Midterm II

Gustavo Esparza

04/02/2019

Question 1

Part A

```
x=c(1.6907,1.7242,1.7552,1.7842,1.8113,1.8369,1.8610,1.8839) # log dose
m=c(59,60,62,56,63,59,62,60) # number of beatles
y=c(6,13,18,28,52,53,61,60) # number of beatles killed
```

We will begin our IRLS algorithm by defining a Jacobian and Weight matrix for our desired parameters, β_1 and β_2. The respective partial derivatives and variance was derived in the in-class portion of the exam.

```
# Start of the IRLS algorithm

JacfW_Q1 = function(beta,X){
  p =(1-exp(-exp(beta[1]+beta[2]*x) ) )
  q=1-p
  f=m*p

  db1 = m*exp(-exp(beta[1]+beta[2]*x))*exp(beta[1]+beta[2]*x)
  db2 = m*exp(-exp(beta[1]+beta[2]*x))*exp(beta[1]+beta[2]*x)*(x)

  J =cbind(db1,db2)
  W = diag(1/(m*p*q))
  list(f=f, J=J, W=W) }
```

Now, we will create our IRLS function to find the MLE's of β_1 and β_2.

```
GN_Q1 = function(y,X, beta0, Jac, Wt = 1, maxit,IRLS = TRUE){
cat("Iteration", "             b1",
    "                    b2", "             MRE\n")

  for (it in 1: maxit){
    a = do.call(Jac, list(beta0, X))
    J = a$J
    f = a$f
    if (IRLS==TRUE){
      Wt = a$W }
    JW = t(J) %*% Wt
    dir = solve(JW %*% J) %*% JW %*% (y-f)
    beta1 = beta0 + dir

    relerr = norm(beta1-beta0)/max(1,norm(beta1))

cat(sprintf(' %2.0f                    %6.6f              %6.6f     %1.1e\n'
                ,it,beta0[1],beta0[2],relerr))
    beta0=beta1
```

```
    if(relerr<1e-8){break}
  }
  MLE_beta<<-beta0
  }
```

Now, we will use 20 iterations and a starting value of $\beta_1 = \beta_2 = 0$.

```
maxit=20
beta0 = c(0,0)
result= GN_Q1(y,x, beta0, 'JacfW_Q1', Wt = 1, maxit, IRLS = TRUE)
```

```
## Iteration             b1                       b2              MRE
##   1                  0.000000              0.000000     1.0e+00
##   2                  -26.085216              14.500290     2.6e-01
##   3                  -35.082296              19.517504     9.6e-02
##   4                  -38.808628              21.610672     1.9e-02
##   5                  -39.549855              22.028456     5.7e-04
##   6                  -39.572454              22.041249     3.6e-06
##   7                  -39.572310              22.041169     2.5e-08
##   8                  -39.572311              22.041170     1.8e-10
```

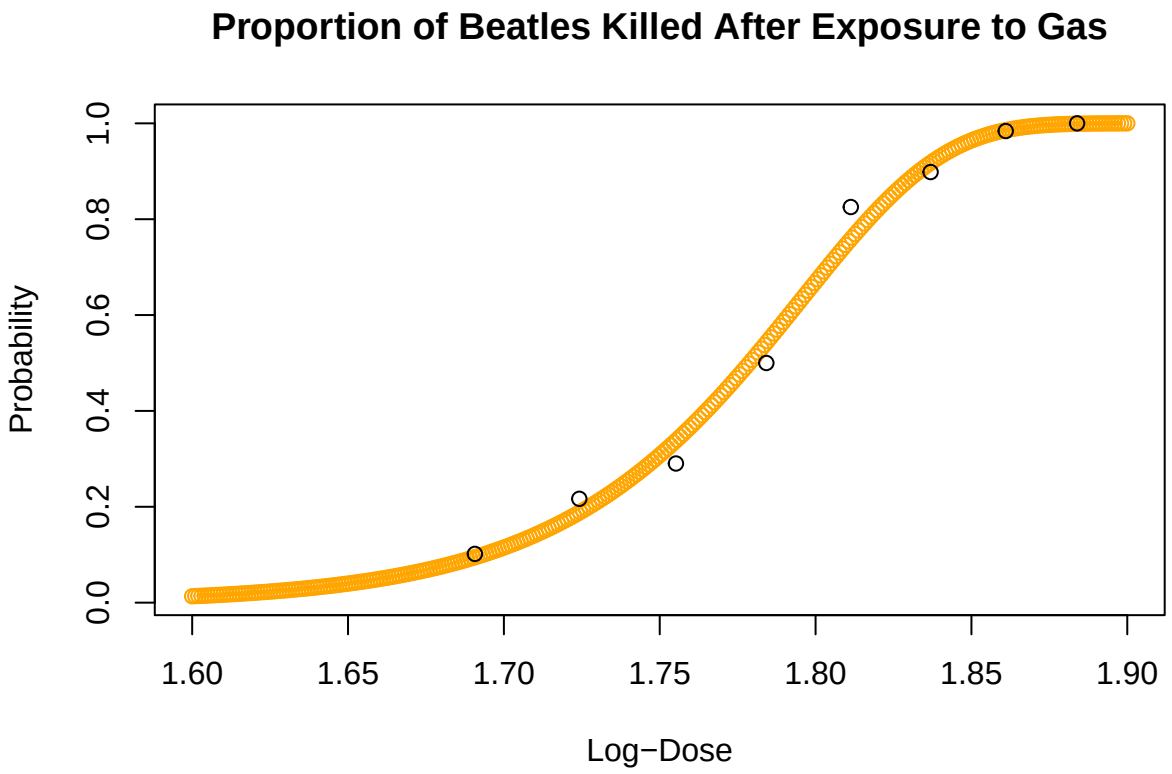
We can see that the MLE estimates are found after 7 iterations. Thus, our MLE estimates are $\hat{\beta}_1$ = -39.57231 and $\hat{\beta}_2$ = 22.04117.

