Lab4

Jun Rao

9/17/2019

## MCMC sampling in R

# Task 1 (25 points)

*In the last lab, I showed that we can create an exact posterior estimate of the mean () of a normally-distributed variate , if we have multiple data observations, and if we have a fixed (known) variance. Given that is a vector of data observations, then the posterior becomes:*

*Here is the average of our data observations, and again, is the number of data observations.*

*Before, we calculated our posterior by creating a sequence of possible values and manually calculating the posterior of each value. As I discussed in lecture, this becomes exceedingly inefficient if we have many parameters in our model. Therefore, we often use MCMC methods to randomly sample from our unknown posterior distribution.*

*On the next page, I have provided code for estimating from a vector of normally distributed data, with fixed variance, using the MH-MCMC algorithm.*

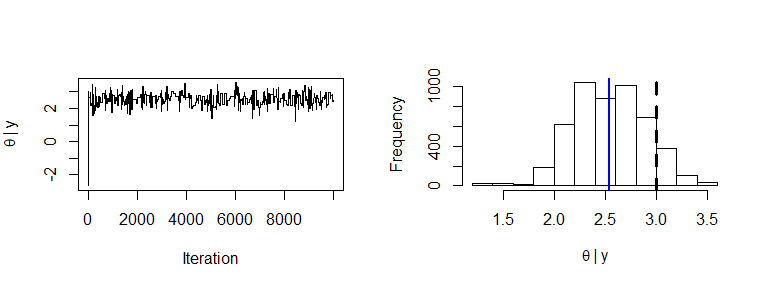
*Complete the following:*

*The MH-MCMC sampler for a model with a single parameter, assuming .*

#-----------------------------------  
#-----------------------------------  
#The mu and sd for the real distribution  
mu\_true = 3  
sd\_fixed = 2  
  
  
#The mu and sd for the prior distribution  
mu\_prior = 5  
sd\_prior = 10  
  
#The mu and sd for the proposal distribution  
mu\_prop = 5  
sd\_prop = sd\_prior \* 1.1  
  
#-----------------------------------  
#-----------------------------------  
#Generate 30 observations from real distributation  
n\_sample = 30  
y\_test = rnorm(n\_sample, mu\_true, sd\_fixed)  
  
#-----------------------------------  
#-----------------------------------  
  
n\_iter = 10000  
chosen\_theta = NULL  
  
for(i in 1:n\_iter){  
   
 # Initializing the old\_theta  
 if(i == 1){  
 old\_theta = rnorm(1, mu\_prop, sd\_prop)  
 }  
   
 # for each loop, it will generate a new theta  
 new\_theta = rnorm(1, mu\_prop, sd\_prop)  
   
 #calculate the probability based on the old theta and new theta which follow proposal distribution  
 old\_prop\_adj = dnorm(old\_theta, mu\_prop, sd\_prop, log = TRUE)  
 new\_prop\_adj = dnorm(new\_theta, mu\_prop, sd\_prop, log = TRUE)  
   
 #calculate the probability based on the old theta and new theta which follow prior distribution  
 old\_prior = dnorm(old\_theta, mu\_prior, sd\_prior, log = TRUE)  
 new\_prior = dnorm(new\_theta, mu\_prior, sd\_prior, log = TRUE)  
   
 #Calculate the likelihood based on the old theta and new theta  
 old\_lik = sum(dnorm(y\_test, old\_theta, sd\_fixed, log = TRUE))  
 new\_lik = sum(dnorm(y\_test, new\_theta, sd\_fixed, log = TRUE))  
   
 # Calculate the posterior  
 old\_post = old\_prior + old\_lik  
 new\_post = new\_prior + new\_lik  
   
 # calculate log\_ratio and ratio  
 log\_ratio = (new\_post - new\_prop\_adj) - (old\_post - old\_prop\_adj)  
 ratio = exp(log\_ratio)  
   
