HW3 Solution

Jitong

2023-02-28

Question 1

a)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.4.0     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(invgamma)
x1 <- scan("http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/school1.dat")
x2 <- scan("http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/school2.dat")
x3 <- scan("http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/school3.dat")
x <- list(x1, x2, x3)
mu <- list()
s2 <- list()
Y <- list()

set.seed(0)
for(i in 1:3){
  n <- length(x[[i]])
  x.bar <- mean(x[[i]])
  prior.mean <- 5
  k.0 <- 1
  nu.0 <- 2
  sigma.sq0 <- 4

  nu.n <- nu.0 + n
  k.n <- k.0 + n
  sigma.sqn <- (nu.0*sigma.sq0 + (n-1)*var(x[[i]]) + k.0*n*(x.bar-prior.mean)^2/(k.n))/(nu.n)
  a <- nu.n / 2
  b <- nu.n * sigma.sqn / 2
  print(paste("Postrior mean of sigma is ", sqrt(b / (a - 1))))
  print("Confidence interval is")
  print(sqrt(qinvgamma(c(0.025, 0.975), a, b)))
```

```
  # Monte Carlo sampling
  S <- 10000
  s2[[i]] <- rinvgamma(S, a, b)
  mu[[i]] <- rnorm(S, (k.0*prior.mean + n*x.bar)/(k.n), sqrt(s2[[i]]/k.n))

  print(paste("Postrior mean of mu is ", mean(mu[[i]])))
  print("Confidence interval is")
  print(quantile(mu[[i]], c(0.025, 0.975)))
}
```

```
## [1] "Postrior mean of sigma is  3.94701741338171"
## [1] "Confidence interval is"
## [1] 3.002789 5.169623
## [1] "Postrior mean of mu is  9.29732812112059"
## [1] "Confidence interval is"
##      2.5%     97.5%
##  7.766568 10.836865
## [1] "Postrior mean of sigma is  4.44509921882711"
## [1] "Confidence interval is"
## [1] 3.343751 5.885496
## [1] "Postrior mean of mu is  6.94447922961877"
## [1] "Confidence interval is"
##     2.5%    97.5%
## 5.122509 8.780117
## [1] "Postrior mean of sigma is  3.79510461616265"
## [1] "Confidence interval is"
## [1] 2.798522 5.121435
## [1] "Postrior mean of mu is  7.80592001831727"
## [1] "Confidence interval is"
##     2.5%    97.5%
## 6.160150 9.426616
```

b)

```
# 123
mean(mu[[1]] < mu[[2]] & mu[[2]] < mu[[3]])
```

```
## [1] 0.0057
```

```
# 132
mean(mu[[1]] < mu[[3]] & mu[[3]] < mu[[2]])
```

```
## [1] 0.0041
```

```
# 213
mean(mu[[2]] < mu[[1]] & mu[[1]] < mu[[3]])
```

```
## [1] 0.0831
```

```
# 231
mean(mu[[2]] < mu[[3]] & mu[[3]] < mu[[1]])
```

```
## [1] 0.6737
```

```
# 312
mean(mu[[3]] < mu[[1]] & mu[[1]] < mu[[2]])
```

```
## [1] 0.0151
```

```
# 321
mean(mu[[3]] < mu[[2]] & mu[[2]] < mu[[1]])
```

```
## [1] 0.2183
```

c)

```
for(i in 1:3){
  Y[[i]] <- rnorm(S, mu[[i]], sqrt(s2[[i]]))
}

# 123
mean(Y[[1]] < Y[[2]] & Y[[2]] < Y[[3]])
```

```
## [1] 0.1065
```

```
# 132
mean(Y[[1]] < Y[[3]] & Y[[3]] < Y[[2]])
```

```
## [1] 0.1059
```

```
# 213
mean(Y[[2]] < Y[[1]] & Y[[1]] < Y[[3]])
```

```
## [1] 0.1881
```

```
# 231
mean(Y[[2]] < Y[[3]] & Y[[3]] < Y[[1]])
```

```
## [1] 0.2628
```

```
# 312
mean(Y[[3]] < Y[[1]] & Y[[1]] < Y[[2]])
```

```
## [1] 0.134
```

```
# 321
mean(Y[[3]] < Y[[2]] & Y[[2]] < Y[[1]])
```

```
## [1] 0.2027
```

d)

```
mean(mu[[2]] < mu[[1]] & mu[[3]] < mu[[1]])
```

```
## [1] 0.892
```

```
mean(Y[[2]] < Y[[1]] & Y[[3]] < Y[[1]])
```

```
## [1] 0.4655
```

Question 2

a)

```
set.seed(0)
x <- scan("http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/agehw.dat", skip = 1, what = list(double(), d
x <- matrix(c(x[[1]], x[[2]]), nrow = 100, ncol = 2)

library(MASS)
```

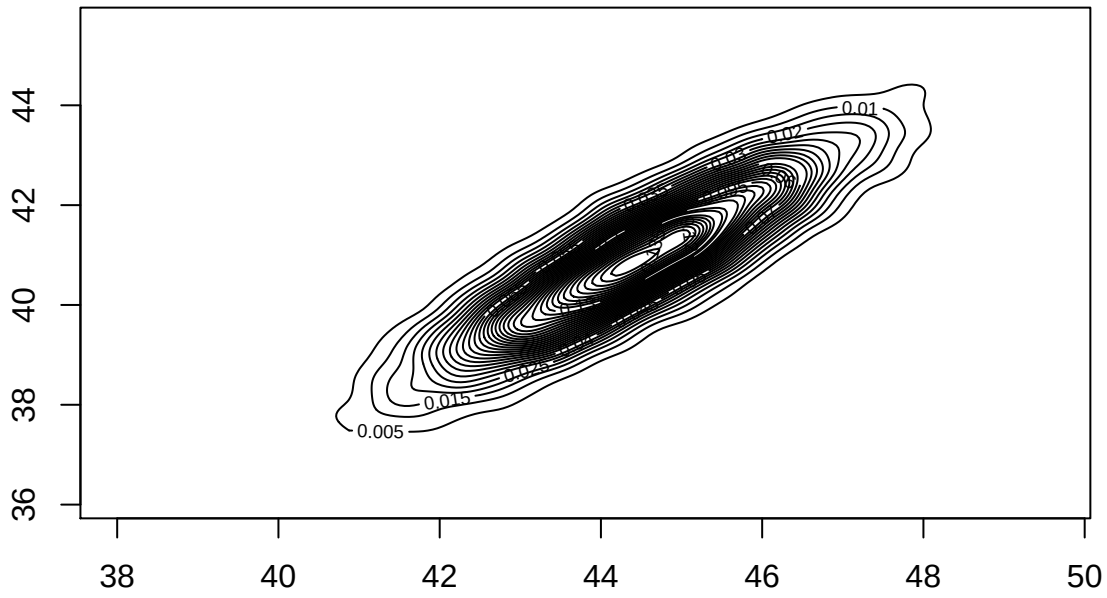
```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(MCMCpack)
```

