Assignment_2

Question 1

```
r)

### make the table
data2 <-
  data <- read.table("data/radon.table.7.3.csv",header=T,sep=",")
library(tidyverse)
attach(data2)
data2$floor_basement <- ifelse(data2$Floor == "Basement",
                                 1,
                                 0)
data2$BlueEarth <- ifelse(data2$County == "BlueEarth",
                               1,
                               0)
data2$Clay <- ifelse(data2$County == "Clay",
                               1,
                               0)
data2$Good <- ifelse(data2$County == "Goodhue",
                               1,
                               0)
data2$floor_basement <- as.factor(data2$floor_basement)
data2$BlueEarth <- as.factor(data2$BlueEarth)
data2$Clay <- as.factor(data2$Clay)
data2$Good <- as.factor(data2$Good)
## outcomes
y <- data2$Radon
y
x <- as.matrix(data2[,4:7])
x

## linear regression model
model <- lm(log(Radon) ~ BlueEarth +
                Clay +
                floor_basement,
              data = data2)
summary(model)


########## attempt 1
# on this first attempt, I used the code from class, but kept getting
# predictive intervals that didn't make sense (had negative values). I
# included the code here for good measure. Perhaps you can see what
# I did wrong here
#########
beta_hat <- model$coef
```

```r
n <- length(weight)
p <- length(beta_hat)
s2 <- (n-p)*summary(model)$sigma^2
V.beta <- summary(model)$cov.unscaled
numsamp <- 10000
beta <- matrix(NA,nrow=numsamp,ncol=p)
sigma <- rep(NA,numsamp)

for (i in 1:numsamp) {
  temp <- rgamma(1, shape = (n-p)/2, rate = s2/2)
  cursigsq <- 1/temp
  curvarbeta <- cursigsq*V.beta
  curvarbeta.chol <- t(chol(curvarbeta))

  z <- rnorm(p,0,1)
  curbeta <- beta_hat+curvarbeta.chol%*%z
  sigma[i] <- cursigsq
  beta[i,] <- curbeta
}
beta
output
output <- exp(cbind(beta[,2], beta[,1] + beta[,2], beta[,3],
                    beta[,1] + beta[,3], beta[,4], beta[,1] + beta[,4], beta[,1], sigma))
for(i in 1:ncol(output)) print(round(quantile(output[,i],c(.25,.5,.75)),1))
beta

## posterior means
postmean.beta <- apply(beta,2,mean)
postmean.sigsq <- mean(sigma)
postmean.beta
postmean.sigsq

## posterior correlation between variables
cor(cbind(beta, sigma))

## 95% posterior intervals
allsamples <- cbind(beta, sigma)
allsamples.sort <- apply(allsamples,2,sort)
allsamples.sort[25,]
allsamples.sort[975,]
quantile(beta[,2], c(0.025, 0.75))
## posterior histograms
par(mfrow=c(2,2))

########## attempt 2
# on the second attempt, I used the code from the Book
# in place of the beta[] value, and this fixed the problems of
# the categorical differences with the beta values.
#########


basement.1 <- c(1,1,1,1, 1, 0, 1,1,1,0,1,1,1,1)
basement.2 <- c(0,1,1,0,1,1,1,1,1,0,1,1,1,0)
basement.3 <- c(1,0,1,0,1,1,1,1,1,1,1,1,1)
counties <- rep(1:3, c(length(y.1), length(y.2), length(y.3)))
y <- c(y.1, y.2, y.3)
x <- cbind(c(basement.1, basement.2, basement.3), make.indicators(counties))
x
class(x)
k <- ncol(x)
ls.out <- lsfit(x, log(y), intercept = F)
nsim <- 10000
lsd <- ls.diag(ls.out)
nsim <- 100000
n <- nrow(x)
k <- ncol(x)
sigma <- rep(NA, nsim)
beta <- array(NA, c(nsim, k))
for(i in 1:nsim) {
  sigma[i] <- lsd$std.dev*sqrt((n-k)/rchisq(1,n-k))
  beta[i,] <- ls.out$coef + (sigma[i]/lsd$std.dev)*lsd$std.err*t(chol(lsd$corr))%*%rnorm(k)}

