MATH 534 EXAM 1

Gustavo Esparza

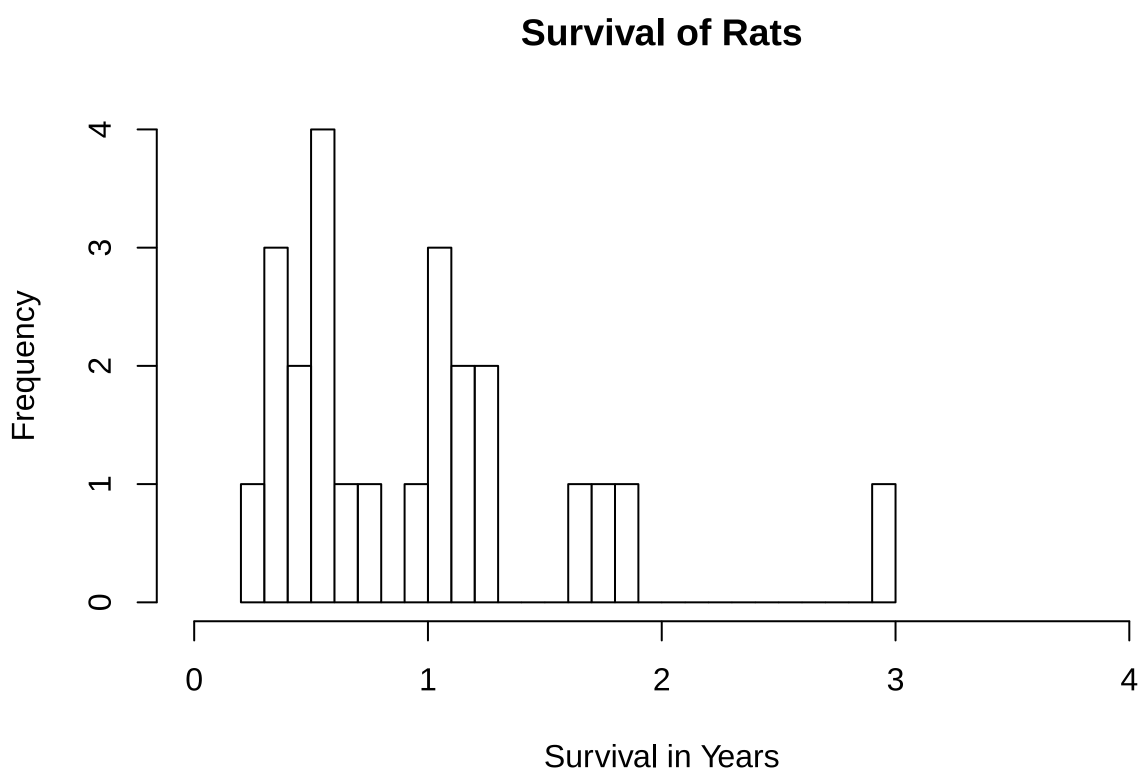
3/3/2019

```r
library(readr)
library(plot3D)

survrat = read.table("/Users/gustavo/Desktop/534 Exam 1/survrat.txt")
surv=survrat$x
```

1.)

A.) Obtain a histogram of the data and commend on its' features.

```r
hist(surv, breaks=30, xlab="Survival in Years", main=("Survival of Rats"),xlim=c(0,4))
```



From the histogram, we can see that the data ranges from about .3 to 3 years of survival. The majority of the data points are located in the first half of the x-interval, with a few outliers residing beyond the 1.5 year mark. Since there is such a high frequency of data residing in this interval, we can assume that the mean should also be located in this interval between about .9 and 1. In regards to the shape of the distribution, we can note about two significant peaks and a skew to the right, which may influence the mean of the data.

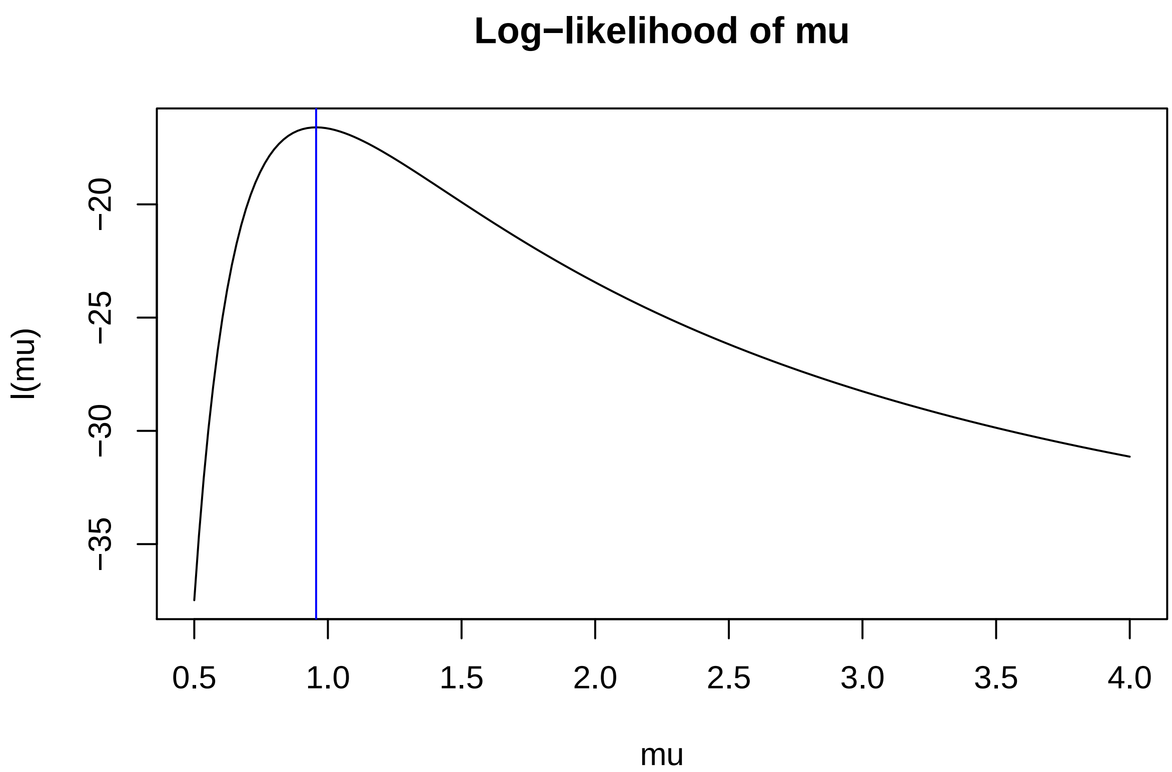
B.) Write R functions having inputs μ,ϕ, and y for the Likelihood,gradient,hessian,Fisher Information matrix and MLE of μ.

```r
#log-likelihood#
likelihood = function(mu,phi,y){
```