```
## Loading required package: coda

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2023 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##

##
## Attaching package: 'MCMCpack'

## The following objects are masked from 'package:invgamma':
##
##     dinvgamma, rinvgamma
```

```
mu0 <- rep(0, 2)
L0 <- matrix(c(10^4, 0, 0, 10^4), nrow=2, ncol=2)
S0 <- matrix(c(10^2, 0, 0, 10^2), nrow=2, ncol=2)

n <- 100
nu0 <- 4
xbar <- apply(x, 2, mean)
Sigma <- cov(x)

S <- 10000
THETA <- matrix(0, nrow = S, ncol = 2)
SIGMA <- matrix(0, nrow = S, ncol = 4)

for(s in 1:S){

  # update theta
  Ln <- solve(solve(L0) + n * solve(Sigma))
  mun <- Ln %*% (solve(L0) %*% mu0 + n * solve(Sigma) %*% xbar)
  theta <- mvrnorm(1, mun, Ln)

  # update Sigma
  Sn <- S0 + (t(x) - c(theta)) %*% t( t(x) - c(theta))
  Sigma <- riwish(nu0 + n, Sn)

  # Update the output
  THETA[s,] <- theta
  SIGMA[s,] <- c(Sigma) # notice the Sigma matrix is vectorized
}
```

```
bivn.kde <- kde2d(THETA[,1], THETA[,2], n = 200)
contour(bivn.kde, nlevels = 50)
```



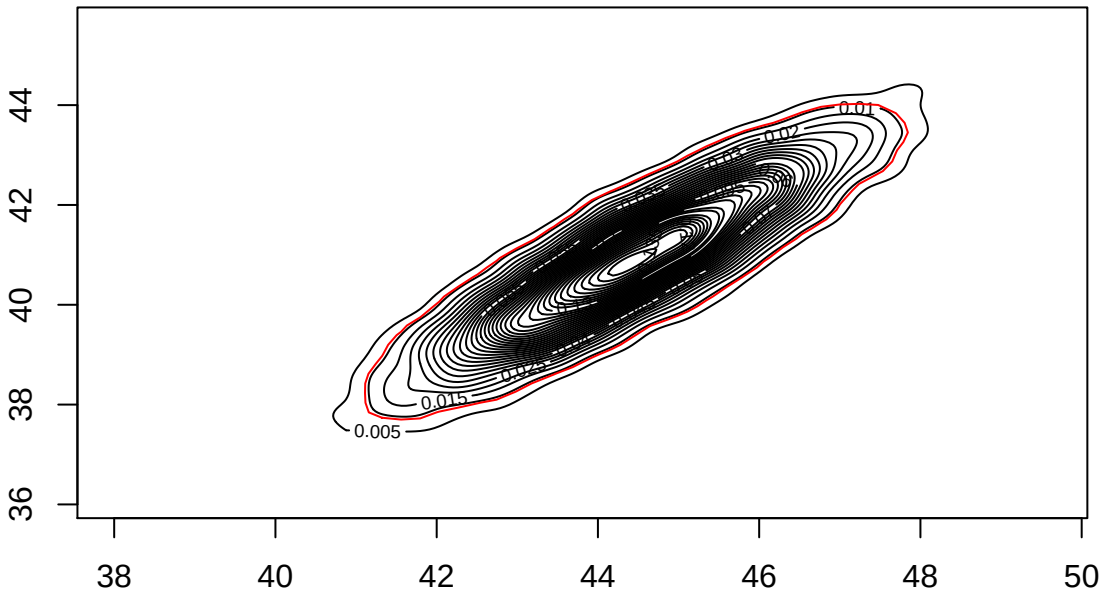

```
library(HDInterval)
hdi(THETA[,1], 0.95)
```

```
##    lower    upper
## 41.77070 47.09545
## attr(,"credMass")
## [1] 0.95
```

