STAT 440: Homework Assignment #7

Due: Thursday, Apr. 17 at 11:59pm on Canvas

Each of these exercises calls for writing code. Please use R, and turn in your code along with your written work and any plots you are asked to produce. Just as you strive for clarity in your writing of language, try to write R code that is easy to read, in the sense that a knowledgeable user of R would be able to understand it easily.

You *may* choose to use LaTeX for your mathematics writing in your homework, in which case I recommend R Markdown as an easy way to combine text, mathematics, R code, and R output. But this is not required; you may choose to format your work however you find convenient, as long as you upload your assignment to Canvas as a single pdf document.

Exercise 1

Let's consider a dataset consisting of batting averages of professional baseball players in 2010 who had 50 or more at bats (i.e., each observation is a proportion in which the denominator is at least 50).

```
mlbbat2010 <- read.csv("https://www.openintro.org/data/csv/mlbbat10.csv")
ba <- mlbbat2010$bat_avg
ab <- mlbbat2010$at_bat
x <- ba[ab >= 50]
```

This exercise asks you to find maximum likelihood estimators θ_1 and θ_2 under the assumption that the batting averages are a simple random sample from a Beta(θ_1, θ_2) distribution. Here is some code that returns the log-likelihood evaluated at θ:

```
ell <- function(theta, data = x) {
   a <- theta[1]
   b <- theta[2]
   n <- length(data)
   return( n * lgamma(a+b) - n * lgamma(a) - n * lgamma(b) +
            (a-1) * sum(log(data)) + (b-1)* sum(log(1 - data)))
}
```

Part (a) Write down the log-likelihood function $\ell(\theta)$ in mathematical notation, using x_i to denote the ith observation. Then find the gradient vector $\nabla\ell(\theta)$, which consists of a 2×1 vector of partial derivatives. You may use a and b in place of θ_1 and θ_2 if you wish.

Hint: The derivative of $\log\Gamma(y)$ with respect to y is called the digamma function and can be written as $\psi(y)$.

Part (b) Find the Hessian matrix $\nabla^2\ell(\theta)$, which consists of a 2×2 matrix of second partial derivatives. Since $\partial^2/\partial a\partial b$ and $\partial^2/\partial b\partial a$ are the same here, your Hessian matrix should be symmetric.

Hint: The derivative of $\psi(y)$ with respect to y is called the trigamma function and can be written as $\psi'(y)$.

Part (c) Write functions `gradient` and `Hessian` in R that return the gradient from part (a) and the Hessian from part (b) as a function of a single 2-dimensional vector `theta`. (Both `gradient` and `Hessian` can be defined to take the same two arguments as `ell` above.)

Hint: In addition to the `lgamma` function, R also has `digamma` and `trigamma` functions. In `R` you can compute the 2×2 Hessian matrix, which equals a constant plus a diagonal matrix, using `k + diag(c(m,n))` where `k` is the constant which is added to all four matrix entries and `m` and `n` are the additional diagonal values. (You should figure out what to substitute for `k` and `m` and `n`.)

Answer

```
mlbbat2010 <- read.csv("https://www.openintro.org/data/csv/mlbbat10.csv")
ba <- mlbbat2010$bat_avg
ab <- mlbbat2010$at_bat
x <- ba[ab >= 50]
```

```
ell <- function(theta, data = x) {
   a <- theta[1]
   b <- theta[2]
   n <- length(data)
   return( n * lgamma(a+b) - n * lgamma(a) - n * lgamma(b) +
            (a-1) * sum(log(data)) + (b-1)* sum(log(1 - data)))
}
```

Part(a)

We know that the Log-Likelihood function for a Beta distribution with parameters θ_1 and θ_2 is:

$n \log(\Gamma(\theta_1 + \theta_2)) - n \log(\Gamma(\theta_1)) - n \log(\Gamma(\theta_2)) + (\theta_1 - 1) \sum_{i=1}^{n} \log(X_i) + (\theta_2 - 1) \sum_{i=1}^{n} \log(1 - X_i)$