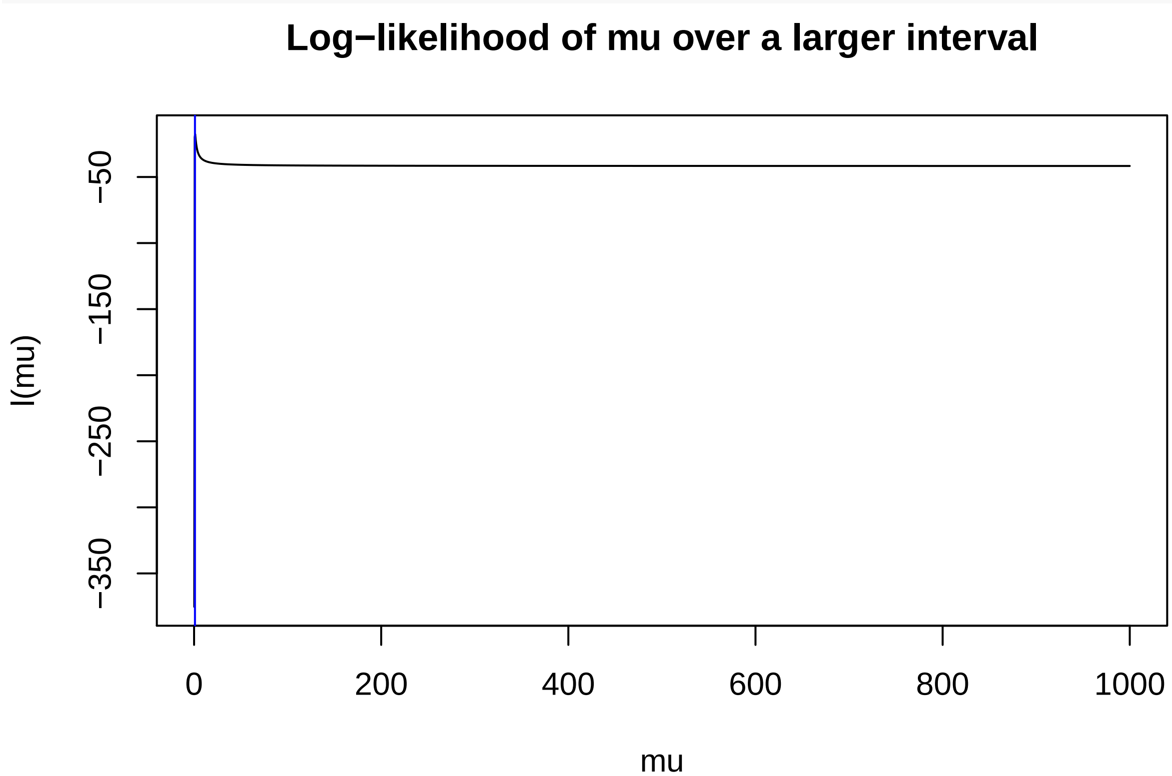
```r
  l = (-1/2)*sum(log(2*pi*phi*(y^3))) + sum((-(y-mu)^2)/(2*phi*(mu^2)*y))
  return(l)}

#gradient#

gradient=function(mu,phi,y){

  n=length(y)

  dmu= (n/((mu^3)*phi))*(mean(y)-mu)
  return(dmu)
}

#Hessian#

hessian=function(mu,phi,y){

  n=length(y)
  ddmu = (n/((mu^4)*phi))*((-3)*mean(y)+2*mu)
  return(ddmu)
}

#Fisher Information#

FisherFun=function(mu,phi,y){

  n=length(y)

  fish=n/(phi*(mu^3))
  return(fish)
}


#Maximum Likelihood Estimate#

Mu_MLE=function(mu,phi,y){

  max=mean(y)
}
```

C.) Graph the log-likelihood function, $l(\mu)$ and the gradient function $\bigtriangledown l(\mu)$ in two seperate graphs. Comment on the plots in regards to where the MLE appears to lie.

```r
mu=seq(0.5,4, length=200)
phi=.5
y=surv
#Likelihood Graph
plot(mu, sapply(X=mu, FUN=function(mu) likelihood(mu,phi,surv)),
type="l",xlab = "mu",ylab="l(mu)",main="Log-likelihood of mu")
abline(v=mean(y),col="blue")
```



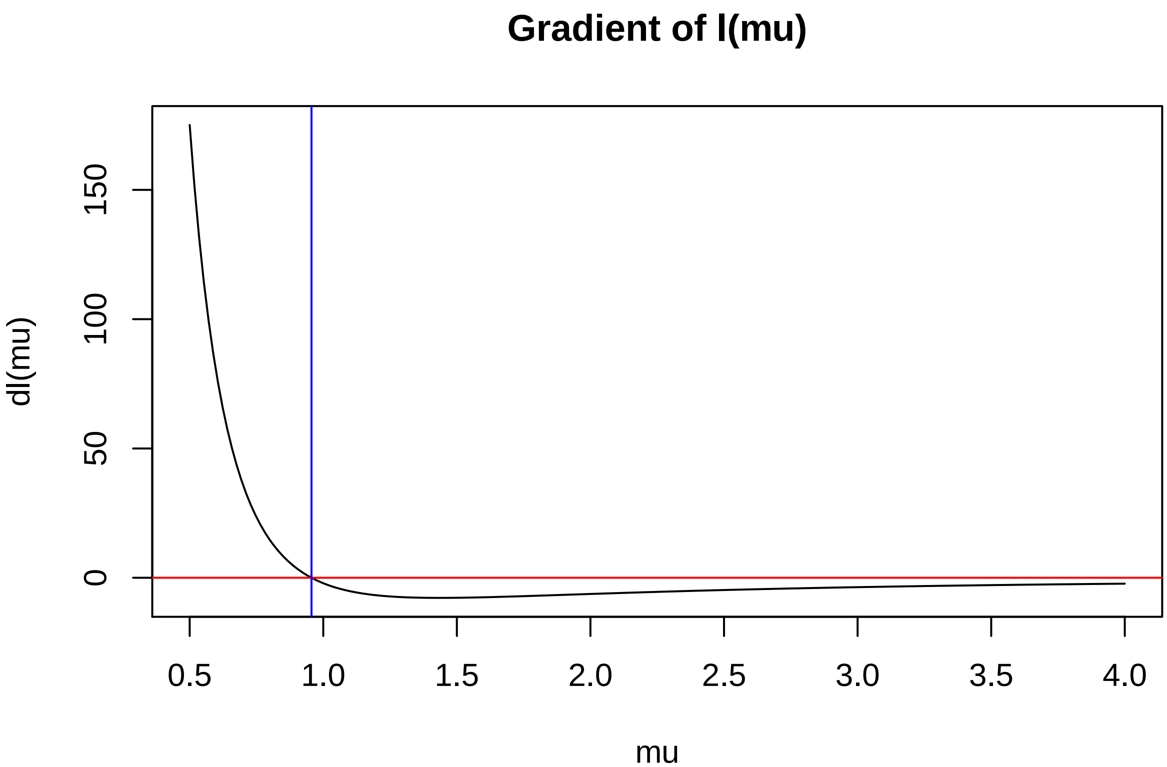

```
mu=seq(0.2,1000, length=2000)
plot(mu, sapply(X=mu, FUN=function(mu) likelihood(mu,phi,surv)),
type="l",xlab = "mu",ylab="l(mu)",main="Log-likelihood of mu over a larger interval")
abline(v=mean(y),col="blue")
```



From the graph of the Likelihood function, we can see a clear maximum value in the interval around $\mu = .95$.

Although there is no upper limit for μ, expanding the interval makes it evident that the maximum value is found near the .95 value. To add clarity, a vertical line represting the MLE of μ, $\bar{y}$, has been added to the graph to support our theory of where the MLE resides.

```
#Gradient Graph
mu=seq(.5,4, length=200)
plot(mu,gradient(mu,phi,surv),type="l",xlab="mu",ylab="dl(mu)",main="Gradient of l(mu)")
abline(h=0,col="red")
abline(v=mean(y),col="blue")
```



From the graph of the gradient of $l(\mu)$, we can see that the MLE estimate we found in the graph of the log-likelihood results in a gradient value of zero. We reinforce this idea by adding lines showing the intersection of the MLE estimate and the x-axis.

