Stochastic Data Processing and Simulation, Assignment A4

Sarah Al-khateeb

1 Introduction

In this lab, we were introduced to Bayesian inference, which is a method of statistical inference. Bayes' theorem is useful since it allows us to use some knowledge or belief that we already have (prior) to help us calculate the probability of a related event. Bayesian inference is the process of concluding properties about a population or probability distribution from data using Bayes' theorem (See Reference). Drawing inferences from observations are interesting but we also care about making rational, informed decisions under uncertainty which we can achieve using Decision theory. We will look more closely at two problems in this lab, Stock optimization where we want to predict future development by modeling daily closing prices, and Age assessment where we try to learn about the age of a person that could be useful in legal contexts.

2 Assignment 4(i)

2.1 Problem

Using stock values data, we will try to model the closing prices of 7 stocks and use decision theory to predict future investments and what stock gives the highest utility to invest in.

2.2 Theory and implementation

2.2.1 Part a

In the stock values data, we have seven stocks with prices each day from 2002-06-03 to 2006-06-01. In total, we have 1006 closing prices for each stock. In this part, we model the daily closing prices for each stock by calculating the percent change that is taking the log of the prices and subtract the previous day data from the current day.

$$Z_{ij} = \log(\frac{X_{ij}}{X_{i,j-1}}) = \log(X_{ij}) - \log(X_{i,j-1})$$

This method helps in understanding the changes in stock prices under the assumption that some stocks move together, but the day-to-day changes are random and independent.

Then, we calculate the mean in each column (γ, using the R function colMeans) and the covariance matrix of log prices (Σ, using the R function cov). The covariance matrix, the off-diagonal elements contain the covariance of each pair of variables. The diagonal elements contain the variances of each variable, which measures how much the data are scattered about the mean.

We are considering the investment over 100 days and generating a 1000 sample from this data. Using the R function rmnorm, we can generate the samples by substituting $n_{days} * colmean$ for (γ), and $n_{days} * cov_{matrix}$ for (Σ). We will consider an approximate solution where we compute an expectation of the utility by averaging over a sample from the relevant distribution.

$$V_q = (V_{q1}, V_{q2}, ..., V_{qm}).\ we\ have\ \ V_q \sim Normal(n\gamma, n\Sigma)$$

2.2.2 Part b

In this part, we are interested in measuring how and where we should invest an amount of money. To do that we consider two scenarios, one where we divide the investment equally by the 7 stocks (all stocks have the same weight and all weights add up to one so the weight for each, in this case, is 1/7).

In the second scenario, we look at the stock with the best performance by calculating the max for each column mean and choose the stock with the maximum mean and that stock will receive all the investment (means all other weights will be zero except for the best performing stock will have a weight of 1).

This is done by using a for loop to calculate the expected utility for each scenario for different k = -0.5, 0.5, 1.5. The utility function basically tells us how useful the outcome that results in investing a specific amount of money, and in terms of stock prices, we can conclude if we are gaining or losing money by choosing a specific action (the amount of investment should be made and how to distribute this investment or which stock should be considered as the best performance). Expected utility function:

$$E(u(T)) \approx \frac{1}{S}\sum_{q=1}^{S}\frac{1}{k}(1-(\sum_{i=1}^{m} w_i \; \exp(V_{qi}))^{-k})$$

2.2.3 Part c

In this part, we limit ourselves to invest in the two best-performing stocks we concluded using the max method. these two stocks are S3 and S4 and using the R function optimize, we try to find the optimal weights each stock should receive using two k values = -0.5 and 1.5 (using the expected utility formula above).

2.3 Results and discussion

2.3.1 Part a

As discussed above, the data was prepared for further investigation. The result is a data table including 1000 sample/vector from a multivariate normal distribution where each vector is independent for different days.

2.3.2 Part b

For different k values the expected utility in the first case (weight for each stock = 1/7):

1. Expected utility with k = -0.5 is $\approx$ 0.053
2. Expected utility with k = 0.5 is $\approx$ 0.043
3. Expected utility with k = 1.5 is $\approx$ 0.033

We can see from the results, with k increasing the expected utility decreasing. When k is larger we are less likely to take risks. This is due to the fact that U(x) is more concave as k increases and so we can conclude that k is a measure of risk aversiveness. When k is large the money gained not much but the money lost affects the utility by decreasing it, which is what we can see from the results above. This could be a good model in case we care about avoiding losing all or most of the money that we invest.
For the second case (where we invest all money to the best-performing stock; weight for S4 = 1 and all other weights = 0), taking the max of the means, S4 seems to have the best performance with mean $\approx$ 0.00053.

1. Expected utility with k = -0.5 is ≈ 0.064
2. Expected utility with k = 0.5 is ≈ 0.035
3. Expected utility with k = 1.5 is ≈ 0.007

We can notice, also with k increasing the utility decreasing and comparing this to results where we invest in all stocks, investing in all stocks for all k values, give more utility comparing to only investing in S4 and so we can say that the usefulness of investing in all stocks is more than investing in the only S4. The main principle of decision theory says that we need to choose an action, which maximizes the expected utility.

