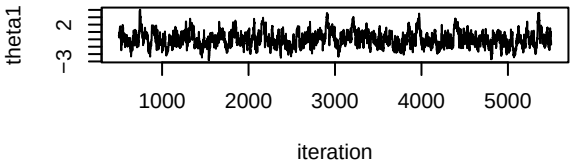
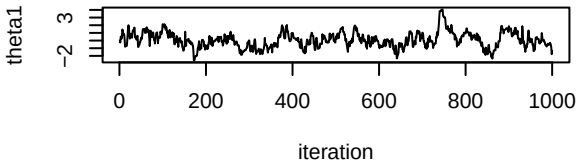
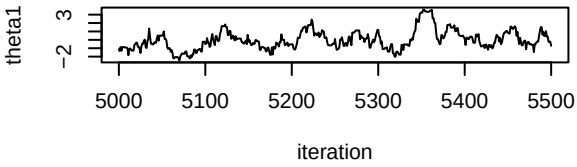
Assignment5

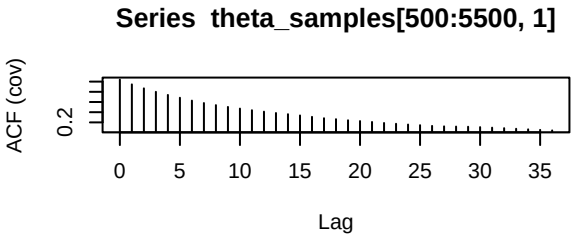
```r
library("mvtnorm")
###Q1.(1)
##Metropolis algorithm
##Bivariate normal distribution p( |y) = N2(0, Σ2×2),where Σ2×2 is the variance-
##covariance matrix with variances equal to 1 ## and correlation equal to 0.8.

### The step for Metropolis algorithm
## Step1: set up theta vector and get init. value
## Step2: (a) Sample theta_star from jumping dist.
##        (b): Calculate ratio of posterior densities.
##        (c): Accept or reject proposed value.

### Set the jumping/proposal distribution is a bivariate normal distribution,
##  centered at the current iteration
##  But with variance-covariance matrix scaled by 0.3(jump size).

## Set a starting value theta^(0)=(0,0), which is the current value.

## Then, for t=1,2,...5500

## Sample a theta^* from proposal distribution

## Calculate the acceptance probability which is

## r(theta new, theta t-1)= Posterior probability of theta new/Posterior probability of theta t-1

## r(theta new, theta t-1)= N2(theta^*, Σ2×2)/N2(theta^t-1, Σ2×2)

##  if the posterior probability is great for the new value of theta, r>1, we will
##  accept the new value of theta.

##  theta t=theta*

##  if the posterior probability is great for the previous value of theta, r<1,
##  we set theta t=theta0 with
##  probability p(theta*|y)/p(theta^(t)|y), if jump, then theta^t=theta*,else theta1=theta t-1

##  We will drop the first 500 accepted samples(assume the Markov Chain does not converge yet.)
##  We will use the last 5000 accepted samples for the value of theta=(theta1,theta2)

###(2) Parameters of a normal distribution used as target distribution

## Yes,the Markov Chain appear to converged.
## For the trace plot at 5000:5500 iterations, there is no clear upward/downward trend
## we conclude the chain has converged.
## The convergence is better for theta2 than for theta1.
```

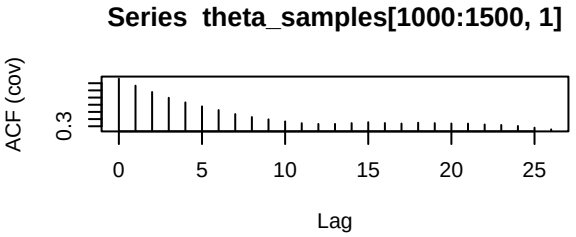
```r
## With the zoom in of both trace and afc plot of theta1 and theta2

## We could see the correlation at 1000:1500 iteration is less than the correlation
## at 5000:5500 iterations.

## There is no visual evidence of high autocorrelation.But somehow bit correlated
## The autocorrelation exponentially decrease with the iterations go on.
r <- 0.8*0.3
S <- diag(2)*0.3
S[1, 2] <- r
S[2, 1] <- r
t <- 0.8
SS <- diag(2)
SS[1, 2] <- t
SS[2, 1] <- t
set.seed(460)
metrop_alg <- function(num_iter){
  theta<-matrix(0,nrow=num_iter,ncol=2 )
  theta[1,1]=0
  theta[1,2]=0
for(i in 2:num_iter)
{

  # Step 2(a): sample theta_star from jumping dist.
  theta_star<-rmvnorm(1,mean=theta[i-1,],S)
  # Step 2(b): Calculate ratio of posterior densities
  post_curr <-theta[i-1,]
  post_prop <-theta_star
  r<-(dmvnorm(post_prop,mean=c(0,0),SS))/(dmvnorm(post_curr,mean=c(0,0),SS))

  if(r>runif(1,min=0,max=1))        ### Actually I am bit confused with why are
  {                                 ##  we compare r with a random draw from uniform(0,1)
    theta[i,]<-post_prop            ## Is this just means we are accept theta^* with
  }                                 ## r probability?
  else
  {
    theta[i,]<-post_curr
  }
}
  return(theta)
}

theta_samples <- metrop_alg(5500)

# make a "trace plot" of the draws of theta
par(mfrow=c(3,2))
plot(x=c(500:5500),y=theta_samples[500:5500,1],type="l",xlab="iteration",ylab="theta1")
plot(x=c(1:1000),y=theta_samples[1:1000,1],type="l",xlab="iteration",ylab="theta1")
plot(x=c(5000:5500),y=theta_samples[5000:5500,1],type="l",xlab="iteration",ylab="theta1")
acf(theta_samples[500:5500,1], type = "covariance")
acf(theta_samples[1000:1500,1], type = "covariance")
acf(theta_samples[5000:5500,1], type = "covariance")
```