D.) Use the analytical expression for the MLE to compute μ^*.

```
mu_star<<-mean(surv)
#or
mu_star<<-Mu_MLE(1,1,y) #no need to input mu or phi

mu_star
```

```
## [1] 0.955966
```

E.) Write an R function that applies the Newton Method for finding the maximum of $l(\mu)$

```
Newton = function (maxit,mu,phi,y,tolerr,tolgrad)
  {
  digits = -log10( abs(mu-mu_star)/abs(mu_star))
  rate=0

  cat(" b" , "    mu(b)", "                CR", "      Digits\n")


     cat(sprintf('%2.0f     %12.12f      %4.2f       %2.2f\n',
                0,mu,rate,digits))

  for (b in 1:maxit){

    dl=gradient(mu,phi,y)
    ddl=hessian(mu,phi,y)

    mub = mu - dl/ddl
    rate = abs(mub-mu_star)/abs(mu-mu_star)
    digits = -log10(abs(mub-mu_star)/abs(mu_star))

    relerr = abs(mub-mu)/max(1,abs(mub))
    dlb=gradient(mub,phi,y)

    if(relerr<tolerr & abs(dlb)<tolgrad){
      break
    }
    cat(sprintf('%2.0f     %12.12f      %4.2f       %2.2f\n',
                b,mub,rate,digits))
    mu=mub
  }
}
```

F.) Run your program using the starting values $\mu^{(0)} = 1$. Comment on your findings. Try two other initial values to compare.
Using $\mu^{(0)} = 1$:

```
maxit=20
mu=1
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Newton(maxit,mu,phi,y,tolerr,tolgrad)
```

```
##  b     mu(b)                 CR       Digits
##  0     1.000000000000       0.00       1.34
##  1     0.949263585341       0.15       2.15
##  2     0.955826947377       0.02       3.84
##  3     0.955965910888       0.00       7.20
```

With an initial value of $\mu_{(0)}$=1, the MLE is reached after 3 iterations and the tolerr is met. This is fairly quick, but it makes sense since the MLE is close to our starting value and the the log-likelihood function does not appear to have a large amount of local maximum values to influence the Newton Method.

Using $\mu^{(0)} = .5$:

```
maxit=20
mu=.5
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Newton(maxit,mu,phi,y,tolerr,tolgrad)
```

```
## b     mu(b)               CR       Digits
## 0     0.500000000000     0.00      0.32
## 1     0.622053236413     0.73      0.46
## 2     0.749970832365     0.62      0.67
## 3     0.862905983614     0.45      1.01
## 4     0.933217697535     0.24      1.62
## 5     0.954415786792     0.07      2.79
## 6     0.955958454612     0.00      5.10
## 7     0.955965971347     0.00      9.73
```

Moving our initial value below the MLE leads to the Newton Method finding the MLE after 7 iterations, which is still a fairly quick path.

Using $\mu^{(0)} = 10$:

```
maxit=20
mu=10
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Newton(maxit,mu,phi,y,tolerr,tolgrad)
```

```
## b     mu(b)               CR       Digits
## 0     10.000000000000     0.00       -0.98
## 1     15.278998445946     1.58       -1.18
## 2     23.182241807307     1.55       -1.37
## 3     35.028111836148     1.53       -1.55
## 4     52.791360471740     1.52       -1.73
## 5     79.432705075776     1.51       -1.91
## 6     119.392442786895     1.51       -2.09
## 7     179.330560946010     1.51       -2.27
## 8     269.236759320714     1.50       -2.45
## 9     404.095410152838     1.50       -2.63
## 10    606.382957813278     1.50       -2.80
## 11    909.813994709472     1.50       -2.98
## 12    1364.960360823856     1.50       -3.15
## 13    2047.679784063439     1.50       -3.33
## 14    3071.758835066255     1.50       -3.51
## 15    4607.877355709637     1.50       -3.68
## 16    6912.055099453476     1.50       -3.86
## 17    10368.321690263660     1.50       -4.04
```

```
## 18      15552.721559945698       1.50        -4.21
## 19      23329.321353448293       1.50        -4.39
## 20      34994.221036355964       1.50        -4.56
```

Now, using a initial value that is far to the right of our MLE for μ shows that the Newton Method is very sensitive to it's starting point and proceeds to grow along the μ axis and never converges to a desired value.

G.) Repeat (e) and (f) using the Fisher's Scoring Method and Secant Method.

Fisher's Scoring Method:

```
Fisher = function (maxit,mu,phi,y,tolerr,tolgrad)
{
  digits = -log10( abs(mu-mu_star)/abs(mu_star))
  rate=0

  cat("b" , "  mu(b)", "            CR", "          Digits\n")

  cat(sprintf('%2.0f   %9.9f     %4.2f         %2.2f\n'
              ,0,mu,rate,digits))

  for (b in 1:maxit){

    dl=gradient(mu,phi,y)
    f=FisherFun(mu,phi,y)

    mub = mu + (dl/f)
    rate = abs(mub-mu_star)/abs(mu-mu_star)
    digits = -log10(abs(mub-mu_star)/abs(mu_star))

    relerr = abs(mub-mu)/max(1,abs(mub))
    dlb=gradient(mub,phi,y)

    if(relerr<tolerr & abs(dlb)<tolgrad){
      break
    }
    cat(sprintf('%2.0f   %10.10f     %4.2f         %2.2f\n'
                ,b,mub,rate,digits))
    mu=mub
  }
}
```

Using $\mu^{(0)} = 1$:

```
maxit=20
mu=1
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Fisher(maxit,mu,phi,y,tolerr,tolgrad)
```

```
## b   mu(b)              CR           Digits
##  0   1.000000000     0.00          1.34
```

```
## 1   0.9559659715     0.00         Inf
```

Using $\mu^{(0)} = .5$:

```
maxit=20
mu=.5
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Fisher(maxit,mu,phi,y,tolerr,tolgrad)
```

```
## b   mu(b)              CR           Digits
##  0   0.500000000     0.00         0.32
##  1   0.9559659715     0.00         15.94
```

Using $\mu^{(0)} = 10$:

```
maxit=20
mu=10
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Fisher(maxit,mu,phi,y,tolerr,tolgrad)
```

```
## b   mu(b)              CR           Digits
##  0   10.000000000     0.00         -0.98
##  1   0.9559659715     0.00         15.46
```

Using the three different initial values used in the Newton Method, the Fisher Method shows to find the MLE instantaneously. Even starting at $\mu^{(0)} = 10$, the Fisher method jumps right to the MLE. If this is not due to an error in the code for the function, then it is assumed that both the strength of the Fisher method and the lack of other maximum values in the Log-likelihood function cause the MLE to be found so quickly.

Secant Method:

```
Secant = function (maxit,mu0,mu1,phi,y,tolerr,tolgrad)
{
  digits = 0
  rate=0

  cat("b" , "  mu(0)", "              mu(1) ", "          CR", "    Digits\n")

  cat(sprintf('%2.0f   %12.12f     %12.12f    %4.2f   %2.2f\n'
              ,0,mu0,mu1,rate,digits))


  cat("b" , "  mu(b)", "          CR", "     Digits\n")

  for (b in 1:maxit){

    dl0=gradient(mu0,phi,y)
    ddl0=hessian(mu0,phi,y)
```

```
    dl1=gradient(mu1,phi,y)
    ddl1=hessian(mu1,phi,y)

    mub = mu1 - dl1*(mu1-mu0)/(dl1-dl0)
    dlb = gradient(mub,phi,y)

    rate = abs(mub-mu_star)/abs(mu1-mu_star)
    digits = -log10(abs(mub-mu_star)/abs(mu_star))
    relerr = abs(mub-mu1)/max(1,abs(mub))
    dlb=gradient(mub,phi,y)

    if(relerr<tolerr & abs(dlb)<tolgrad){
      break
    }
    cat(sprintf('%2.0f   %4.4f      %4.4f     %2.2f\n',
                b,mub,rate,digits))
    mu0=mu1
    mu1=mub
  }
}
```