2.3.3 Part c

In this part, we consider investing in S3 and S4 which have the best performance, we want to optimize and choose the optimal weights to assign for each of these stocks.

1. Optimized weight with k = -0.5 is ≈ 0.6 for S3 and 0.4 for S4.
   The expected utility, in this case, is ≈ 0.0008
2. Optimized weight with k = 1.5 is ≈ 0.3 for S3 and 0.7 for S4.
   The expected utility, in this case, is ≈ 0.0003

We can say from these results that putting more weight in S3 with k = -0.5 gives more utility while when k = 1.5, S3 has less weight and the expected utility decreased. As we mentioned before, with higher k money lost decreases the utility, and here for k = 1.5 we have more weight in S4 and it seems that this causes more money loss (less utility).

2.3.4 Part d

We started by an assumption that the data is independent and we generated samples from a multivariate normal distribution, a typical assumption but what if the data is not normally distributed? In this case, the optimization is not really valid. Maybe investigating this a bit more will be more appropriate for later computations (although this would invalidate the simplification done).

The utility function is not guaranteed to be perfect, but in this case, it simplifies the problem and works the way we want it to, so there's no reason to reject it.

We should be careful with the value of k, large values make the optimum solution invest a lot in fairly constant stock (like we saw in part c). While this is good from one point of view, it does take very long before any significant profit is made from the investment.

We are using average and this might be not very accurate since average tends to be sensitive to outliers, and that can shift the value in an undesired way. Moreover, we only made 1 sample from 4 years of data, and in order to make decisions on a 100-day investment we estimated the mean and the covariance from the data, then simulated 100-day returns and used this value to estimate the expected utilities, this is not enough to have accurate results.

3 Assignment 4(ii)

3.1 Problem

In this problem, we investigate using a biological feature "maturity of knees" to assess a person's age and measure the cost of having a miss-classification. Using the decision theory, we will analyze the optimal selection of a decision rule (for each reporting state, whether persons with a medical report should be classified as adults or children).

3.2 Theory and implementation

3.2.1 Part a

In order to fit a logistic regression to the data, we first implement an optimization method using the R function optim, which performs minimization using the negative log-likelihood (for simplicity, we use NLL in calculations, we typically minimize loss functions, so we use negative log-likelihood because we can minimize it, and so when we minimize the negative log-likelihood, we are performing maximum likelihood estimation). This optimization aims to find the optimal parameters a and b to fit a logistic regression. Likelihood function:

$$L(a,b) = \prod_{i=1}^{n} \left(\frac{\exp(a + b\, x_i)}{1 + \exp(a + b\, x_i)}\right)^{y_i} \left(1 - \frac{\exp(a + b\, x_i)}{1 + \exp(a + b\, x_i)}\right)^{1-y_i}$$

After finding the parameters a and b that maximize the likelihood function, we use the R function ggplot to plot the regression along with data points.

3.2.2 Part b

In this part, we use Bayesian inference for the model parameters a and b. The probability distribution of these parameters is derived as a posterior distribution. Thus, we are using Bayesian decision theory. To do this, we consider using a 2D discrete prior that is uniform on the grid. To consider the uncertainty in these parameters, we consider a range of values (evenly spaced using the R function seq, a=[-0.5,2], b=[0.5,3]) for both a and b, plot a grid ($21 * 21$) of these values considering a uniform distributed prior that is for each pair of (a,b) values in this grid, we have a probability of 1/441 (using the R function matrix).

To obtain a posterior distribution, we multiply the prior matrix with the likelihood matrix (which is calculated by using a nested for loop to implement the likelihood function for each pair (a,b) for each of the data points (x_i, y_i). Using the R function image, we plot the resulted posterior distribution. An adjustment was used in the likelihood function by subtracting 18 from x so that the resulting values are both positive and negative (This adjustment decreases the dependency between a and b and improves the numerical properties).

3.2.3 Part c

Here we assume that the report is "Mature knee" and we make a function using optimal parameters a and b found in part a, the logistic report probability function f(x), the two cost functions $c_1(x), c_2(x)$ and the gamma density $\pi(x; \mu, \alpha)$ to compute the difference between the cost of classifying as a child and the cost of classifying as an adult.

$$f(x) = \frac{\exp(a + b\,(x - 18))}{1 + \exp(a + b\,(x - 18))}$$

$$c_1(x) = \begin{cases} B & if\, x \leq 18 \\ 1 & if\, x > 18 \end{cases} \qquad c_2(x) = \begin{cases} B\,(18-x) & if\, x \leq 18 \\ x-18 & if\, x > 18 \end{cases}$$

$$\pi(x;\mu,\alpha) = Gamma(x-14;\alpha,\alpha/(\mu-14)) = \begin{cases} \frac{(\frac{\alpha}{\mu-14})^{\alpha}}{\Gamma(\alpha)}\,(x-14)^{\alpha-1}\,\exp(-\frac{\alpha}{\mu-14}(x-14)) & if\, x \geq 14 \\ 0 & if\, x < 14 \end{cases}$$

For this goal we use three possibilities:

1. Age distribution 1: $\mu = 18.5$, $\alpha = 3$.
2. Age distribution 2: $\mu = 19.5$, $\alpha = 6$.
3. Age distribution 3: $\mu = 20.58$, $\alpha = 3$.