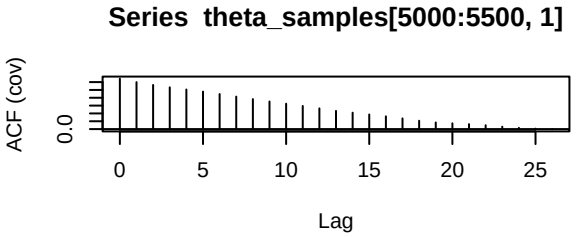




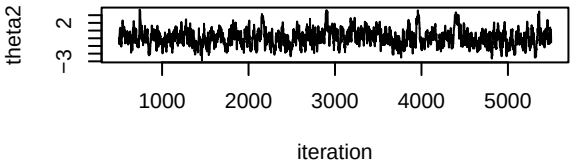


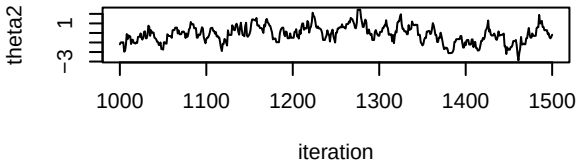


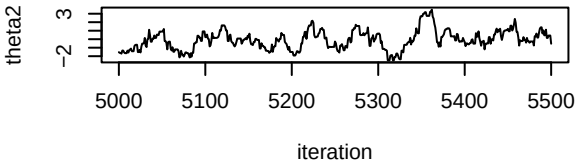


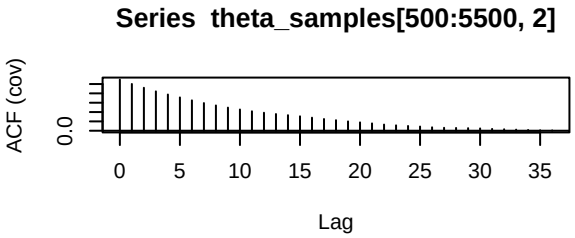


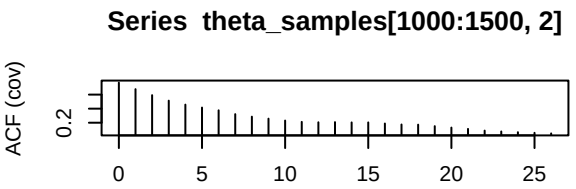

```
par(mfrow=c(3,2))
plot(x=c(500:5500),y=theta_samples[500:5500,2],type="l",xlab="iteration",ylab="theta2")
plot(x=c(1000:1500),y=theta_samples[1000:1500,2],type="l",xlab="iteration",ylab="theta2")
plot(x=c(5000:5500),y=theta_samples[5000:5500,2],type="l",xlab="iteration",ylab="theta2")
acf(theta_samples[500:5500,2], type = "covariance")
acf(theta_samples[1000:1500,2], type = "covariance")
acf(theta_samples[5000:5500,2], type = "covariance")

##(3) Yes, the draws appear to match the bivariate normal target density.
library("ggplot2")
```