Now we need to find the gradient vector of the Log-Likelihood:

$$\nabla \ell(\theta) = \begin{pmatrix} \frac{\partial \ell}{\partial \theta_1} \\ \frac{\partial \ell}{\partial \theta_2} \end{pmatrix}$$

$$\frac{\partial \ell}{\partial \theta_1} = n\psi(\theta_1 + \theta_2) - n\psi(\theta_1) + \sum_{i=1}^{n} \log(X_i)$$

$$\frac{\partial \ell}{\partial \theta_2} = n\psi(\theta_1 + \theta_2) - n\psi(\theta_2) + \sum_{i=1}^{n} \log(1 - X_i)$$

So then we will get that $\nabla \ell(\theta) = \begin{pmatrix} n\psi(\theta_1 + \theta_2) - n\psi(\theta_1) + \sum_{i=1}^{n} \log(X_i) \\ n\psi(\theta_1 + \theta_2) - n\psi(\theta_2) + \sum_{i=1}^{n} \log(1 - X_i) \end{pmatrix}$

Part(b)

Now to find the Hessian Matrix, we need to find the partial derivatives with respect to each parameter of both of the partials in the gradient vector. i.e. the second partial derivatives.

$$\nabla^2 \ell(\theta) = \begin{pmatrix} \frac{\partial^2 \ell}{\partial \theta_1^2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_2} \\ \frac{\partial^2 \ell}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_2^2} \end{pmatrix}$$

$$\frac{\partial^2 \ell}{\partial \theta_1^2} = n\psi'(\theta_1 + \theta_2) - n\psi'(\theta_1)$$

$$\frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_2} = n\psi'(\theta_1 + \theta_2)$$

$$\frac{\partial^2 \ell}{\partial \theta_2 \partial \theta_1} = n\psi'(\theta_1 + \theta_2)$$

$$\frac{\partial^2 \ell}{\partial \theta_2^2} = n\psi'(\theta_1 + \theta_2) - n\psi'(\theta_2)$$

$$\nabla^2 \ell(\theta) = \begin{pmatrix} n\psi'(\theta_1+\theta_2) - n\psi'(\theta_1) & n\psi'(\theta_1+\theta_2) \\ n\psi'(\theta_1+\theta_2) & n\psi'(\theta_1+\theta_2) - n\psi'(\theta_2) \end{pmatrix}$$

Part(c)

```
gradient <- function(theta, data = x){
  a <- theta[1]
  b <- theta[2]
  n <- length(data)
  return( c(n * digamma(a+b) - n * digamma(a) + sum(log(data)),
    n * digamma(a+b) - n * digamma(b) + sum(log(1-data)))
  )
}

Hessian <- function(theta, data = x) {
  a <- theta[1]
  b <- theta[2]
  n <- length(data)
  return( n * trigamma(a+b) - n*diag(c(trigamma(a), trigamma(b))))
}
```

Exercise 2

Next you'll find a maximum likelihood estimator $\hat{\theta}_{\text{MLE}}$ for the dataset in Exercise 1 in two different ways. Each requires that you enter a starting value of θ. Use $\theta_0 = (20, 60)^\top$.

Part (a) Use `optim` to find $\hat{\theta}_{\text{MLE}}$. Be sure to read the help file for `optim` and use the `control` argument with `list(fnscale = -1)` so that it finds a maximum instead of a minimum. Display enough of the output from the `optim` function so that you can see both the final parameter values as well as the maximized value of the log-likelihood.

Part (b) Use Newton-Raphson to find $\hat{\theta}_{\text{MLE}}$. The idea is the same as for the one-dimensional Newton method: You're searching for a zero of the gradient function by approximating that gradient function by a linear function passing through the current point and having a constant derivative (in this case, the Hessian matrix). In other words,

$$\theta_{\text{next}} = \theta_{\text{current}} - [\nabla^2 \ell(\theta_{\text{current}})]^{-1} \nabla \ell(\theta_{\text{current}}).$$

In R, this might look like the following:

```
print(new <- theta - solve(Hessian(theta)) %*% gradient(theta))
theta <- new
```

The above two lines of code can be repeated until convergence.

Starting from `theta <- c(20, 60)`, document the value of `theta` at each iteration using the algorithm above. Compare your final MLE with the value obtained in Part (a).

Part (c) Produce a histogram of the original dataset and overlay a beta density using the value of $\hat{\theta}_{\text{MLE}}$ as the parameter vector. Since the MLE represents in some sense the "best possible" fit from the assumed class of beta distributions, what do you think of the modeling choice to use a beta distribution for this dataset?