Using R functions Vectorize (a function wrapper) and integrate to formulate decision theory for this case. We will compute the expected cost of miss-classification as children (C_c) and the expected cost of miss-classification as adult (C_a).

Using the two formulas above, we can classify as children if $C_c < C_a$ while we can classify as adult if $C_c > C_a$.

$$C_c = \int_{18}^{\infty} \pi(x;\mu,\alpha) f_k(x) c(x)\, dx \qquad C_a = \int_{0}^{18} \pi(x;\mu,\alpha) f_k(x) c(x)\, dx$$

3.2.4 Part d

In this part, we consider the uncertainty in the values of parameters a and b. In the previous part, we used the values we obtained from part a with optimal a,b values resulted from the optimization. So to consider the uncertainty we use the posterior distribution we got in part b with the results obtained from the function in part c and we average these results over this distribution. The expected cost will be calculated for the three population distributions to compare the results with the ones we got from using optimal a and b. The implementation is pretty similar to part c but here we loop through ranges of a and b values (same used in part b) and then we multiply results from expected cost and the posterior distribution and average these results.

3.3 Results and discussion

3.3.1 Part a

Implementing an optimization algorithm using the negative log likelihood, the optimal parameters that maximize the likelihood for the data are: a ≈ 0.68 and b ≈ 1.72.
Using these parameters we fit a logistic regression to the data and plot the curve along with data points

Figure 1 displays the logistic regression, and we can see that the model finds the correct decision boundary between maturity and immaturity depending. The x-axis represents the age of a person and the y-axis indicates a response for each personage. A response of 1 means a mature knee and 0 indicates the immature knee report. Moreover, we can notice that with age increasing the response is 1 (is most common) while for younger people we have the response 0 becomes more common, but in the middle, we have a bit confusing/mixed classes and it seems a bit hard to classify on these ages. We can say that the value of a yields Pr when X is zero, and b adjusts how quickly the probability changes with changing X one unit.

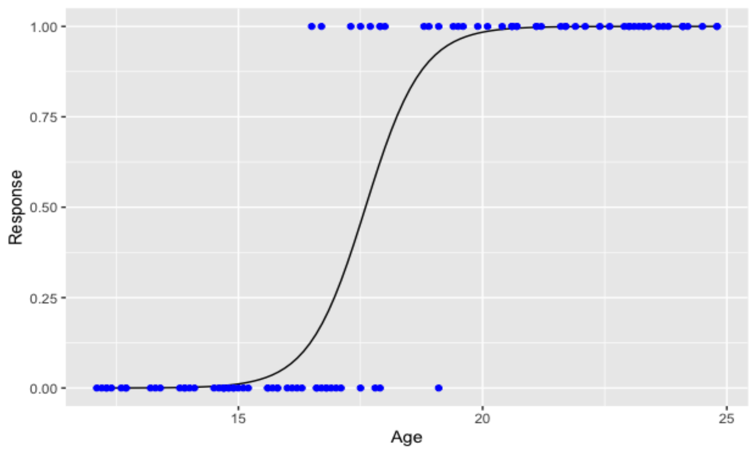

Figure 1: The "Maturity of knees" data together with LR curve using optimal parameters a, b.

3.3.2 Part b

Figure 2 shows the posterior distribution resulted from multiplying the uniform prior and the likelihood, and the concentration is around a ≈ 0.7 and b ≈ 1.7, these values are pretty similar to the values we obtained in part a.

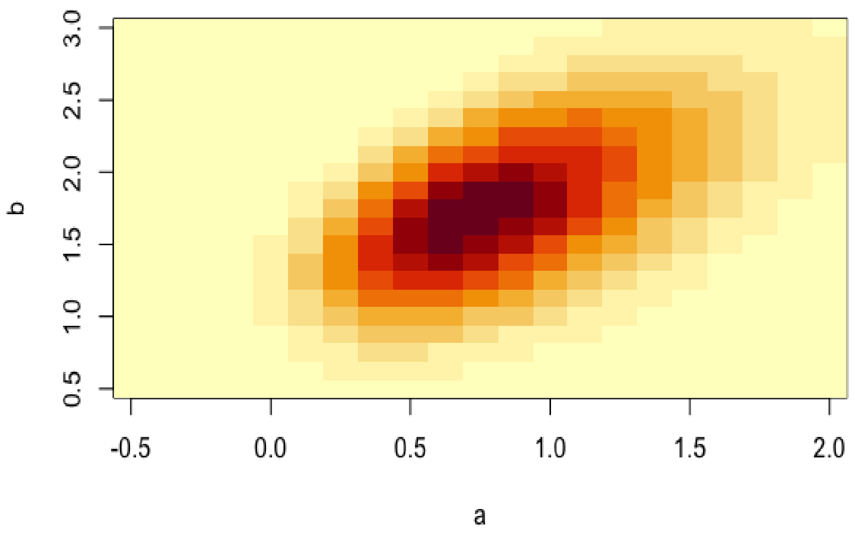

Figure 2: The posterior over a range of parameters a and b in the logistic regression.