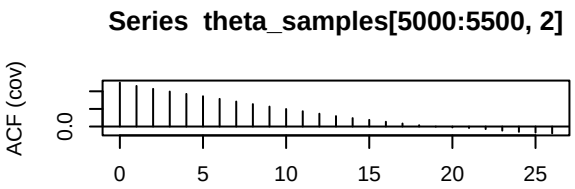




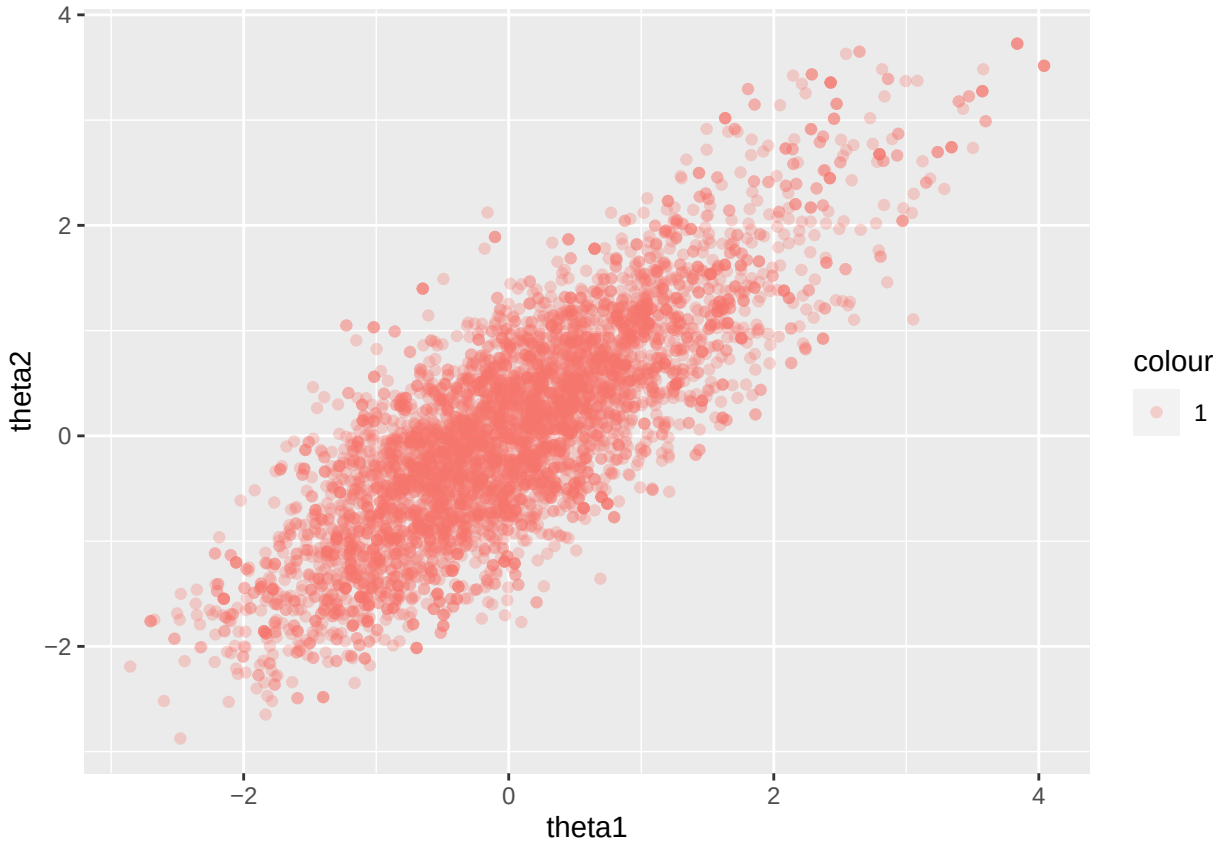









```
theta_samples<-as.data.frame(theta_samples)
colnames(theta_samples)[1] <- "theta1"
colnames(theta_samples)[2] <- "theta2"
ggplot()+geom_point(data =as.data.frame(theta_samples[500:5500,]) ,aes(theta1,theta2, color = '1'), alp
```

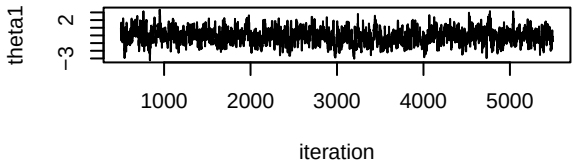


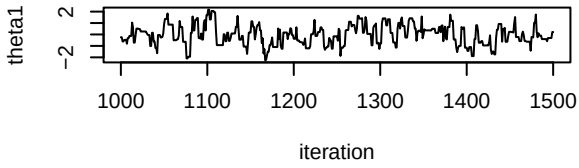

```
##Q2
library("mvtnorm")
##(1).
SSS<-var(theta_samples[500:5500,])


## All four steps from Q1.

## (1) The algorithm step is same as the one states in Q1 (1) with the difference
##     of jumping distribution. In Q1 (1), we have symmetric jumping distribution
##     which has property Jt(theta*|theta^(t-1))=Jt(theta^(t-1)|theta^*).
##     For here, the jumping distribution is N2(0,Σ2×2),where Σ2×2 is the variance-
##      covariance matrix with variances we estimate in Q2(1) with variance function.


##(2).
library("bayestestR")
map_estimate(theta_samples[500:5500,],precision = 2^10, method = "kernel")
```

