STAT 428 – Homework #4
Luis Steven Lin

Problem 1

R-code

```
# Newton method to find root
# Inputs: function, derivative, initial guess
#         maximum number of iterations

newton = function(number,x0,maxiter){

      # Define Function
      f = function(x){x^(2)-number}

      # Define Derivative
      df = function(x){2*x}

      # Initialize variables
      maxiter = maxiter
      iter=1
      err=1
      x = x0

      # Loop
      while( err > 10^-12 & iter <= maxiter) {
            xnew = x - f(x)/df(x)
            err = abs( (xnew-x)/xnew )*100
            x = xnew
            iter = iter + 1;
      }

      # Results
      print(paste("Iterations:", iter))
      print(paste("Root of ", number, "is:"))
      return(x)
}
```

R-output

```
> # Find root of 897 using newton
> newton(897,30,100)
[1] "Iterations: 5"
[1] "Root of  897 is:"
[1] 29.94996
>
> # Check with built-in R function
> sqrt(897)
[1] 29.94996
```

Problem 2

Part a

Density of the data:

$$f(x_i|k) = k\, x_i^{k-1}\, e^{-x_i^k}$$

Likelihood function:

$$l(k; x_i) = \prod_{i=1}^{n} f(x_i|k) = \prod_{i=1}^{n} k\, x_i^{k-1}\, e^{-x_i^k}$$

Log Likelihood function:

$$L(k; x_i) = \ln\left(\prod_{i=1}^{n} k\, x_i^{k-1}\, e^{-x_i^k}\right) = n\ln(k) + (k-1)\sum_{i=1}^{n}\ln(x_i) - \sum_{i=1}^{n} x_i{}^k$$

First derivative of Log Likelihood function:

$$L'(k; x_i) = \frac{d\,L(k;x_i)}{dk} = \frac{\mathrm{n}}{k} + \sum_{i=1}^{n}\ln(x_i) - \sum_{i=1}^{n} x_i{}^k \ln(x_i)$$

Second derivative of Log Likelihood function:

$$L''(k; x_i) = \frac{d^2 L(k;x_i)}{dk^2} = -\frac{n}{k^2} - \sum_{i=1}^{n} x_i{}^k \ln(x_i)^2$$

Find the maximum likelihood estimate for k using Newton's Method (finding the root of the derivative of a log likelihood) by programming a function in R that takes as input the data X and returns as output the MLE of k.

R-code

```
x = c(1.4413414, 1.0559832, 1.2363259, 0.2419938, 0.4254919,
0.1195233, 1.1289144, 0.2114744, 1.4926782, 0.5879979)

newton = function(f,df,x0,maxiter){

      # Initialize variables
      maxiter = maxiter
      iter=1
      err=1
      x = x0

      # Netwon loop

      while( err > 10^-12 & iter <= maxiter) {
            xnew = x - f(x)/df(x)
            err = abs( (xnew-x)/xnew )*100
            x = xnew
            iter = iter + 1;
      }
      return(list(x=x,iter=iter))
}

weibullMLE = function(x,k0,maxiter,print=FALSE){

      n = length(x)
      lnx = log(x)

      # Define first derivative of log likelihood

      dL1 = function(k){ n/k + sum(lnx) - sum((x^k)*lnx) }

      # Define second derivative of log likelihood

      dL2 = function(k){ -n/k^2 - sum((x^k)*lnx^2) }

      # Newton to find root of first derivative

      mle = newton(dL1,dL2,k0,maxiter)

      # Results
      if (print==TRUE){
            print(paste("Iterations:", mle$iter))
            print(paste("MLE estimate of k: ", mle$x))}
      return(mle$x)
}
```

R-output

```
> k = weibullMLE(x,1,100,print=TRUE)
[1] "Iterations: 7"
[1] "MLE estimate of k:  1.56233996691347"
> k
[1] 1.56234
```

Part b

Simulate data from a Weibull distribution with parameter k in R and use function from part a to find the MLE of k for this simulated random sample x