Part B

We will plot the probability of positive response by using our MLE estimates and our provided probability function.

```
beta_mle = c(-39.572311,22.04117)
```

```
dose = seq(1.6,1.9, length=300)
probability= (1-exp(-exp(beta_mle[1]+beta_mle[2]*dose) ) )

plot(dose,probability, xlab="Log-Dose", ylab="Probability",
     main="Proportion of Beatles Killed After Exposure to Gas",
     col="orange")
points(x,y/m)
```



Thus, we can see that there is a positive relationship between Log-Dose and the probability of death. The relationship also seems to increase as the dosage increases, before ultimately plateauing at a 100% kill-rate. We can also see that the model using the MLE estimates fits the actual data quite well.

Part C

We will predict the probability of death by using our provided probability function, and inputting our MLE estimates and desired Log-Dose.

```
prediction = (1-exp(-exp(beta_mle[1]+beta_mle[2]*1.8) ) )

prediction
```

```
## [1] 0.6695026
```

Thus, we can see that our predicted death rate for a log-dose level of 1.8 is about 67%.

Part D

In order to find the dose level at which 50% of the beetles experience death, we must rearrange our probability expression to solve for the log-dose value, x_i. The final result is as follows:

$$x_{.50} = \frac{log(-log(1-\pi_{.50})) - \beta_1}{\beta_2} = \frac{log(-log(.50)) - \beta_1}{\beta_2}$$

```
LD50 = (log(-log(.5))-beta_mle[1])/beta_mle[2]

LD50
```

```
## [1] 1.778753
```

Thus, we can expect 50% of the beetles to die when the Log-Dose value is equivalent to 1.778753. The data provided has 1.7842 valued to have a kill rate of 50%, so our model estimate shows to be quite accurate but since the dose is on a log scale with few observations we can expect some innacuracy in our model.

Question 2

Part A

NOTE: An attempt was made for extra-credit part A of this problem. The attempt is located at an additional page at the end of this exam.

Before beginning our problem, we will input the data from the table provided.

```
x= c(0,62.5,125,250,500)
y1=c(15,17,22,38,144)
y2=c(1,0,7,59,132)
y3=c(281,225,283,202,9)
m=c(297,242,312,299,285)
m2=m-y1
```

Part B

We need to show that the Multinomial model can be expressed as two seperate binomials. Then, we have the Multinomial distribution defined as follows:

$$f(y_{i1}, y_{i2}, y_{i3}; m_i, \pi_{i1}, \pi_{i2}, \pi_{i3}) = \frac{m_i!}{y_{i1}! y_{i2}! y_{i3}!} \pi_{i1}^{y_{i1}} \pi_{i2}^{y_{i2}} \pi_{i3}^{y_{i3}}$$

Our first Binomial model will have a success determined by whether the mice dies or did not die. Thus, we will ignore the other y values for the moment and make the following algebraic alterations to make the equation equivalent to a Binomial distribution.

$$\frac{m_i!}{y_{i1}!} \pi_{i1}^{y_{i1}} \Longrightarrow \frac{m_{i1}!}{y_{i1}!(m_i - y_{i1})!} \pi_{i1}^{y_{i1}} (1 - \pi_{i1})^{m_i - y_{i1}}$$

Thus, we have fit our first Binomial model with y_{i1}. Now, using the added expressions used to fit the Binomial, we will look at the resulting terms for y_{i2} and y_{i3}. That is, we are left with

$$\frac{(m_i - y_{i1})!}{y_{i2}! y_{i3}!} \frac{\pi_{i2}^{y_{i2}} \pi_{i3}^{y_{i3}}}{(1 - \pi_{i1})^{m_i - y_{i1}}}$$

We would like to fit this model to be a Binomial with success determined by whether or not the mice are malformed or normal GIVEN that they are already determined to be alive (not from y_{i1}). Thus, we can use the following equivalencies to help simplify the model:

$y_{i3} = m_i - y_{i1} - y_{i2}$ and $1 - \pi_{i1} = \pi_{i2} + \pi_{i3}$. Thus, we now have the following result:

$$\frac{(m_i - y_{i1})!}{(y_{i2})!(m_i - y_{i1} - y_{i2})!} * \frac{\pi_{i2}^{y_{i2}} \pi_{i3}^{y_{i3}}}{(\pi_{i2} + \pi_{i3})^{y_{i2} + y_{i3}}}$$

