Homework 2

Yin Haoyu

08 February 2018

Exercise 1

(a)

From the known conditions, we have the probability density function of the samples as follow:

$$p(x;\theta) = \frac{1}{\pi[1+(x-\theta)^2]}$$

And as we know that the samples — $\{x_1, ..., x_n\}$ are i.i.d., we can deduce the formula of log-likelihood function of θ:

$$\begin{aligned} l(\theta) &= \ln(\prod_{i=1}^{n} p(x_i;\theta)) \\ &= \ln(\prod_{i=1}^{n} \frac{1}{\pi[1+(x_i-\theta)^2]}) \\ &= \sum_{i=1}^{n} \ln(\frac{1}{\pi[1+(x_i-\theta)^2]}) \\ &= -n\ln\pi - \sum_{i=1}^{n} \ln[1+(\theta-x_i)^2] \end{aligned}$$

Then we can get the first derivative of $l(\theta)$:

$$l'(\theta) = -2\sum_{i=1}^{n} \frac{\theta - x_i}{1+(\theta-x_i)^2}$$

Next we can get the second derivative of $l(\theta)$:

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

Last but not least, for the fisher scoring value $I(\theta)$:

$$I(\theta) = n\int \frac{\{p'(x)\}^2}{p(x)} dx$$

And for the first derivative of $p(x)$:

$$p'(x) = \frac{-2(x-\theta)}{\pi[1+(x-\theta)^2]^2}$$

Therefore, we get:

$$I(\theta) = \frac{4n}{\pi}\int_{-\infty}^{\infty}\frac{x^2dx}{(1+x^2)^3}$$

Using variable substitution: $x = \tan(t)$ where $t \in (-\frac{\pi}{2}, \frac{\pi}{2})$, thus we can update the formula of the fisher information of θ:

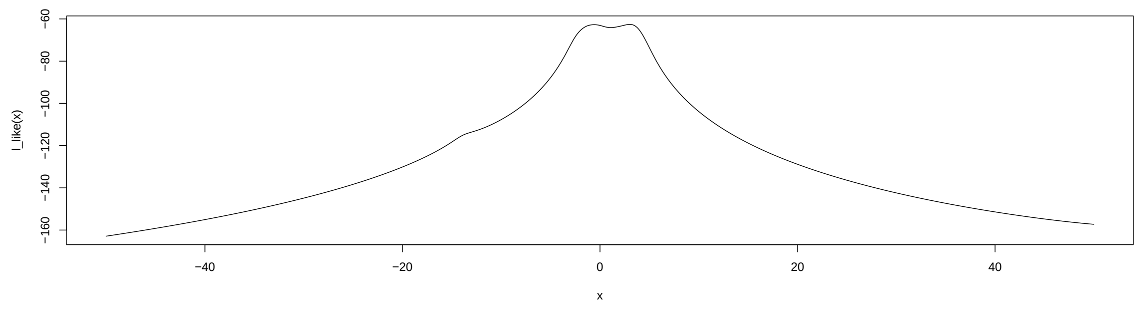
$$\begin{aligned} I(\theta) &= \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{\tan^2(t)d(\tan(t))}{(1+tan^2(t))^3} \\ &= \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{\tan^2(t)}{(\frac{1}{\sec^2(t)})^3}(\frac{1}{\sec^2(t)})dt \\ &= \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{\sin^2(t)}{\cos^2(t)}\cos^4(t)dt \\ &= \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\sin^2(t)\cos^2(t)dt \\ &= \frac{4n}{\pi} * \frac{\pi}{8} = \frac{n}{2} \end{aligned}$$

In conclusion, $I(\theta) = \frac{n}{2}$.

(b)

Firstly, we graph the log-likelihood function:

```r
#Exercise 1(b)
x <- c(1.77,-0.23,2.76,3.80,3.47,56.75,-1.34,4.24,-2.44,
       3.29,3.71,-2.40,4.53,-0.07,-1.05,-13.87,-2.53,-1.75)
#l(theta)
log_like <- function(theta,x. = x){
  sum(-log(pi)-log(1+(x. - theta)^2))
}
l_like <- Vectorize(log_like)
#graph the log-likelihood function, limit from -100 to 100
curve(l_like, from = -50, to = 50, n = 1000)
```



The results shown below are the numbers of iterations and the values of log-likelihood function:

```
count_1b
```

```
## [1] 31  4  5  5  5  5 32 33 29
```

```
lval_1b
```

```
## [1] -857.28520  -62.73186  -62.73186  -64.10186  -62.59032  -62.73186
## [7] -850.77506 -852.82452 -848.12593
```

In addition, if we choose the sample mean as another starting point:

```
count_1b2
```

```
## [1] 5
```

```
lval_1b2
```

```
## [1] -62.59032
```

To sum up, when we choose the sample mean as a new starting point, the number of Newton method iteration is 5 and the log-likelyhood function value is approximate to maximum, which meas a well-performed starting point.