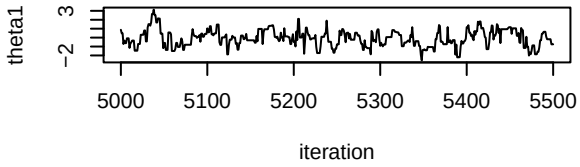
```
## MAP Estimate
##
## Parameter | MAP_Estimate
## ------------------------
## theta1    |    -4.59e-03
## theta2    |        -0.13
```

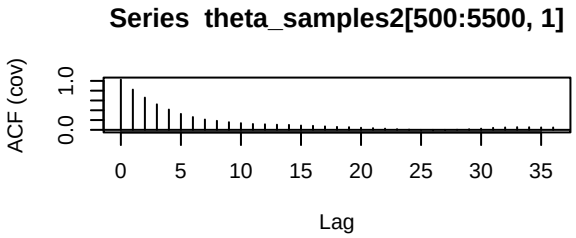
```
t <- 0.8
SS <- diag(2)
```

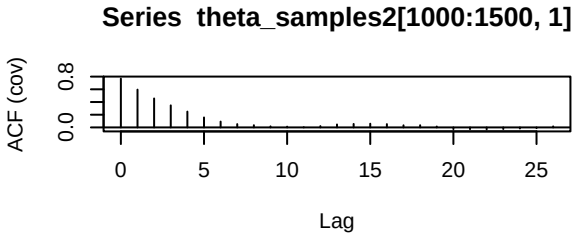
```r
SS[1, 2] <- t
SS[2, 1] <- t

metrop_alg <- function(num_iter){
  theta<-matrix(0,nrow=num_iter,ncol=2 )
  theta[1,1]=0 ## I used 0 as suggested in tutorial.
  theta[1,2]=0
for(i in 2:num_iter)
{

  # Step 2(a): sample theta_star from jumping dist.
  theta_star<-rmvnorm(1,mean=theta[i-1,],SSS) ##make it jumping distribution
  # Step 2(b): Calculate ratio of posterior densities
  post_curr <-theta[i-1,]
  post_prop <-theta_star
  r<-(dmvnorm(post_prop,mean=c(0,0),SS))/(dmvnorm(post_curr,mean=c(0,0),SS))

  if(r>runif(1,min=0,max=1))
  {
    theta[i,]<-post_prop
  }
  else
  {
    theta[i,]<-post_curr
  }
}
  return(theta)
}

theta_samples2 <- metrop_alg(5500)
# make a "trace plot" of the draws of theta

## Yes,the Markov Chain appear to converged.
## For the trace plot at 5000:5500 iterations, there is no clear upward/downward trend
## so we conclude the chain has converged.
## The convergence is better for theta2 than for theta1. (same as previous sampling)

## With the zoom in of both trace and afc plot of theta1 and theta2

## We could see the correlation at 1000:1500 iteration is less than the correlation
## at 5000:5500 iterations.

## There is no visual evidence of high autocorrelation.But somehow bit correlated
## The autocorrelation exponentially decrease with the iterations go on.

par(mfrow=c(3,2))
plot(x=c(500:5500),y=theta_samples2[500:5500,1],type="l",xlab="iteration",ylab="theta1")
plot(x=c(1000:1500),y=theta_samples2[1000:1500,1],type="l",xlab="iteration",ylab="theta1")
plot(x=c(5000:5500),y=theta_samples2[5000:5500,1],type="l",xlab="iteration",ylab="theta1")
acf(theta_samples2[500:5500,1], type = "covariance")
acf(theta_samples2[1000:1500,1], type = "covariance")
acf(theta_samples2[5000:5500,1], type = "covariance")
```