Further simplifying our equatiion to get a Binomial distribution, we have

$$\frac{(m_i - y_{i1})!}{(y_{i2})!(m_i - y_{i1} - y_{i2})!} * \frac{\pi_{i2}^{y_{i2}} \pi_{i3}^{y_{i3}}}{(\pi_{i2} + \pi_{i3})^{y_{i2}} (\pi_{i2} + \pi_{i3})^{y_{i3}}} = \frac{(m_i - y_{i1})!}{(y_{i2})!(m_i - y_{i1} - y_{i2})!} \left(\frac{\pi_{i2}}{\pi_{i2} + \pi_{i3}}\right)^{y_{i2}} \left(\frac{\pi_{i3}}{\pi_{i2} + \pi_{i3}}\right)^{y_{i3}}$$

Finally, getting the expressions for success and failure consistent, we have the final Binomial model for y_{i2} as

$$\frac{(m_i - y_{i1})!}{(y_{i2})!(m_i - y_{i1} - y_{i2})!}\Big(\frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}}\Big)^{y_{i2}}\Big(\frac{\pi_{i3}}{\pi_{i2}+\pi_{i3}}\Big)^{m_i - y_{i1} - y_{i2}}$$

Thus, we have seen that our Multinomial model can be fit by two seperate Binomial models. Specifically, we have $Bin(y_{i1}; m_i, p_{i1})Bin(y_{i2}; m_i - y_{i1}, p_{i2})$ with $p_{i1} = \pi_{i1}$ and $p_{i2} = \frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}}$

This problem specifically asks to show that the Multinomial involving the **Likelihood** can be fit by two seperate Binomial Models.We have already shown that the Multinomial is equivalent to two sperate Binomials. So, showing the likelihood can be fit is simply stating the following.

$$\prod_{i=1}^{n} \frac{m_i!}{y_{i1}!y_{i2}!y_{i3}!}\pi_{i1}^{y_{i1}}\pi_{i2}^{y_{i2}}\pi_{i3}^{y_{i3}} =$$

$$\left(\prod_{i=1}^{n} \frac{m_{i1}!}{y_{i1}!(m_i - y_{i1})!}\pi_{i1}^{y_{i1}}(1-\pi_{i1})^{m_i - y_{i1}}\right)\left(\prod_{i-1}^{n} \frac{(m_i - y_{i1})!}{(y_{i2})!(m_i - y_{i1} - y_{i2})!}\Big(\frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}}\Big)^{y_{i2}}\Big(\frac{\pi_{i3}}{\pi_{i2}+\pi_{i3}}\Big)^{m_i - y_{i1} - y_{i2}}\right)$$

So, we have shown that we can use two seperate Binomial models to fit the data and now we are ready to obtain the MLEs of our paramters through two different IRLS algorithms.

Part C

Now that we have fit our two seperate Binomial models, we will implement IRLS to obtain our MLE estimates. First we will use our $binom(y_{i1}; m_i, \pi_{i1})$ model to find the MLE estimates of α_1 and β_1. We will use the following equality for our necessary response probability:

$$log\Big(\frac{\pi_{i1}}{\pi_{i2}+\pi_{i3}}\Big) = log\Big(\frac{\pi_{i1}}{1-\pi_{i1}}\Big) = \alpha_1 + \beta_1 x_i$$

Thus, we will solve for our π_{i1} value using our equivalence to obtain the final value of

$$\pi_{i1} = \frac{\exp(\alpha_1 + \beta_1 x_i)}{1 + \exp(\alpha_1 + \beta_1 x_i)}$$

Thus, our response y_{i1} is simply $m_i\pi_{i1}$. Now, we can begin our IRLS algorithm.

Solving for α_1 and β_1

Given our response y_{i1} for our Binomial model, we will derive the Jacobian matrix elements as follows:

$$\frac{\partial y_{i1}}{\partial \alpha_1} = m_i * \frac{\exp(\alpha_1 + \beta_1 x_i) * (1 + exp(\alpha_1 + \beta_1 x_i)) - \exp^2(\alpha_1 + \beta_1 x_i)}{(1 + \exp(\alpha_1 + \beta_1 x_i))^2}$$

$$\frac{\partial y_{i1}}{\partial \beta_1} = m_i * x_i * \frac{\exp(\alpha_1 + \beta_1 x_i) * (1 + \exp(\alpha_1 + \beta_1 x_i)) - \exp^2(\alpha_1 + \beta_1 x_i)}{(1 + \exp(\alpha_1 + \beta_1 x_i))^2}$$

In addition, the weight matrix is defined as having off-diagonals equivalent to $cov(y_{i1}, y_{j1}) = 0$ and diagonal values equivalent to the variance $\frac{1}{m_i * \pi_{i1} * (1-\pi_{i1})}$

Now, we can begin our IRLS algorithm. The following is our Jacobian/Weight function.

```
JacfW_Binom1 = function(theta,x){

  a1=theta[1]
  b1=theta[2]

  exp1 = exp(a1+b1*x)

  p = exp1/(1+exp1)
  q=1-p
  f=m*p

  da1 = m*(exp1*(1+exp1)-(exp1^2))/((1+exp1)^2)

  db1 = m*x*(exp1*(1+exp1)-(exp1^2))/((1+exp1)^2)

  J =cbind(da1,db1)
  W = diag(1/(m*p*q))

  list(f=f, J=J, W=W) }
```