output <- exp(cbind(beta[,2], beta[,1] + beta[,2], beta[,3], beta[,1] +
                      beta[,3], beta[,4], beta[,1] + beta[,4], beta[,1], sigma))
for(i in 1:ncol(output)) print(round(quantile(output[,i],c(.25,.5,.75)),1))

beta
### part 2 of this question
## house in Blue Earth county
Xstar <- c(1,1,0,0)
Xstar <- t(Xstar)

## use posterior samples from before:
numsamp <- 10000
ystar.samp <- rep(NA,numsamp)
for (i in 1:numsamp){
  xstarbeta <- Xstar%*%t(t(beta[i,]))
  ystar.samp[i] <- rnorm(1,mean=xstarbeta,sd=sqrt(sigma[i]))
}

ystar.postmean <- mean(ystar.samp)
ystar.postmean

par(mfrow=c(2,1))
xmin <- min(Radon,ystar.postmean)
xmax <- max(Radon,ystar.postmean)
hist(Radon,main="Dataset Weights",xlim=c(xmin,xmax), breaks = 40)
abline(v=mean(Radon),col=2)
hist(ystar.samp,main="Predicted Point of Blue Earth in Basement",xlim=c(xmin,xmax))
abline(v=ystar.postmean,col=2)

## house in Blue Earth county
Xstar <- c(0,1,0,0)
Xstar <- t(Xstar)

## use posterior samples from before:
```

```
numsamp <- 10000
ystar.samp <- rep(NA,numsamp)
for (i in 1:numsamp){
  xstarbeta <- Xstar%*%t(t(beta[i,]))
  ystar.samp[i] <- rnorm(1,mean=xstarbeta,sd=sqrt(sigma[i]))
}

ystar.postmean <- mean(ystar.samp)
ystar.postmean

par(mfrow=c(2,1))
xmin <- min(Radon,ystar.postmean)
xmax <- max(Radon,ystar.postmean)
hist(Radon,main="Dataset Weights",xlim=c(xmin,xmax), breaks = 40)
abline(v=mean(Radon),col=2)
hist(ystar.samp,main="Predicted Point of Blue Earth in Basement",xlim=c(xmin,xmax))
abline(v=ystar.postmean,col=2)



### again, the code from the book gives a slightly different interpretation of this
nsim <- 10000
theta <- rbeta(nsim, 3, 13)
b <- rbinom(nsim, 1, theta)
logy.rep <- rnorm(nsim, beta[,3] + b*beta[,1], sigma)
y.rep <- exp(logy.rep)
print(round(quantile(y.rep, c(.025, .25, .5, .75, .975)),1))
hist(y.rep[y.rep<40], yaxt = "n", breaks = 0:40,
     xlab = "radon measurement", main = "Posterior Predictive samples of Blue Earth County First Floor"

### also can't figure out how to change this to the basement measure,
#but I think we are supposed to change the index of the betas!

data_blue <- data2%>%
  filter(BlueEarth == 1)
class(data_blue$floor_basement)
data_blue %>%
  filter(Floor == "Basement") %>%
  summarise(me = mean(Radon))
data_blue %>%
  filter(Floor == "First") %>%
  summarise(me = mean(Radon))
```

Question 2, 3, 4

Redo this by looking at the Contour Plot code

```
### question 2
### question 2

data <- read.table("data/batting.1975-2016.csv",header=T,sep=",")
data <- data[data$AB >=100,]
dim(data)
```

```r
data <- data[data$HR >=1, ]
HR <- data$HR
AB <- data$AB
player <- data$player
year <- data$year
HR.average <- HR/AB
data$HR.average <- data$HR/data$AB

numgrid <- 100
## define what the domain of alpha and beta are
## step 1: set up the grid
alpha <- ppoints(numgrid)*.5+1.9
alpha
beta <- ppoints(numgrid)*6+69  # beta between 69 and 75
beta
full <- matrix(NA, nrow = 100, ncol = 100)
for( i in 1:100){
  for(j in 1:100){
    full[i,j] <- sum(dbeta(HR.average, alpha[i], beta[j], log = T))
  }
}
#evaluate the joint posterior
full <- exp(full - max(full))
full
full <- full/sum(full)
full
contour(alpha, beta, full, xlab = "alpha", ylab = "beta", drawlabels = F)


## calculating probabilities for grid sampler:

alphamarginal <- rep(NA,numgrid)
for (i in 1:numgrid){
  alphamarginal[i] <- sum(full[i,])
}
