Homework 2

Hannah Zmuda

10/27/2020

Question 1

Bisection Method

```
bisection <- function(f, a, b, nMax, tol)
{
  #initiate the a and b value, assume intervals will be proper
  iteration <- 0
  #check bounds
  if(f(a) == 0.0){
    return(a)
  }
  if(f(b) == 0.0){
    return(b)
  }
  # Begin method's loop
  for (i in 1:nMax){
    c <- (a + b)/2 #Calc the midpoint
    if(f(c) != 0) {
      #TRUE: f(c) > tol AND i <=NMAX
      if((abs(f(c)) > tol)) {
        if(sign(f(c)) == sign(f(a))) {
          a <- c
          b <- b
        }
        else {
          a <- a
          b <- c
        }
        c <- (a + b)/2
        iteration = iteration + 1
      }
      else {
        #the f(c) is within the range of tolerance
        break
      }
    }
    else {
      #FALSE: f(c) is a root
```

```
      break
    }
  }
  return(list("it" = iteration, "root" = c))
}

fcn <- function(x){sqrt(x)-cos(x)}
bmMe <- bisection(fcn, 0, 2, 3, 1e-7)
bmMe$root
```

```
## [1] 0.625
```

Newton-Raphson Method

```
newton <- function(f, dx, a, b, inital, nMax, tol){
  #set initial value
  x0 <- inital
  rootArray <- nMax

  #Check bounds
  if(f(a) == 0.0){
    return(a)
  }
  if(f(b) == 0.0){
    return(b)
  }

  #begin loop for loop method
  for (i in 1:nMax) {
    x1 = x0 - (f(x0)/dx(x0))
    rootArray[i] <- x1
    if (abs(x1 - x0) <= tol){
      root <- tail(rootArray,n = 1)
      result <- list('root' = root, 'iterations' = rootArray)
      return(result)
    }
    x0 <- x1
  }
}

f <- function(x){sqrt(x)-cos(x)}
dx <- function(x){0.5*(x^(-0.5)) + sin(x)}
newMe <- newton(f, dx, 0, 2, 1, 3, 1e-3)
newMe$root
```

```
## [1] 0.6417144
```

The Newton-Raphuson Method finds the root within the three iterations, compared to the Bisection Method. The Bisection Method found 0.625, which is not even within the tolerance.

Question 2

Part a: Deriving the Newton-Raphson Method

In the problem, we are told we can use the Poisson process assumption so we can have the likelihood function:

$$L(N|\lambda_i) = \sum_{i=1}^{n} \frac{\lambda^{N_i} e^{-\lambda}}{N!} \tag{1}$$

and because $\lambda_i = \alpha_1 b_{i1} + \alpha_2 b_{i2}$ we can substitute λ_i into Eq (1):

$$L(N|\alpha_1, \alpha_2) = \sum_{i=1}^{n} \frac{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^{N_i} e^{-(\alpha_1 b_{i1} + \alpha_2 b_{i2})}}{N!} \tag{2}$$

In order to find the parameters α_1 and α_2 we can use the Newton-Raphson update which needs to become Eq (3):

$$\begin{bmatrix} \alpha_1(t+1) \\ \alpha_2(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_1(t) \\ \alpha_2(t) \end{bmatrix} - \frac{L\prime}{L\prime\prime} \tag{3}$$

In order to get Eq (3), we first need to get the log likelihood of Eq (2):

$$l(N|\alpha_1, \alpha_2) = \sum_{i=1}^{n} N_i \ln(\alpha_1 b_{i1} + \alpha_2 b_{i2}) - \sum_{i=1}^{n} \alpha_1 b_{i1} + \alpha_2 b_{i2} - \sum_{i=1}^{n} \ln(N!) \tag{4}$$

We can then use Eq (4) and take the partial first derivative in regard to both parameters α_1 and α_2:

$$l\prime(N|\alpha_1, \alpha_2) = \begin{bmatrix} \sum_{i=1}^{n} \frac{N_i b_{i1}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i1} \\ \sum_{i=1}^{n} \frac{N_i b_{i2}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i2} \end{bmatrix} \tag{5}$$