Using $\mu^{(0)} = 1$, $\mu^{(1)}$ =1.1, since the Secant method requires two initial values.

```
maxit=20
mu1=1.1
mu0=1
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Secant(maxit,mu0,mu1,phi,y,tolerr,tolgrad)
```

```
## b   mu(0)               mu(1)            CR     Digits
##  0   1.000000000000     1.100000000000    0.00   0.00
## b   mu(b)           CR      Digits
##  1   0.9314      0.1706     1.59
##  2   0.9684      0.5053     1.89
##  3   0.9569      0.0761     3.01
##  4   0.9559      0.0395     4.41
##  5   0.9560      0.0030     6.94
```

Using $\mu^{(0)} = .5$, $\mu^{(1)}$ =1.5, where the MLE is between the two values.

```
maxit=20
mu1=1.5
mu0=.5
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Secant(maxit,mu0,mu1,phi,y,tolerr,tolgrad)
```

```
## b   mu(0)               mu(1)            CR     Digits
##  0   0.500000000000     1.500000000000    0.00   0.00
```

```
## b   mu(b)           CR      Digits
##  1   1.4577      0.9222      0.28
##  2   10.1592      18.3435     -0.98
##  3   10.6577      1.0542     -1.01
##  4   15.8927      1.5396     -1.19
##  5   20.4300      1.3038     -1.31
##  6   27.6396      1.3702     -1.45
##  7   36.5715      1.3347     -1.57
##  8   48.7145      1.3409     -1.70
##  9   64.6393      1.3334     -1.82
## 10   85.8152      1.3325     -1.95
## 11   113.8249      1.3301     -2.07
## 12   150.9503      1.3289     -2.20
## 13   200.1197      1.3278     -2.32
## 14   265.2604      1.3271     -2.44
## 15   351.5504      1.3265     -2.56
## 16   465.8614      1.3260     -2.69
## 17   617.2905      1.3257     -2.81
## 18   817.8915      1.3255     -2.93
## 19   1083.6310      1.3253     -3.05
## 20   1435.6608      1.3251     -3.18
```

Using $\mu^{(0)} = .5$, $\mu^{(1)} =$.8, where the MLE is not between the two values.

```
maxit=20
mu1=.8
mu0=.5
phi=.5
y=surv
tolerr=1e-6
tolgrad=1e-9

Secant(maxit,mu0,mu1,phi,y,tolerr,tolgrad)
```

```
## b   mu(0)                mu(1)              CR     Digits
##  0   0.500000000000     0.800000000000    0.00   0.00
## b   mu(b)           CR      Digits
##  1   0.8273      0.8247      0.87
##  2   0.9075      0.3770      1.29
##  3   0.9395      0.3391      1.76
##  4   0.9536      0.1424      2.61
##  5   0.9558      0.0506      3.91
##  6   0.9560      0.0073      6.04
##  7   0.9560      0.0004      9.47
```

Using the Secant method, we can see that it is also sensitive to the initial values. When using values that reside in a range before the maximum is achieved or near the MLE, the Secant method seems to converge to our desired MLE. When using initial values that are distant from the MLE, we see the method unableis to find a converging value. It is also important to remember that the maximum is located around $\mu = .9$ and starting values located after that point may not be able to converge to the MLE since the maximum is so sharp in this instance.

2.)

A.) Derive the log-likelihood of $\theta = (\mu, \phi)$: $l(\mu, \phi)$

$$L(\mu, \phi) = \prod_{i=1}^{n} (2\pi\phi y_i^3)^{-\frac{1}{2}} * \prod_{i=1}^{n} \left(exp\Big(\frac{-(y_i - \mu)^2}{2\phi\mu^2 y_i} \Big) \right)$$

$$l(\mu, \phi) = \frac{-1}{2} \sum_{i=1}^{n} log(2\pi\phi y_i^3) + \sum_{i=1}^{n} \left(\frac{-(y_i - \mu)^2}{2\phi\mu^2 y_i} \right)$$

B.) Derive the gradient vector for $l(\mu, \phi)$: $\bigtriangledown l(\mu, \phi)$

$$\frac{\partial l}{\partial \phi} = \frac{-1}{2} \sum_{i=1}^{n} \frac{2\pi y_i^3}{2\pi\phi y_i^3} + \sum_{i=1}^{n} \frac{(y_i - \mu)^2}{2\phi^2\mu^2 y_i} = \sum_{i=1}^{n} \left(\frac{-1}{2\phi} + \frac{(y_i - \mu)^2}{2\phi^2\mu^2 y_i} \right)$$

Thus, the gradient vector of $l(\mu, \phi)$ is

$$\bigtriangledown \mathbf{l}(\mu, \phi) = \begin{bmatrix} \frac{n}{\mu^3\phi}[\bar{y} - \mu] \\ \\ \sum_{i=1}^{n} \left(\frac{-1}{2\phi} + \frac{(y_i-\mu)^2}{2\phi^2\mu^2 y_i} \right) \end{bmatrix}$$

C.) Derive the Hessian for $l(\mu, \phi)$: $\bigtriangledown^2 l(\mu, \phi)$

$$\frac{\partial^2 l}{\partial \mu \partial \phi} = \frac{\partial^2 l}{\partial \phi \partial \mu} = \frac{-n}{\mu^3\phi^2}[\bar{y} - \mu]$$

$$\frac{\partial^2 l}{\partial \phi^2} = \sum_{i=1}^{n} \frac{1}{2\phi^2} + \sum_{i=1}^{n} \frac{-2(y_i - \mu)^2}{2\phi^3\mu^2 y_i} = \sum_{i=1}^{n} \left(\frac{1}{2\phi^2} - \frac{(y_i - \mu)^2}{\phi^3\mu^2 y_i} \right)$$

Thus, the Hessian for $l(\mu, \phi)$ is as follows

$$\bigtriangledown^{\mathbf{2}} \mathbf{l}(\mu, \phi) = \begin{bmatrix} \frac{n}{\mu^4\phi}[-3\bar{y} + 2\mu] & \frac{-n}{\mu^3\phi^2}[\bar{y} - \mu] \\ \\ \frac{-n}{\mu^3\phi^2}[\bar{y} - \mu] & \sum_{i=1}^{n} \left(\frac{1}{2\phi^2} - \frac{(y_i-\mu)^2}{\phi^3\mu^2 y_i} \right) \end{bmatrix}$$

D.) Derive the Fisher Information Matrix for $l(\mu,\phi)$, $I(\mu,\phi)$

$$E\left(\frac{\partial^2 l}{\partial\mu\partial\phi}\right) = E\left(\frac{\partial^2 l}{\partial\phi\partial\mu}\right) = \frac{-n}{\mu^3\phi^2}[E(\bar{y})-\mu] = \frac{-n}{\mu^3\phi^2}[\mu-\mu] = 0$$

$$E\left(\frac{\partial^2 l}{\partial\phi^2}\right) = \sum_{i=1}^{n}\left(\frac{1}{2\phi^2} - E\Big(\frac{y_i^2-2y_i\mu+\mu^2}{\phi^3\mu^2 y_i}\Big)\right) = \sum_{i=1}^{n}\left(\frac{1}{2\phi^2} - \frac{1}{\phi^3\mu^2}\Big(E\big(y_i - 2\mu + \frac{\mu^2}{y_i}\big)\Big)\right) =$$

$$\sum_{i=1}^{n}\left(\frac{1}{2\phi^2} - \frac{\mu-2\mu+\mu+\phi\mu^2}{\phi^3\mu^2}\right) = \sum_{i=1}^{n}\frac{-1}{2\phi^2} \Longrightarrow -E\Big(\frac{\partial^2 l}{\partial\phi^2}\Big) = \frac{n}{2\phi^2}$$

Thus, the Fisher Information Matrix, $I(\mu,\phi)$ is as follows:

$$\mathbf{I}(\mu,\phi) = \begin{bmatrix} \frac{n}{\mu^3\phi} & 0 \\ \\ 0 & \frac{n}{2\phi^2} \end{bmatrix}$$

E.) Write R functions for each of the functions in (A) - (D) having inputs μ,ϕ, and y.