betaconditional <- matrix(NA,nrow=numgrid,ncol=numgrid)
for (i in 1:numgrid){
  for (j in 1:numgrid){
    betaconditional[i,j] <- full[i,j]/sum(full[i,])
  }
}

## plotting marginal distribution of alpha
par(mfrow=c(1,1))
plot(alpha,alphamarginal,type="l",main="marginal dist. of alpha")

## plotting conditional distribution of beta given alpha
alpha[25]
alpha[50]
alpha[75]
par(mfrow=c(3,1))
plot(beta,betaconditional[25,],type="l",main="dist. of beta for alpha = 24.9")
plot(beta,betaconditional[50,],type="l",main="dist. of beta for alpha = 29.9")
plot(beta,betaconditional[75,],type="l",main="dist. of beta for alpha = 34.9")

## sampling grid values:

alpha.samp <- rep(NA,10000)
beta.samp <- rep(NA,10000)
for (m in 1:10000){
  a <- sample(1:100,size=1,replace=T,prob=alphamarginal)
  b <- sample(1:100,size=1,replace=T,prob=betaconditional[a,])
  alpha.samp[m] <- alpha[a]
  beta.samp[m] <- beta[b]
}

par(mfrow=c(1,1))
contour(alpha, beta,full,xlab="alpha",ylab="beta",drawlabels=F,col=2)
points(alpha.samp,beta.samp)



## distribution of alpha, beta, beta covers 0, so it **could** increase over time
par(mfrow=c(2,1))
hist(alpha.samp,main="Alpha Samples")
hist(beta.samp,main="Beta Samples")

##posterior means
mean(alpha.samp)
mean(beta.samp)
alpha.sampsort <- sort(alpha.samp)
beta.sampsort <- sort(beta.samp)
alpha.sampsort[250]
alpha.sampsort[9750]
beta.sampsort[250]
beta.sampsort[9750]


#### question 3
#use a grid search algorithm on the log likelihood to find maximum estimates of alpha and beta
max(full)
which(full == max(full), arr.ind = T)
full
alpha_mle <- alpha.samp[40] ## about 2.0975
beta_mle <- beta.samp[51] ## about 72.03

#MLE estimates variance
nr <- (alpha_mle*beta_mle)
dr1 <- (alpha_mle + beta_mle)^2
dr2 <- (alpha_mle + beta_mle +1)
var_mle <- nr/(dr1*dr2)
var_mle

#actual data variance
var(average_run)
```

```r
### question 4

loglik_beta <- function(start_val, x){
  sum(-dbeta(x,
             start_val[1],
             start_val[2],
             log = TRUE))
}

out <- optim(par = c(0.1,0.1),
             fn = loglik_beta,
             x = average_run,
             method = "Nelder-Mead")
out$par

out <- optim(par = c(0.1,0.1),
             fn = loglik_beta,
             x = average_run,
             method = "BFGS")
out$par

out <- optim(par = c(0.1,0.1),
             fn = loglik_beta,
             x = average_run,
             method = "CG")
out$par
```

Question 5,6,7

Explain Here

```r
### Part 1: read in data
data <- read.table("data/batting.1975-2016.csv",header=T,sep=",")
data <- data[data$AB >=100,]
dim(data)
data <- data[data$HR >=1, ]
HR <- data$HR
AB <- data$AB
player <- data$player
year <- data$year
HR.average <- HR/AB
## distribution of homeruns is questionable
#for using two normal components
par(mfrow=c(1,1))
hist(HR.average, main = "Histogram of HomeRun Averages", xlab = "HomeRun Average")


#Part 2: Expectation function

Estep2 <- function(y,alpha,mu1,sigsq){
  n <- length(y)
  ind <- rep(NA,n)
  ##prob0 <- (1-alpha)*dnorm(y[i],mean=mu0,sd=sqrt(sigsq))
  prob0 <- 2*(1-alpha)*dnorm(y,mean=0,sd=sqrt(sigsq))
  prob1 <- alpha*dnorm(y,mean=mu1,sd=sqrt(sigsq))
  ind <- prob1/(prob0+prob1)
  ind
}

# observed data log likelihood function
loglik.mix <- function(y,ind,alpha,mu1,sigsq){
  loglik <- sum(log(alpha*dnorm(y,mu1,sqrt(sigsq))+2*(1-alpha)*dnorm(y,mean=0,sd=sqrt(sigsq))))
  loglik
}
loglik