Now that we have defined our Jacobian, we can create a program that performs the IRLS algorithm.

```
GN_Binom1 = function(y,X, theta0, Jac, Wt = 1, maxit,IRLS = TRUE){
  cat("Iteration", "              a1",
      "               b1","             MRE\n")
  for (it in 1: maxit){
    a = do.call(Jac, list(theta0, X))
    J = a$J
    f = a$f
    if (IRLS==TRUE){
      Wt = a$W }
    JW = t(J) %*% Wt
    dir = solve(JW %*% J) %*% JW %*% (y-f)
    theta1 = theta0 + dir

    relerr = norm(theta1-theta0)/max(1,norm(theta1))

    cat(sprintf(' %2.0f                    %6.6f       %6.6f        %1.1e\n'
                ,it,theta0[1],theta0[2],relerr))
    theta0=theta1

    if(relerr<1e-7){break}
  }
  MLE_theta<<-theta0}
```

Now, we can implement our algorithm.

```
maxit=20
theta0 = c(0,0)

AlphaBeta1 = GN_Binom1(y1,x, theta0, 'JacfW_Binom1', Wt = 1, maxit, IRLS = TRUE)
```

```
## Iteration              a1               b1              MRE
##   1                   0.000000       0.000000        1.0e+00
##   2                   -2.034678       0.003662        3.0e-01
```

```
##    3                -2.892696       0.005605        9.9e-02
##    4                -3.210050       0.006307        1.2e-02
##    5                -3.247465       0.006388        1.4e-04
##    6                -3.247934       0.006389        2.2e-08
```

Our algorithm converges to the MLE estimates (with desired MRE threshold) after 6 iterations. Our MLE estimates are $\hat{\alpha}_1$ = -3.247934 and $\hat{\beta}_1 = 0.006389$.

Solving for α_2 and β_2

Now, we can use our other binomial model, $binom(y_{i2}; m_i - y_{i1}, \frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}})$ to find the MLE estimates of α_2 and β_2.We will use the following equality for our necessary response probability:

$$log\Big(\frac{\pi_{i2}}{\pi_{i3}}\Big) = log\left(\frac{\frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}}}{\frac{\pi_{i3}}{\pi_{i2}+\pi_{i3}}}\right) = log\Big(\frac{p_{i2}}{1-p_{i2}}\Big) = \alpha_2 + \beta_2 x_i$$

Thus, we will solve for our p_{i2}, ($\frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}}$) value using our equivalence to obtain the final value of

$$p_{i2} = \frac{\exp(\alpha_2 + \beta_2 x_i)}{1 + \exp(\alpha_2 + \beta_2 x_i)}$$

Thus, our response y_{i2} is simply $(m_i - y_{i1}) * p_{i2}$. Now, we can begin our IRLS algorithm.

Given our response y_{i2} for our Binomial model, we will derive the Jacobian matrix elements as follows:

NOTE: For convenience, let $m_i - y_{i1} = m_{2i}$

$$\frac{\partial y_{i2}}{\partial \alpha_2} = m_{2i} * \frac{\exp(\alpha_2 + \beta_2 x_i) * (1 + \exp(\alpha_2 + \beta_2 x_i)) - \exp^2(\alpha_2 + \beta_2 x_i)}{(1 + \exp(\alpha_2 + \beta_2 x_i))^2}$$

$$\frac{\partial y_{i2}}{\partial \beta_2} = m_{2i} * x_i * \frac{\exp(\alpha_2 + \beta_2 x_i) * (1 + \exp(\alpha_2 + \beta_2 x_i)) - exp^2(\alpha_2 + \beta_2 x_i)}{(1 + \exp(\alpha_2 + \beta_2 x_i))^2}$$

In addition, the weight matrix is defined as having off-diagonals equivalent to $cov(y_{i2}, y_{j2}) = 0$ and diagonal values equivalent to the variance

$$\sigma^2 = \frac{1}{m_{2i} * \frac{\pi_{i2}}{\pi_{i2}+\pi_{i3}} * \frac{\pi_{i3}}{\pi_{i2}+\pi_{i3}}} = \frac{1}{m_{2i} * p_{i2} * (1 - p_{i2})}$$

Now, we can begin our IRLS algorithm. The following is our Jacobian/Weight function.

```
JacfW_Binom2 = function(theta,x){

  a2=theta[1]
  b2=theta[2]

  exp2 = exp(a2+b2*x)

  p = exp2/(1+exp2)
  q=1-p
  f=m2*p    #use resulting sample

  da1 = m2*(exp2*(1+exp2)-(exp2^2))/((1+exp2)^2)
  db1 = m2*x*(exp2*(1+exp2)-(exp2^2))/((1+exp2)^2)
```

```
  J =cbind(da1,db1)
  W = diag(1/(m2*p*q))

  list(f=f, J=J, W=W) }
```

Now that we have defined our Jacobian, we can create a program that performs the IRLS algorithm.

```
GN_Binom2 = function(y,X, theta0, Jac, Wt = 1, maxit,IRLS = TRUE){
  cat("Iteration", "              a2",
      "                  b2","           MRE\n")
  for (it in 1: maxit){
    a = do.call(Jac, list(theta0, X))
    J = a$J
    f = a$f
    if (IRLS==TRUE){
      Wt = a$W }
    JW = t(J) %*% Wt
    dir = solve(JW %*% J) %*% JW %*% (y-f)
    theta1 = theta0 + dir

    relerr = norm(theta1-theta0)/max(1,norm(theta1))

    cat(sprintf(' %2.0f                    %6.6f       %6.6f        %1.1e\n'
                ,it,theta0[1],theta0[2],relerr))
    theta0=theta1
    if(relerr<1e-7){break}
  }
  MLE_theta<<-theta0
}
```