To get the double derivative we use the Hessian matrix

$$l\prime\prime(N|\alpha_1, \alpha_2) = \begin{bmatrix} \frac{\partial^2 l}{\partial \alpha_1^2} & \frac{\partial^2 l}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial^2 l}{\partial \alpha_2^2} & \frac{\partial^2 l}{\partial \alpha_2 \partial \alpha_1} \end{bmatrix} \tag{6}$$

$$l\prime\prime(N|\alpha_1, \alpha_2) = \begin{bmatrix} \sum_{i=1}^{n} -\frac{N_i b_{i1}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \sum_{i=1}^{n} -\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \\ \sum_{i=1}^{n} -\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \sum_{i=1}^{n} -\frac{N_i b_{i2}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \end{bmatrix} \tag{7}$$

with this we can get the equation to be used in the Newton-Raphson update to get a final equation

$$\begin{bmatrix} \alpha_1(t+1) \\ \alpha_2(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_1(t) \\ \alpha_2(t) \end{bmatrix} - \begin{bmatrix} \sum_{i=1}^{n} -\frac{N_i b_{i1}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \sum_{i=1}^{n} -\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \\ \sum_{i=1}^{n} -\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \sum_{i=1}^{n} -\frac{N_i b_{i2}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} \frac{N_i b_{i1}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i1} \\ \sum_{i=1}^{n} \frac{N_i b_{i2}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i2} \end{bmatrix} \tag{8}$$

Part b: Deriving the Fisher Scoring Method

Similar to part (a), we will use the log likelihood to find the equation to find the parameters α_1 and α_2. Instead of only taking the second derivative, we will take the variance of the first derivative to get the Fisher Information. This output will be a two by two matrix as well but instead is the Covariance matrix instead of the Hessian matrix:

$$\begin{bmatrix} Var(\sum_{i=1}^{n} \frac{N_i b_{i1}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i1}) & Covariance \\ Covariance & Var(\sum_{i=1}^{n} \frac{N_i b_{i2}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} - \sum_{i=1}^{n} b_{i2}) \end{bmatrix} \tag{9}$$

Or we could take the (negative) expectation of the second derivative of the log likelihood function.

$$I(\alpha_1, \alpha_2) = -E[l''(N(\alpha_1, \alpha_2))] = \begin{bmatrix} \sum_{i=1}^{n} E[\frac{N_i b_{i1}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2}] & \sum_{i=1}^{n} E[\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2}] \\ \sum_{i=1}^{n} E[\frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2}] & \sum_{i=1}^{n} E[\frac{N_i b_{i2}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2}] \end{bmatrix} \tag{10}$$

Equation 10 then simplifies down to the Fisher Scoring Approach:

$$\begin{bmatrix} \alpha_1(t+1) \\ \alpha_2(t+1) \end{bmatrix} = \begin{bmatrix} \alpha_1(t) \\ \alpha_2(t) \end{bmatrix} - I(\theta^t)^{-1} l'(\theta^t) \tag{11}$$

Part c: Implementing Newton and Fisher Methods in R

```
#import data set from Givens et al.
data.oil <- read.table("oilspills.dat",header = TRUE)
#likelihood function
l <- function(N, a, b){
  result <- sum(N*log(a[1]*b[1]+a[2]*b[2])) - sum((a[1]*b[1]+a[2]*b[2])) - sum(log(factorial(N)))
  return(result)
}
#derivative of the likelihood function
dl <- function(N, a, b){
  result1 <- sum((N*b[1])/(a[1]*b[1] + a[2]*b[2])-sum(b[1]))
  result2 <- sum((N*b[2])/(a[1]*b[1] + a[2]*b[2])-sum(b[2]))
  return(matrix(data = list(result1,result2), nrow = 2, ncol = 1, dimnames = NULL))
}
#double derivative of the likelihood function
d2l <- function(N, a, b){
  result11 <- sum(-(N*b[1]^2)/(a[1]*b[1] + a[2]*b[2])^2)#1st row, 1st col
  result12 <- sum(-(N*b[1]*bi2)/(a[1]*b[1] + a[2]*b[2])^2)#1st row,2nd col
  result22 <- sum(-(N*b[2]^2)/(a[1]*b[1] + a[2]*b[2])^2)#2nd row, 2nd col
  result21 <- sum(-(N*b[2]*b[1])/(a[1]*b[1] + a[2]*b[2])^2)#2nd row, 1st col
  return(matrix(data = list(result11,result21,result12,result22), nrow = 2, ncol = 2, dimnames = NULL))
}
#Fisher Information
I <- function(N, a, b){
  result11 <- sum(mean((N*b[1]^2)/(a[1]*b[1] + a[2]*b[2])^2))#1st row, 1st col
  result12 <- sum(mean((N*b[1]*bi2)/(a[1]*b[1] + a[2]*b[2])^2))#1st row,2nd col
  result22 <- sum(mean((N*b[2]^2)/(a[1]*b[1] + a[2]*b[2])^2))#2nd row, 2nd col
  result21 <- sum(mean((N*b[2]*b[1])/(a[1]*b[1] + a[2]*b[2])^2))#2nd row, 1st col
  return(matrix(data = list(result11,result21,result12,result22), nrow = 2, ncol = 2, dimnames = NULL))
}
new.oil <- function(N,a,b,l,dl,d2l,tol){
  n <- length(N)
  i <- 1
  a0 <- matrix(c(1,1),nrow = 2,ncol = 1)
  for(i in 1:n){
    a = a0 - dl(N,a,b)/d2l(N,a,b)
    if(abs(a-a0) <= tol){
      output.new <- a
      return(output.new)
    }
    a0 <- a
```

```
  }
}

fish.oil <- function(N,a,b,l,dl,I,tol){
  n <- length(N)
  i <- 1
  a0 <- matrix(c(1,1),nrow = 2,ncol = 1)
  for(i in 1:n){
    a = a0 - dl(N,a,b)/I(N,a,b)
    if(abs(a-a0) <= tol){
      output.fish <- a
      return(output.fish)
    }
    a0 <- a
  }
}
N <- data.oil$spills
```

