Question 1

Jin Barai

```
library(carData)
data(TitanicSurvival)
Titanic = na.omit(TitanicSurvival)
Titanic = Titanic[Titanic$sex == "female", ]
Titanic$survived1 = as.numeric(Titanic$survived == "yes")
head(Titanic)
```

```
##                                  survived    sex age passengerClass survived1
## Allen, Miss. Elisabeth Walton        yes female  29            1st         1
## Allison, Miss. Helen Loraine          no female   2            1st         0
## Allison, Mrs. Hudson J C (Bessi       no female  25            1st         0
## Andrews, Miss. Kornelia Theodos      yes female  63            1st         1
## Appleton, Mrs. Edward Dale (Cha      yes female  53            1st         1
## Astor, Mrs. John Jacob (Madelei      yes female  18            1st         1
```

(a)

```
par(mfcol = c(1,2))
age = Titanic$age
rx = range(age)
hist(age, breaks=seq(0, 80, 8), prob=TRUE, xlab="Female passenger's Age", main="Bin width equal to 8 yea
hist(age, breaks=quantile(age, p=seq(0,1,length.out=11)), prob=TRUE, main="Varying bin width", xlab="Fem
```

(b)

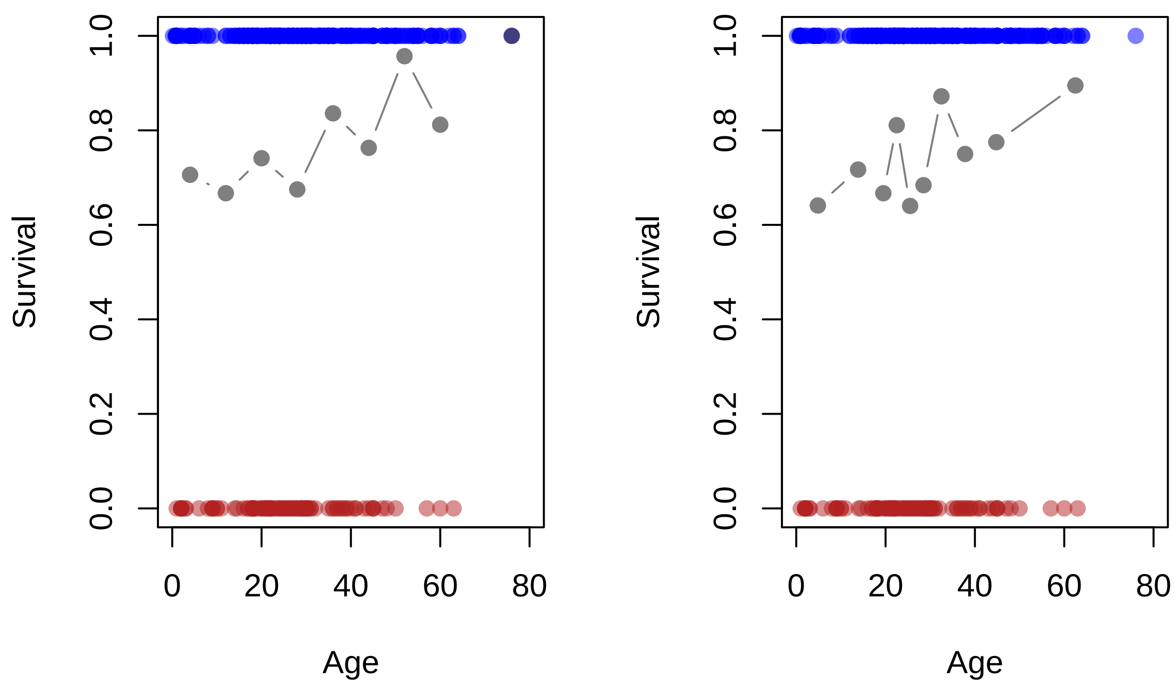
```
survived = Titanic$survived1

plot.fun <- function (agegroup){
  plot(age, survived, pch=19,col=c(adjustcolor("firebrick",0.5), adjustcolor("blue", 0.5))[survived +1]
  xlim=c(0,80), xlab="Age", ylab="Survival")
  x = matrix(0, nrow=length(agegroup)-1, ncol=5)
  dimnames(x)[[2]] = c("Lower", "Upper", "Total", "Num.Survived", "Prop.Survived")
  for (i in 1:nrow(x)) {
  x[i,1:2] = c(agegroup[i], agegroup[i+1])
  x[i,3] = sum(age > agegroup[i] & age <= agegroup[i+1])
  x[i,4] = sum(survived[age > agegroup[i] & age <= agegroup[i+1]] )
  x[i,5] = round(x[i,4]/x[i,3],3)
}
  points(agegroup[-length(agegroup)]+ diff(agegroup)/2, x[,5], pch=19, col=adjustcolor("black", 0.5),ty
  x
}
```

```
par(mfcol = c(1,2))
agegroup1 = seq(0,80,8)
agegroup2 = quantile(age, seq(0,1,length.out=11))
agegroup2[1] = 0
plot.fun(agegroup1)
```

```
##       Lower Upper Total Num.Survived Prop.Survived
##  [1,]     0     8    34           24         0.706
##  [2,]     8    16    30           20         0.667
##  [3,]    16    24   108           80         0.741
##  [4,]    24    32    77           52         0.675
##  [5,]    32    40    61           51         0.836
##  [6,]    40    48    38           29         0.763
##  [7,]    48    56    23           22         0.957
##  [8,]    56    64    16           13         0.812
##  [9,]    64    72     0            0           NaN
## [10,]    72    80     1            1         1.000
```

```r
plot.fun(agegroup2)
```




```
##       Lower Upper Total Num.Survived Prop.Survived
##  [1,]   0.0   9.7    39           25         0.641
##  [2,]   9.7  18.0    53           38         0.717
##  [3,]  18.0  21.0    27           18         0.667
##  [4,]  21.0  24.0    53           43         0.811
##  [5,]  24.0  27.0    25           16         0.640
##  [6,]  27.0  30.0    38           26         0.684
##  [7,]  30.0  35.0    39           34         0.872
##  [8,]  35.0  40.6    36           27         0.750
##  [9,]  40.6  49.0    40           31         0.775
## [10,]  49.0  76.0    38           34         0.895
```