 # make a decision   
 if(ratio > 1){  
 chosen\_theta[i] = new\_theta  
 }else{  
   
 rand = runif(1, min = 0, max = 1)  
   
 if(ratio >= rand){  
 chosen\_theta[i] = new\_theta  
 }else{  
 chosen\_theta[i] = old\_theta  
 }  
   
 }  
   
 #assign the new value to the old\_theta  
 old\_theta = chosen\_theta[i]  
   
}

*Here are some example plots of the output of a single chain:*

par(mfrow = c(1, 2))  
  
# Plot the full trace:  
plot(chosen\_theta ~ c(1:n\_iter), type = "l",  
 xlab = "Iteration", ylab = expression(theta~"|"~y))  
  
# Discard the burn-in  
# Plot the sample of the estimated posterior of \theta  
n\_burn = n\_iter / 2  
hist(chosen\_theta[n\_burn:n\_iter], main = "",  
 xlab = expression(theta~"|"~y), ylab = "Frequency")  
  
# Add the mean estimate  
mu\_est = mean(chosen\_theta[n\_burn:n\_iter])  
abline(v=mu\_est, col = "blue", lwd = 2)  
  
# Add the true mean  
abline(v = mu\_true, col = "black", lwd = 3, lty = 2)



par(mfrow = c(1,1))

*Run multiple MCMC chains, remove the burn-in () from each chain, and store the resulting output in an R object (e.g., a data frame). Then, create a function to calculate the Gelman-Ruben statistic, based on the equations in class. Use this function to determine if your chains have converged (15 points).*

library(matrixStats)  
n\_iter = 10000  
  
 #-----------------------------------  
#-----------------------------------  
#Generate 30 observations from real distributation  
n\_sample = 30  
y\_test = rnorm(n\_sample, mu\_true, sd\_fixed)  
  
theta <- function(){  
 #-----------------------------------  
 #-----------------------------------  
 #The mu and sd for the real distribution  
 mu\_true = 3  
 sd\_fixed = 2  
  
 #The mu and sd for the prior distribution  
 mu\_prior = 5  
 sd\_prior = 10  
  
 #The mu and sd for the proposal distribution  
 mu\_prop = 5  
 sd\_prop = sd\_prior \* 1.1  
  
 #-----------------------------------  
 #-----------------------------------  
  
   
 chosen\_theta = NULL  
 result = NULL  
  
 for(i in 1:n\_iter){  
   
 # Initializing the old\_theta  
 if(i == 1){  
 old\_theta = rnorm(1, mu\_prop, sd\_prop)  
 }  
   
 # for each loop, it will generate a new theta  
 new\_theta = rnorm(1, mu\_prop, sd\_prop)  
   
 #calculate the probability based on the old theta and new theta which follow proposal distribution  
 old\_prop\_adj = dnorm(old\_theta, mu\_prop, sd\_prop, log = TRUE)  
 new\_prop\_adj = dnorm(new\_theta, mu\_prop, sd\_prop, log = TRUE)  
   
 #calculate the probability based on the old theta and new theta which follow prior distribution  
 old\_prior = dnorm(old\_theta, mu\_prior, sd\_prior, log = TRUE)  
 new\_prior = dnorm(new\_theta, mu\_prior, sd\_prior, log = TRUE)  
   
 #Calculate the likelihood based on the old theta and new theta  
 old\_lik = sum(dnorm(y\_test, old\_theta, sd\_fixed, log = TRUE))  
 new\_lik = sum(dnorm(y\_test, new\_theta, sd\_fixed, log = TRUE))  
   
 # Calculate the posterior  
 old\_post = old\_prior + old\_lik  
 new\_post = new\_prior + new\_lik  
   
 # calculate log\_ratio and ratio  
 log\_ratio = (new\_post - new\_prop\_adj) - (old\_post - old\_prop\_adj)  
 ratio = exp(log\_ratio)  
   
 # make a decision   
 if(ratio > 1){  
 chosen\_theta[i] = new\_theta  
 }else{  
   
 rand = runif(1, min = 0, max = 1)  
   
 if(ratio >= rand){  
 chosen\_theta[i] = new\_theta  
 }else{  
 chosen\_theta[i] = old\_theta  
 }  
   