Part d: Standard Error

Part e: Quasi-newton Method

Question 3

##Part a based on the problem, we know we are given an observed data set $X = (X_1, X_2, ..., X_n)$ where $X_i \sim \alpha N(\mu_1, \sigma_1^2) + (1-\alpha)N(\mu_2, \sigma_2^2)$. with this, and knowing we have a two-component mixture model, we can derive the EM algorithm to find $\hat{\theta}$ where $\theta = (\alpha, \mu_1, \mu_2, \sigma_1, \sigma_2)$. For the Q function we have:

$$Q(\theta|\hat{\theta}) = E[log(L(\theta|X))]$$

$$Q(\theta|\hat{\theta}) = \Sigma_{i=1}^{n} log(\alpha N(X_i, \mu_1, \sigma_1) + (1-\alpha)N(X_i, \mu_2, \sigma_2))$$

From here, we can say that the Maximum Likelihood extimate for the parameters μ an dσ is as follows:

$$\hat{\mu_j} = \frac{\Sigma_i X_i P(\theta_j|X_i)}{\Sigma_i P(\theta_j|X_i)}$$

$$\hat{\sigma_j} = \frac{\Sigma_i (X_i - \mu_j)^2 P(\theta_j|X_i)}{\Sigma_i P(\theta_j|X_i)}$$