(c)

```
counts_1c
```

```
##        1 0.64 0.25
## -11  292    1    1
## -1  1000   15    1
## 0   1000   14    1
## 1.5 1000   15    1
## 4   1000   14    1
## 4.7 1000   13    1
## 7   1000   13    1
## 8   1000   15    1
## 38  1000   15    1
```

```
thetas_1c
```

```
##              1       0.64       0.25
## -11 -0.5914735 -0.5914735 -0.5914735
## -1  -1.1713919 -0.5914735 -0.5914735
## 0    0.1035079 -0.5914735 -0.5914735
## 1.5 -1.1713919 -0.5914735 -0.5914735
## 4    0.1035079 -0.5914735 -0.5914735
## 4.7  0.2417269 -0.5914735 -0.5914735
## 7    0.2417269 -0.5914735 -0.5914735
## 8   -1.1063091 -0.5914735 -0.5914735
## 38  -1.1063091 -0.5914735 -0.5914735
```

(d)

```
counts_1d
```

```
## [1]  99  76  78  47  35  36  39  40 229
```

```
thetas_1d
```

```
## [1] -0.5914735 -0.5914735 -0.5914735  3.0213454  3.0213454  3.0213454
## [7]  3.0213454  3.0213454  3.0213454
```

(e)

In a nutshell, we can do a comparison among the three different methods that we have applied above with respect to counts and θ:

```
count_1b
```

```
## [1] 31  4  5  5  5  5 32 33 29
```

```
knitr::kable(counts_1c)
```

	1	0.64	0.25
-11	292	1	1
-1	1000	15	1
0	1000	14	1
1.5	1000	15	1
4	1000	14	1
4.7	1000	13	1
7	1000	13	1
8	1000	15	1
38	1000	15	1

```
counts_1d
```

```
## [1]  99  76  78  47  35  36  39  40 229
```

```
theta_1b
```

```
## [1] -1.240187e+10 -5.914735e-01 -5.914735e-01  1.092730e+00  3.021345e+00
## [6] -5.914735e-01  1.035024e+10  1.095657e+10  9.615946e+09
```

```
knitr::kable(thetas_1c)
```

	1	0.64	0.25
-11	-0.5914735	-0.5914735	-0.5914735
-1	-1.1713919	-0.5914735	-0.5914735
0	0.1035079	-0.5914735	-0.5914735
1.5	-1.1713919	-0.5914735	-0.5914735

	1	0.64	0.25
4	0.1035079	-0.5914735	-0.5914735
4.7	0.2417269	-0.5914735	-0.5914735
7	0.2417269	-0.5914735	-0.5914735
8	-1.1063091	-0.5914735	-0.5914735
38	-1.1063091	-0.5914735	-0.5914735

```
thetas_1d
```

```
## [1] -0.5914735 -0.5914735 -0.5914735  3.0213454  3.0213454  3.0213454
## [7]  3.0213454  3.0213454  3.0213454
```

As it is shown in the tables above, we notice that Newton-Raphson method converges faster towards the maximum value of the log-likelihood than other methods in most of starting values, the only one exception is when comparing with fixed-point method in the condition that alpha equals to 0.25, the latter method has a better performance of approaching. In addition, there is one phenomenon which we cannot ignore is that the iteration numbers of Newton-Raphson method will be much bigger and the final estimated thetas diverge to infinite as we may use some other different starting points to do the iteration, which means the non-ideal stability of that method.

Then for the fixed-point iteration, we can find that its approaching performance depends strongly on what value of α we have selected. For instance, when α equals to 1, the number of iteration times is more than ten thousand.

Finally, as for Fisher Scoring, we find that the speed of convergency is faster than the other two methods in the first a few steps of iteration, however, it will use more times to reach the MLE than the other two methods, so it is often used in the beginning to make rapid improvements, and Newton's method for refinement near the end.