R-code

```
set.seed(1)
k = 2
n = 100
x = rweibull(n,k)
```

R-output

```
> k = weibullMLE(x,1,100,print=TRUE)
[1] "Iterations: 8"
[1] "MLE estimate of k:  2.29429468647011"
> k
[1] 2.294295
```

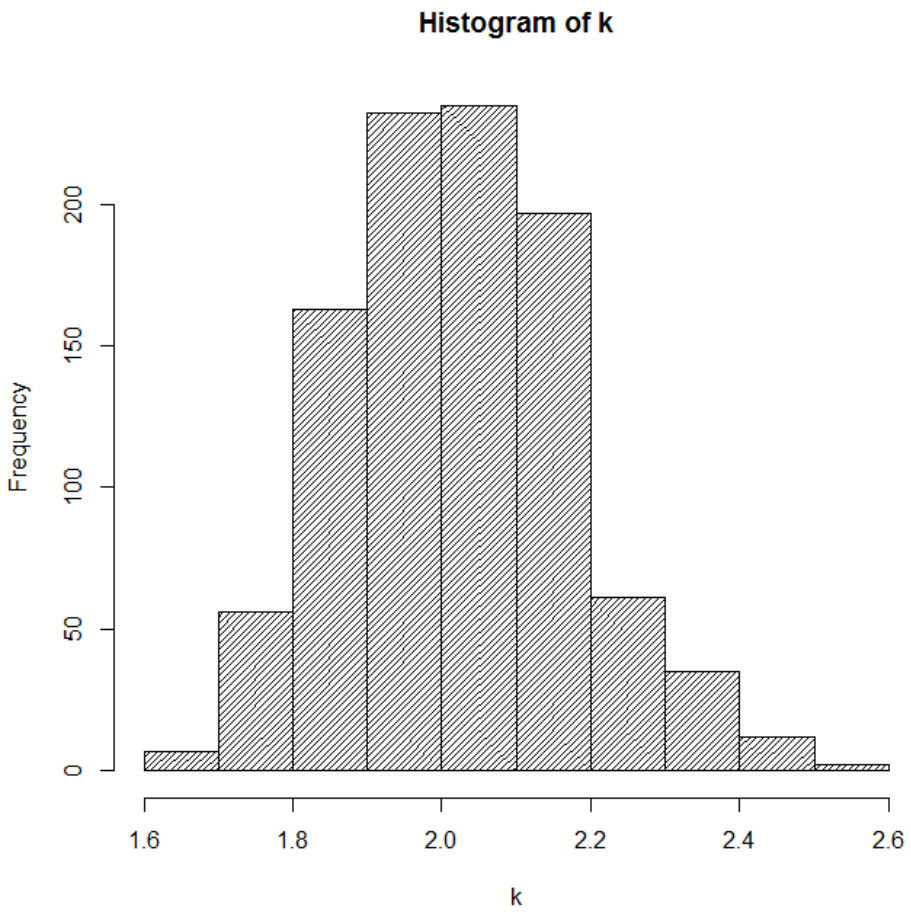
Part c

Generate 1000 such samples of X in a loop. For each sample, find the MLE for k, say k and record it in a vector. Plot a histogram of the k after the loop runs.

R-code

```
n=1000
k = 1:n
for (i in 1:n){
      set.seed(i)
      x = rweibull(100,2)
      k[i] = weibullMLE(x,1,1000);
      }
hist(k,density=30)
```