```
hdi(THETA[,2], 0.95)
```

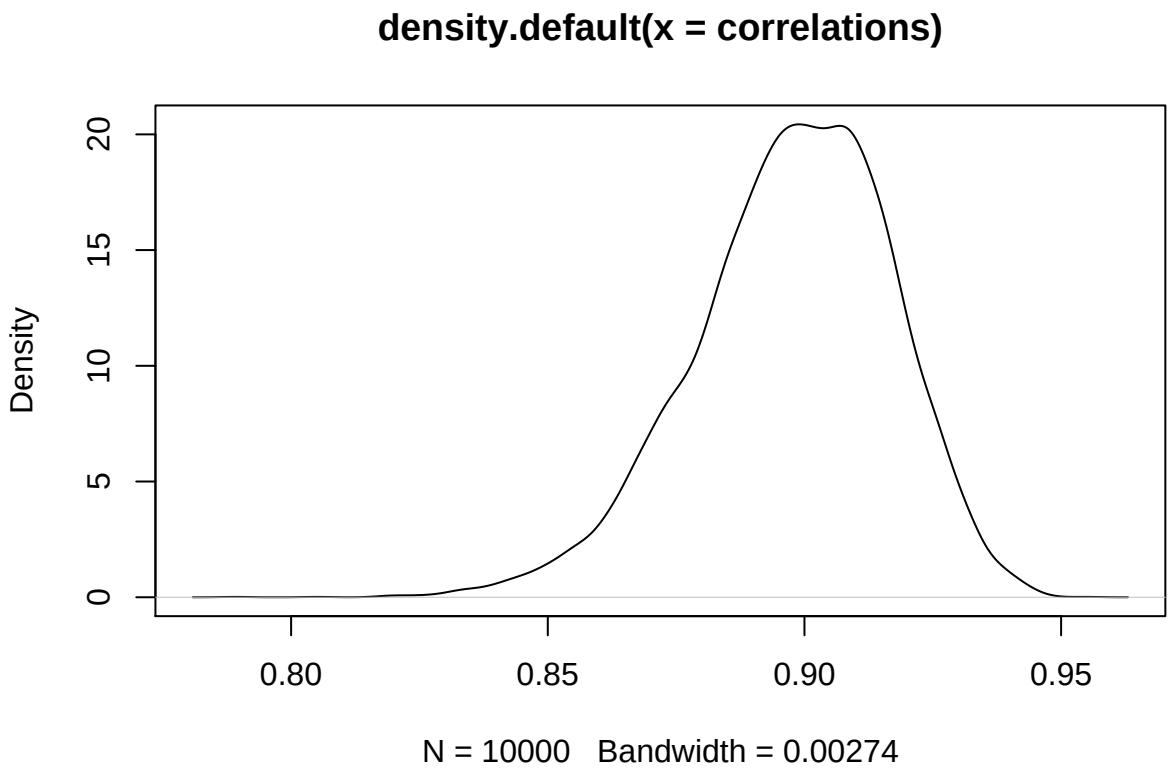
```
##    lower    upper
## 38.43052 43.41666
## attr(,"credMass")
## [1] 0.95
```

```
library(emdbook)
contour(bivn.kde, nlevels = 50)
HPDregionplot(mcmc(THETA), add = T, col = "red")
```



b)

```
correlations <- SIGMA[,2] / sqrt(SIGMA[,1] * SIGMA[,4])
plot(density(correlations))
```

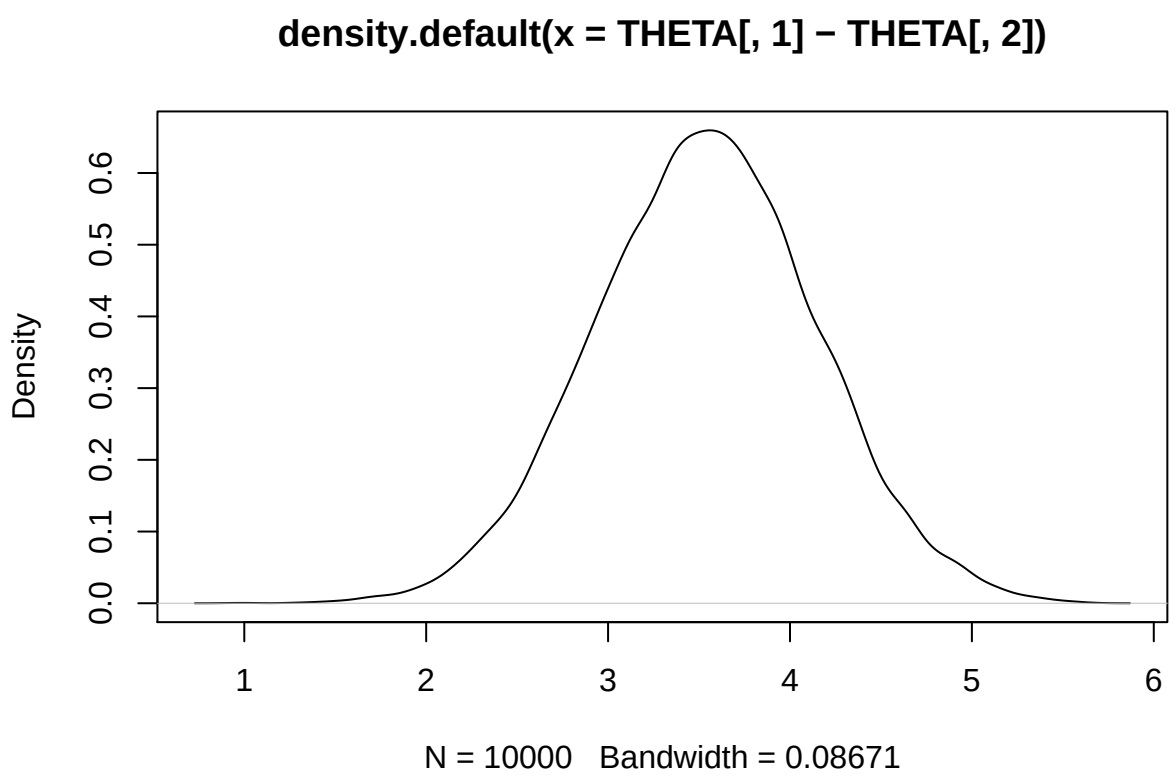



```
hdi(correlations, 0.95)
```

```
##     lower     upper
## 0.8594552 0.9328851
## attr(,"credMass")
## [1] 0.95
```

c)