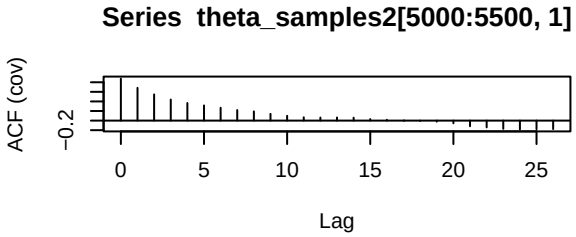




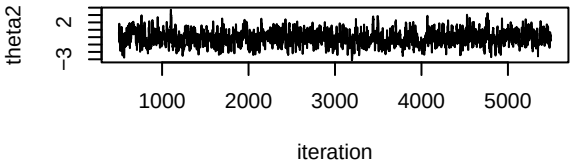


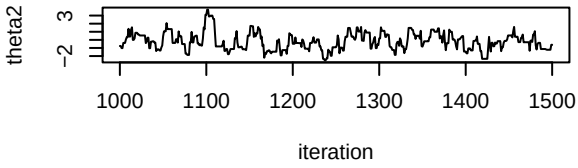


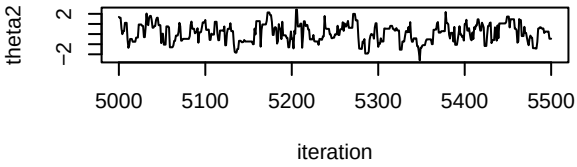


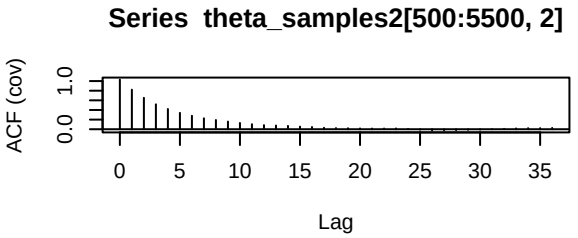


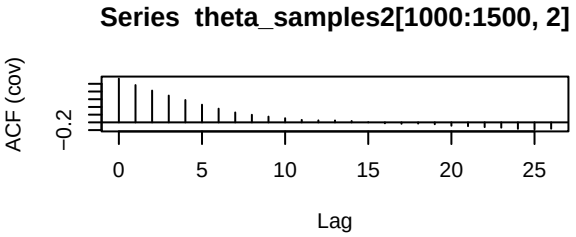

```r
par(mfrow=c(3,2))
plot(x=c(500:5500),y=theta_samples2[500:5500,2],type="l",xlab="iteration",ylab="theta2")
plot(x=c(1000:1500),y=theta_samples2[1000:1500,2],type="l",xlab="iteration",ylab="theta2")
plot(x=c(5000:5500),y=theta_samples2[5000:5500,2],type="l",xlab="iteration",ylab="theta2")
acf(theta_samples2[500:5500,2], type = "covariance")
acf(theta_samples2[1000:1500,2], type = "covariance")
acf(theta_samples2[5000:5500,2], type = "covariance")
```