Now, we can implement our algorithm.

```
maxit=20
theta0 = c(0,0)

alphabeta2 = GN_Binom2(y2,x, theta0, 'JacfW_Binom2', Wt = 1, maxit, IRLS = TRUE)
```

```
## Iteration              a2                 b2            MRE
##   1                   0.000000       0.000000        1.0e+00
##   2                   -2.457071       0.007224        3.7e-01
##   3                   -3.901897       0.011570        2.2e-01
##   4                   -5.008118       0.015014        1.0e-01
##   5                   -5.574542       0.016907        2.1e-02
##   6                   -5.696621       0.017354        9.3e-04
##   7                   -5.701891       0.017375        1.9e-06
##   8                   -5.701902       0.017375        8.2e-12
```

Our algorithm converges to the MLE estimates (with desired MRE threshold) after 8 iterations. Our MLE estimates are $\hat{\alpha}_2$ = -5.701902 and $\hat{\beta}_1$ = 0.017375.

Part D

We will use our MLE estimates to plot the estimated response curves for π_1, π_2, π_3.

NOTE: Although we have found our MLE estimates using two Binomial fits, our original proportions π_i still belong to the multinomial distribution originally stated. Thus, we must derive the equations for our proportions in the following manner:

$$log\Big(\frac{\pi_{i1}}{\pi_{i2}+\pi_{i3}}\Big) = log\Big(\frac{\pi_{i1}}{1-\pi_{i1}}\Big) = \alpha_1 + \beta_1 x_i$$

Thus, we will solve for our π_{i1} value using our equivalence to obtain the final value of

$$\pi_{i1} = \frac{exp(\alpha_1 + \beta_1 x_i)}{1 + \exp(\alpha_1 + \beta_1 x_i)}$$

Now, we will find π_2 in a similar fashion.

$$log\Big(\frac{\pi_{i2}}{\pi_{i3}}\Big) = log\Big(\frac{\pi_{i2}}{1-\pi_{i1}-\pi_{i2}}\Big)$$

This gives us the following:

$$\pi_{i2} = (1-\pi_{i1}-\pi_{i2}) * \exp(\alpha_2 + \beta_2 x_i)$$

$\Rightarrow$

$$\pi_{i2}(1 + exp(\alpha_2 + \beta_2 x_i)) = exp(\alpha_2 + \beta_2 x_i) - \pi_{i1} * exp(\alpha_2 + \beta_2 x_i)$$

Using our derived value for π_1 and simplifying, we have

$$\pi_2 = \frac{\exp(\alpha_2 + \beta_2 x_i)}{(1 + \exp(\alpha_2 + \beta_2 x_i)) * (1 + \exp(\alpha_1 + \beta_1 x_i))}$$

The value of π_3 is simply $1 - \pi_1 - \pi_2$.

Now that we have derived each of our proportions for the Multinomial population, we can plot the estimated response curves. We will use our MLE estimates for our parameter values.

```
a1=-3.247934
a2=-5.701902
b1= .006389
b2= .017375
```

We will now create R functions for each of our proportions.

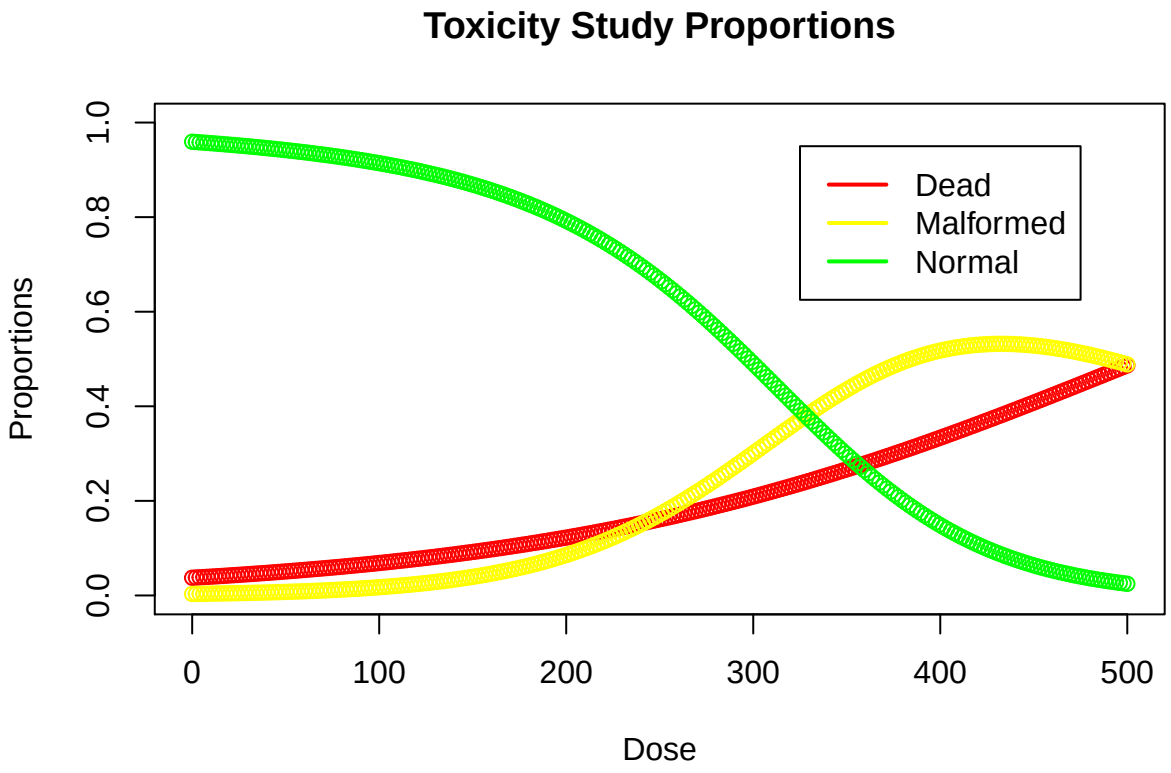
```
pie_1 =function(x){

  pi = exp(a1+b1*x)/(1+exp(a1+b1*x))

  return(pi)
}
```

```
pie_2 = function(x){

  pi= (exp(a2+b2*x))/ ( (1+exp(a2+b2*x))*(1+exp(a1+b1*x)) )

  return(pi)
}
```

```
pie_3=function(x){

  pi = 1-pie_1(x)-pie_2(x)

  return(pi)
}
```

