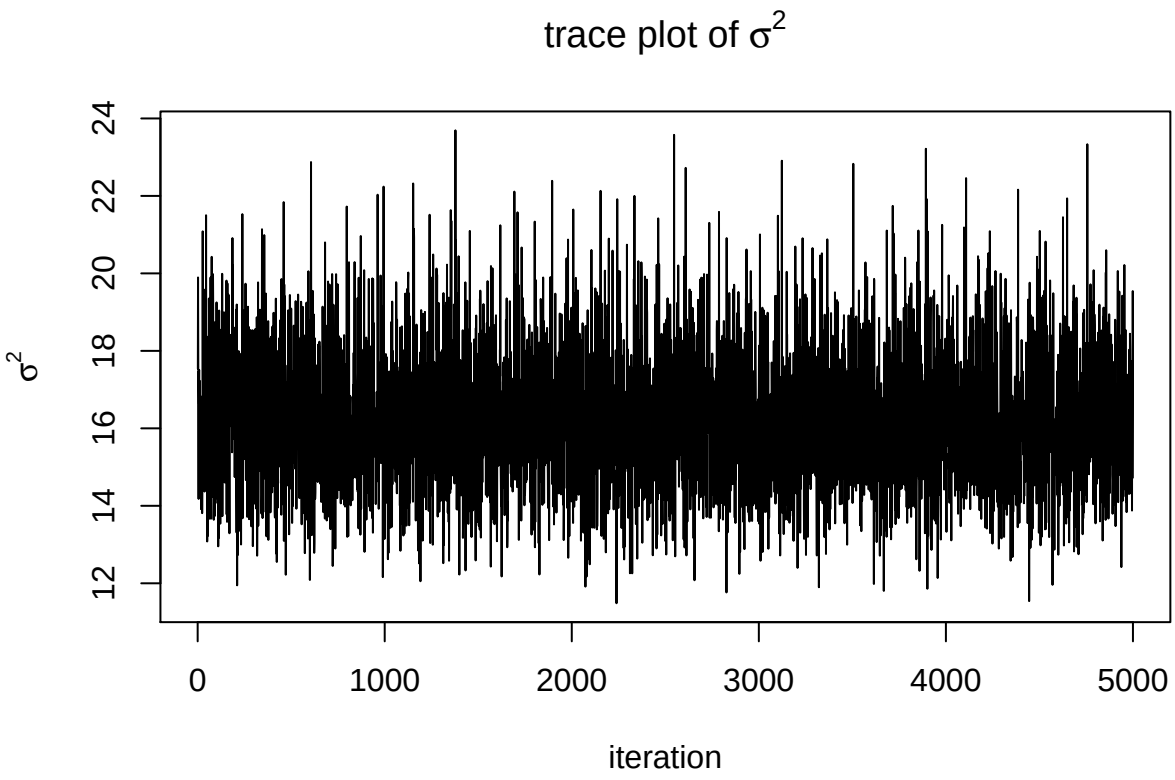
Trace plots of the draws for μ and σ^2. Also comment on the trace plots. Is the chain mixing well, or is there (visual) evidence of high autocorrelation? (You do not need to make the autocorrelation function plot, just to provide a visual qualitative judgement. You also do not need to comment on whether the chain has converged because the 5000 draws are supposed to be after convergence.)

```
source("Gibbs sampler.R")
```