3.3.3 Part c

Figure 3 shows the age distribution for the three possibilities. The μ parameter is the average mean while the α parameter related to the spread of the data, and we can notice with larger α the red curve is more spread and more wide around the average age.

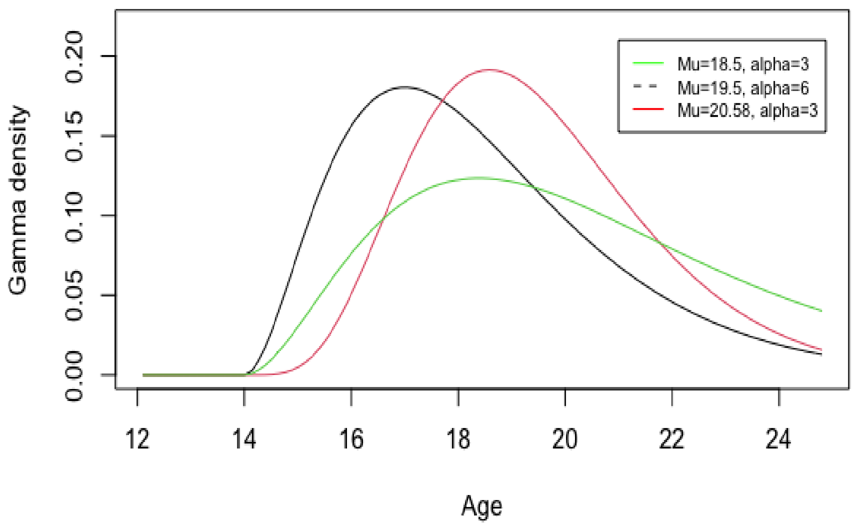

Figure 3: Age distribution. The red curve shows population distributions with expectation $\mu = 19.5$ and $\alpha = 6$. The black curve shows population distributions with expectation $\mu = 18.5$ and $\alpha = 3$. The green curve shows population distributions with expectation $\mu = 20.58$ and $\alpha = 3$.

Table 1: Results of classification using $c_1(x)$ function.

Age distribution	$C_c > C_a$
$\mu = 18.5,\ \alpha = 3$	False
$\mu = 19.5,\ \alpha = 6$	False
$\mu = 20.58,\ \alpha = 3$	False

Table 2: Results of classification using $c_2(x)$ function.

Age distribution	$C_c > C_a$
$\mu = 18.5,\ \alpha = 3$	True
$\mu = 19.5,\ \alpha = 6$	True
$\mu = 20.58,\ \alpha = 3$	True

Comparing the results (From **Table 1** and **Table 2**) and we only using the optimal values of a and b, when using $\boldsymbol{c_1(x)}$ and $\boldsymbol{c_2(x)}$, We can notice that using $\boldsymbol{c_1(x)}$ results in classifying as children in all the age distributions while using $\boldsymbol{c_2(x)}$ lead to classifying as adult in all these possibilities.

In $\boldsymbol{c_1(x)}$, this cost function doesn't depend on the actual age, the ration of this cost is B=10. In most legal systems, it considered as most problematic to classify a child as an adult rather than classify an adult as a child. But in $\boldsymbol{c_2(x)}$ for example, if we classify x=17.9 as an adult is not as problematic for that person as if classifying a person who is 16 as an adult.

3.3.4 Part d

Table 3 and **Table 4** show the results taking into consideration the uncertainty in a and b. We can notice that for $\boldsymbol{c_1(x)}$ we have similar results where we can classify as children for age distribution 18.5 and 19.5, but the last value with age distribution 20.85 we got here a classification as an adult.

In $\boldsymbol{c_2(x)}$ we have similar results in all age distribution. We can say that, considering the uncertainty leads to clearer results especially in the case when we consider the cost of classifying a child as an adult rather than classify an adult as a child is more problematic.

We know that the age for adults is above 19 (unless national law defines a person as an adult at an earlier age) and considering the uncertainty seems better looking at the results, since using only one value of a and b (optimal values that maximize the likelihood for the data) could lead to inaccurate results, for example in case the data changed a little bit these values might not represent this new data or fit it correctly.

Table 3: Results of classification using $c_1(x)$ function.

Age distribution	$C_c > C_a$
$\mu = 18.5$, $\alpha = 3$	False
$\mu = 19.5$, $\alpha = 6$	False
$\mu = 20.58$, $\alpha = 3$	True

Table 4: Results of classification using $c_2(x)$ function.