Now, we will plot each of our response curves over the desired interval of Concentration.

```
Concentration = seq(0,500, length=250)

plot(Concentration,pie_1(Concentration), xlab="Dose",
     ylab="Proportions", main="Toxicity Study Proportions",
     ylim=c(0,1),col="red")
points(Concentration,pie_2(Concentration),col="yellow")
points(Concentration,pie_3(Concentration),col="green")

legend(325,.95, c("Dead", "Malformed", "Normal"), col =c("red", "yellow", "green"),
       lty=1, lwd=2)
```



Going over the plot of the response curves, we can first note that proportion of Normal mice is much higher than the proportion of dead and malformed mice for the first 250 concentration levels. This is very reflective of the data provided in the tables. We can then see that as the concentration goes beyond 300, the proportion of normal mice begins to approach zero. Thus, we can see that there is a threshold for which mice can be exposed to the particular chemicals before disproportianitely succumbing to malformations or death.For the response curves of Malformed and Dead mice, we can see that the rate of death is slightly higher than the rate of malformation up until they are both equal at the 250 dose level. Afterwards, we see that malformation occurs more often than death until they are nearly equivalent again at the 500 dose level. All of these comparisons are equivalent to the results from the actual data provided. Thus, we can say that our MLE estimates model the Multinomial Data provided from the Toxicity study.

Question 3

For this problem, we will be considering the TREATMENT group data from the table provided in the textbook. Thus, we will be ommitting any observations from the control group.

```
survival = c(6, 6, 6, 6, 7, 9, 10, 10, 11, 13, 16, 17, 19, 20, 22, 23, 25, 32, 32, 34, 35)

censored =  c(0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0)
```

Part A

For the Newton algorithm, we have been provided with the log-likelihood function. Thus, we must derive the gradient and hessian in order to proceed with the algorithm.

Gradient:

$$\frac{\partial l}{\partial \theta} = \sum_{i=1}^{n} \delta_i(\frac{1}{\theta} - z_i) + (1 - \delta_i) * (-z_i)$$

Hessian:

$$\frac{\partial^2 l}{\partial \theta^2} = \sum_{i=1}^{n} \delta_i(\frac{-1}{\theta^2})$$

Now that the gradient and hessian elements have been derived, we can proceed with our Newton Algorithm.

```
likelihood = function(theta,gcomp,hesscomp){

d=censored
z=survival

l = sum(d*(log(theta) - theta*z) + (1-d)*(-theta*z))

if(gcomp==TRUE) {

grad = sum(d*( (1/theta) -z) + (1-d)*(-z))

}

 if(hesscomp==TRUE){

   hess = sum( (-d/(theta^2)))
 }

list(l=l,grad=if(gcomp) grad,hess=if(hesscomp)hess)
  }
```

```
Newton = function (theta,maxit) {

  cat("Iteration", "       log-likelihood", "         theta       MRE\n")

  for(it in 1:maxit){
```

```
  a =likelihood(theta,TRUE,TRUE)
  gradient = a$grad
  hessian = a$hess
  theta1 = theta - (gradient/hessian)
  atmp = likelihood(theta1,FALSE,FALSE)

  relerr= abs(theta1-theta)/max(1,abs(theta1))


  cat(sprintf(' %2.0f                %6.6f         %6.6f      %2.2e\n'
              ,it, atmp$l, theta, relerr))

  theta = theta1
  if(relerr<1e-7){break}
  }
  list(theta =theta)
  }
```

Now, we will implement our Newton Algorithm with initial settings.

```
theta=0.01
Newton(theta,10)
```

```
## Iteration        log-likelihood          theta       MRE
##    1                -42.958240         0.010000      6.01e-03
##    2                -42.258997         0.016011      5.79e-03
##    3                -42.176203         0.021796      2.85e-03
##    4                -42.174881         0.024642      4.20e-04
##    5                -42.174880         0.025062      7.28e-06
##    6                -42.174880         0.025070      2.12e-09
```

```
## $theta
## [1] 0.02506964
```

Our MLE estimate is $\hat{\theta} = 0.025070$, and it was reached in 6 iterations.