 }  
   
 #assign the new value to the old\_theta  
 old\_theta = chosen\_theta[i]  
 }  
   
 #remove the burn-in (n/2) from each chain  
 result = tail(chosen\_theta,length(chosen\_theta)/2)  
   
 return(result)  
}  
   
  
#Create an Matrix as a objeCt  
object <- matrix(0, 15, ncol = (n\_iter/2))  
for(row in 1:15){  
 result <- theta()  
 object[row, ] <- result  
}  
  
str(object)

## num [1:15, 1:5000] 3.54 3.34 3.13 2.32 3.25 ...

#create a function to calculate the Gelman-Ruben statistic  
GR <- function(M){  
   
 #mean value for each row in the matrix   
 Chain.mean = rowMeans(M)  
 #Variance for each row in the matrix  
 Chain.Var = rowVars(M)  
   
 #Calculte W in our notes  
 W = sum(Chain.Var)/nrow(M)  
 #Calculate Beta  
 sum = 0  
 for (i in 1:nrow(M)) {  
 sum = sum +(Chain.mean[i] - mean(M))\*\*2  
 }  
 Beta = (ncol(M) \* sum)/(nrow(M)-1)  
   
 temp = ((ncol(M)-1)\*W)/ncol(M) + Beta/ncol(M)  
   
 result = sqrt(temp/W)  
   
 return(result)  
}  
   
GR(object)

## [1] 1.004094

***Since the value at here is less than 1.1, hence we can conclude that our chains are converged.***

*Validate that the MCMC sampler correctly approximates the theoretical (i.e., analytically-derived) mean and variance of the posterior distribution of the unknown parameter, . Look at the equation above to help (3 points).*

#the mean of MCMC sampler and the mean variance of the MCMC sampler  
mean(rowMeans(object))

## [1] 3.246702

mean(rowVars(object))

## [1] 0.1339936

#Now, we will calculate the mean and variance based on the equation  
mu = 5  
tau = 10  
sigma = 2  
n=30  
y = mean(y\_test)  
#Based on the equation, I am calculating the mu  
mu.equation = ((mu/tau\*\*2) + (y\*n/sigma\*\*2))/(1/tau\*\*2 + n/sigma\*\*2)  
mu.equation

## [1] 3.247648

#Based on the equation I am calculate the variance  
sigma.equation = (1/((1/tau\*\*2) + (n/sigma\*\*2)))\*\*2  
sigma.equation

## [1] 0.01773047

***We can find at here：***

***the mean in the MCMC sample is 2.684517, and the variance is 0.1357709***

***the mean from the equation is 2.694757, and the variance is 0.1587276***

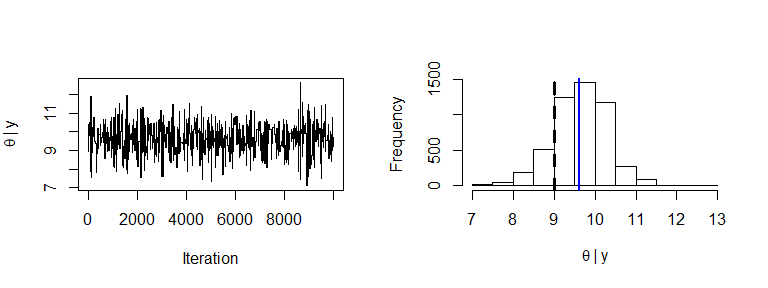
*4.Conduct some synthetic experiments with your MCMC sampler. Specifically, use your sampler with different prior and proposal distributions. How does the prior and proposal affect the MCMC sampler’s efficiency and how quickly it reaches a “good” probability space? Do the same with differently sized data sets. I do not want to see the output of these simulations. Just use the MCMC on your own as a “toy” model. Then, using these anecdotal simulations as support, briefly comment on how the priors, proposals, and data set size influence the MCMC sampler (5 points).*

