5361 HW2

Tianshu Zhao

February 7, 2018

Question 1

1(a)

Proof.

$$l(\theta) \tag{1}$$
$$= \ln \prod_{i=1}^{n} \frac{1}{\pi[1+(x-\theta)^2]} \tag{2}$$
$$= \sum_{i=1}^{n} \ln \frac{1}{\pi[1+(x-\theta)^2]} \tag{3}$$
$$= \sum_{i=1}^{n} [\ln \frac{1}{\pi} + \ln \frac{1}{1+(x-\theta)^2}] \tag{4}$$
$$= -n \ln \pi - \sum_{i=1}^{n} \ln[1+(x-\theta)^2] \tag{5}$$
$$\tag{6}$$

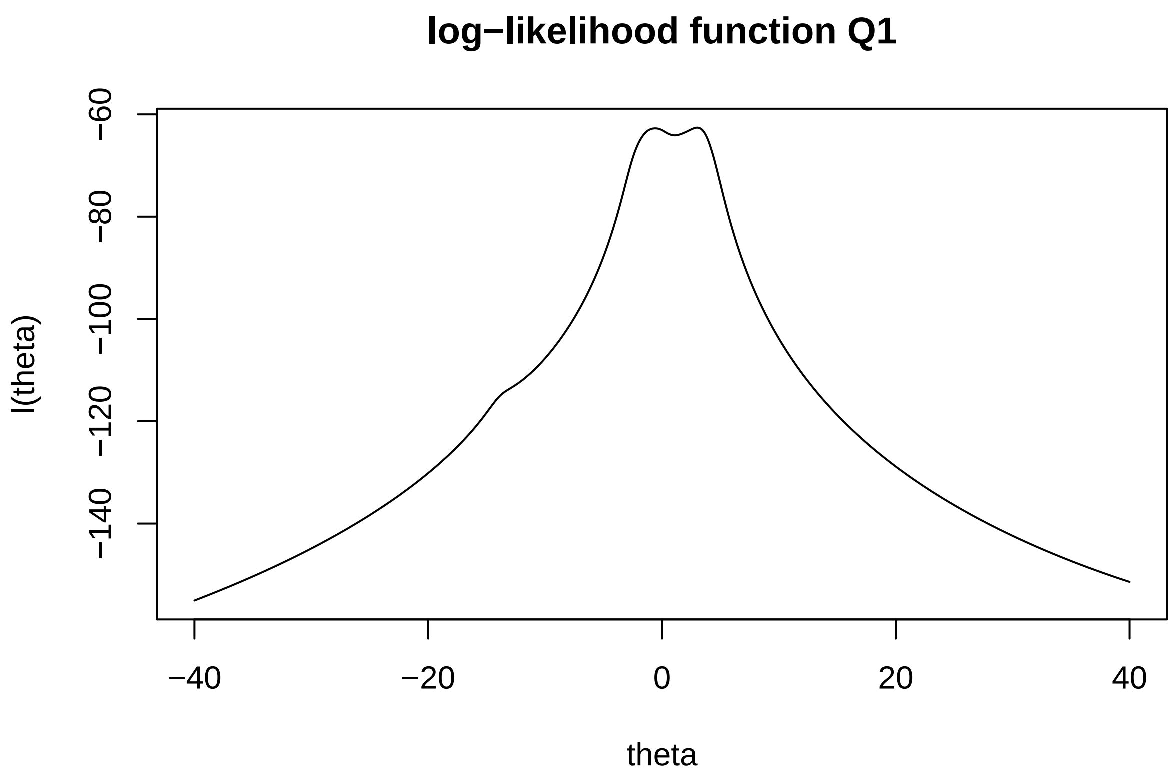
$$l'(\theta) \tag{7}$$
$$= 0 - \sum_{i=1}^{n} \frac{2(\theta - x_i)}{1+(\theta - x_i)^2} \tag{8}$$
$$= -2 \sum_{i=1}^{n} \frac{\theta - x_i}{1+(\theta - x_i)} \tag{9}$$
$$\tag{10}$$

$$l''(\theta) \tag{11}$$
$$= -2 \sum_{i=1}^{n} \frac{1+(\theta - x_i)^2 - 2(\theta - x_i)(\theta - x_i)}{[1+(\theta - x_i)^2]^2} \tag{12}$$
$$= -2 \sum_{i=1}^{n} \frac{1-(\theta - x_i)^2}{[1+(\theta - x_i)^2]^2} \tag{13}$$
$$\tag{14}$$