We would prefer the age range with varying lengths. This is because there are roughly the same number of units within each age group. For Equal bin widths, the total number of units in each age group varies significantly.

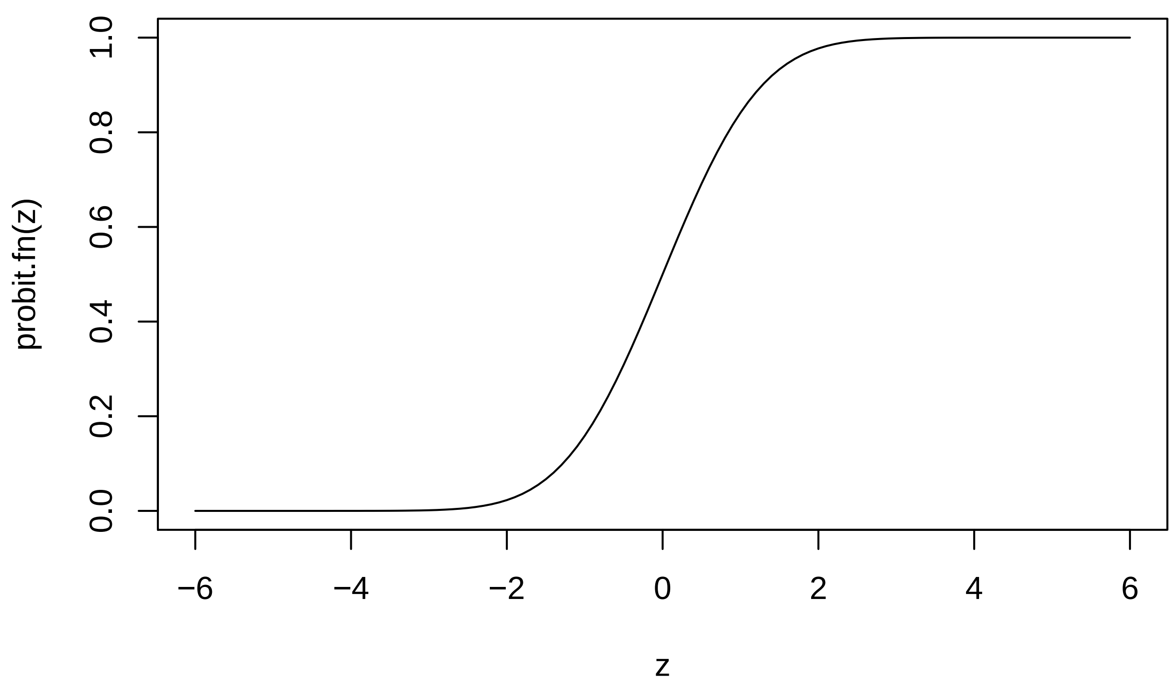
(c)