#Part 3: Maximization function
Mstep2 <- function(y,ind){
  n <- length(y)
  alpha <- sum(ind)/n
  mu1 <- sum(ind*y)/sum(ind)
  ## get rid of the mu0
  sigsq <- sum(ind*((y-mu1)^2))
  ## make this y
  sigsq <- sigsq+sum((1-ind)*((y-0)^2))
  sigsq <- sigsq/n
  c(alpha,mu1,sigsq)
}


#Part 4: Running EM iterations

curalpha <- 0.6
curmu1 <- 0.001
cursigsq <- 1
curind <- Estep2(HR.average,curalpha,curmu1,cursigsq)
loglik <- loglik.mix(HR.average,curind, curalpha, curmu1, cursigsq)
itermat2 <- c(curalpha,curmu1,cursigsq,loglik)
loglik
## while loop that runs this for 100 iterations
## will stop when the successive iterations of the algorithm don't
#change by a certain amount
diff <- 1
numiters <- 1
while (diff > 0.001 || numiters <= 100){
  curind <- Estep2(HR.average,curalpha,curmu1,cursigsq)
  curparam <- Mstep2(HR.average,curind)
  curalpha <- curparam[1]
  curmu1 <- curparam[2]
  cursigsq <- curparam[3]
  loglik <- loglik.mix(HR.average,curind,curalpha,curmu1,cursigsq)
  itermat2 <- rbind(itermat2,c(curparam,loglik))
  numiters <- numiters + 1
  diff <- max(abs(itermat2[numiters,]-itermat2[numiters-1,]))
  print (c(numiters,loglik))
```

```r
}


#Tracking iterations
parametertext <- c("alpha","mu0","sigsq","loglik")
par(mfrow=c(2,3))
for (i in 1:5){
  plot(1:numiters,itermat2[,i],type="l",main=parametertext[i],xlab="Iterations",ylab="Value")
}

## alpha converges to 0.55, mu0 is 0.04

### EM code above with different starting values

curalpha <- 0.2
curmu1 <- 0.1

# Part 7
lastiter <- length(itermat2[,1])
EM.alpha.1 <- itermat2[lastiter,][1]
Em.mu1.1 <- itermat2[lastiter,][2]
Em.sigsq1.1 <- itermat2[lastiter,][3]
EM.alpha.beta.1 <- itermat2[lastiter,][4]
EM.beta.1 <- itermat2[lastiter,][5]
results_MLE_EM <- matrix(NA, 6,1)
results_MLE_EM[ , 1] <- itermat[lastiter,]
rownames(results_MLE_EM) <- c("Alpha", "Mean", "Variance", "alpha of the beta", "beta", "lik")
colnames(results_MLE_EM) <- "MLE Values"

#Part 8: Plotting
finalparam<-itermat2[numiters,]
alpha <- finalparam[1]
mu1 <- finalparam[2]
sigsq <- finalparam[3]
par(mfrow=c(1,1))
hist(HR.average,prob=T, main = "Histogram of Home Run Averages", xlab = "Home Run Average")
x <- ppoints(1000)*0.15
# y1 <- (1-alpha)*dnorm(x,mu0,sqrt(sigsq0))
# y2 <- alpha*dnorm(x,mu1,sqrt(sigsq1))
y1 <- (1-alpha)*dnorm(x,mu1, sqrt(sigsq))
y2 <- 2*alpha*dnorm(x,0, sqrt(sigsq))
lines(x,y1,col=2)
lines(x,y2,col=3)
alpha

##Individuals

finalindprops <- Estep2(HR.average,alpha,mu1,sigsq)

hist(finalindprops)
sum(finalindprops > 0.9999)
players.topHR<-data[finalindprops > 0.9999,1:5]
players.topHR
```

```r
##ortiz
### ortiz
y <- HR.average
id <- data$playerID
player_index <- id == "ortizda01"
ind <- curind
Estep2(y[player_index], alpha, mu1, sigsq)
```

Question 8,9

Explain Here

```r
### Part 1: read in data
data <- read.table("data/batting.1975-2016.csv",header=T,sep=",")
data <- data[data$AB >=100,]
dim(data)
data <- data[data$HR >=1, ]
HR <- data$HR
AB <- data$AB
player <- data$player
year <- data$year
HR.average <- HR/AB

y<- HR.average
n <- length(HR.average)
mu.data <- mean(HR.average)
sd.data <- sd(HR.average)
var.data <-sd.data^2
alpha.data <- mu.data*(mu.data*(1-mu.data)/var.data - 1)
beta.data <- (1-mu.data)*(mu.data*(1-mu.data)/var.data -1)

#Part 2: Expectation function
Estep <- function(y,alpha, mu1, sigsq1, alpha.beta, beta){
  n <- length(y)