Exercise 2

(a)

According to the given condition, the probability density with respect to the parameter θ is:

$$p(x;\theta) = \frac{1-\cos(x-\theta)}{2\pi}$$

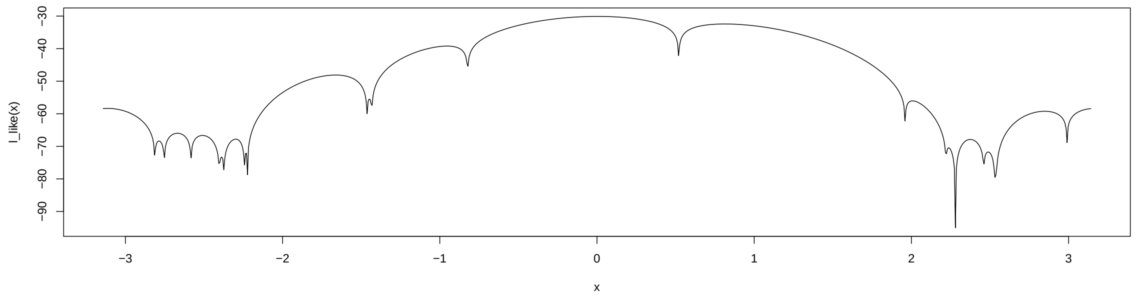
Thus, we can get the log-likelihood function of θ:

$$l(\theta) = -n\ln 2\pi + \sum_{i=1}^{n} \ln[1-\cos(x_i-\theta)]$$

Therefore, we can draw the graph of that function between $-\pi$ and π:

```r
rm(list=ls())
x <- c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96,
       2.53, 3.88, 2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52)
```

```r
#l(theta)
log_like <- function(theta,x. = x){
  sum(-log(2*pi)+log(1-cos(x. - theta)))
}
l_like <- Vectorize(log_like)
#graph the log-likelihood function, limit from -pi to pi
curve(l_like, from = -pi, to = pi, n = 1000)
```



(b)

According to the definition of the expectation of a random variable x, we can compute it regarding θ as a given constant in this situation:

$$\begin{aligned} E[X|\theta] &= \int_0^{2\pi} x * \frac{1-\cos(x-\theta)}{2\pi} dx \\ &= \int_0^{2\pi} \frac{x}{2\pi} dx - \int_0^{2\pi} \frac{x * \cos(x-\theta)}{2\pi} dx \\ &= \frac{1}{2\pi} * \frac{x^2}{2}|_0^{2\pi} - \int_0^{2\pi} \frac{x}{2\pi} d[\sin(x-\theta)] \\ &= \pi - \frac{x}{2\pi} * \sin(x-\theta)|_0^{2\pi} - \int_0^{2\pi} \sin(x-\theta) d(\frac{x}{2\pi}) \\ &= \pi - \sin(2\pi-\theta) - \frac{1}{2\pi}\int_{-\theta}^{2\pi-\theta} \sin u du \\ &= \pi + \sin\theta + \frac{1}{2\pi}\cos u|_{-\theta}^{2\pi-\theta} \\ &= \pi + \sin\theta \end{aligned}$$

Therefore, we get the so-called "Method-of-moments" estimator of θ:

```r
#Exercise 2(b)
estimator <-function(theta, x_mean){
  pi+sin(theta)-x_mean
}
print(root1<-uniroot(estimator,c(0,pi/2),x_mean=mean(x))$root)
```

```
## [1] 0.09539388
```

```r
print(root2<-uniroot(estimator,c(pi/2,pi),x_mean=mean(x))$root)
```

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

(c) & (d)

As the question (c) and the question (d) are just different with starting points, we can put all of the required starting points together:

```r
#starting points
sta_p <- c(root1, root2, -2.7, 2.7)
```

Through those iterations above, we finally get:

```r
thetas_2cd
```

```
##               0.1     3.05      -2.7      2.7
## [1,] 0.003118157 3.170715 -2.668857 2.848415
```

```r
counts_2cd
```

```
##      0.1 3.05 -2.7 2.7
## [1,]   4    6    5   6
```

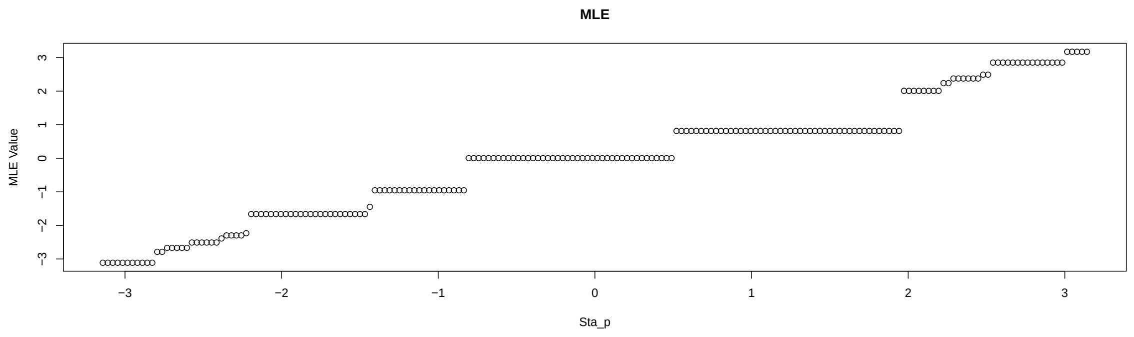
```r
lvals_2cd
```

```
##             0.1      3.05      -2.7       2.7
## [1,] -30.09493 -58.34181 -65.98183 -59.22775
```