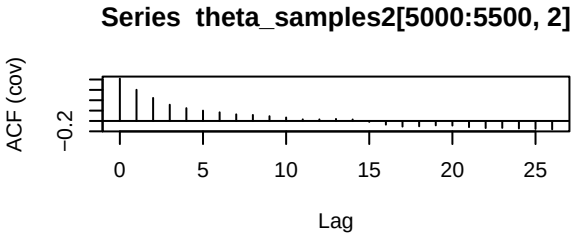




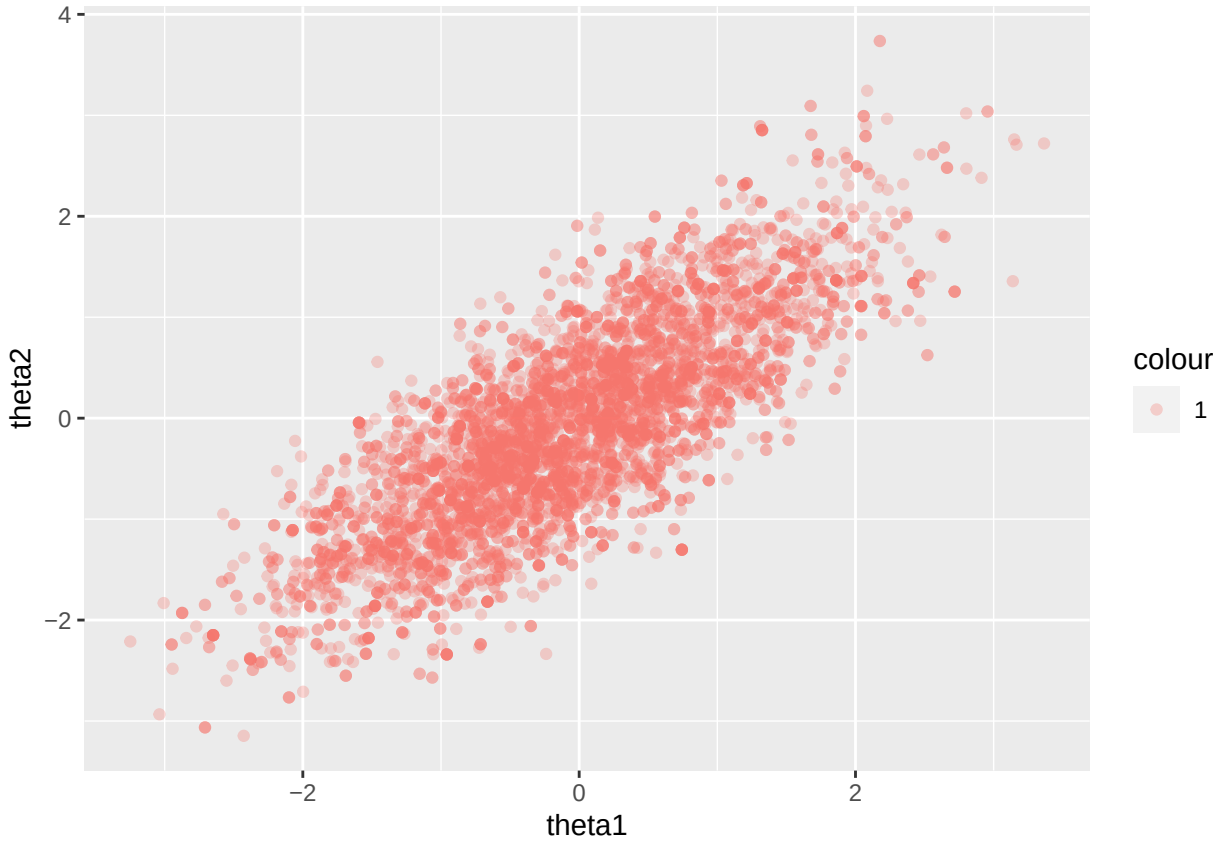








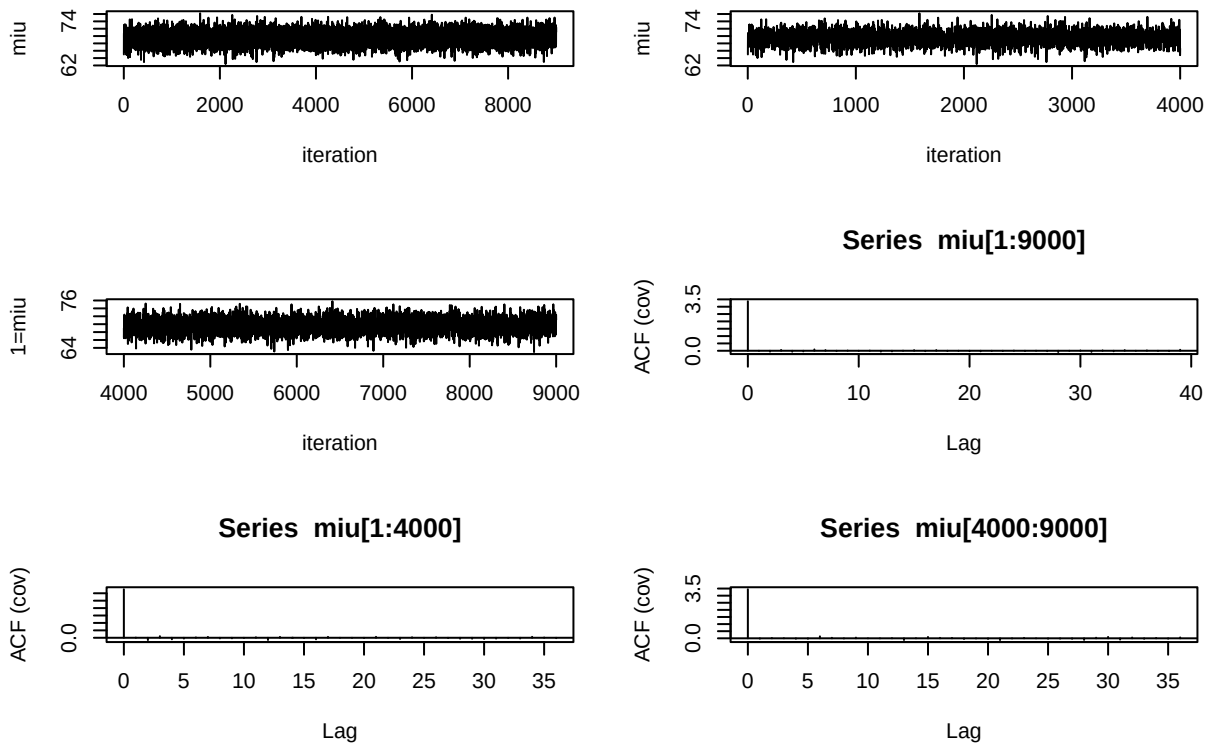

```
##(3) Yes, the draws appear to match the bivariate normal target density.
library("ggplot2")
theta_samples2<-as.data.frame(theta_samples2)
colnames(theta_samples2)[1] <- "theta1"
colnames(theta_samples2)[2] <- "theta2"
ggplot()+geom_point(data =as.data.frame(theta_samples2[500:5500,]) ,aes(theta1,theta2, color = '1'), al
```