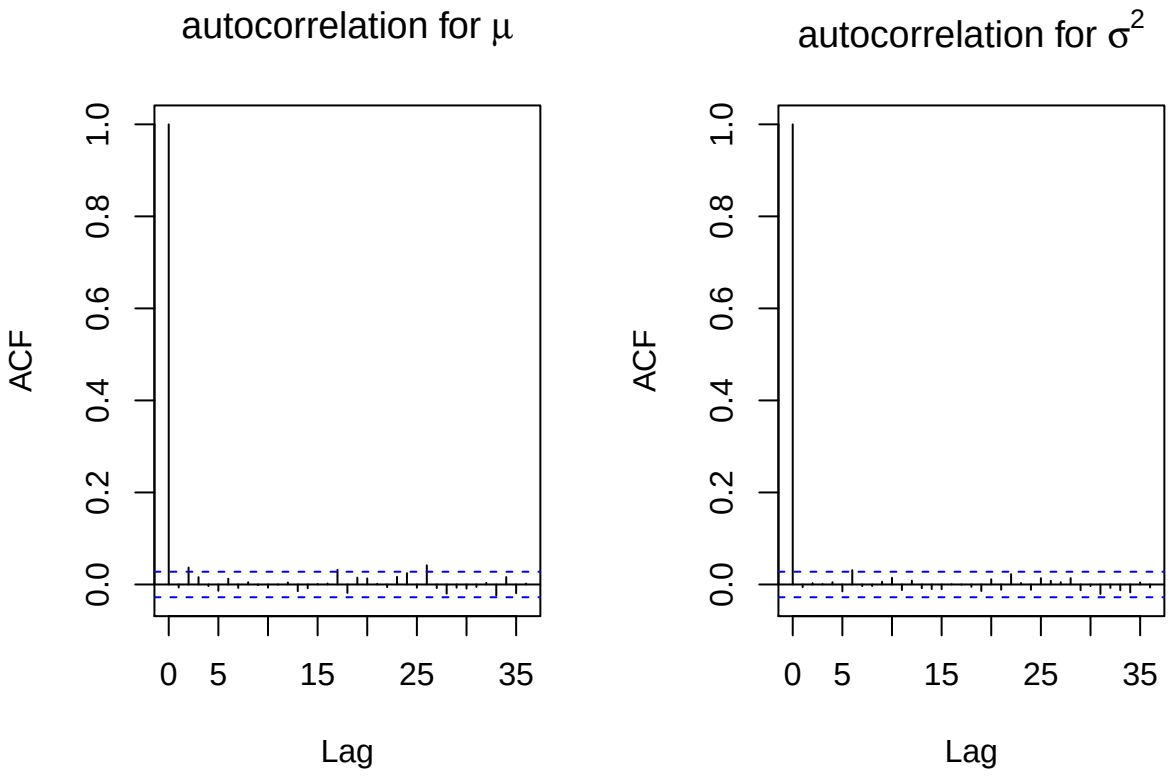
```
plot(c(1:5000), mu,
     type="l",
     xlab="iteration",
     ylab=expression(mu),
     main = expression(paste("trace plot of ",mu)))
```




```
plot(c(1:5000), sig2,
     type="l",
     xlab="iteration",
     ylab=expression(sigma^2),
     main = expression(paste("trace plot of ",sigma^2)))
```




```
par(mfrow=c(1,2))
acf(mu, main = expression(paste("autocorrelation for ",mu)))
acf(sig2, main = expression(paste("autocorrelation for ",sigma^2)))
```