```
plot(density(THETA[,1] - THETA[,2]))
```

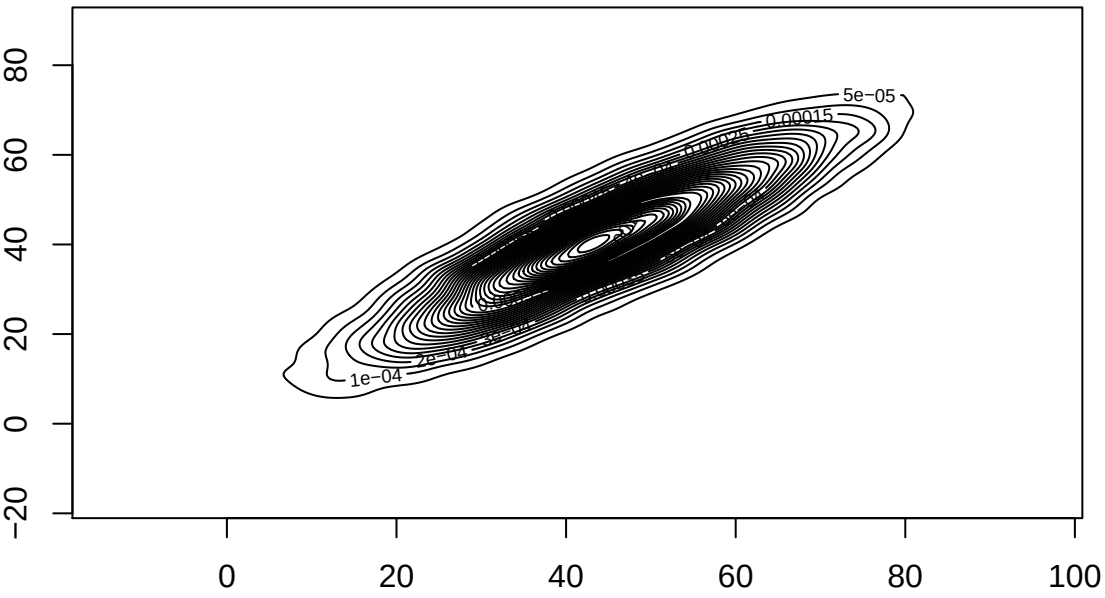



```
mean(THETA[,1] > THETA[,2])
```

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

d)

```
x_new <- matrix(0, nrow = S, ncol = 2)
for(s in 1:S){
  x_new[s,] <- mvrnorm(1, mu = THETA[s,], Sigma = matrix(SIGMA[s,], 2, 2))
}

bivn.kde <- kde2d(x_new[,1], x_new[,2], n = 200)
contour(bivn.kde, nlevels = 50)
```




```
mean(x_new[,1] > x_new[,2])
```

```
## [1] 0.7217
```

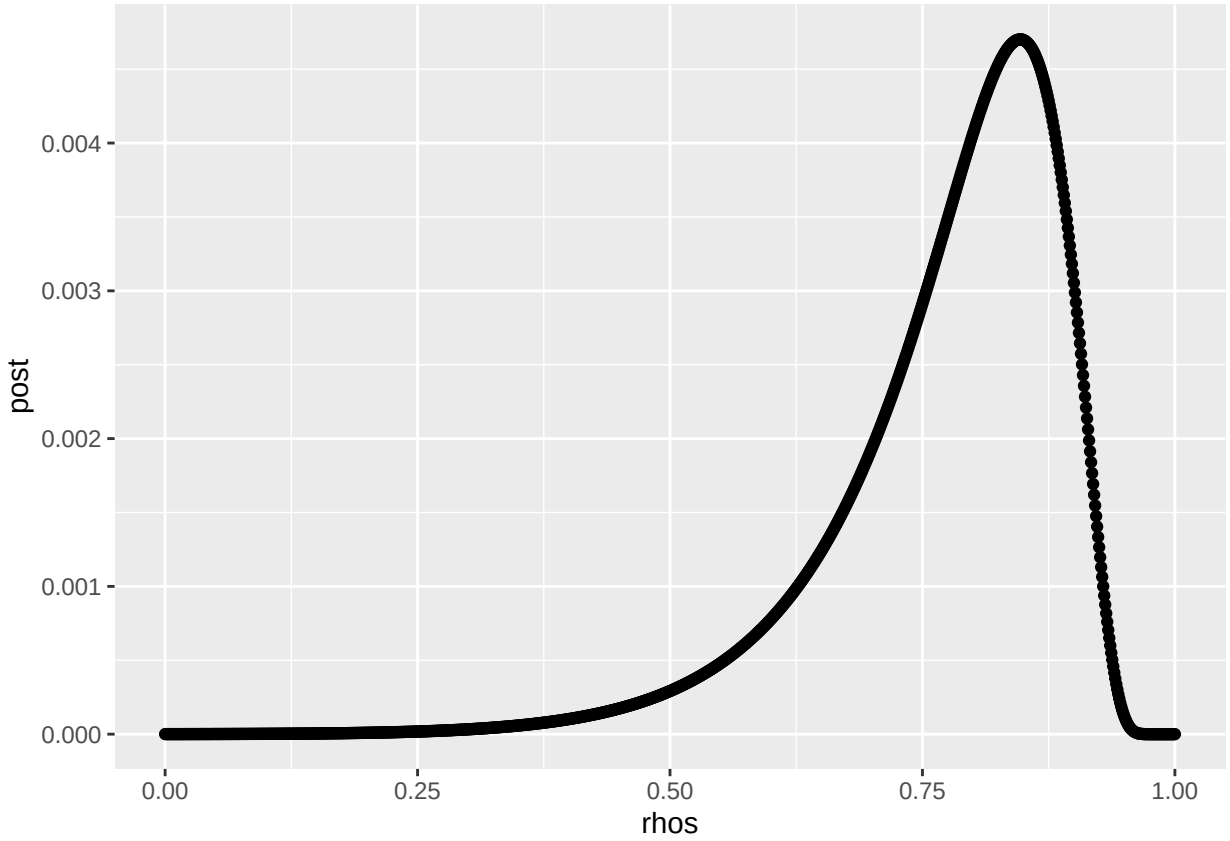
The posterior predictive distribution is much more spread than the posterior distribution of mean parameters.

Question 3

a)