Log-Likelihood:

```
likelihood2 = function(mu,phi,y){

  l = (-1/2)*sum(log(2*pi*phi*(y^3))) + sum((-(y-mu)^2)/(2*phi*(mu^2)*y))
  list(l=l)
}
```

Gradient Vetor:

```
gradient2=function(mu,phi,y){

  n=length(y)
  a=mu^2
  b=phi^2
  c=mu^3

  grad=matrix(c(0),2,1)

  grad[1]= (n/(c*phi))*(mean(y)-mu) #Dmu

  grad[2] = sum( (-1/(2*phi)) + ( (y-mu)^2/(2*phi^2*mu^2*y)) ) #Dphi

  return(grad)
}
```

Hessian Matrix:

```
hessian2=function(mu,phi,y){
  n=length(y)

  hess=matrix(c(0),2,2)


  hess[1,1] = (n*(-3*mean(y)+2*mu))/(mu^4*(phi)) #ddmumu
```

```r
  hess[1,2] =  (-n*(mean(y)-mu))/(mu^3*(phi^2))  #ddmuphi

  hess[2,1] = hess[1,2]    #ddphimu

  hess[2,2] = sum( (1/(2*phi^2)) + (-(y-mu)^2/(phi^3*mu^2*y)) )   #ddphiphi

  return(hess)
}
```

Fisher Information Matrix:

```r
fisher2=function(mu,phi,y){
  n=length(y)

  fish=matrix(c(0),2,2)

  fish[1,1]=n/(phi*(mu^3)) #mu

  fish[1,2]=0              #muphi

  fish[2,1]=fish[1,2]      #phimu

  fish[2,2]=n/((2*phi^2)) #phi

  return(fish)
}
```

F.) Derive the MLE of ϕ as a function of μ.

$$\frac{\partial l}{\partial \phi} = \sum_{i=1}^{n} \frac{-1}{2\phi} + \sum_{i=1}^{n} \frac{(y_i - \mu)^2}{2\phi^2\mu^2 y_i} = 0 \implies \frac{-n}{2\phi} + \sum_{i=1}^{n} \frac{(y_i - \mu)^2}{2\phi^2\mu^2 y_i} = 0$$

$$\implies \frac{n}{2\phi} = \frac{1}{2\phi^2\mu^2} \sum_{i=1}^{n} \frac{(y_i - \mu)^2}{y_i} \implies n\phi\mu^2 = \sum_{i=1}^{n} \frac{(y_i)^2}{y_i} \implies \hat{\phi} = \frac{1}{n\mu^2} \sum_{i=1}^{n} \frac{(y_i - \mu)^2}{y_i}$$

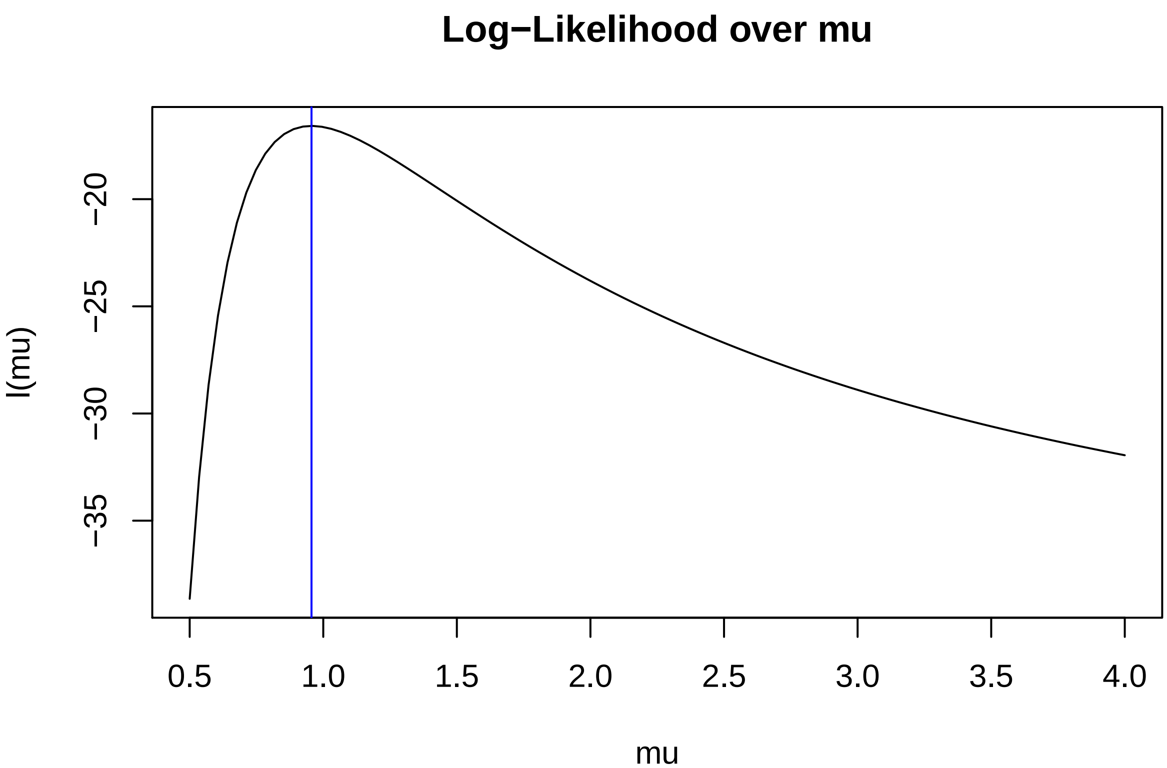
Using the dataset, plug in the mle for $\hat{\mu}$ to obtain the value of $\hat{\phi}$ according to these data.

```r
phi_mle= function(mu,phi,y){
  n=length(y)

  p=(1/((mu^2)*n))*sum(((y-mu)^2)/y)

  return(p)
}

phi_star<<-phi_mle(mu_star,phi,surv)
phi_star
```

```
## [1] 0.4731563
```

G.) Obtaim the following plots using the log-likelihood function.
i.) log-likelihood over μ.

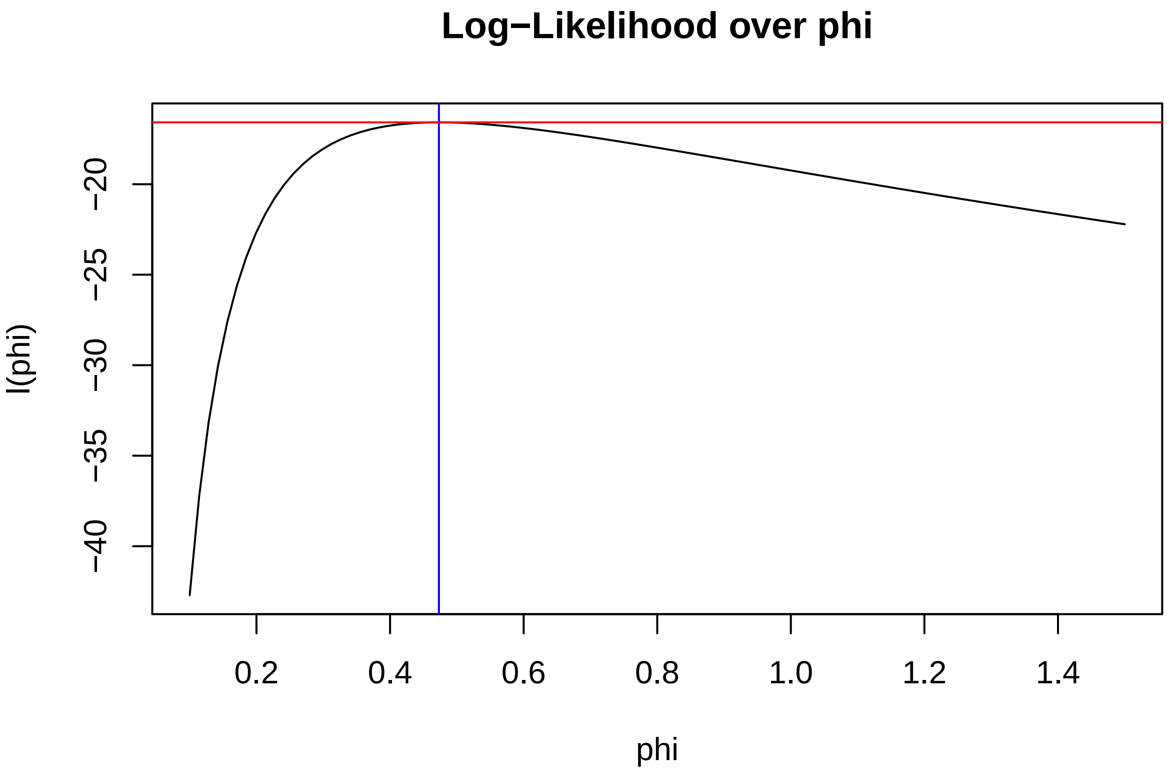
```
mu=seq(0.5,4, length=100)
phi=phi_star
y=surv

plot(mu, sapply(X=mu, FUN=function(mu) likelihood2(mu,phi,surv)),
type="l",xlab = "mu",ylab="l(mu)",main="Log-Likelihood over mu")
abline(v=mean(y),col="blue")
```