  ind <- rep(NA,n)
  for(i in 1:n) {
    prob0 <- (1-alpha)*dbeta(y[i], alpha.beta, beta)
    prob1 <- alpha*dnorm(y[i],mean=mu1,sd=sqrt(sigsq1))
    ind[i] <- prob1/(prob0+prob1)
  }
  ind
}


#negative log likelihood
negative.log.likelihood <- function(theta, y, alpha, mu1, sigsq1, ind){
  n <- length(y)
  alpha.beta <- theta[1]
  beta <- theta[2]
  total <- sum(log((alpha*dnorm(y,mu1, sqrt(sigsq1))) + ((1-alpha)*dbeta(y,alpha.beta, beta))))
  return(-total)
}
```

```r
#Part 3: Maximization function
Mstep <- function(y,ind, alpha.initial, beta.initial){
  n <- length(y)
  alpha <- sum(ind)/n
  mu1 <- sum(ind*y)/sum(ind)
  sigsq1 <- sum(ind*((y-mu1)^2))/sum(ind)
  results.optim <- optim(par = c(alpha.initial, beta.initial),
                         fn = negative.log.likelihood,
                         y = y, alpha = alpha,
                         mu1 = mu1, sigsq1 = sigsq1)
  c(alpha,mu1,sigsq1, results.optim$par[1], results.optim$par[2])
}


#Part 4: Running EM iterations

curalpha <- 0.05
curmu1 <- 0.02
cursigsq1 <- 0.03
alpha.initial <- alpha.data
beta.initial <- beta.data
itermat <- c(curalpha, curmu1, cursigsq1, alpha.initial, beta.initial)

diff <- 1
numiters <- 1
while (diff > 0.00001 || numiters <= 100){
  numiters <- numiters + 1
  curind <- Estep(HR.average,curalpha,curmu1, cursigsq1, alpha.beta = alpha.initial,
                  beta = beta.initial)
  curparam <- Mstep(HR.average,curind, alpha.initial, beta.initial)
  curalpha <- curparam[1]
  curmu1 <- curparam[2]
  cursigsq1 <- curparam[3]
  alpha.initial <- curparam[4]
  beta.initial <- curparam[5]
  loglik <- -negative.log.likelihood(c(alpha.initial, beta.initial), y, alpha = curalpha,
                                     mu1 = curmu1, sigsq1 = cursigsq1)
  itermat <- rbind(itermat, c(curparam, loglik))
  diff <- max(abs(itermat[numiters,]-itermat[numiters-1,]))
  print (c(numiters,loglik))
}

#Tracking iterations
parametertext <- c("alpha", "mu1", "sigsq1", "alpha",
                   "beta", "log likelihood")
par(mfrow = c(2,3))
for (i in 1:6){
  plot(1:length(itermat[,1]),itermat[,i],main=parametertext[i],xlab="Iterations",ylab="Value")
}


# Part 7
lastiter <- length(itermat[,1])
```

```
EM.alpha.1 <- itermat[lastiter,][1]
Em.mu1.1 <- itermat[lastiter,][2]
Em.sigsq1.1 <- itermat[lastiter,][3]
EM.alpha.beta.1 <- itermat[lastiter,][4]
EM.beta.1 <- itermat[lastiter,][5]
results_MLE_EM <- matrix(NA, 6,1)
results_MLE_EM[ , 1] <- itermat[lastiter,]
rownames(results_MLE_EM) <- c("Alpha", "Mean", "Variance", "alpha of the beta", "beta", "lik")
colnames(results_MLE_EM) <- "MLE Values"

#Part 8: Plotting
par(mfrow = c(1,1))
hist(y, prob = T, main = "Histogram of Home Run Averages", ylim = c(0,25), xlab = "Home Run Average")
x <- ppoints(1000)*0.15
y1 <- EM.alpha.1*dnorm(x, mean = Em.mu1.1, sd = sqrt(Em.sigsq1.1))
y2 <- (1-EM.alpha.1)*dbeta(x,EM.alpha.beta.1, EM.beta.1)
lines(x,y1, col =2)
lines(x, y2, col =3)
legend(0.07, 20, col = c("red", "green"), lty = 1:1, cex = 1)



##Individuals

finalindprops <- Estep(HR.average,alpha,mu1,sigsq)
hist(finalindprops)
sum(finalindprops > 0.9999)
players.topHR<-data[finalindprops > 0.9999,1:5]
players.topHR

##ortiz
y <- HR.average
id <- data$playerID
player_index <- id == "ortizda01"
ind <- curind
Estep(y[player_index], EM.alpha.1,
      Em.mu1.1, Em.sigsq1.1,
      EM.alpha.beta.1, EM.beta.1)
```