```
x <- c(-3.3, -2.6, 0.1, -0.2, -1.1, -1.5, 2.7, 1.5, 2.0, 1.9, -0.4, -0.3)
x <- matrix(x, ncol = 2, byrow = T)

mu <- c(0, 0)

rhos <- seq(0, 1, by = 0.001)
post <- map_dbl(rhos, ~ prod(mvtnorm::dmvnorm(x, mu, sigma =  matrix(c(1, .x, .x, 1), nrow = 2))))
post <- post / sum(post)
qplot(rhos, post)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```




```
# Mean
sum(rhos * post)
```

```
## [1] 0.7754083
```

```
# Median
rhos[which(cumsum(post) > 0.5)[1]]
```

```
## [1] 0.8
```

```
# Mode
rhos[which.max(post)]
```

```
## [1] 0.847
```

```
# CI
rhos[which(cumsum(post) > 0.025)[1]]
```

```
## [1] 0.493
```

```
rhos[which(cumsum(post) > 0.975)[1]]
```

```
## [1] 0.918
```

Question 4

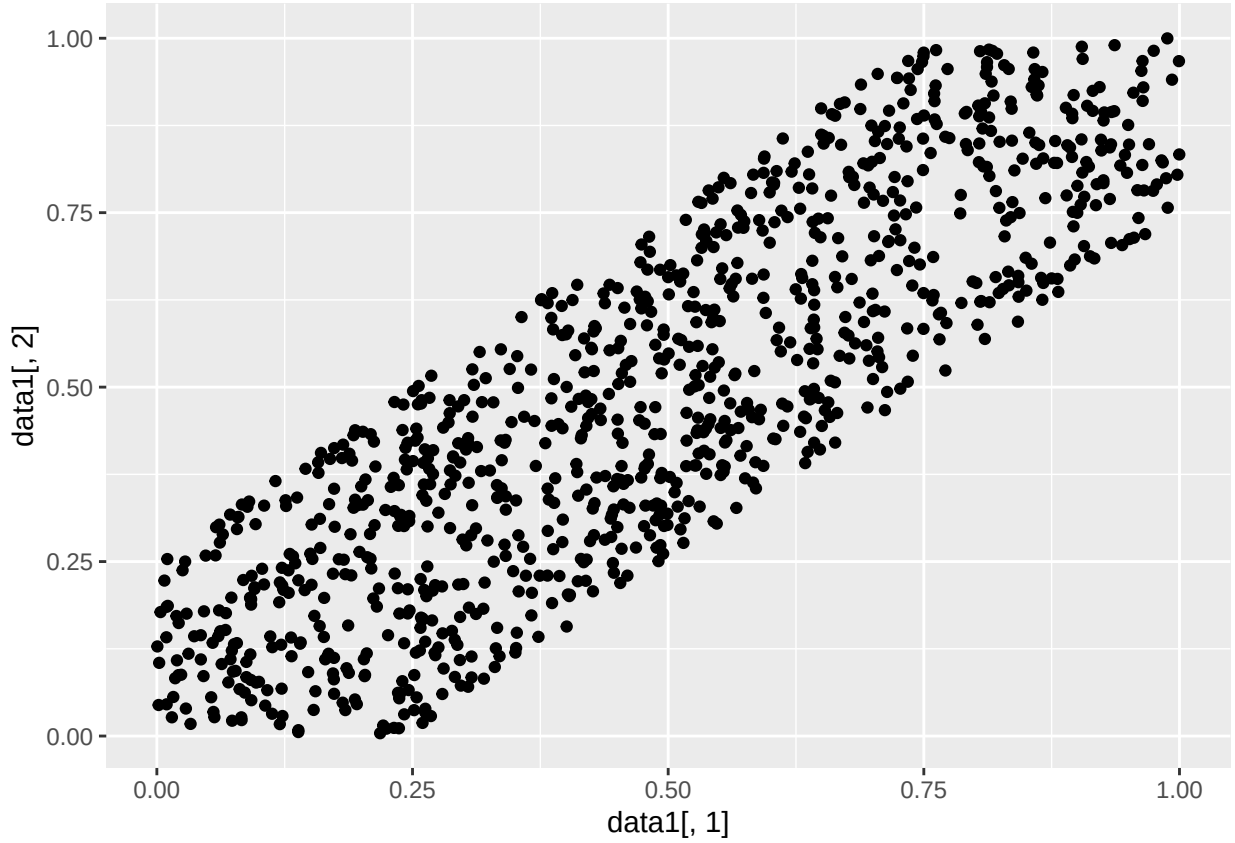
b)

```
q4_mcmc_xy <- function(c, S = 1000, init = 0.5){
  stopifnot(c > 0)
```

```
  X <- Y <- rep(0, S)
  X[1] <- init
  for(i in 1:S){
    if(i > 1){
      X[i] <- runif(1, max(0, Y[i - 1] - c), min(Y[i - 1] + c, 1))
    }
    Y[i] <- runif(1, max(0, X[i] - c), min(X[i] + c, 1))
  }
  return(cbind(X, Y))
}

data1 <- q4_mcmc_xy(0.25)
data2 <- q4_mcmc_xy(0.05)
data3 <- q4_mcmc_xy(0.02)
```