Again, we can see that the analytical MLE computed matches the visual Maximum found in the graph.

ii.) log-likelihood over ϕ.

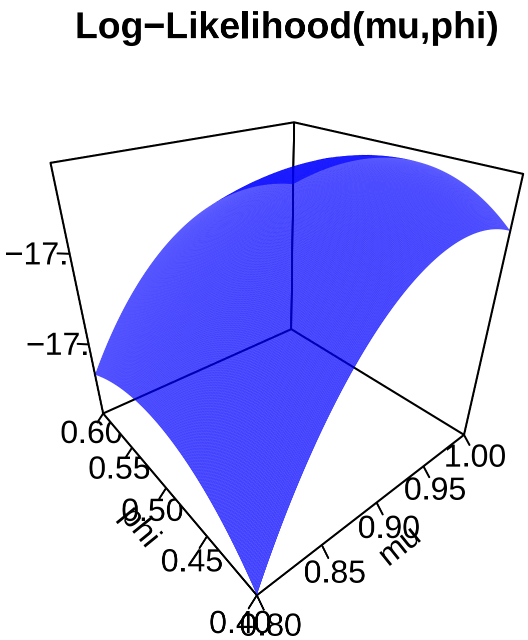
```
phi=seq(.1,1.5, length=100)
y=surv
mu=mu_star
max=likelihood2(mu_star,phi_star,surv)

plot(phi, sapply(X=phi, FUN=function(phi) likelihood2(mu,phi,surv)),
type="l",xlab = "phi",ylab="l(phi)",main="Log-Likelihood over phi")
abline(v=phi_star,col="blue")
abline(h=max,col="red")
```



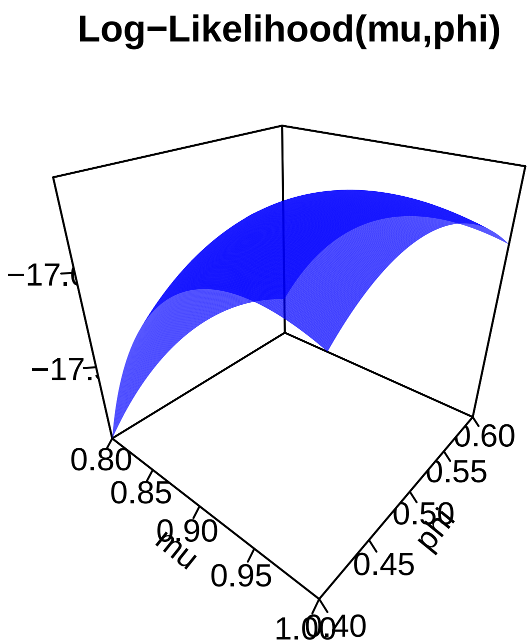
Now, using ϕ as a support, we can see that the MLE computed using $\hat{\mu}$ matches our graph of the likelihood function, supporting our estimate of $\hat{\phi}$.

iii.) 3-D plot over μ and ϕ.

```
y<<-surv

loglikevector = function(m,p) {
  n1 = length(m)
  l = numeric(length(n1))
  for (i in 1:n1){
    l[i]= likelihood2(m[i],p[i],surv)$l
  }
  l
}

m = seq(.8,1,len=300)
p =seq(.4,.6,len=300)

l = outer(m,p,loglikevector)

persp3D(m,p, l, xlab = "mu", ylab = "phi", zlab ="",
        main = "Log-Likelihood(mu,phi)" ,
        col = 4 ,ticktype="detailed",theta = -40, phi =30,alpha= 0.5)
```

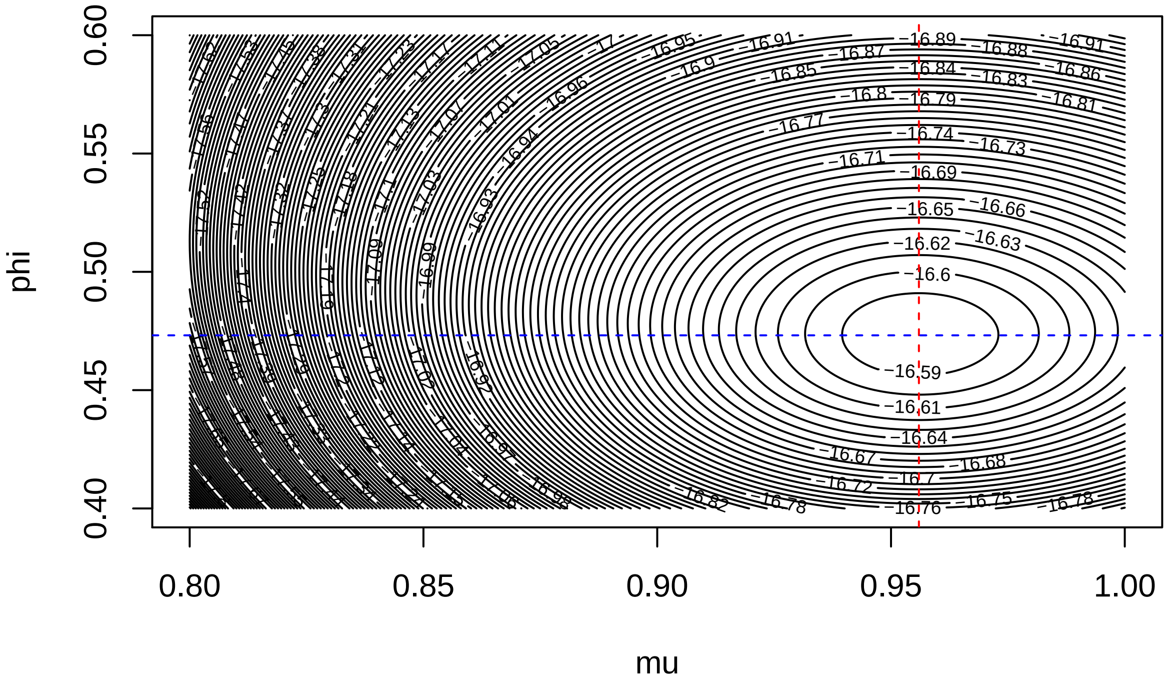



```r
persp3D(m,p, l, xlab = "mu", ylab = "phi", zlab ="",
        main = "Log-Likelihood(mu,phi)" ,
        col = 4 ,ticktype="detailed",theta = 40, phi =30,alpha= 0.5)
```



iv.) Contour plot of the log-likelihood

```r
contour(m,p,l,nlevels=100,xlab='mu',ylab='phi')
abline(v=mean(y),lty=2,col='red')
abline(h=phi_star,lty=2,col='blue')
```



From the 3-D plots and the contour plots, we have a visual perspective of where the MLE resides with respect to both μ and ϕ. The 3-D plots show us that the Likelihood is maximized in a region near the coordinates of our $\hat{\mu}$ and $\hat{\phi}$ estimates. The Contour plot shows us that the highest point in our log-likelihood occurs at the intersection of our respective MLE estimates, justifying our calculations.