Answer

```r
theta_0 <- c(20,60)
```

Part(a)

```r
optim(par=theta_0, fn = ell, control = list(fnscale=-1))
```

```
## $par
## [1] 14.38685 45.14657
##
## $value
## [1] 823.8554
##
## $counts
## function gradient
##       79       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

According to the `optim` function, the maximum likelihood estimators for the beta distribution are $\theta_1 \approx 14.387$ and $\theta_2 \approx 45.147$. The value of the Log-Likelihood function at the maximum is 823.855

Part(b)

Now we are going to use the Newton-Raphson Method to solve for the MLE of the Beta distribution. Using this iterative method, we can find the MLEs by repeating the following step until the parameter values converge:

$\theta_{\text{next}} = \theta_{\text{current}} - [\nabla^2 \ell(\theta_{\text{current}})]^{-1} \nabla \ell(\theta_{\text{current}})$

In R, this would look like:

```r
#set initial parameter estimates to 20, 60
theta <- theta_0
iter <- 6

# create matrix for storing the parameter values
iter_table <- matrix(NA, nrow = iter, ncol = 2)
colnames(iter_table) <- c("theta1", "theta2")

for (i in 1:iter) {
  new <- theta - solve(Hessian(theta)) %*% gradient(theta)
  iter_table[i, ] <- new
  theta <- new
}

iter_df <- as.data.frame(iter_table)
iter_df$Iteration <- 1:iter

iter_df <- iter_df[, c("Iteration", "theta1", "theta2")]

print(iter_df)
```

```
##   Iteration   theta1   theta2
## 1         1 11.65549 37.64771
## 2         2 13.79229 43.45748
## 3         3 14.36247 45.07688
## 4         4 14.38933 45.15430
## 5         5 14.38939 45.15445
## 6         6 14.38939 45.15445
```

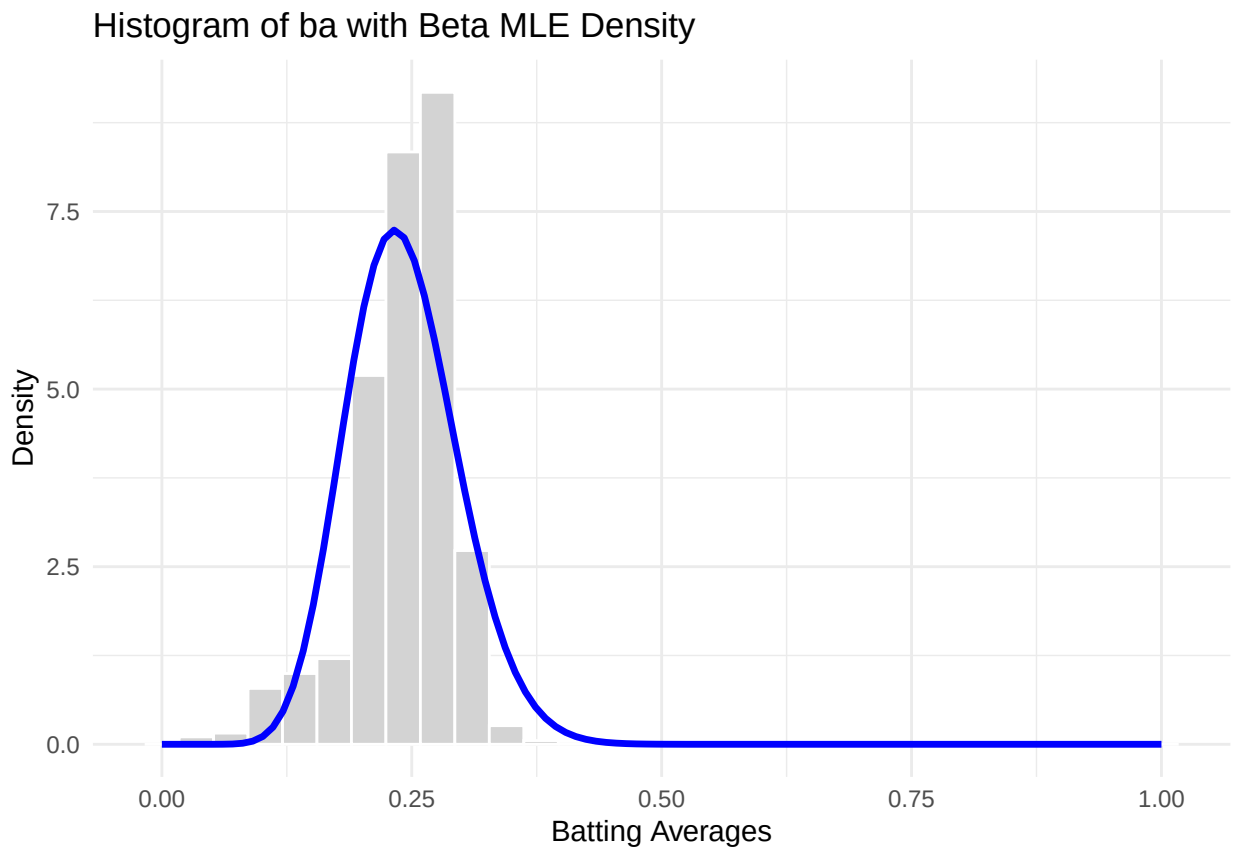
In this table we can see the values of the parameter values during each iteration of the Newton-Rahpson method.

Part(c)