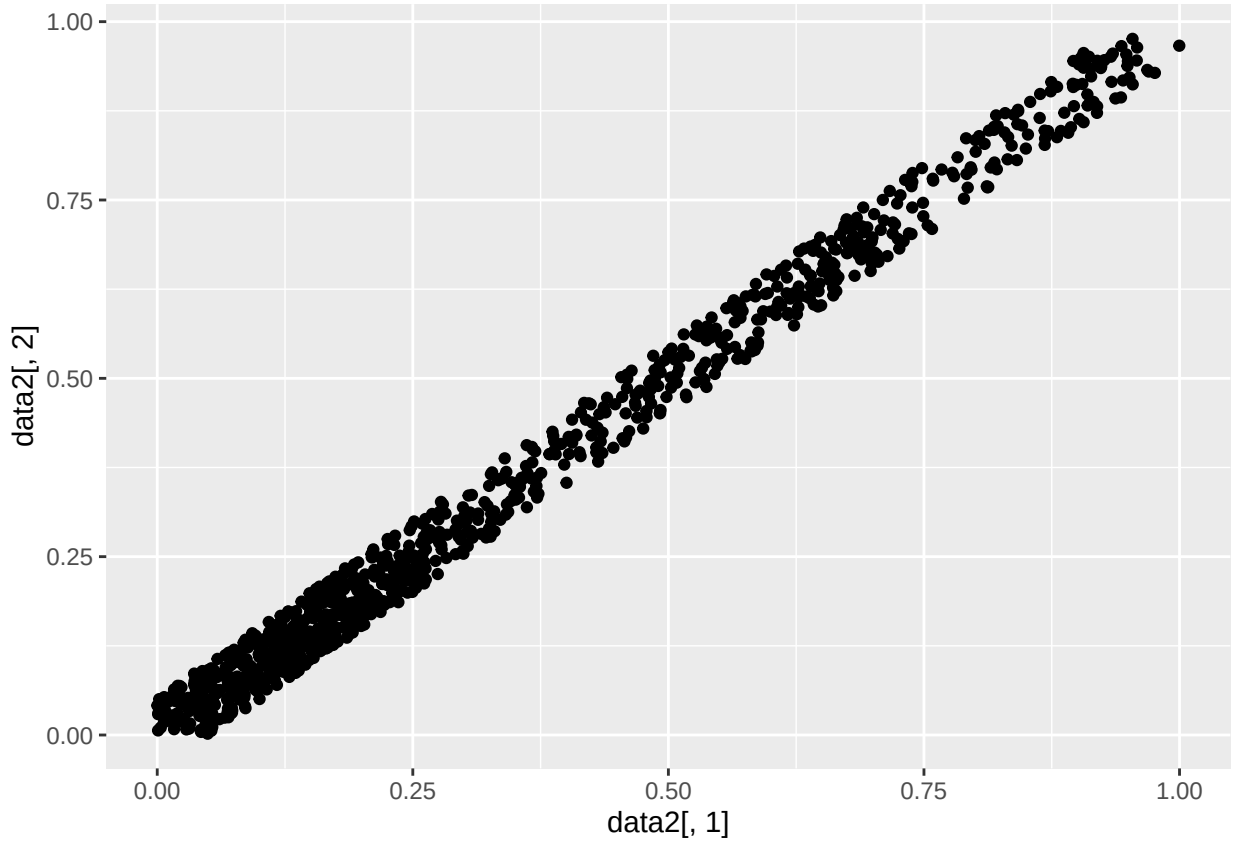
c)

```
par(mfrow = c(1, 3))
ts.plot(data1[,1])
ts.plot(data1[,2])
qplot(data1[,1], data1[,2])
```

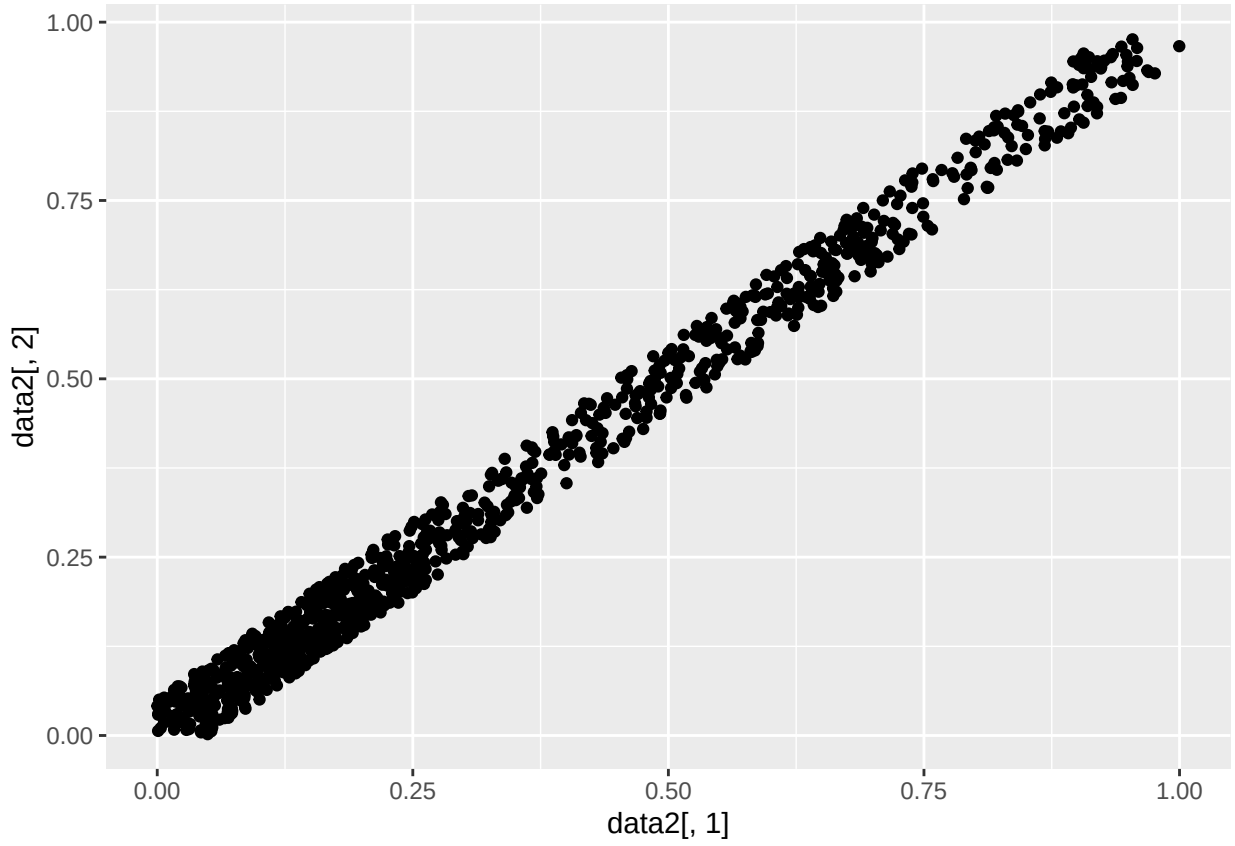



```
par(mfrow = c(1, 3))
ts.plot(data2[,1])
ts.plot(data2[,2])
```

```
qplot(data2[,1], data2[,2])
```




```
par(mfrow = c(1, 3))
ts.plot(data1[,1])
ts.plot(data1[,2])
qplot(data2[,1], data2[,2])
```



d)