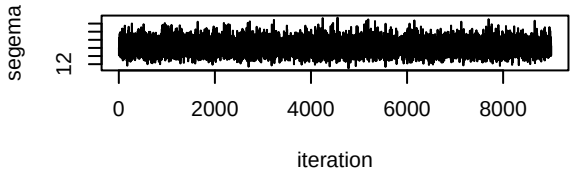

```
##Q3

##(1) At iteration t, the Gibbs sampler draws Thetaj^t~p(thetaj|theta-j^(t-1),y),
##    for j=1,...,d, where theta-j^(t-1) represents all the components of theta,
##    expect for thetaj, at their current value:
##    Theta-j^(t-1)=(theta1^(t),..., thetaj-1^(t),thetaj+1^(t-1),...,thetad^(t-1))

## Based on the lecture note, we know that miu|,y ~
## N(((miu0/Tao^2)+(n*y-bar/segema^2))/ (1/Tao^2)+(n/segema^2)), 1/(1/tao^2+n/segema^2)))

##  segema^2| miu,y ~ Inv-X^2 (v+n, ((v*segema^2+nv)/(v+n))), v=1/n sum(yi-miu^t)^2

##  Initializing (miu^(0),segema^2(0))=(65,16)

##  For each iteration/update,
##  miu^(t) ~ miu| segema^(t-1),y
##  segema^2(t) ~segema^2| miu^t, y

## and we have data y1 = 70, y2=75 and y3=72.

## Code for sampling
library("invgamma") ## I will use rinvchisq in this package to sample
##                     from inverse chi-square distribution
y1<-70
y2<-75
y3<-72
ybar<-mean(c(70,75,72))
```

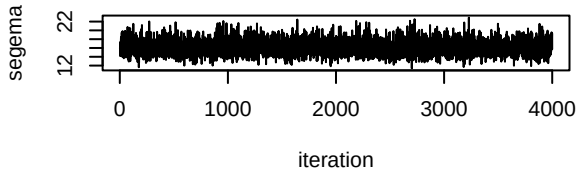
```r
ysum<-sum(c(70,75,72))
rho<-0.8
N_iter<-9000


miu<-rep(0,N_iter)
segema<-rep(0,N_iter)
X<-rep(0,N_iter)
miu[1]<-65
segema[1]<-16

for(t in 2:N_iter){
   miu1<-((65/9+ysum/segema[t-1])/(1/9+3/segema[t-1]))
   segema1<-(1/(1/9+3/segema[t-1]))
   miu[t]<-rnorm(1,miu1,sqrt(segema1))

   VV<-function(x,x0){
        return(sum((x-x0)^2))}

   V<-VV(c(70,75,72),miu[t])

   X<-rchisq(1,178)
   segema2<-(175*16+V)/(178)
   segema[t]<-((178*segema2)/X)


}

### Yes, the chain mixing well, there is no visual evidence of high autocorrelation.
# make a "trace plot" of the draws of theta

## Yes,the Markov Chain appear to converged after 4000 burn-in samples.(because of the
## low correlation)
## I will used first 4000 as burn-in data.
## For the trace plot at 4000:9000 iterations, there is no clear upward/downward trend
## so we conclude the chain has converged.
## The convergence is better for segema than for miu.

## With the zoom in of both trace and afc plot of miu and segema

## We could see the correlation at 4000:9000 iterations and at 1:4000 iterations
## all have very low autocorrelation and less correlation at 4000:9000 iterations.


## There is no visual evidence of high autocorrelation.But somehow bit correlated
## The autocorrelation drop down with the iterations go on.

par(mfrow=c(3,2))
plot(x=c(1:9000),y=miu[1:9000],type="l",xlab="iteration",ylab="miu")
plot(x=c(1:4000),y=miu[1:4000],type="l",xlab="iteration",ylab="miu")
plot(x=c(4000:9000),y=miu[4000:9000],type="l",xlab="iteration",ylab="1=miu")
acf(miu[1:9000], type = "covariance")
acf(miu[1:4000], type = "covariance")
```

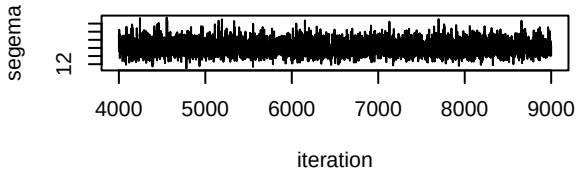
```
acf(miu[4000:9000], type = "covariance")
```

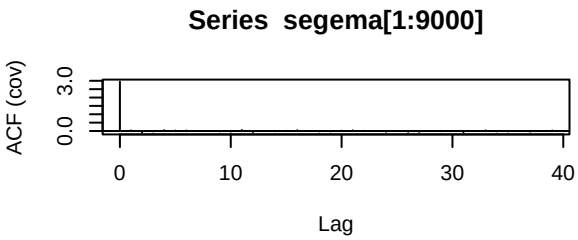


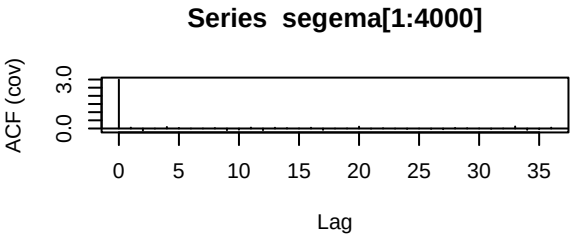

```
par(mfrow=c(3,2))
plot(x=c(1:9000),y=segema[1:9000],type="l",xlab="iteration",ylab="segema")
plot(x=c(1:4000),y=segema[1:4000],type="l",xlab="iteration",ylab="segema")
plot(x=c(4000:9000),y=segema[4000:9000],type="l",xlab="iteration",ylab="segema")
acf(segema[1:9000], type = "covariance")
acf(segema[1:4000], type = "covariance")
acf(segema[4000:9000], type = "covariance")
```