Age distribution	$C_c > C_a$
$\mu = 18.5$, $\alpha = 3$	True
$\mu = 19.5$, $\alpha = 6$	True
$\mu = 20.58$, $\alpha = 3$	True

Appendix - code

```
################################################################################################

### Q1

#load a package
library('LearnBayes')
library('mnormt')

## part (a)

# load the "stockvalues" dataset
stock_values <- as.matrix(read.csv("stockvalues.txt"))

log_stock <- log(stock_values) #logged prices

log_stock1 <- log_stock[-1,] # delete the first row from the log_stock data
#new_row <- tail(log_stock, n=1)
log_stock2 <- log_stock[-1006,] # delete the last row from the log_stock
    data

percent_change <- log_stock1 - log_stock2 #subtract the previous day data
    from current day data
#calculate the expectation and covariance matrix from logged data
#summary(log_stock)
colmean <- colMeans(percent_change) #find the mean in each column
mean <- mean(colmean) # the mean of the log prices
#mean

x = percent_change
cov_matrix <- cov(x) # the covariance matrix of log prices
#Generate a sample of V1,......V1000
n_days = 100
n_samples = 1000
Vq <- rmnorm(n_samples, n_days*colmean, n_days*cov_matrix)
#dimnames(Vq) <- NULL

## part (b)

## 1) each stock receives equal investment, w1=w2....=w7 = 1/7 since all
    weights add up to 1
w = cbind(1/7,1/7,1/7,1/7,1/7,1/7,1/7)
m = 7
k1 <- -0.5
k2 <- 0.5
k3 <- 1.5

#vq <- as.matrix(Vq)
#Vq <- matrix(1,2,7)
#the expected utility for k1
a2 <- 0.0
for (q in 1:n_samples){
  a <- 0.0
  for (i in 1:m){
    tmp <- (w[i]*exp(Vq[q,i]))
```

```
      a <- a + tmp
    }
    tmp <- 1 - (a^-k1)
    tmp <- tmp / k1

    a2 <- a2 + tmp
}
output1 <- a2 / n_samples


#the expected utility for k2
b2 = 0.0
for (q in 1:n_samples){
    b <- 0.0
    for (i in 1:m){
      tmp <- (w[i]*exp(Vq[q,i]))
      b <- b + tmp
    }
    tmp <- 1 - (b^-k2)
    tmp <- tmp / k2

    b2 <- b2 + tmp
}
output2 = b2 / n_samples


#the expected utility for k3
c2 <- 0.0
for (q in 1:n_samples){
    c <- 0.0
    for (i in 1:m){
      tmp <- (w[i]*exp(Vq[q,i]))
      c <- c + tmp
    }
    tmp <- 1 - (c^-k3)
    tmp <- tmp / k3

    c2 <- c2 + tmp
}
output3 = c2 / n_samples

as.double(output1) #print the output as number
as.double(output2)
as.double(output3)

## 2) The stock with the best expected performance receives all the
    investment

best_stock <- max(colmean)
best_stock #S4 has the highest mean so we can say it has the best
    performance

sw <- 1
#the expected utility for S4 with the 3 k's
d = e = f = 0
for (i in (1:1000)){
      d = d + ((1-((sw*exp(Vq[i,4]))^-k1)) / k1)
      e = e + ((1-((sw*exp(Vq[i,4]))^-k2)) / k2)
    f = f + ((1-((sw*exp(Vq[i,4]))^-k3)) / k3)}

u4_k1 <- as.double(d/n_samples)
u4_k2 <- as.double(e/n_samples)
u4_k3 <- as.double(f/n_samples)
#print results in a nice way :)
cat('S4_k1:', u4_k1, '\nS4_k2:', u4_k2, '\nS4_k3:', u4_k3)


## part (c)

library('Rcpp')
library('optimization')
library('ggplot2')

#convert percent_change data to a dataframe
pc <- as.data.frame(percent_change)

#use the optimize function to find optimal weights for S3 and S4
#first we define functions for both k's and for both stocks x1= pc$S3 and
    x2=pc$S4,
#w is the weight and since weights add up to 1 if S3 has w then S4 will
    have 1-w
u_stock1 <- function(w, x1, x2) {
  sum((((1-(((w*exp(x1)) + (((1-w)*exp(x2))))^0.5)) / -0.5)) / n_samples
  }
opt_w1 <-  optimize(u_stock1, lower = 0, upper = 1, x1= pc$S3, x2=pc$S4,
    maximum = TRUE)

u_stock2 <- function(w, x1, x2) {
  sum((((1-(((w*exp(x1)) + (((1-w)*exp(x2))))^-1.5)) / 1.5)) / n_samples
  }
opt_w2 <- optimize(u_stock2, lower = 0, upper = 1, x1= pc$S3, x2=pc$S4,
    maximum = TRUE)

opt_w1
opt_w2

## part (d)
#Written question

################################################################################

### Q2

# load the "stockvalues" dataset

mature_Knee <- read.table("matureKnee.txt")