i.

```
probit.fn <- function(z) {
  pnorm(z)
}

z = seq(-6,6,0.1)

plot(z,probit.fn(z),type='l')
```



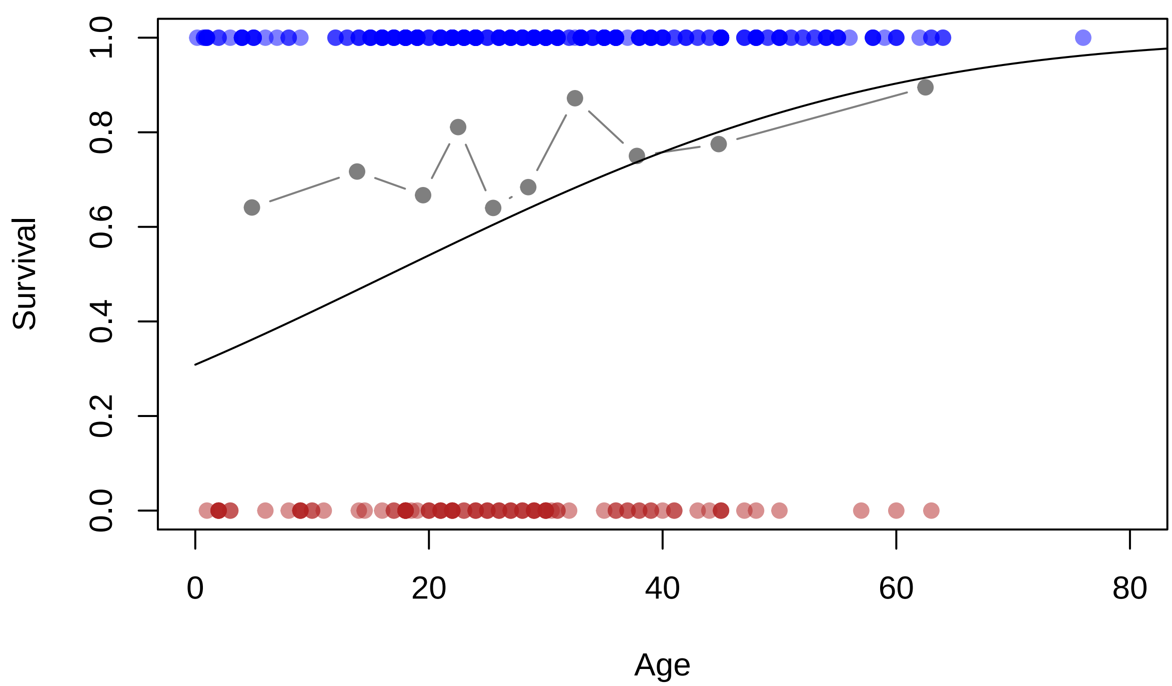
ii.

```
plot.fun(agegroup2)
```

```
##      Lower Upper Total Num.Survived Prop.Survived
## [1,]   0.0   9.7    39           25         0.641
## [2,]   9.7  18.0    53           38         0.717
## [3,]  18.0  21.0    27           18         0.667
## [4,]  21.0  24.0    53           43         0.811
```

```
## [5,] 24.0 27.0  25          16          0.640
## [6,] 27.0 30.0  38          26          0.684
## [7,] 30.0 35.0  39          34          0.872
## [8,] 35.0 40.6  36          27          0.750
## [9,] 40.6 49.0  40          31          0.775
## [10,] 49.0 76.0 38          34          0.895
```

```r
z = seq(0,300,0.1)
lines(z,probit.fn(-1/2 + 0.03*z))
```