$$
\begin{aligned}
&I(\theta) && (15)\\
&= n\int \frac{\{p'(x)\}^2}{p(x)}dx && (16)\\
&= n\int \frac{4(x-\theta)^2}{\pi[1+(x-\theta)^2]^4} * \pi[1+(x-\theta)^2]dx && (17)\\
&= \frac{4n}{\pi}\int_{-\infty}^{\infty} \frac{(x-\theta)^2}{[1+(x-\theta)^2]^3}dx && (18)\\
&= \frac{4n}{\pi}\int_{-\infty}^{\infty} \frac{x^2}{(1+x^2)^3}dx && (19)\\
&= \frac{4n}{\pi}(\int_{-\infty}^{\infty} \frac{1}{(1+x^2)^2}dx - \int_{-\infty}^{\infty} \frac{1}{(1+x^2)^3}dx) && (20)\\
&= \frac{4n}{\pi}[\int_{-\infty}^{\infty} \frac{1}{(1+x^2)^2}dx - (\frac{x}{4(1+x^2)^2}|_{-\infty}^{\infty} + \frac{3}{4}\int_{-\infty}^{\infty} \frac{dx}{(1+x^2)^2})] && (21)\\
&= \frac{4n}{\pi}(\int_{-\infty}^{\infty} \frac{1}{4(1+x^2)^2}dx - \frac{x}{4(1+x^2)^2}|_{-\infty}^{\infty}) && (22)\\
&= \frac{4n}{\pi}[\frac{1}{4}(\frac{x}{2(1+x^2)}|_{-\infty}^{\infty} + \frac{1}{2}\int_{-\infty}^{\infty} \frac{1}{1+x^2}dx) - \frac{x}{4(1+x^2)^2}|_{-\infty}^{\infty}] && (23)\\
&= \frac{4n}{\pi}(\frac{x(x^2-1)}{8(1+x^2)^2}|_{-\infty}^{\infty} + \frac{1}{8}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\sec^2 t}{1+\tan^2 t}dt) && (24)\\
&= \frac{4n}{\pi}(0+\frac{\pi}{8}) && (25)\\
&= \frac{n}{2} && (26)\\
& && (27)
\end{aligned}
$$

1(b)

The gragh of likelihood function



Find MLE using Newton-Raphson method:

Below is the code “nlminb” for Newton-Raphson method:

```
nlminb_1b <- function(x0){
  X <- array()
  X[1] <- x0
  i <- 1
  difference <- 1
  while(abs(difference)>= 0.0001){
    X[i+1] <- X[i]-gr_l(X[i])/hess_l(X[i])
    difference <- X[i+1]-X[i]
    i <- i+1
  }
  return(X[i])
}
```

Where gr_l and hess_l are gradient and Hessian for the log-likelihood function:

```
gr_l <- function(t){
  grad_l <- -2*sum((t-x)/(1+(t-x)^2))
  return(grad_l)
}
hess_l <- function(t){
  he_l <- -2*sum((1-(t-x)^2)/(1+(t-x)^2))^2
  return(he_l)
}
```

Find MLE for the following starting points

```
sp <- c(-11,-1, 0, 1.5, 4, 4.7, 7, 8, 38)
```

```
MLE_1b <- array()
for (k in 1:length(sp)){
  MLE_1b[k] <- nlminb_1b(x0=sp[k])
}
```

MLE vector is

```
## [1] -0.5987182 -0.5987148 -0.5842149  3.0189234  3.0237233  3.0237384
## [7]  3.0236737  3.0237394  3.0236906
```

If we use sample mean as the starting point, then MLE equals to:

```
## [1] 3.023685
```

So, sample mean is a good starting point compared to each sample.

1(c)

Apply fixed-point iterations using G(x) = alpha*l'(theta)+theta, with scaling choices of ?? ??? {1, 0.64, 0.25}, for the same starting points above.

Code "nlminb" for Newton-Raphson method is:

```
#nlminb for fixed point iteration
nlminb_1c <- function(x0,alpha){
  X <- array()
  X[1] <- x0
  i=1
  difference <- 1
  while(abs(difference)>= 0.0001 & i<100){
    X[i+1] <- X[i]+gr_l(X[i])*alpha
    difference <- X[i+1]-X[i]
    i <- i+1
  }
  return(X[i])
}
```

MLE array is

```
##             [,1]       [,2]       [,3]
##  [1,] -0.5907336 -0.5914836 -0.5915732
##  [2,]  0.1035079 -0.5914824 -0.5915348
##  [3,] -1.1063091 -0.5914659 -0.5913732
##  [4,]  0.1035079  3.2398379  3.0213435
##  [5,] -1.1063091 -0.5914671  3.0213445
##  [6,] -1.1713919 -0.5914885  3.0213440
##  [7,] -1.1713919  2.5915066  3.0213452
##  [8,]  0.2417269 -0.5914885  3.0213442
##  [9,]  0.2417269  2.5916269  3.0213435
```

1(d)

Next, we use Fisher scoring to find the MLE for theta.