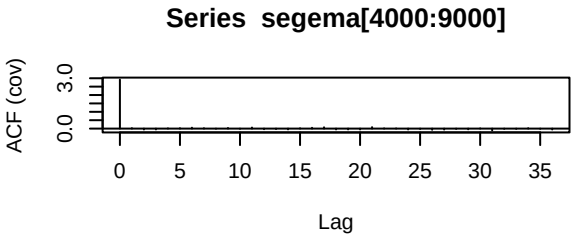




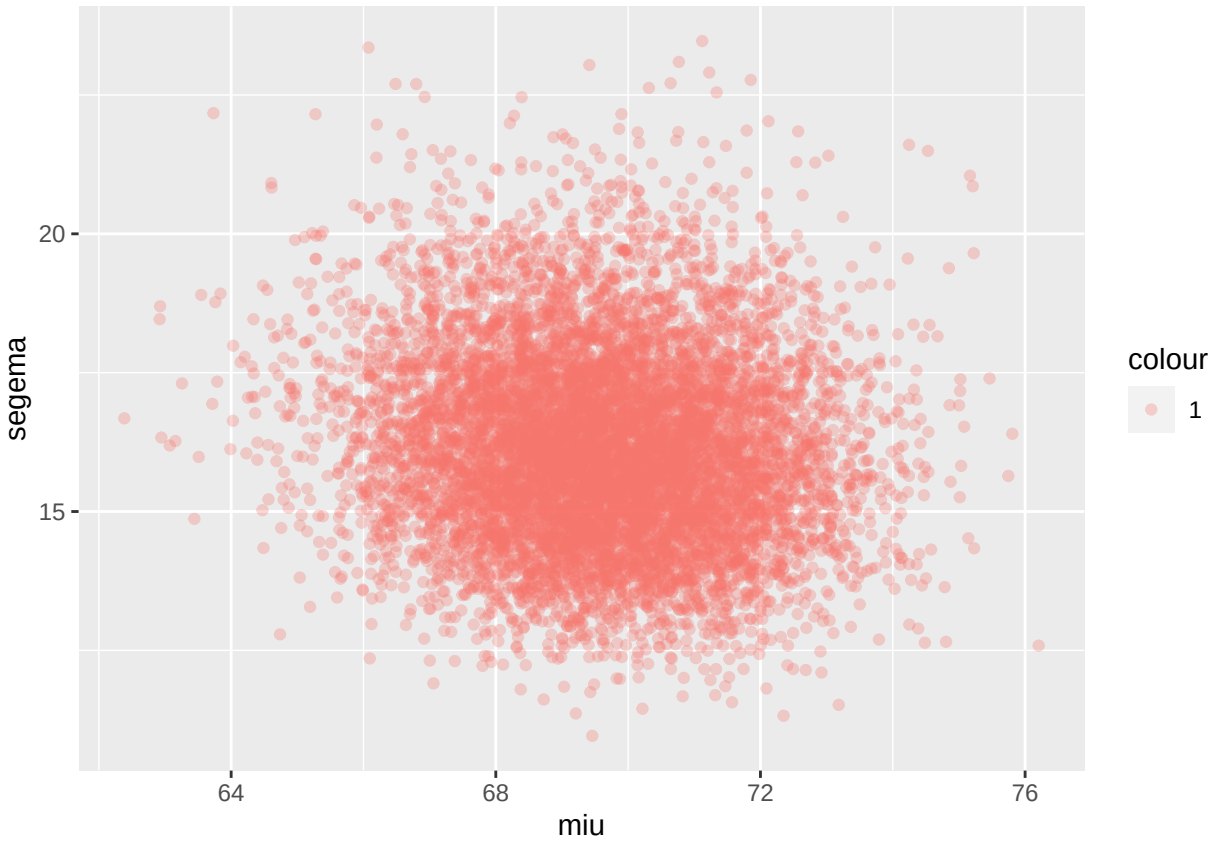









```r
### (3) two-demensional plot for miu and segema.
dataa<-as.data.frame(cbind(miu,segema))
library("ggplot2")

colnames(dataa)[1] <- "miu"
colnames(dataa)[2] <- "segema"
ggplot()+geom_point(data =dataa ,aes(miu,segema, color = '1'), alpha = 0.3)
```


segema
20
15
64
68
72
76
miu
colour
1