(d)

The log-likelihood is given by

$$l(\boldsymbol{\theta}) = l(\alpha, \beta) = \sum_{i=1}^{N} \left[y_i \log \frac{p_i}{1 - p_i} + \log(1 - p_i) \right]$$

where

$$p_i = \boldsymbol{\Phi}(\hat{y}) = \boldsymbol{\Phi}(\alpha + \beta[x_i - \overline{x}]) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(\alpha + \beta[x_i - \overline{x})^2]}{2}}$$

Now we proceed by finding partial derivatives with respect to α and β

$$\frac{\partial l}{\partial \alpha} = \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \alpha}$$

$$\frac{\partial l}{\partial \beta} = \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \beta}$$

$$\frac{\partial l}{\partial p_i} = \frac{\partial}{\partial p_i} \left[y_i log \frac{p_i}{1-p_i} - \frac{1}{1-p_i} \right] = \frac{y_i - p_i}{p_i(1-p_i)}$$

$$\mathbf{\Phi}(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \Rightarrow \mathbf{\Phi}'(z) = \phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$$

$$p = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \Rightarrow \frac{\partial p}{\partial z} = \frac{1}{\sqrt{2\pi}} e^{-z^2/2} = \phi(z)$$

$$\frac{\partial z}{\partial \alpha} = \frac{\partial}{\partial \beta}(\alpha + \beta[x_i - \overline{x}]) = 1$$

$$\frac{\partial z}{\partial \beta} = \frac{\partial}{\partial \beta}(\alpha + \beta[x_i - \overline{x}]) = (x_i - \overline{x})$$

$$\frac{\partial l}{\partial \alpha} = \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \alpha} = \sum_{i=1}^{N} \frac{y - p_i}{p_i(1-p_i)} \times \phi(\hat{y}) \times 1$$

$$\frac{\partial l}{\partial \beta} = \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \beta} = \sum_{i=1}^{N} \frac{y - p_i}{p_i(1-p_i)} \times \phi(\hat{y}) \times (x_i - \overline{x})$$

$$\frac{\partial l}{\partial (\alpha, \beta)} = \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \alpha} + \sum_{i=1}^{N} \frac{\partial l}{\partial p_i} \frac{\partial p_i}{\partial z} \frac{\partial z}{\partial \beta} = \sum_{i=1}^{N} \frac{y - p_i}{p_i(1-p_i)} \times \phi(\hat{y}) \times \begin{bmatrix} 1 \\ (x_i - \overline{x}) \end{bmatrix}$$

(e)

i.

```
createObjBinary <- function(x,y) {
  ## local variable
  xbar <- mean(x)
  ## Return this function
  function(theta) {
  alpha <- theta[1]
  beta <- theta[2]
  y_hat = alpha + beta * (x - xbar)
  pi = probit.fn( y_hat )
  -1*sum( y*log(pi/(1-pi)) + log(1-pi) )
  }
}
```

ii.

```
createBinaryLogisticGradient <- function(x,y) {
  ## local variables
  xbar <- mean(x)
  ybar <- mean(y)
  N <- length(x)
  function(theta) {
    alpha <- theta[1]
    beta <- theta[2]
    y_hat = alpha + beta * (x - xbar)
    pi = probit.fn( y_hat )
    resids = y - pi
    -1*c( sum(resids), sum((x - xbar) * resids) )
  }
}
```