Fisher information is

```r
#Fisher information for x
I_theta <- length(x)/2
```

Code "nlminb" for Fisher scoring is:

```r
#nlminb for Fisher scoring
nlminb_1d <- function(x0){
  X <- array()
  X[1] <- x0
  i=1
  difference <- 1
  while(abs(difference)>= 0.0001){
    X[i+1] <- X[i]+gr_l(X[i])/I_theta
    difference <- X[i+1]-X[i]
    i <- i+1
  }
  return(X[i])
}
```

MLE array is

```
## [1] -0.5918082 -0.5917988 -0.5911707  3.0212553  3.0214278  3.0214653
## [7]  3.0214557  3.0214375  3.0214548
```

And then we run Newton-Raphson method to refine the estimate above and we get refined MLE:

```
## [1] -0.5918036 -0.5917943 -0.5911748  3.0212589  3.0214245  3.0214605
## [7]  3.0214514  3.0214338  3.0214505
```

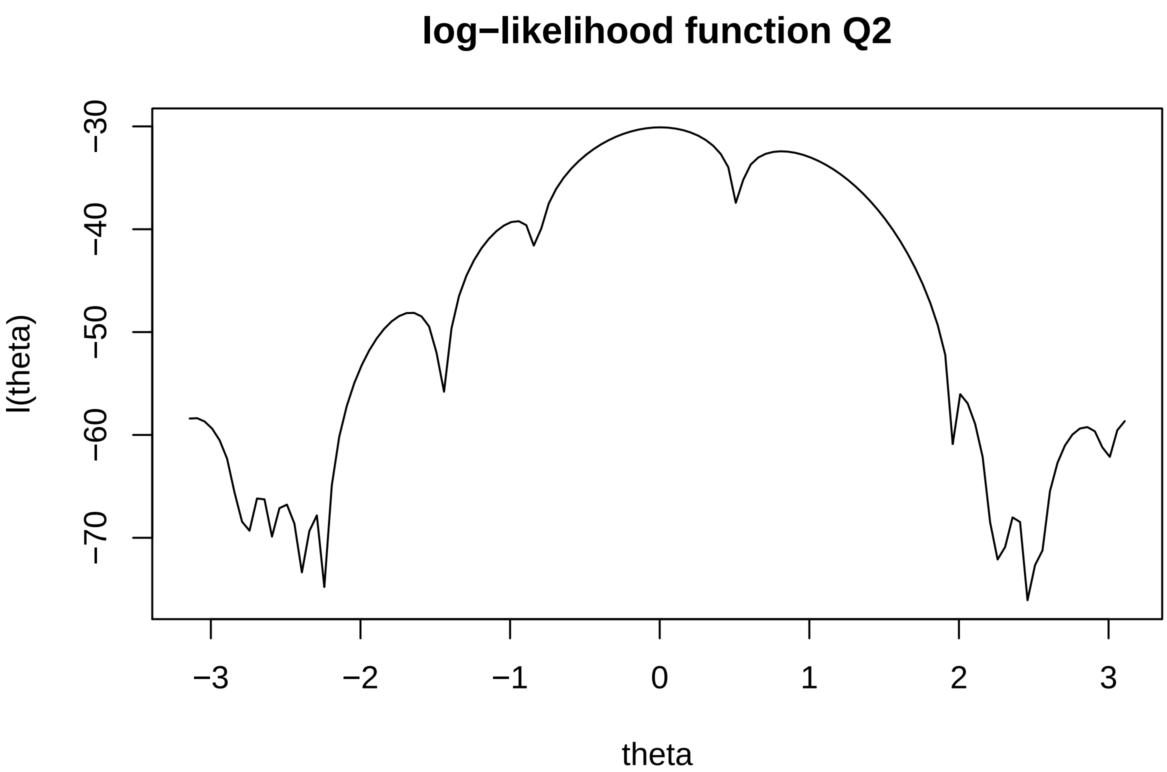
1(e)

The convergent speed of Newton-Raphson method is the fastest among the three, while the fixed-point method has more fluctuation than the other two functions.