#add y of 1 to the data as a column and add it to the data
mm <- rep(1, 50)
mk <- cbind(mature_Knee, mm)
#change column name to match the column name for the immatureKnee.txt below
colnames(mk)[2] <- "m"

immature_Knee <- read.table("immatureKnee.txt")
#add y of 0 to the data as a column and add it to the data
im <- rep(0, 50)
imk <- cbind(immature_Knee, im)
```

```
#change column name to match the column name for the matureKnee.txt above
colnames(imk)[2] <- "m"

#concatenate both data into one dataframe
all_ages <- rbind(imk,mk)
attach(all_ages)

#mean age for each report
m_mean <- mean(as.matrix(read.csv(('matureKnee.txt'))))
im_mean <- mean(as.matrix(read.csv(('immatureKnee.txt'))))
#print means in each data
cat('Mature_knee mean age:', m_mean, '\nImmature_knee mean age:', im_mean)

## part (a)

#define start points for a and b to use in optim function
betas <- c(0, 0)
#define a function for the NLL to minimize
loglike <- function(par,x,y) {
  a <- par[1]
  b <- par[2]
  # Define the logistic function
  logit <- function(x,a,b) {
    exp(a+b*(x-18))/(1 + exp(a+b*(x-18)))
  }
  p <- logit(x,a,b)
  - sum(y*log(p) + (1-y)*log(1-p))
}
#find the optimal a and b using optim function
MLE <- optim(betas, loglike, x = all_ages$V1, y = all_ages$m)
MLE$par #print the optimal values of a and b

x <- all_ages$V1 #the age values
y <- all_ages$m # the 0,1 values

best_a <- MLE$par[1] #optimal a value
best_b <- MLE$par[2] #optimal b value

#fit logistic regression to the data with optimal a and b and plot
p <- ggplot(data=all_ages, aes(V1, m))
scurve <- function(x){
  y <- exp(best_a+best_b*(x-18))/(1 + exp(best_a+best_b*(x-18)))
  return(y)
}
p + stat_function(fun = scurve, n = 100) + xlab('Age') +
  ylab('Response') + geom_point(color="blue")



## part (b)

#define seq for a and b equally distanced by 21
a <- seq(from = -0.5, to = 2, length.out = 21)
b <- seq(from = 0.5, to = 3, length.out = 21)

#define prior uniform with prob = 1/441
uniform <- matrix(1/441, nrow=21, ncol = 21)
```

```r
prob <- matrix(0, nrow=21, ncol = 21) #21*21 matrix to store the likelihood
    function results
#find likelihood values for each (a,b) pair
for (i in (1:21)){
  for (k in (1:21)){
    p <- 1
    for (j in (1:100)){
      p <- p * (((exp(a[i] + b[k]*(x[j]-18)) / (1 + (exp(a[i] + b[k]*(x[j
          ]-18)))))^y[j]) *
                   ((1 - (exp(a[i] + b[k]*(x[j]-18)) / (1 + (exp(a[i] + b[k
                      ]*(x[j]-18))))))^(1-y[j])))
    }
    prob[k, i] <- p
  }
}

#find posterior by multiplying prior and likelihood matrices
pp <- uniform * prob
pp <- pp / sum(pp)

#plot the posterior
image(a, b, pp)


## part (c)

#age distributions
mu_ <- cbind(18.5, 19.5, 20.58)
alpha_ <- cbind(3, 6, 3)


#plot(x, dgamma(x=x-14, shape = alpha_[1], rate = alpha_[1]/(mu_[1]-14)),
 #    ylab = "Gamma density", xlab = 'Age')
#plot(x, dgamma(x=x-14, shape = alpha_[2], rate = alpha_[2]/(mu_[2]-14)),
 #    ylab = "Gamma density", xlab = 'Age')
#plot(x, dgamma(x=x-14, shape = alpha_[3], rate = alpha_[3]/(mu_[3]-14)),
 #    ylab = "Gamma density", xlab = 'Age')

#plot age distributions
plot(0, 0, xlim = c(min(x), max(x)), ylim = c(0, 0.22), type = "n",
     ylab = "Gamma density", xlab = 'Age')
for(i in 1:length(mu_))
  curve(dgamma(x=x-14, shape = alpha_[i], rate = alpha_[i]/(mu_[i]-14)),
        from = min(x), to = max(x), col = i, add = TRUE)

legend(21, 0.21, legend=c("Mu=18.5, alpha=3", "Mu=19.5, alpha=6", "Mu
    =20.58, alpha=3"),
       col=c("green", "black", "red"), lty=1:2, cex=0.7)


#for classification children
#gamma density function
agedens <- function(x, alpha, mu) {dgamma(x=x-14, shape=alpha, rate=alpha/(
    mu-14))}
#cost function
c1 <- function(x) {if (x<=18) {10} else {1}}
c2 <- function(x) {if (x<=18) {10*(18-x)} else {(x-18)}}
```

```r
#Probability function
f1 <- function(x, a, b) {(1 / (1 + exp(-a-b*(x-18))))}

#using c1