```r
library(ggplot2)
theta_hat <- theta[,1]

df <- data.frame(x = x)

# Step 2: Create beta density curve using MLE parameters
x_vals <- seq(0, 1, length=100)
beta_density <- data.frame(
  x = x_vals,
  y = dbeta(x_vals, shape = theta_hat[1], shape2 = theta_hat[2])
)

# Step 3: Plot
ggplot(df, aes(x = x)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightgray", color = "white") +
  geom_line(data = beta_density, aes(x = x, y = y), color = "blue", size = 1.2) +
  labs(title = "Histogram of ba with Beta MLE Density",
       x = "Batting Averages", y = "Density") +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Exercise 3

In a paper called "Living Histograms" (*International Statistical Review*, 1975), Brian Joiner measured heights of undergraduate statistics students in inches. Here are the data as an R object:

```
joiner <-
  rep(59:75,
  c(3, 5, 7, 12, 16, 23, 23, 18, 15, 16, 23, 21, 13, 19, 8, 9, 3))
```

Let's suppose that this group of students is a representative sample from a population we're interested in studying.

Use the `normalmixEM2comp` function in the package called `mixtools` to find the MLE for population mean male height and mean female height, along with the accompanying standard deviation estimates and the proportion of males, under the following two assumptions:

- Both male heights and female heights follow normal distributions.
- The standard deviation is the same for both males and females.

Create a histogram of the heights, overlaid with density curves for the male and female normal distributions (each of these should be multiplied by the corresponding estimated proportion). Also plot the sum of these two curves, which should closely follow the dataset if you've done things correctly.

Hint: If you dig up the paper, you will definitely recognize the location where the photographs were taken. This probably will not help with your assignment but you might find it amusing.

Answer

```
joiner <-
  rep(59:75,
  c(3, 5, 7, 12, 16, 23, 23, 18, 15, 16, 23, 21, 13, 19, 8, 9, 3))
```

```
set.seed(440)
library(mixtools)
```

```
## Warning: package 'mixtools' was built under R version 4.3.3
```