Part B

Now, we will use the BFGS method in the R function optim to find the MLE of θ. We will first need to create seperate R functions for the log-likelihood and gradient of our model.

```
likelihood2 = function(theta){

d=censored
z=survival
l = sum(d*(log(theta) - theta*z) + (1-d)*(-theta*z))

return(l)}
```

```
gradient2 = function(theta){

d=censored
z=survival
gradient = sum(d*( (1/theta) -z) + (1-d)*(-z))

return(gradient)}
```

Now, we will implement the BFGS method.

```
optim(par=c(.01),fn=likelihood2,gr=gradient2,method="BFGS",control=c(fnscale=-1))
```

```
## $par
## [1] 0.02506963
##
## $value
## [1] -42.17488
##
## $counts
## function gradient
##       43        8
##
## $convergence
## [1] 0
##
## $message
## NULL
```

We can see that the MLE estimate for θ is found to be .0250696, which is equivalent to the estimate that was found in the prior Newton Algorithm.

Part C

Now, we will use the EM algorithm to fit the data and estimate $\hat{\theta}$. We will be following the steps outlined in the in-class exam.

NOTE: A page been added to the end of this exam, which contains derivations of the required log-likelihood, e-step and m-step of the EM algorithm for this problem. This is attached in order to aquire partial credit points for the in-class portion of the exam.

```
EM = function (theta, maxit){

  d=censored
  z=survival
  n=length(z)

   cat("Iteration", "           theta",      "         MRE\n")

  for (it in 1:maxit){

    ### The E Step
    xstar = (1/theta) + z

    ### The M Step
    theta1 = n/sum(d*z + (1-d)*xstar)

    relerr = abs(theta1-theta)/max(1,abs(theta1))


    cat(sprintf(' %2.0f                   %6.6f        %2.2e\n'
              ,it, theta,relerr))
```

```
    theta = theta1

    if(relerr<1e-7){break}

  }
  print('_____________________________________________')
  print(sprintf('Theta                 Actual Theta'))
  print(sprintf('%6.6f       %6.6f',
                theta, 0.02507))

}
```

```
EM(.01,25)
```

```
## Iteration            theta          MRE
##   1                   0.010000       3.47e-03
##   2                   0.013470       3.33e-03
##   3                   0.016802       2.77e-03
##   4                   0.019568       2.03e-03
##   5                   0.021599       1.36e-03
##   6                   0.022961       8.58e-04
##   7                   0.023820       5.20e-04
##   8                   0.024340       3.07e-04
##   9                   0.024647       1.79e-04
##  10                   0.024827       1.04e-04
##  11                   0.024930       5.96e-05
##  12                   0.024990       3.42e-05
##  13                   0.025024       1.96e-05
##  14                   0.025043       1.12e-05
##  15                   0.025055       6.40e-06
##  16                   0.025061       3.66e-06
##  17                   0.025065       2.09e-06
##  18                   0.025067       1.20e-06
##  19                   0.025068       6.83e-07
##  20                   0.025069       3.90e-07
##  21                   0.025069       2.23e-07
##  22                   0.025069       1.27e-07
##  23                   0.025069       7.29e-08
## [1] "_____________________________________________"
## [1] "Theta                 Actual Theta"
## [1] "0.025070       0.025070"
```

Thus, the EM-algorithm reaches the same $\hat{\theta}$ value as the two previous Newton methods. It is worth noting that EM converges sublinearly, so it does take a few more iterations to achieve the same accuracy as the newton algorithm which converges quadratically.

Part D

We have seen through each of our algorithms that the MLE estimate of θ is reached fairly quickly for our starting value. However, we can see differing results when the starting θ value is changed. The first Newton algorithm is particularly sensitive to starting values. If a θ value is chosen to be distant from the actual MLE estimate, then the Newton algorithm will get stuck searching for a maximum value and will often reach the wrong optimization goal or fail to find an MLE estimate. For the built in Newton method, we can see that it

is a bit more flexible in regards to initial values since we have set the function to search for a maximum value.Thus, choosing a distant starting value still finds the MLE but it takes many iterations to arrive at this point. We can also note that the built in function will not work if a starting value is chosen outside of the parameter bounds.This difference is likely due to the fact that the BFGS method is quasi-newton and does not require the hessian element for any computations. The EM algorithm does not seem to take issue with starting values or parameter bounds. Trying extreme values still shows that the EM algorithm can find the MLE estimate in less than 20 iterations. That is a sign of a strong method for the given problem.

In addition to computation time, we can also compare the amount of effort that goes into creating the algorithms. For the Newton methods, we had to utilize the Log-likelihood, Gradient and (for the first method) the Hessian. For problems that include many parameters, this could become cumbersome. The EM algorithm, does not require the Log-likelihood in the algorithm and utilizes a very simple method for iterations that also accounts for missing information.

For these reasons, this problem seems to work best with the EM algorithm.

Extra Credit

2A

For the multinomial IRLS, I attempted to utilize π_{i2} from the Multinomial distribution. This response probability includes all four of our parameters to estimate. The jacobian was created with four seperate partial derivatives. The IRLS was then created and executed. There seemed to be an issue with the inverse of (JW * J), which would not allow the algorithm to iterate.