integrate_me <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c1(x)}
integrate_me <- Vectorize(integrate_me, "x")
# Define alpha1, mu1, a and b
mus <- cbind(18.5, 19.5, 20.58); alphas <- cbind(3, 6, 3)
mu <- mus[1]; alpha <- alphas[1]; a <- best_a; b <- best_b
int_c1<- integrate(integrate_me, 18, Inf)
# Define alpha2, mu2, a and b
mu <- mus[2]; alpha <- alphas[2]; a <- best_a; b <- best_b
int_c2<- integrate(integrate_me, 18, Inf)
# Define alpha3, mu3, a and b
mu <- mus[3]; alpha <- alphas[3]; a <- best_a; b <- best_b
int_c3<- integrate(integrate_me, 18, Inf)

cls_child1 = int_c1[[1]]; cls_child2 = int_c2[[1]]; cls_child3 = int_c3
    [[1]]

#using adults c1
integrate_m <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c1(x)}
integrate_m <- Vectorize(integrate_m, "x")
# Define alpha1, mu1, a and b
mu <- mus[1]; alpha <- alphas[1]; a <- best_a; b <- best_b
int_a1<- integrate(integrate_m, 0, 18)
# Define alpha2, mu2, a and b
mu <- mus[2]; alpha <- alphas[2]; a <- best_a; b <- best_b
int_a2<- integrate(integrate_m, 0, 18)
# Define alpha3, mu3, a and b
mu <- mus[3]; alpha <- alphas[3]; a <- best_a; b <- best_b
int_a3<- integrate(integrate_m, 0, 18)

cls_adult1 = int_a1[[1]];cls_adult2 = int_a2[[1]];cls_adult3 = int_a3[[1]]

#compare results using c1 cost

cat("Using c1:",
  "\nClassify as adult for mu = 18.5?", cls_child1 > cls_adult1,
    "\nClassify as adult for mu = 19.5?", cls_child1 > cls_adult1,
    "\nClassify as adult for mu = 20.58?", cls_child1 > cls_adult1)

#using children c2
integrate_me <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c2(x)}
integrate_me <- Vectorize(integrate_me, "x")
# Define alpha1, mu1, a and b
mu <- mus[1]; alpha <- alphas[1]; a <- best_a; b <- best_b
int_c11<- integrate(integrate_me, 18, Inf)
# Define alpha2, mu2, a and b
mu <- mus[2]; alpha <- alphas[2]; a <- best_a; b <- best_b
int_c22<- integrate(integrate_me, 18, Inf)
# Define alpha3, mu3, a and b
mu <- mus[3]; alpha <- alphas[3]; a <- best_a; b <- best_b
int_c33<- integrate(integrate_me, 18, Inf)

cls_child11 = int_c11[[1]]; cls_child22 = int_c22[[1]]; cls_child33 =
    int_c33[[1]]
```

```
#using c2 adults
integratee <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c2(x)}
integratee <- Vectorize(integratee, "x")
# Define alpha1, mu1, a and b
mu <- mus[1]; alpha <- alphas[1]; a <- best_a; b <- best_b
int_a11<- integrate(integratee, 0, 18)
# Define alpha2, mu2, a and b
mu <- mus[2]; alpha <- alphas[2]; a <- best_a; b <- best_b
int_a22<- integrate(integratee, 0, 18)
# Define alpha3, mu3, a and b
mu <- mus[3]; alpha <- alphas[3]; a <- best_a; b <- best_b
int_a33<- integrate(integratee, 0, 18)

cls_adult11 = int_a11[[1]]; cls_adult22 = int_a22[[1]]; cls_adult33 =
    int_a33[[1]]

#compare results using c1 cost

cat("Using c2:",
    "\nClassify as adult for mu = 18.5?", cls_child11 > cls_adult11,
    "\nClassify as adult for mu = 19.5?", cls_child22 > cls_adult22,
    "\nClassify as adult for mu = 20.58?", cls_child33 > cls_adult33)



## part (d) (similar to part c but we average over the posterior
    distribution)

a_seq <- seq(from = -0.5, to = 2, length.out = 21)
b_seq <- seq(from = 0.5, to = 3, length.out = 21)

mus <- cbind(18.5, 19.5, 20.58); alphas <- cbind(3, 6, 3)
##children with c1
integrate_me <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c1(x)}
integrate_me <- Vectorize(integrate_me, "x")
c1_seq = matrix(, nrow = 21, ncol = 21)
c2_seq = matrix(, nrow = 21, ncol = 21)
c3_seq = matrix(, nrow = 21, ncol = 21)
#here we loop through all a and b values to consider uncertainty
for(i in 1:length(a_seq)){
  for(j in 1:length(b_seq)){
    # Define alpha1, mu1, a and b
    a_val = a_seq[i]
    b_val = b_seq[j]
    mu <- mus[1]; alpha <- alphas[1]; a <- a_val; b <- b_val
    int_c1<- integrate(integrate_me, 18, Inf)
    c1_seq[j,i] <- int_c1[[1]]
    # Define alpha2, mu2, a and b