iii.

```
gradientDescent <- function(theta = 0,
                            rhoFn, gradientFn,
                            lineSearchFn, testConvergenceFn,
                            maxIterations = 100, # maximum number of iterations
                            # in gradient descent loop
                            #
                            tolerance = 1E-6, # parameters for the test
                            relative = FALSE, # for convergence function
                            #
                            lambdaStepsize = 0.01, # parameters for the line search
                            lambdaMax = 0.5 # to determine lambda
                            ) {
  ## Initialize
  converged <- FALSE
  i <- 0
  ## LOOP
  while (!converged & i <= maxIterations) {
    ## gradient
    g <- gradientFn(theta)
    ## gradient direction
    glength <- sqrt(sum(g^2))
    if (glength > 0) g <- g /glength

    ## line search for lambda
    lambda <- lineSearchFn(theta, rhoFn, g,
                          lambdaStepsize = lambdaStepsize,
                          lambdaMax = lambdaMax)
    ## Update theta
    thetaNew <- theta - lambda * g
    ##
    ## Check convergence
    converged <- testConvergenceFn(thetaNew, theta,
    tolerance = tolerance,
    relative = relative)
    ## Update
```

```r
    theta <- thetaNew
    i <- i + 1
  }
  ## Return last value and whether converged or not
  list(theta = theta,
       converged = converged,
       iteration = i,
       fnValue = rhoFn(theta)
       )
}

### line searching could be done as a simple grid search
gridLineSearch <- function(theta, rhoFn, g,
                           lambdaStepsize = 0.01,
                           lambdaMax = 1) {
  ## grid of lambda values to search
  lambdas <- seq(from = 0,
                 by = lambdaStepsize,
                 to = lambdaMax)

  ## line search
  rhoVals <- Map(function(lambda) {rhoFn(theta - lambda * g)},
                 lambdas)
  ## Return the lambda that gave the minimum
  lambdas[which.min(rhoVals)]
}

### Where testCovergence might be (relative or absolute)
testConvergence <- function(thetaNew, thetaOld, tolerance = 1E-10, relative=FALSE) {
  sum(abs(thetaNew - thetaOld)) < if (relative) tolerance * sum(abs(thetaOld)) else tolerance
}
gradient <- createBinaryLogisticGradient(age, survived)
rho <- createObjBinary(age, survived )

# c(-1.32,0.02609972 )
result <- gradientDescent(theta = c(0, 0),
                          rhoFn = rho, gradientFn = gradient,
                          lineSearchFn = gridLineSearch,
                          testConvergenceFn = testConvergence,
                          lambdaStepsize = 0.0001,
                          lambdaMax = 0.01,
                          maxIterations = 10^5)
### Print the results
Map(function(x){if (is.numeric(x)) round(x,4) else x}, result)
```

```
## $theta
## [1] 0.6787 0.0131
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 161
```

```
##
## $fnValue
## [1] 213.4404
```

The algorithm converges.

iv.

- If age has no effect on survival then $\beta = 0$
- If the proportion of females who survived the titanic is 0.717 then

$$0.717162 = \Phi(\alpha) = \int_{-\infty}^{\alpha} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \Rightarrow \alpha = 0.574$$

Hence the starting value is $(\alpha, \beta) = (0.574, 0)$

```r
result <- gradientDescent(theta = c(0.574, 0),
                          rhoFn = rho, gradientFn = gradient,
                          lineSearchFn = gridLineSearch,
                          testConvergenceFn = testConvergence,
                          lambdaStepsize = 0.0001,
                          lambdaMax = 0.01,
                          maxIterations = 10^5)
### Print the results
Map(function(x){if (is.numeric(x)) round(x,4) else x}, result)
```

```
## $theta
## [1] 0.6898 0.0133
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 76
##
## $fnValue
## [1] 213.4148
```

The number of iterations went from 161 to 76, hence an improvement.

(f)

i.

```r
temp = plot.fun(agegroup2)
z = seq(0, 80, length.out=100)
lines(z, probit.fn(0.6898 + 0.0133*(z-mean(age))))
```

ii.

```
x = apply(temp[,1:2],1,mean)
probit.probit.prop = probit.fn(0.6898 + 0.0133*(x - mean(age)))
propx = cbind(temp, probit.probit.prop)
round(propx,3)
```

iii.

- The parametric model assumes monotonic increasing or decreasing survival proportion
- The non-parametric model assumes constant survival proportions over each interval

iv.

$$p = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$

where $z = 0.6898 + 0.0133 \times (x - \overline{x})$. When there is a 50-50 chance p = 1/2 so we have

$$1/2 = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \Rightarrow z = 0$$

$$0 = 0.6898 + 0.0133 \times (x - \overline{x}) \Rightarrow x = \overline{x} - \frac{0.6898}{0.0133} = 28.68707 - \frac{0.6898}{0.0133}$$