I have attempted to rework the model to be that of a multivariate response (ie $y = (1,0,0),(0,1,0),(0,0,1)$), but I could not figure out how to create the Jacobian to include serveral partial derivatives of different response proportions. The following is my coded attempt at computing the MLE estimate for all four parameters at once.

```
theta=c(0,0,0,0)
JacfW_Exam2 = function(theta,x){

  a1=theta[1]
  a2=theta[2]
  b1=theta[3]
  b2=theta[4]

  exp1 = exp(a1+b1*x)
  exp2= exp(a2+b2*x)

  pi2 = exp2/(1+exp2)
  pi1 = 1/(1+exp1)
  p = (exp2)/( (1+exp2)*(1+exp1))

  q=1-p
  f=m*p

  da1 = m*(-exp1*exp2*(1+exp2))/( (1+exp2)^2 * (1+exp1)^2)

  da2 = m*( exp2*(1+exp2)*(1+exp1) - (exp2^2)*(1+exp1))/( (1+exp2)^2 * (1+exp1)^2)

  db1 = m*(-x*exp1*exp2*(1+exp2))/( (1+exp2)^2 * (1+exp1)^2)

  db2 = m*( x*exp2*(1+exp2)*(1+exp1) - x*(exp2^2)*(1+exp1))/( (1+exp2)^2 * (1+exp1)^2)


  J =cbind(da1,da2,db1,db2)
  W = diag(1/(m*p*q))

  list(f=f, J=J, W=W)
}
```

```
GN_multinomial = function(y,X, theta0, Jac, Wt = 1, maxit,IRLS = TRUE){
  cat("Iteration", "           a1",
      "                    a2", "                 b1","                    b2",  "
      MRE\n")
  for (it in 1: maxit){
    a = do.call(Jac, list(theta0, X))
    J = a$J
```

```
    f = a$f
    if (IRLS==TRUE){
      Wt = a$W }
    JW = t(J) %*% Wt
    dir = solve(JW %*% J) %*% JW %*% (y-f)
    theta1 = theta0 + dir

    relerr = norm(theta1-theta0)/max(1,norm(theta1))

    cat(sprintf(' %2.0f                    %6.6f       %6.6f     %6.6f
                %6.6f     %1.1e\n'
                ,it,theta0[1],theta0[2],theta0[3],theta0[4],relerr))
    theta0=theta1
  }
  MLE_theta<<-theta0
  }
```

```
maxit=20
theta0 = c(-3,-5,0,0)

#GN_multinomial(y2,x, theta0, 'JacfW_Exam2', Wt = 1, maxit, IRLS = TRUE)
```

Derivations For Problem 3 on Take Home

Complete Log-Likelihood

Our complete data has right censored data, so our complete density is defined as

$$f(x;\theta) = [f(z_i;\theta)]^{\delta_i}[1 - F(z_i;\theta)]^{1-\delta_i}$$

Thus if $\delta_i = 1$ our data was observed and the density is simply $f(z_i;\theta)$. If $\delta_i = 0$, our z_i was right censored and the true x_i survival time must occur after the recorded survivial time. So our density is $[1-F(z_i,\theta)]$.

Thus, our Likelihood function is

$$L(\theta, x) = \prod_{i=1}^{n}[f(z_i,\theta)]^{\delta_i}[1 - F(z_i;\theta)]^{1-\delta_i} = \prod_{i=1}^{n}[\theta \exp(-\theta z_i)]^{\delta_i}[\exp(-\theta z_i)]^{1-\delta_i}$$

And therefore, our log-likelihood is as follows:

$$l(\theta, x) = \sum_{i=1}^{n}\left(\delta_i(log(\theta) - \theta z_i) + (1 - \delta_i)(-\theta z_i)\right)$$

E-Step

Now, we must use our complete log-likelihood to derive $Q(\theta', \theta)$. Thus we have

$$E[l(\theta, x)] = \sum_{i=1}^{n}\left(\delta_i(log(\theta') - \theta' x_i^*) + (1 - \delta_i)(-\theta' x_i^*)\right)$$

We must find the values of x_i^*.

If z_i was recorded ($\delta_i = 1$), then $x_i^* = x_i$.

If z_i was censored ($\delta_i = 0$), then x_i must have occured after z_i. Thus, using the memoryless property, we have

$$x_i^* = E[x_i] + z_i = \frac{1}{\theta} + z_i$$

Thus, we have

$$Q(\theta', \theta) = \sum_{i=1}^{n}\left(\delta_i(log(\theta') - \theta' x_i) + (1 - \delta_i)(-\theta' x_i^*)\right)$$

M-Step

Now, we must take the derivative of Q with respect to θ' and solve for our new θ'.

$$\frac{\partial Q}{\partial \theta'} = \sum_{i=1}^{n} \left(\delta_i (\frac{1}{\delta'} - x_i) + (1 - \delta_i)(-x_i^*) \right) = 0$$

This gives us the following result:

$$\sum_{i=1}^{n} \frac{\delta_i}{\theta'} = \sum_{i=1}^{n} \delta_i x_i + (1 - \delta_i) x_i^* \Longrightarrow \theta' = \sum_{i=1}^{n} \frac{\delta_i}{\delta_i x_i + (1 - \delta_i) x_i^*} = \frac{n}{\sum_{i=1}^{n} \delta_i x_i + (1 - \delta_i) x_i^*}$$

Now that we have our steps established, we can iterate through our steps to find the MLE estimate of θ. This was completed in problem 3.