    mu <- mus[2]; alpha <- alphas[2]; a <- a_val; b <- b_val
    int_c2<- integrate(integrate_me, 18, Inf)
    c2_seq[j,i] <- int_c2[[1]]
    # Define alpha3, mu3, a and b
    mu <- mus[3]; alpha <- alphas[3]; a <- a_val; b <- b_val
    int_c3<- integrate(integrate_me, 18, Inf)
    c3_seq[j,i] <- int_c3[[1]]
  }
}
##adults with c1
```

```
integratee <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c1(x)}
integratee <- Vectorize(integratee, "x")
a1_seq = matrix(, nrow = 21, ncol = 21)
a2_seq = matrix(, nrow = 21, ncol = 21)
a3_seq = matrix(, nrow = 21, ncol = 21)
for(i in 1:length(a_seq)){
  for(j in 1:length(b_seq)){
    # Define alpha1, mu1, a and b
     a_val = a_seq[i]
     b_val = b_seq[j]
    mu <- mus[1]; alpha <- alphas[1]; a <- a_val; b <- b_val
     int_c1<- integrate(integratee, 0, 18)
     a1_seq[j,i] <- int_c1[[1]]
    # Define alpha2, mu2, a and b
    mu <- mus[2]; alpha <- alphas[2]; a <- a_val; b <- b_val
     int_c2<- integrate(integratee, 0, 18)
     a2_seq[j,i] <- int_c2[[1]]
    # Define alpha3, mu3, a and b
    mu <- mus[3]; alpha <- alphas[3]; a <- a_val; b <- b_val
     int_c3<- integrate(integratee, 0, 18)
     a3_seq[j,i] <- int_c3[[1]]
  }
}
c1_children1 = mean(c1_seq * pp); c1_children2 = mean(c2_seq * pp);
    c1_children3  = mean(c3_seq * pp)

c1_adults1  = mean(a1_seq * pp); c1_adults2 = mean(a2_seq * pp);c1_adults3
   = mean(a3_seq * pp)

#compare results with c1

cat("Using c1:",
     "\nClassify as adult for mu = 18.5?", c1_children1 > c1_adults1,
     "\nClassify as adult for mu = 19.5?", c1_children2 > c1_adults2,
     "\nClassify as adult for mu = 20.58?", c1_children3 > c1_adults3)


#children with c2
integrate_me2 <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c2(x)}
integrate_me2 <- Vectorize(integrate_me2, "x")
c12_seq = matrix(, nrow = 21, ncol = 21)
c22_seq = matrix(, nrow = 21, ncol = 21)
c32_seq = matrix(, nrow = 21, ncol = 21)
for(i in 1:length(a_seq)){
  for(j in 1:length(b_seq)){
    # Define alpha1, mu1, a and b
     a_val = a_seq[i]
     b_val = b_seq[j]
    mu <- mus[1]; alpha <- alphas[1]; a <- a_val; b <- b_val
     int_c1<- integrate(integrate_me2, 18, Inf)
     c12_seq[j,i] <- int_c1[[1]]
    # Define alpha2, mu2, a and b
    mu <- mus[2]; alpha <- alphas[2]; a <- a_val; b <- b_val
     int_c2<- integrate(integrate_me2, 18, Inf)
     c22_seq[j,i] <- int_c2[[1]]
    # Define alpha3, mu3, a and b
    mu <- mus[3]; alpha <- alphas[3]; a <- a_val; b <- b_val
     int_c3<- integrate(integrate_me2, 18, Inf)
    c32_seq[j,i] <- int_c3[[1]]
  }
}

#adults with c2
integratee1 <- function(x) {agedens(x, alpha, mu) * f1(x, a, b) * c2(x)}
integratee1 <- Vectorize(integratee1, "x")
a12_seq = matrix(, nrow = 21, ncol = 21)
a22_seq = matrix(, nrow = 21, ncol = 21)
a32_seq = matrix(, nrow = 21, ncol = 21)
for(i in 1:length(a_seq)){
  for(j in 1:length(b_seq)){
    # Define alpha1, mu1, a and b
    a_val = a_seq[i]
    b_val = b_seq[j]
    mu <- mus[1]; alpha <- alphas[1]; a <- a_val; b <- b_val
    int_c1<- integrate(integratee1, 0, 18)
    a12_seq[j,i] <- int_c1[[1]]
    # Define alpha2, mu2, a and b
    mu <- mus[2]; alpha <- alphas[2]; a <- a_val; b <- b_val
    int_c2<- integrate(integratee1, 0, 18)
    a22_seq[j,i] <- int_c2[[1]]
    # Define alpha3, mu3, a and b
    mu <- mus[3]; alpha <- alphas[3]; a <- a_val; b <- b_val
    int_c3<- integrate(integratee1, 0, 18)
    a32_seq[j,i] <- int_c3[[1]]
  }
}
c2_children1 = mean(c12_seq * pp); c2_children2 = mean(c22_seq * pp);
   c2_children3 = mean(c32_seq * pp)

c2_adults1 = mean(a12_seq * pp); c2_adults2 = mean(a22_seq * pp);
   c2_adults3= mean(a32_seq * pp)

#compare results with c2

cat("Using c2:",
    "\nClassify as adult for mu = 18.5?", c2_children1 > c2_adults1,
    "\nClassify as adult for mu = 19.5?", c2_children2 > c2_adults2,
    "\nClassify as adult for mu = 20.58?", c2_children3 > c2_adults3)
```