R-output



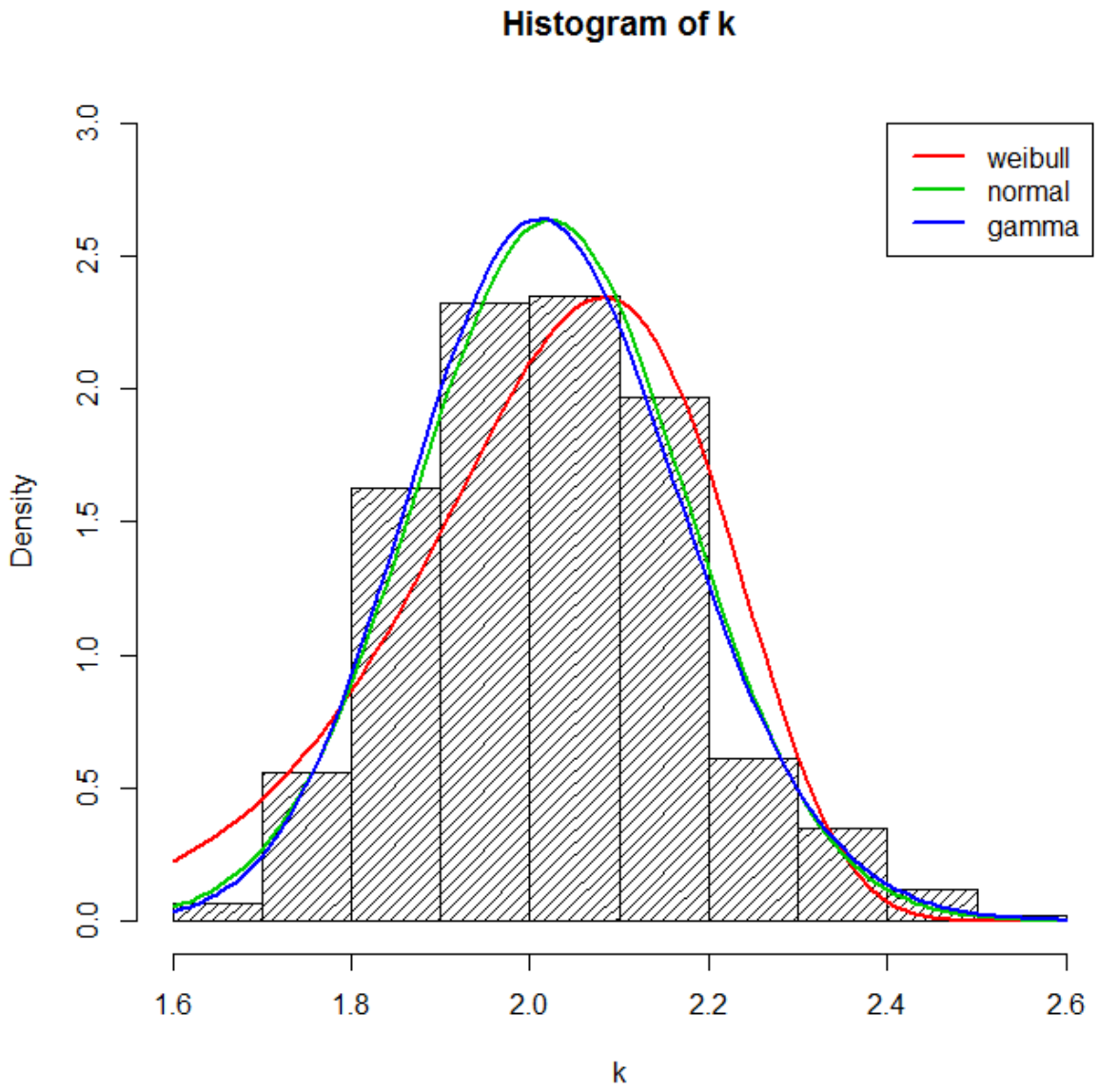
Part d

What distribution does this histogram of k (obtained in c) look like?

R-code

```
library(MASS)

x=k
weibMLE = fitdistr(k,"weibull")$estimate
normalMLE = fitdistr(k,"normal")$estimate
gammaMLE = fitdistr(x,"gamma")$estimate

hist(k,freq=FALSE,ylim=c(0,3),density=20)

curve(dweibull(x,weibMLE["shape"],weibMLE["scale"]),col=2,lwd=2,add=TRUE)
curve(dnorm(x,normalMLE["mean"],normalMLE["sd"]),col=3,lwd=2,add=TRUE)
curve(dgamma(x,gammaMLE["shape"],gammaMLE["rate"]),col=4,lwd=2,add=TRUE)

legend(2.4,3,c("weibull","normal","gamma"),col=c(2,3,4),lty=1,lwd=2)
```

R-output



From the plot showing the density curves over the histogram, it can be seen that the histogram does not look like a weibull distribution. The histogram looks more like a normal or gramma distribution. However, goodness of fit tests should be conducted to determine which is a better fit or if any fit is appropriate.