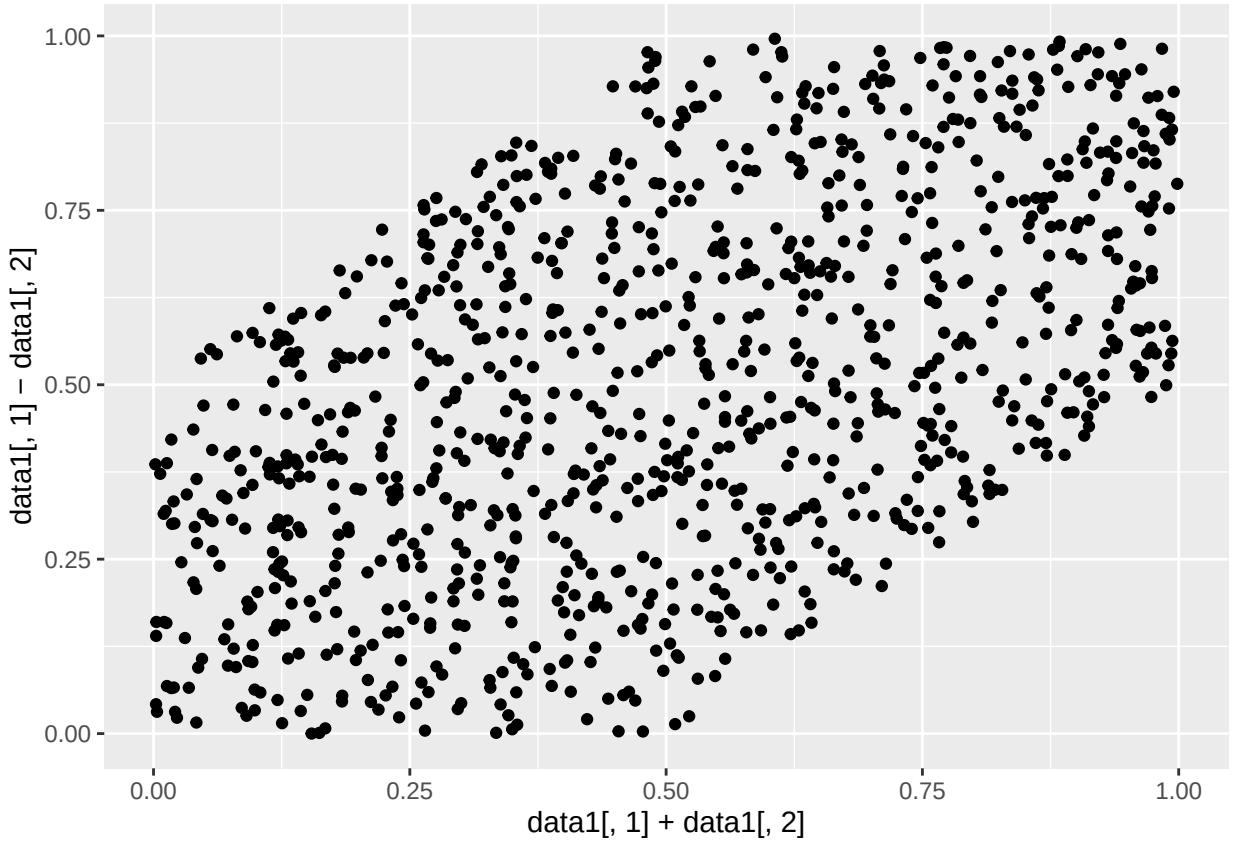
As c get smaller, X and Y become more and more correlated. $c = 1$ means X and Y are independent.

e)

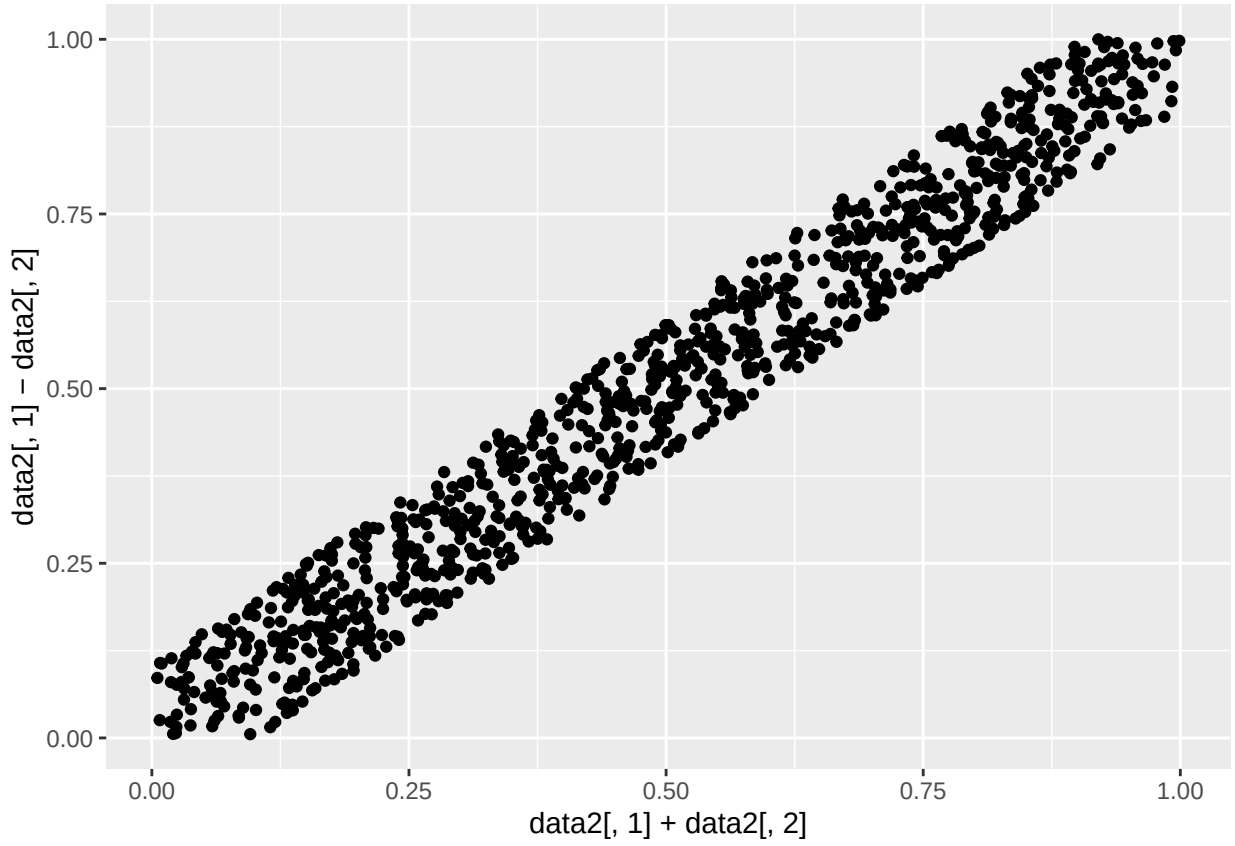
```
q4_mcmc_uv <- function(c, S = 1000, init = 0.5){
  stopifnot(c > 0)
  U <- V <- rep(0, S)
  U[1] <- init
  for(i in 1:S){
    if(i > 1){
      U[i] <- runif(1, abs(V[i-1]), 1 - abs(V[i-1]))
    }
    V[i] <- runif(1, -min(c, U[i], 1 - U[i]), min(c, U[i], 1 - U[i]))
  }
  return(cbind(U, V))
}

data1 <- q4_mcmc_uv(0.25)
data2 <- q4_mcmc_uv(0.05)
data3 <- q4_mcmc_uv(0.02)
```

```
par(mfrow = c(2, 2))
ts.plot(data1[,1])
ts.plot(data1[,2])
```