H.) Write an R function to apply Fisher's Scoring Method and Newton's Method to μ and ϕ.

Fisher:

```
FisherMethod = function(maxit,mu,phi,y){

cat("Iteration", " mu(n)", "               phi(n)",    "     C.R.Mu",    "     C.R.Phi\n")

  v_star=matrix(c(mu_star,phi_star)) #Vectorized MLEs
  v=matrix(c(mu,phi),2,1)            #Vectorized parameters

  cat(sprintf(' %2.0f       %4.4f            %4.4f            %4.4f   %4.4f\n'
            ,0,v[1],v[2],0,0))

  for(i in 1:maxit){
    g=gradient2(v[1],v[2],y)
    f=fisher2(v[1],v[2],y)
    finv=solve(f)

    v_n = v + (finv)%*%g              #Fisher method

    halve=0

    while(min(v_n[1],v_n[2]) <= 0 | likelihood(v_n[1],v_n[2],y) < likelihood(v[1],v[2],y))
    {
    halve=halve+1                         #Halve Stepping to ensure positive parameters
    v_n = v_n - (finv%*%g/(2^halve))      #Final value for the iteration
```

```
    }

    cvm = abs(v_n[1]-v_star[1])/abs(v[1]-v_star[1])  #Convergence Ratio for Mu

    cvp = abs(v_n[2]-v_star[2])/abs(v[2]-v_star[2])  #Convergence Ratio for Phi

    cat(sprintf(' %2.0f       %12.12f     %12.12f     %4.4f   %4.4f\n'
                ,i,v_n[1],v_n[2],cvm,cvp))

    v=v_n                                           #update parameter vector
  }
}
```

Using $\mu^{(0)}$=$\phi^{(0)}$=1

```
FisherMethod(20,1,1,surv)
```

```
## Iteration  mu(n)               phi(n)      C.R.Mu      C.R.Phi
##    0       1.0000             1.0000            0.0000   0.0000
##    1       0.955965971525      0.475184599164      0.0000   0.0038
##    2       0.955965971525      0.473156288825       NaN   0.0000
##    3       0.955965971525      0.473156288825       NaN   0.0000
##    4       0.955965971525      0.473156288825       NaN    Inf
##    5       0.955965971525      0.473156288825       NaN   1.0000
##    6       0.955965971525      0.473156288825       NaN   1.0000
##    7       0.955965971525      0.473156288825       NaN   1.0000
##    8       0.955965971525      0.473156288825       NaN   1.0000
##    9       0.955965971525      0.473156288825       NaN   1.0000
##   10       0.955965971525      0.473156288825       NaN   1.0000
##   11       0.955965971525      0.473156288825       NaN   1.0000
##   12       0.955965971525      0.473156288825       NaN   1.0000
##   13       0.955965971525      0.473156288825       NaN   1.0000
##   14       0.955965971525      0.473156288825       NaN   1.0000
##   15       0.955965971525      0.473156288825       NaN   1.0000
##   16       0.955965971525      0.473156288825       NaN   1.0000
##   17       0.955965971525      0.473156288825       NaN   1.0000
##   18       0.955965971525      0.473156288825       NaN   1.0000
##   19       0.955965971525      0.473156288825       NaN   1.0000
##   20       0.955965971525      0.473156288825       NaN   1.0000
```

Using $\mu^{(0)}$= .5 $\phi^{(0)}$=.3 , where the initial values are below the MLEs.

```
FisherMethod(20,.5,.3,surv)
```

```
## Iteration  mu(n)               phi(n)      C.R.Mu      C.R.Phi
##    0       0.5000             0.3000            0.0000   0.0000
##    1       0.955965971525      1.343082513738      0.0000   5.0239
##    2       0.955965971525      0.473156288825       NaN   0.0000
##    3       0.955965971525      0.473156288825       NaN    Inf
##    4       0.955965971525      0.473156288825       NaN   1.0000
##    5       0.955965971525      0.473156288825       NaN   1.0000
##    6       0.955965971525      0.473156288825       NaN   1.0000
##    7       0.955965971525      0.473156288825       NaN   1.0000
##    8       0.955965971525      0.473156288825       NaN   1.0000
##    9       0.955965971525      0.473156288825       NaN   1.0000
##   10       0.955965971525      0.473156288825       NaN   1.0000
```

```
## 11       0.955965971525     0.473156288825      NaN   1.0000
## 12       0.955965971525     0.473156288825      NaN   1.0000
## 13       0.955965971525     0.473156288825      NaN   1.0000
## 14       0.955965971525     0.473156288825      NaN   1.0000
## 15       0.955965971525     0.473156288825      NaN   1.0000
## 16       0.955965971525     0.473156288825      NaN   1.0000
## 17       0.955965971525     0.473156288825      NaN   1.0000
## 18       0.955965971525     0.473156288825      NaN   1.0000
## 19       0.955965971525     0.473156288825      NaN   1.0000
## 20       0.955965971525     0.473156288825      NaN   1.0000
```

Using $\mu^{(0)}$= 1.2 $\phi^{(0)}$=1.5, where the initial values are above the MLEs.

```r
FisherMethod(20,1.2,1.5,surv)
```

```
## Iteration  mu(n)              phi(n)      C.R.Mu      C.R.Phi
##    0       1.2000            1.5000          0.0000   0.0000
##    1       0.955965971525     0.516417218976     0.0000   0.0421
##    2       0.955965971525     0.473156288825      NaN   0.0000
##    3       0.955965971525     0.473156288825      NaN    Inf
##    4       0.955965971525     0.473156288825      NaN   1.0000
##    5       0.955965971525     0.473156288825      NaN   1.0000
##    6       0.955965971525     0.473156288825      NaN   1.0000
##    7       0.955965971525     0.473156288825      NaN   1.0000
##    8       0.955965971525     0.473156288825      NaN   1.0000
##    9       0.955965971525     0.473156288825      NaN   1.0000
##   10       0.955965971525     0.473156288825      NaN   1.0000
##   11       0.955965971525     0.473156288825      NaN   1.0000
##   12       0.955965971525     0.473156288825      NaN   1.0000
##   13       0.955965971525     0.473156288825      NaN   1.0000
##   14       0.955965971525     0.473156288825      NaN   1.0000
##   15       0.955965971525     0.473156288825      NaN   1.0000
##   16       0.955965971525     0.473156288825      NaN   1.0000
##   17       0.955965971525     0.473156288825      NaN   1.0000
##   18       0.955965971525     0.473156288825      NaN   1.0000
##   19       0.955965971525     0.473156288825      NaN   1.0000
##   20       0.955965971525     0.473156288825      NaN   1.0000
```

Once again, we see that the Fisher Method is very reliable in this situation. In the three scenarios we conducted, the MLE estimates are found very quickly. Again, if all calculations and scripts are conducted correctly and effeciently, then the strength of Fisher's method and the lack of maximums in the likelihood function cause the maximum to be found immediatly.