Problem 3

Simulate from a mixture of Exponentials. Write an Expectation-Maximization function to fit the data X simulated above (treat the parameters as unknown). It should return the history of the parameters at each iteration.

R-code:

```
set.seed(1)
N = 100 #Number of observations
p = .5 #Mixing parameter
lambdas = c(1.2, 4.8) #Exponential rate parameter for each distribution
Z = sample(c(1,2), N, replace = TRUE, prob=c(1-p,p))
X = sapply(Z,function(c) rexp(1,rate = lambdas[c])) #goes through each
#entry in Z and applies the function to it.
hist(X) #it is pretty hard to see a mixture of exponentials

EM = function(X,print=FALSE){
      Max.iter=10000
      L=c(1,5)
      tol=.Machine$double.eps^0.5
      L.old=L+1
      for(j in 1:Max.iter){
      #Estep
            f1=dexp(X,L[1])
            f2=dexp(X,L[2])
            p1=f1/(f1+f2)
            p2=1-p1
      #Mstep (note that the mean is the inverse of the pararmeter)
      phat=1/(sum(p1)/length(p1))
      L[1]=1/(sum(p1*X)/sum(p1))
      L[2]=1/(sum(p2*X)/sum(p2))
      if (print==TRUE){
      print(paste("Iteration: ", j, ", lamda1: ", L[1],
                  ", lamda2: ", L[2]))   }
      #stop?
      if(sum(abs(L-L.old)/L.old)<tol) break
      L.old=L
    }
      return(L)
 }

ret = EM(X,print=TRUE)
ret
```

R-output:

```
> ret = EM(X,print=TRUE)
[1] "Iteration:  1 , lamda1:  1.21378 , lamda2:  4.61506"
[1] "Iteration:  2 , lamda1:  1.25992 , lamda2:  4.43599"
[1] "Iteration:  3 , lamda1:  1.27098 , lamda2:  4.34202"
[1] "Iteration:  4 , lamda1:  1.27469 , lamda2:  4.29279"
[1] "Iteration:  5 , lamda1:  1.27636 , lamda2:  4.26693"
[1] "Iteration:  6 , lamda1:  1.27722 , lamda2:  4.25327"
[1] "Iteration:  7 , lamda1:  1.27768 , lamda2:  4.24603"
[1] "Iteration:  8 , lamda1:  1.27792 , lamda2:  4.24218"
[1] "Iteration:  9 , lamda1:  1.27806 , lamda2:  4.24013"
[1] "Iteration:  10 , lamda1:  1.27813 , lamda2:  4.23903"
[1] "Iteration:  11 , lamda1:  1.27817 , lamda2:  4.23845"
[1] "Iteration:  12 , lamda1:  1.27819 , lamda2:  4.23814"
[1] "Iteration:  13 , lamda1:  1.2782 , lamda2:  4.23797"
[1] "Iteration:  14 , lamda1:  1.2782 , lamda2:  4.23788"
[1] "Iteration:  15 , lamda1:  1.27821 , lamda2:  4.23783"
[1] "Iteration:  16 , lamda1:  1.27821 , lamda2:  4.23781"
[1] "Iteration:  17 , lamda1:  1.27821 , lamda2:  4.23779"
[1] "Iteration:  18 , lamda1:  1.27821 , lamda2:  4.23779"
[1] "Iteration:  19 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  20 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  21 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  22 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  23 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  24 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  25 , lamda1:  1.27821 , lamda2:  4.23778"
[1] "Iteration:  26 , lamda1:  1.27821 , lamda2:  4.23778"
> ret
[1] 1.278209 4.237778
```

If repeated with a larger number of observations:

```
> ret
[1] 1.207118 4.815175
```