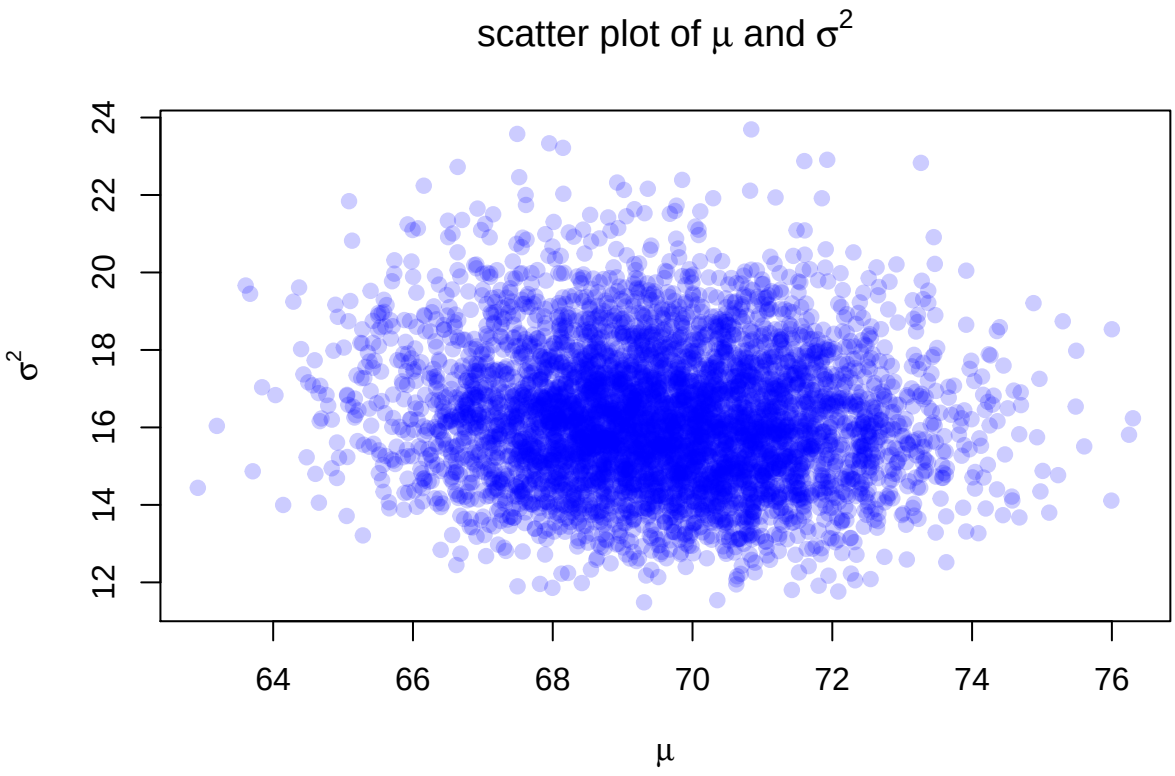


The trace plots indicate the chain is mixing well, and there is no (visual) evidence of high autocorrelation which is also proved by acf plot.

3.

A two-dimensional scatterplot of the draws for μ and σ^2.

```
plot(mu, sig2, col=rgb(0,0,1,alpha=0.2),pch=19,
     xlab = expression(mu),
     ylab = expression(sigma^2),
     main = expression(paste("scatter plot of ", mu, " and ", sigma^2)))
```



4.

Your code, which you may provide in an Appendix if you wish. Your code should be commented and readable, particularly if in a language other than R.

```
# Gibbs sampler.R

# student height example

# observed data
y <- c(70,75,72)
n <- length(y)
ybar <- mean(y)

# prior
mu0 <- 65
tau0sq <- 9
nu0 <- 175
sig0sq <- 16

# number of iterations
N_iter <- 10000

# set up vectors
mu <- rep(NA, N_iter)
sig2 <- rep(NA, N_iter)

# initialize
mu[1] <- 65
sig2[1] <- 16


# start iteration
for (t in 2:N_iter){

  a <- (mu0/tau0sq) + (n*ybar/(sig2[t-1]))
  b <- (1/tau0sq) + (n/(sig2[t-1]))


  # draw mu from conditional distribution (normal)
  mu[t] <- rnorm(1, mean = (a/b), sd = sqrt((1/b)))

  # draw sigma2 from conditional distribution (inverse chi-square)
  nu <- mean((y - mu[t])^2)
  deg_free <- nu0 + n
  s2 <- (nu0*sig0sq + nu*n)/(nu0 + n)
  X <- rchisq(1, df = deg_free)
  sig2[t] <- deg_free*s2/X
}

# discard the first 500 draws as burn-in.
mu <- mu[-(1:5000)]
sig2 <- sig2[-(1:5000)]
```