*The MH-MCMC sampler for a model with a single parameter, assuming .*

#-----------------------------------  
#-----------------------------------  
#The mu and sd for the real distribution  
mu\_true = 9  
sd\_fixed = 4  
  
  
#The mu and sd for the prior distribution  
mu\_prior = 5  
sd\_prior = 10  
  
#The mu and sd for the proposal distribution  
mu\_prop = 5  
sd\_prop = sd\_prior \* 1.1  
  
#-----------------------------------  
#-----------------------------------  
#Generate 30 observations from real distributation  
n\_sample = 30  
y\_test = rnorm(n\_sample, mu\_true, sd\_fixed)  
  
#-----------------------------------  
#-----------------------------------  
  
n\_iter = 10000  
chosen\_theta = NULL  
  
for(i in 1:n\_iter){  
   
 # Initializing the old\_theta  
 if(i == 1){  
 old\_theta = rnorm(1, mu\_prop, sd\_prop)  
 }  
   
 # for each loop, it will generate a new theta  
 new\_theta = rnorm(1, mu\_prop, sd\_prop)  
   
 #calculate the probability based on the old theta and new theta which follow proposal distribution  
 old\_prop\_adj = dnorm(old\_theta, mu\_prop, sd\_prop, log = TRUE)  
 new\_prop\_adj = dnorm(new\_theta, mu\_prop, sd\_prop, log = TRUE)  
   
 #calculate the probability based on the old theta and new theta which follow prior distribution  
 old\_prior = dnorm(old\_theta, mu\_prior, sd\_prior, log = TRUE)  
 new\_prior = dnorm(new\_theta, mu\_prior, sd\_prior, log = TRUE)  
   
 #Calculate the likelihood based on the old theta and new theta  
 old\_lik = sum(dnorm(y\_test, old\_theta, sd\_fixed, log = TRUE))  
 new\_lik = sum(dnorm(y\_test, new\_theta, sd\_fixed, log = TRUE))  
   
 # Calculate the posterior  
 old\_post = old\_prior + old\_lik  
 new\_post = new\_prior + new\_lik  
   
 # calculate log\_ratio and ratio  
 log\_ratio = (new\_post - new\_prop\_adj) - (old\_post - old\_prop\_adj)  
 ratio = exp(log\_ratio)  
   
 # make a decision   
 if(ratio > 1){  
 chosen\_theta[i] = new\_theta  
 }else{  
   
 rand = runif(1, min = 0, max = 1)  
   
 if(ratio >= rand){  
 chosen\_theta[i] = new\_theta  
 }else{  
 chosen\_theta[i] = old\_theta  
 }  
   
 }  
   
 #assign the new value to the old\_theta  
 old\_theta = chosen\_theta[i]  
   
}

Here are some example plots of the output of a single chain:

par(mfrow = c(1, 2))  
  
# Plot the full trace:  
plot(chosen\_theta ~ c(1:n\_iter), type = "l",  
 xlab = "Iteration", ylab = expression(theta~"|"~y))  
  
# Discard the burn-in  
# Plot the sample of the estimated posterior of \theta  
n\_burn = n\_iter / 2  
hist(chosen\_theta[n\_burn:n\_iter], main = "",  
 xlab = expression(theta~"|"~y), ylab = "Frequency")  
  
# Add the mean estimate  
mu\_est = mean(chosen\_theta[n\_burn:n\_iter])  
abline(v=mu\_est, col = "blue", lwd = 2)  
  
# Add the true mean  
abline(v = mu\_true, col = "black", lwd = 3, lty = 2)



par(mfrow = c(1,1))

*Both prior and proposal can affect the MCMC sample’s efficiency. Little change of the mean will not change the sample distribution too much. However, the low standard deviation of prior can make the MCMC sampler trace plot varies frequently. It means the estimate mean will be far away the true mean.*

## Task 2 (25 points)

Complete the following:

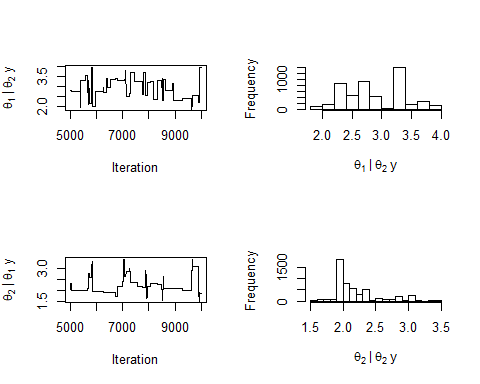
1. Extend your MCMC sampler to accommodate 2 parameters. Here we will estimate both the mean and the standard deviation of the data (15 points). A hint for the proposal distributions: you can sum the proposal densities of the two parameters together on the log scale to get the total proposal adjustment. Also, be sure to store your output correctly. Your code should store the marginal posteriors of both parameters of interest.

#-----------------------------------  
#-----------------------------------  
  
mu\_true = 3 # Parameter 1 (unknown)  
sd\_true = 2 # Parameter 2 (unknown)  
  
##### FOR PARAMETER 1  
# Prior for Parameter 1   
# Normal Distribution:  
mu\_prior = 5  
sd\_prior = 10  
  
# Proposal for Parameter 1  
# Normal Distribution  
mu\_prop = 5  
sd\_prop = sd\_prior \* 1.1  
  
##### FOR PARAMETER 2  
# Prior for Parameter 2   
# Gamma distribution:  
shape\_prior = 1  
rate\_prior = 0.5  
  
# Proposal for Parameter 2  
# Gamma distribution  
shape\_prop = 1  
rate\_prop = rate\_prior\*0.5  
  
#-----------------------------------  
#-----------------------------------  
n\_sample = 30  
y\_test = rnorm(n\_sample, mu\_true, sd\_true)  
#-----------------------------------  
#-----------------------------------  
  
  
n\_iter = 10000  
  
H <- function(n\_iter=n\_iter){  
  
 chosen\_mu = NULL  
 chosen\_sd = NULL  
   
 for(i in 1:n\_iter){  
   
 if(i==1){  
 old\_mu = rnorm(1, mu\_prop, sd\_prop)  
 old\_sd = rgamma(1, shape\_prop, rate\_prop)  
 }  
   
 new\_mu = rnorm(1, mu\_prop, sd\_prop)  
 new\_sd = rgamma(1, shape\_prop, rate\_prop)  
   
 old\_mu\_prop\_adj = dnorm(old\_mu, mu\_prop, sd\_prop, log = TRUE)  
 new\_mu\_prop\_adj = dnorm(new\_mu, mu\_prop, sd\_prop, log = TRUE)  
   
 old\_sd\_prop\_adj = dgamma(old\_sd, shape\_prop, rate\_prop, log = TRUE)  
 new\_sd\_prop\_adj = dgamma(new\_sd, shape\_prop, rate\_prop, log = TRUE)  
   
 old\_total\_prop\_adj = old\_mu\_prop\_adj + old\_sd\_prop\_adj  
 new\_total\_prop\_adj = new\_mu\_prop\_adj + new\_sd\_prop\_adj  
 #####################################################################  
 old\_mu\_prior = dnorm(old\_mu, mu\_prior, sd\_prior, log = TRUE)  
 new\_mu\_prior = dnorm(new\_mu, mu\_prior, sd\_prior, log = TRUE)  
 old\_sd\_prior = dgamma(old\_sd, shape\_prior, rate\_prior, log = TRUE)  
 new\_sd\_prior = dgamma(new\_sd, shape\_prior, rate\_prior, log = TRUE)  
   
   
 old\_lik = sum(dnorm(y\_test, old\_mu, old\_sd, log = TRUE))  
 new\_lik = sum(dnorm(y\_test, new\_mu, new\_sd, log = TRUE))  
   
   
 old\_mu\_post = old\_mu\_prior + old\_sd\_prior + old\_lik  
 new\_mu\_post = new\_mu\_prior + old\_sd\_prior + new\_lik  
   
 old\_sd\_post = old\_mu\_prior + old\_sd\_prior + old\_lik  
 new\_sd\_post = old\_mu\_prior + new\_sd\_prior + new\_lik  
   