Newton:

```r
Newton2 = function(maxit,mu,phi,y){

  cat("Iteration", " mu(n)", "          phi(n)",    "               C.R.Mu",    "         C.R.Phi\n")

  v_star=matrix(c(mu_star,phi_star),2,1) #Vectorized MLEs
  v=matrix(c(mu,phi),2,1)                #Vectorized parameters

  cat(sprintf(' %2.0f       %4.4f              %4.4f           %4.4f          %4.4f\n'
              ,0,v[1],v[2],0,0))

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

```
    g=gradient2(v[1],v[2],y)
    h=hessian2(v[1],v[2],y)
    hinv=solve(h)


    v_n = v - hinv%*%g              #Newton method

    halve=0

while(min(v_n[1],v_n[2]) <= 0 | likelihood(v_n[1],v_n[2],y) < likelihood(v[1],v[2],y) )

   {
   halve=halve+1                        #Halve Stepping to ensure positive parameters/direction
   v_n = v_n - ((hinv%*%g)/(2^halve))     #Final value for the iteration

   if(halve>100){
     break
   }

   }

    cvm = abs(v_n[1]-v_star[1])/abs(v[1]-v_star[1])  #Convergence Ratio for Mu

    cvp = abs(v_n[2]-v_star[2])/abs(v[2]-v_star[2])  #Convergence Ratio for Phi

    cat(sprintf(' %2.0f       %12.12f     %12.12f     %12.12f   %12.12f\n'
                ,i,v_n[1],v_n[2],cvm,cvp))

    v=v_n                                          #update parameter vector
  }
}
```

Using $\mu^{(0)}=\phi^{(0)}=1$

```
Newton2(20,1,1,surv)
```

```
## Iteration  mu(n)           phi(n)              C.R.Mu          C.R.Phi
##    0       1.0000              1.0000          0.0000          0.0000
##    1       1.891296937554    20.567203822868     21.241094635532   38.140433505833
##    2       1.739491311655    63.316025218555     0.837698492392   3.127437059321
##    3       1.689601045693    191.364376677463     0.936325906250   3.037595571935
##    4       1.672429308200    575.448601625582     0.976593625227   3.012058093432
##    5       1.666600508213    1727.680944081799     0.991864482537   3.003967911676
##    6       1.664643562634    5184.371145233793     0.997246199729   3.001316937998
##    7       1.663989589705    15554.439465655671     0.999077192595   3.000438346306
##    8       1.663771410603    46664.643664968113     0.999691847707   3.000146045101
##    9       1.663698663185    139995.256008812721     0.999897221166   3.000048673885
##   10       1.663674411701    419987.092955636734     0.999965733554   3.000016223760
##   11       1.663666327613    1259962.603767870925     0.999988577092   3.000005407824
##   12       1.663663632888    3779889.136195160449     0.999996192279   3.000001802597
##   13       1.663662734643    11339668.733473889530     0.999998730750   3.000000600865
##   14       1.663662435227    34019007.525309026241     0.999999576916   3.000000200288
##   15       1.663662335422    102057023.900814071298     0.999999858972   3.000000066763
##   16       1.663662302154    306171073.027329087257     0.999999952991   3.000000022254
```

```
## 17          1.663662291064      918513220.406874060631      0.999999984330    3.000000007418
## 18          1.663662287368      2755539662.545508384705      0.999999994777    3.000000002473
## 19          1.663662286136      8266618988.961410522461      0.999999998259    3.000000000824
## 20          1.663662285725      24799856968.209114074707      0.999999999420    3.000000000275
```

Using $\mu^{(0)}$= .5 $\phi^{(0)}$=.3 , where the initial values are below the MLEs.

```
Newton2(20,.5,.3,surv)
```

```
## Iteration  mu(n)            phi(n)              C.R.Mu          C.R.Phi
##   0        0.5000             0.3000          0.0000          0.0000
##   1        0.609596953068     0.330616844857    0.759637867929    0.823183754601
##   2        0.725046895257     0.363190251946    0.666685136262    0.771477941948
##   3        0.829921931922     0.403830808772    0.545836410052    0.630426284525
##   4        0.907852056403     0.444222533298    0.381723049132    0.417361055478
##   5        0.947174175323     0.467672461780    0.182728763181    0.189530427183
##   6        0.955631337883     0.472945715031    0.038062033536    0.038399058155
##   7        0.955965471745     0.473155973506    0.001493513838    0.001497424667
##   8        0.955965971523     0.473156288824    0.000002234752    0.000002239506
##   9        0.955965971525     0.473156288825    0.000000000000    0.000078610172
##  10        0.955965971525     0.473156288825              NaN    1.000000000000
##  11        0.955965971525     0.473156288825              NaN    1.000000000000
##  12        0.955965971525     0.473156288825              NaN    1.000000000000
##  13        0.955965971525     0.473156288825              NaN    1.000000000000
##  14        0.955965971525     0.473156288825              NaN    1.000000000000
##  15        0.955965971525     0.473156288825              NaN    1.000000000000
##  16        0.955965971525     0.473156288825              NaN    1.000000000000
##  17        0.955965971525     0.473156288825              NaN    1.000000000000
##  18        0.955965971525     0.473156288825              NaN    1.000000000000
##  19        0.955965971525     0.473156288825              NaN    1.000000000000
##  20        0.955965971525     0.473156288825              NaN    1.000000000000
```

Using $\mu^{(0)}$= 1.2 $\phi^{(0)}$=1.5, where the initial values are above the MLEs.

```
Newton2(20,1.2,1.5,surv)
```

```
## Iteration  mu(n)            phi(n)              C.R.Mu          C.R.Phi
##   0        1.2000             1.5000          0.0000          0.0000
##   1        2.203844542146     6.905897912890    5.113543297293    6.264577125085
##   2        1.611081727421     23.009667530251    0.524983577184    3.503406876644
##   3        1.425846870211     70.459629376216    0.717248660954    3.105470600025
##   4        2.012619624988     212.378834064138    2.248769116638    3.027809066914
##   5        1.999317372644     638.673961099361    0.987410962618    3.011721118144
##   6        1.994924489300     1917.542704210461    0.995789641592    3.003865763677
##   7        1.993464608633     5754.143458073241    0.998594861449    3.001284073404
##   8        1.992978466145     17263.943902134484    0.999531428312    3.000427525451
##   9        1.992816472249     51793.344629316038    0.999843787903    3.000142453133
##  10        1.992762480231     155381.546609286364    0.999947926902    3.000047478231
##  11        1.992744483552     466146.152482016245    0.999982642034    3.000015825394
##  12        1.992738484733     1398439.970077813370    0.999994213982    3.000005275056
##  13        1.992736485134     4195321.422857739963    0.999998071324    3.000001758343
##  14        1.992735818602     12585965.781195031479    0.999999357108    3.000000586114
##  15        1.992735596425     37757898.856206074357    0.999999785702    3.000000195371
##  16        1.992735522366     113273698.081238925457    0.999999928567    3.000000065124
##  17        1.992735497680     339821095.756337344646    0.999999976189    3.000000021708
##  18        1.992735489451     1019463288.781632423401    0.999999992063    3.000000007236
```

```
## 19    1.992735486708    3058389867.857517719269    0.999999997354   3.000000002412
## 20    1.992735485794    9175169605.085172653198    0.999999999118   3.000000000804
```

Similar to the univariate case of the Newton method, the MLE estimates of μ and ϕ are found only when the initial values are within a close neighborhood of the actual calculated MLE values. If initial values are too distant from the MLE of this case, we see the iterations jump to extreme values. In a sense, the method is searching for the next maximum value but is unable to detect such value. Also, in regards to the convergence ratio for previous cases mentioned, it is important to note that most methods that converge will converge to a value of 0,1, or infinity. We see this different depending on the accuracy of the fraction evaluated in our Convergence formula. For μ, we see the convergence go to infinity or NA since the estimate is exact and for ϕ we see the estimate go to 1 since we found $\hat{\phi}$ by using $\hat{\mu}$ as a starting point.