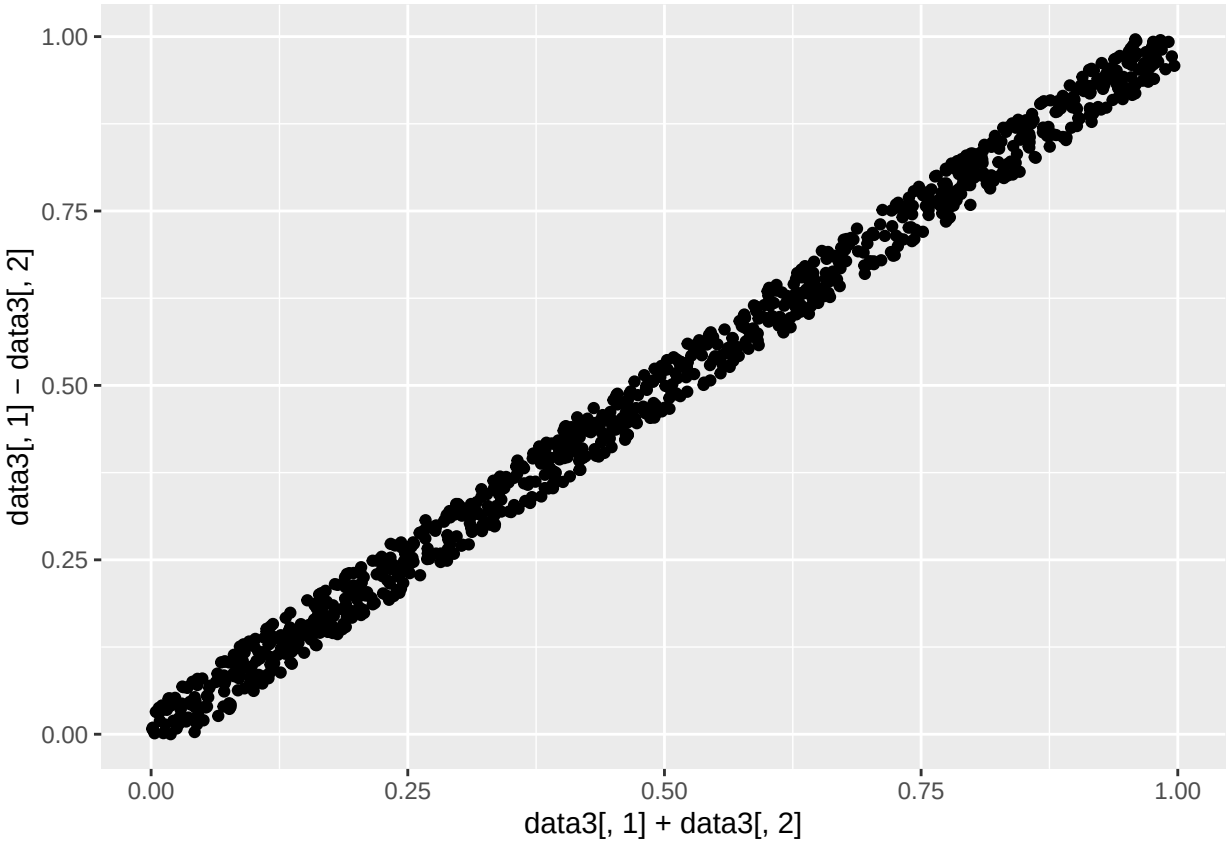
```
qplot(data1[,1], data1[,2])
qplot(data1[,1] + data1[,2], data1[,1] - data1[,2])
```




```
par(mfrow = c(2, 2))
ts.plot(data2[,1])
ts.plot(data2[,2])
qplot(data2[,1], data2[,2])
qplot(data2[,1] + data2[,2], data2[,1] - data2[,2])
```




```
par(mfrow = c(2, 2))
ts.plot(data1[,1])
ts.plot(data1[,2])
qplot(data2[,1], data2[,2])
qplot(data3[,1] + data3[,2], data3[,1] - data3[,2])
```



As c get smaller, U and V become more and more uncorrelated.