Question 2

2(a)

The gragh of likelihood function



2(b)

expression for E[X|theta]: E[X|theta]=pi-sin(theta),which is equivalent to the expression below, so that theta_moment can be determined:

```
theta_mo1 <- asin(mean(x2)-pi)
theta_mo2 <- pi-asin(mean(x2)-pi)
```

And theta_moments are:

```
theta_mo1
```

```
## [1] 0.09539407
```

```
theta_mo2
```

```
## [1] 3.046199
```

2(c)

Find MLE using Newton-Raphson method with theta_moment:

Below is the code "nlminb" for Newton-Raphson method:

```
#nlminb for Newton-Raphson method
nlminb_2c <- function(x0){
  X <- array()
  X[1] <- x0
  i <- 1
  difference <- 1
  while(abs(difference)>= 0.0001){
```

```
    X[i+1] <- X[i]-gr_l2(X[i])/hess_l2(X[i])
    difference <- X[i+1]-X[i]
    i <- i+1
  }
  return(X[i])
}
```

Where gr_l2 and hess_l2 are gradient and Hessian for the log-likelihood function:

```
#gradient and Hessian for the log-likelihood function
gr_l2 <- function(t){
  grad_l <- sum(sin(x2-t)/(1-cos(x2-t)))
  return(grad_l)
}
hess_l2 <- function(t){
  he_l <- sum(1/(1-cos(x2-t)))
  return(he_l)
}
```

Find MLEs with theta_moments:

```
MLE_2c1 <- array()
MLE_2c1 <- nlminb_2c(theta_mo1)
MLE_2c1
```

```
## [1] 0.003118157
```

```
MLE_2c2 <- array()
MLE_2c2 <- nlminb_2c(theta_mo2)
MLE_2c2
```

```
## [1] 3.170715
```

2(d)

When we start at theta0 = ???2.7 and theta0 = 2.7, we find:

```
#find MLE theta0 = +/-2.7
MLE_2d1 <- array()
MLE_2d1 <- nlminb_2c(2.7)
MLE_2d1
```

```
## [1] 2.848415
```

```
MLE_2d2 <- array()
MLE_2d2 <- nlminb_2c(-2.7)
MLE_2d2
```

```
## [1] -2.668857
```

Question 3

3(a)

Fit the population growth model to the beetles data using the Gauss-Newton approach:

First, subsitute data for the model, where t denotes time, y denotes observerd value of the population:

```
beetles <- data.frame(
  days = c(0, 8, 28, 41, 63, 69, 97, 117, 135, 154),
  beetles = c(2, 47, 192, 256, 768, 896, 1120, 896, 1184, 1024))
t <- beetles$days
y <- beetles$beetles
N0 <- y[1]
```

Then, define functions for f(t), partial derivative of f(t) for K and r, and define functions for A and z in Gauss-Newton approach:

```
f_t <- function(K,r){
  f <- K*N0/(N0+(K-N0)*exp(-r*t))
  return(f)
}
f_deri_K <- function(K,r){
  f <- N0^2*(1-exp(-r*t))/(N0+(K-N0)*exp(-r*t))^2
  return(f)
}
f_deri_r <- function(K,r){
  f <- t*K*N0*(K-N0)*exp(-r*t)/(N0+(K-N0)*exp(-r*t))^2
  return(f)
}
A <- function(K,r){
  At <- array(dim=c(length(t),2))
  At[,1] <- t(f_deri_K(K,r))
  At[,2] <- t(f_deri_r(K,r))
  return(At)
}
z <- function(K,r){
  Z <- array(dim=c(length(t),1))
  Z <- y-f_t(K,r)
  return(Z)
}
```

Next, we can define "nls" function for Gauss-Newton method:

```
#nls for Gauss-Newton approach
nls_3a <- function(K0,r0){
  Kr <- array(dim=c(2,1000))
  Kr[,1] <- c(K0,r0)
  i <- 1
  difference <- 1
  while(difference >= 0.0001){
    Kr[,i+1] <- Kr[,i]+(solve((t(A(Kr[1,i],Kr[2,i]))%*%A(Kr[1,i],Kr[2,i])))%*%t(A(Kr[1,i],Kr[2,i]))%*%
    difference <- sum(abs(Kr[,i+1]-Kr[,i]))
    i <- i+1
  }
  return(Kr[,i])
}
```

Finally, we find model prediction by giving an initial pair of (K0,r0) = (1000,0.5):

```
## [1] 1049.4072354    0.1182684
```

So the prediction of K and r given by this model is 1049.4072354 and 0.1182684.