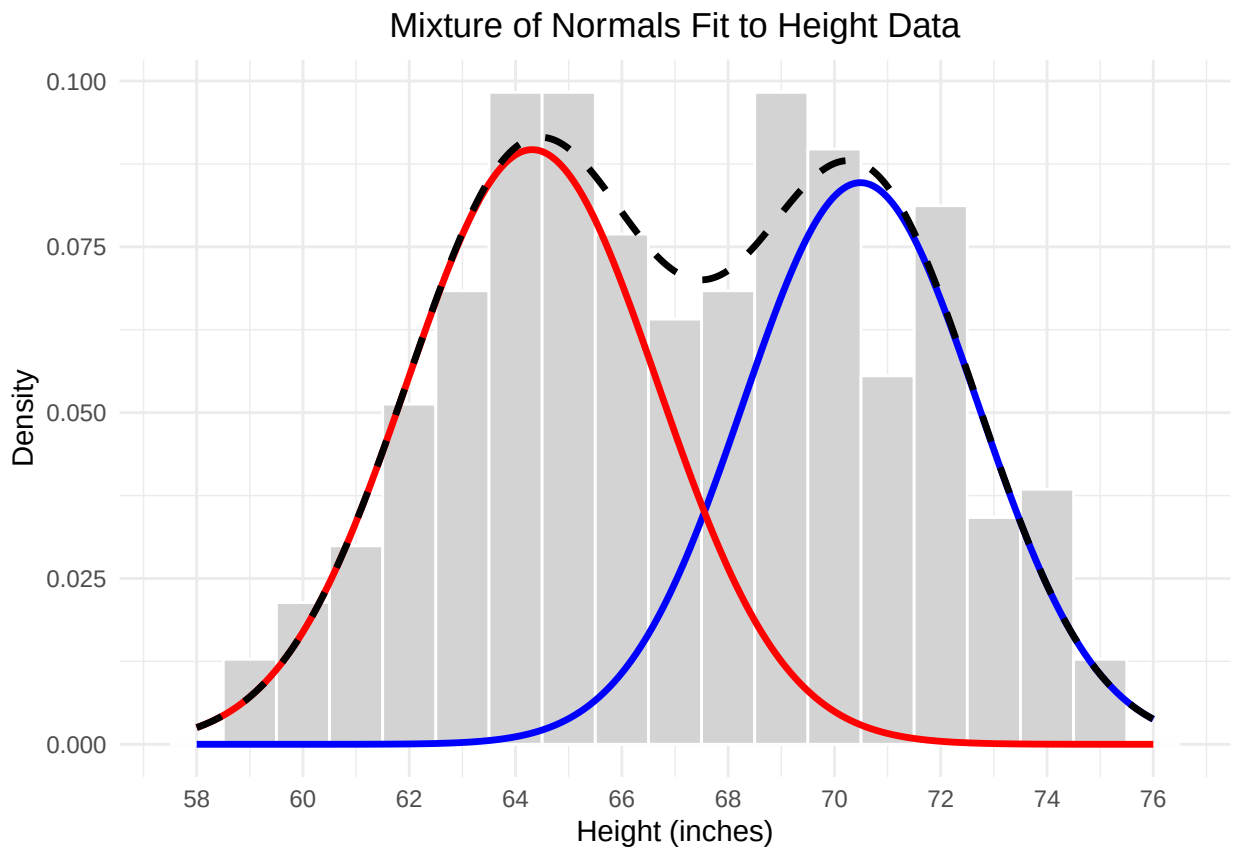
```
## mixtools package, version 2.0.0.1, Released 2022-12-04
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```
out <- normalmixEM2comp(x=joiner,lambda = 0.5, mu=c(60,70), sigsqrd = c(1,1))
```

```
## number of iterations= 213
```

```
#give the function initial arbitrary estimates for the mixing proportion, mean, and variance of the dat

FMsplit <- out$lambda
avgHeights <- out$mu
sdHeights <- out$sigma
```

```
x_vals <- seq(min(joiner) - 1, max(joiner) + 1, length = 500)
density_df <- data.frame(
  x = x_vals,
  female = FMsplit[1] * dnorm(x_vals, mean = avgHeights[1], sd = sdHeights[1]),
  male = FMsplit[2] * dnorm(x_vals, mean = avgHeights[2], sd = sdHeights[2])
)
density_df$total <- density_df$male + density_df$female

# Original data as data frame
df <- data.frame(height = joiner)

# Plot with ggplot
ggplot(df, aes(x = height)) +
  geom_histogram(aes(y = ..density..), bins = 19, fill = "lightgray", color = "white") +
  geom_line(data = density_df, aes(x = x, y = male), color = "blue", size = 1.2) +
  geom_line(data = density_df, aes(x = x, y = female), color = "red", size = 1.2) +
  geom_line(data = density_df, aes(x = x, y = total), color = "black", size = 1.2, linetype = "dashed")
  labs(title = "Mixture of Normals Fit to Height Data",
       x = "Height (inches)", y = "Density") +
  theme_minimal() +
  scale_x_continuous(breaks = seq(58, 76, by = 2)) +
  theme(plot.title = element_text(hjust = 0.5))
```


Mixture of Normals Fit to Height Data
Density
0.100
0.075
0.050
0.025
0.000
58
60
62
64
66
68
70
72
74
76
Height (inches)