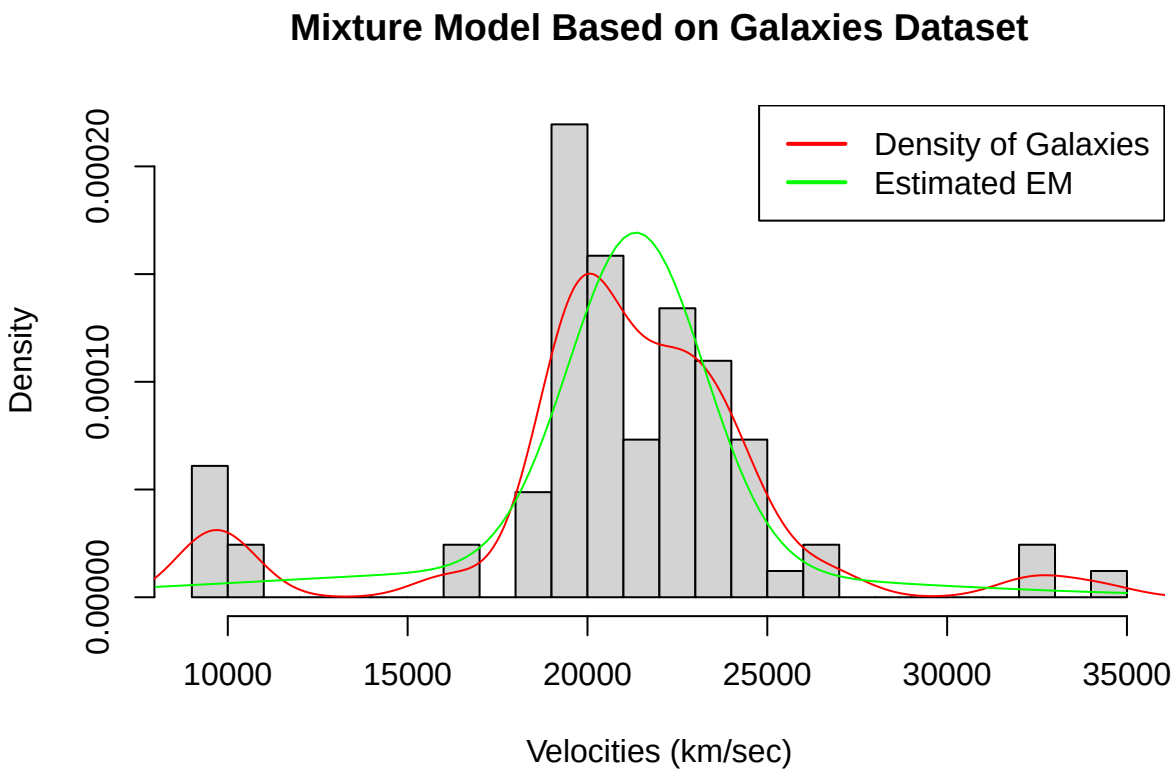
##Part b

```
#call the galaxies dataset
data(galaxies)
x <- galaxies
n <- length(x)
#make an estimation for the mu, sigma, and alpha variables. Can use k clustering based on the remark
kCluster <- kmeans(x,2)$cluster
mu1 <- mean(x[kCluster == 1])
mu2 <- mean(x[kCluster == 2])
sigma1 <- sd(x[kCluster == 1])
sigma2 <- sd(x[kCluster == 2])
alpha <- sum(kCluster == 1)/length(kCluster) #Make alpha based on the first mixture
#Other variables
i <- 2 #number of mixture models (two-mixture model)
tol <- 1e-10 #tolerance for the EM Algorithm
#Calculate Q
#initalize Q
Q <- 0
Q[2] <- sum(log(alpha*dnorm(x,mu1,sqrt(sigma1)))) + sum(log((1-alpha)*dnorm(x,mu2,sqrt(sigma2))))
#E step: Compute Q(theta | theta^t) where theta = (alpha,mu1,mu2,sig1,sig2)
while(abs(Q[i]-Q[i-1])>=tol){
  #Find the conditional probability for each mixture model
  mix1 <- alpha*dnorm(x,mu1,sigma1)
  mix2 <- (1-alpha)*dnorm(x,mu2,sigma2)
  totalMix <- mix1 + mix2
  cp1 <- mix1/totalMix#Conditional Probability for mixture 1
  cp2 <- mix2/totalMix#Conditional probability for mixture 2
  #M step
  alpha <- sum(cp1)/n
  mu1 <- sum(x*cp1)/sum(cp1)
  mu2 <- sum(x*cp2)/sum(cp2)
  sigma1 <- sqrt(sum(((x-mu1)^2)*cp1)/sum(cp1))
  sigma2 <- sqrt(sum(((x-mu2)^2)*cp2)/sum(cp2))
  #update probability values
  cp1 <- alpha
  cp2 <- 1-alpha
```

```
  #update counter
  i <- i + 1
  #Return E-step, unless stopping criteria has been met
  Q[i] <- sum(log(totalMix))
}
theta <- list("alpha" = alpha, "mu1" = mu1, "mu2" = mu2, "sigma1" = sigma1,"sigma" = sigma2)
print(theta)
```

```
## $alpha
## [1] 0.2547571
##
## $mu1
## [1] 19276.54
##
## $mu2
## [1] 21358.59
##
## $sigma1
## [1] 8189.965
##
## $sigma
## [1] 1890.294
```

```
hist(x, prob = T, breaks = 20, xlab = "Velocities (km/sec)", main = "Mixture Model Based on Galaxies Da
lines(density(x), col = "red")
xfit <- seq(8000,35000,200)
EMEstimate <- (alpha * dnorm(xfit,mu1,sigma1)) + ((1-alpha) * dnorm(xfit,mu2,sigma2))
lines(xfit, EMEstimate, col = "green",ylim = max(EMEstimate))
legend('topright', col = c("red","green"), lwd = 2, legend = c("Density of Galaxies", "Estimated EM"))
```


Mixture Model Based on Galaxies Dataset
Density of Galaxies
Estimated EM
Density
0.00000
0.00010
0.00020
10000
15000
20000
25000
30000
35000
Velocities (km/sec)

Question 4