 mu\_log\_ratio = (new\_mu\_post - new\_total\_prop\_adj) - (old\_mu\_post - old\_total\_prop\_adj)  
 sd\_log\_ratio = (new\_sd\_post - new\_total\_prop\_adj) - (old\_sd\_post - old\_total\_prop\_adj)   
   
 mu\_ratio = exp(mu\_log\_ratio)  
 sd\_ratio = exp(sd\_log\_ratio)  
   
 if(mu\_ratio > 1){  
 chosen\_mu [i] = new\_mu  
 }  
 else{  
 rand = runif(1,min=0,max=1)  
 if(mu\_ratio >= rand){  
 chosen\_mu[i] = new\_mu  
 }else{  
 chosen\_mu[i] = old\_mu  
 }  
 }  
   
 if(sd\_ratio > 1){  
 chosen\_sd [i] = new\_sd  
 }  
 else{  
 rand = runif(1,min=0,max=1)  
 if(sd\_ratio >= rand){  
 chosen\_sd[i] = new\_sd  
 }else{  
 chosen\_sd[i] = old\_sd  
 }  
 }  
   
   
 old\_mu = chosen\_mu[i]  
   
 old\_sd = chosen\_sd[i]  
 }  
 # Discard your burn-in  
 mu\_theta = tail(chosen\_mu,length(chosen\_mu)/2)  
 sd\_theta = tail(chosen\_sd,length(chosen\_sd)/2)  
  
 temp <- matrix(0, nrow=2, ncol = length(mu\_theta))  
 temp[1,] <- mu\_theta  
 temp[2,] <- sd\_theta  
   
 return(temp)  
}

*2.Run multiple MCMC chains and store the output appropriately in an object. This can be done in a data frame object if you are careful, but ’s is also helpful. You would have dimensions in the array, where is the number of chains, is the number of iterations in each chain, and is the number of parameters.*

*3.Discard your burn-in, plot the traces of each parameter, and use your function to calculate the for each parameter (5 points).*

num.chains=15  
  
Multiple\_MCM <- function(){  
 object <- array(0,dim = c(2,15,5000))  
 for(i in 1:15){  
 temp <- H(10000)  
   
 object[,i,] <- temp  
 }  
 return(object)  
}  
  
  
obj <- Multiple\_MCM()  
  
par(mfrow=c(2,2))  
  
  
plot(obj[1,1,] ~ c(5001:10000),  
 type="l",  
 xlab = "Iteration",  
 ylab =expression(theta[1]~"|"~theta[2]~y)  
 )  
  
hist(obj[1,1,], main = "",  
 xlab =expression(theta[1]~"|"~theta[2]~y),  
 ylab = "Frequency")  
  
  
plot(obj[2,1,] ~ c(5001:10000),  
 type="l",  
 xlab = "Iteration",  
 ylab =expression(theta[2]~"|"~theta[1]~y)  
 )  
  
hist(obj[2,1,], main = "",  
 xlab =expression(theta[2]~"|"~theta[2]~y),  
 ylab = "Frequency")



Now, we will compute

R.hat.mu <- GR(obj[1,,])  
R.hat.sd <- GR(obj[2,,])   
R.hat.mu

## [1] 1.024216

R.hat.sd

## [1] 1.041624

*4.Calculate the mean, marginal posterior estimates for each parameter. Plot the joint posterior samples of parameter 1 and parameter 2 and place a large “point” at the location of these mean estimates. Hint: You can use the function. (5 points)*

result <- H(500000)  
mu\_estimate <- mean(result[1])  
sd\_estimate <- mean(result[2])  
mu\_estimate

## [1] 3.225676

sd\_estimate

## [1] 2.068743

plot(x=result[1,],  
 y = result[2,],  
 xlab = expression(theta[1]~"|"~theta[2]~y),  
 ylab = expression(theta[2]~"|"~theta[1]~y),  
 )  
  
points(x=mu\_estimate, y=sd\_estimate, col='red', pch=19)