(e)

Then we can do the MLE and show the results in the following figure:

```r
plot(sta_p, thetas_2e, main='MLE', ylab='MLE Value', xlab='Sta_p')
```



According to R's manual, We find that the "unique" function is used to eliminate the repetitive values of starting points which have the same estimator of θ, in that case we can do the desired partition as follow:

```r
thetas_2e_p <- as.data.frame(round(t(thetas_2e), digits = 4))
colnames(thetas_2e_p) <- c("MLE")
```

```r
unq <- unique(thetas_2e_p)
knitr::kable(unq)
```

	MLE
-3.14	-3.1125
-2.79	-2.7866
-2.73	-2.6689
-2.57	-2.5094
-2.38	-2.3883
-2.35	-2.2979
-2.23	-2.2322
-2.19	-1.6627
-1.44	-1.4475
-1.41	-0.9544
-0.81	0.0031
0.52	0.8126
1.97	2.0072
2.23	2.2370
2.29	2.3747
2.48	2.4884
2.54	2.8484
3.02	3.1707

Exercise 3

(a)

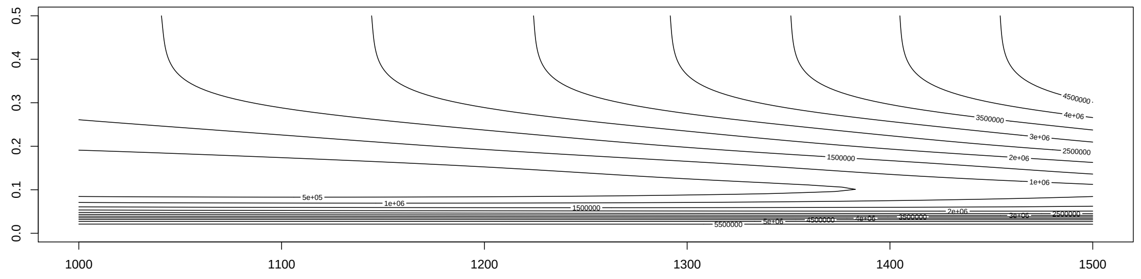
```r
rm(list=ls())
beetles <- data.frame(
  days = c(0, 8, 28, 41, 63, 69, 97, 117, 135, 154),
  beetles = c(2, 47, 192, 256, 768, 896, 1120, 896, 1184, 1024))

 bias <- rep(NA, nrow(beetles))

nls(beetles ~ (K*beetles[1])/(beetles[1]+(K-beetles[1])*exp(-r*days)),
    start = list(K = 500, r = 0.5), data = beetles, trace = TRUE)
```

```
## 1668466 :  500.0   0.5
## 980314.2 :  792.1048944   0.4065344
## 540042.5 :  795.2648096   0.1948644
## 117871.3 :  945.016260   0.128334
## 74127.87 :  1036.6071401    0.1196954
## 73430.9 :  1048.3342674    0.1185133
## 73420.01 :  1049.2500441    0.1183099
## 73419.71 :  1049.3812074    0.1182754
## 73419.7 :  1049.4028582    0.1182696
## 73419.7 :  1049.4065029    0.1182686
```

```
## Nonlinear regression model
##   model: beetles ~ (K * beetles[1])/(beetles[1] + (K - beetles[1]) * exp(-r *     days))
##    data: beetles
##         K         r
## 1049.4065    0.1183
##  residual sum-of-squares: 73420
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 9.083e-06
```

According to the lecture notes, we can use "nls" function which built-in Guass-Newton method as the default approach to do the required iteration. And through that function, we fit the population growth model with: K = 1049.4065, r = 0.1183 and the residual sum-of-sqaures = 73420.

(b)



(c)

In order to apply Newton method in high-dimensional situation, we should first compute the partial derivatives with respect to the parameters K, r and sigma.

Next, taking the results of question (a) in this exercise into consideration, we can set K = 1050, r = 0.12 and sigma = 0.6 as the initial values:

```r
krsig <- matrix(c(1050, 0.12, 0.6))
row.names(krsig) <- c("K", "r", "sigma")
knitr::kable(krsig)
```

K	1050.00
r	0.12
sigma	0.60

```
## [1] 8
```

K	r	sigma_2
820.38	0.1926401	0.4148444

After 8 times iteration, we get the MLE values of K, r and σ^2. Based on the lecture notes and previous knowledge, we can get the variance of the estimators that we have just estimated above from the information matrix of those parameters, which can be obtained by computing the minus inverse of fisher information matrix. That is, each element in the diagonal of the information matrix means the variance of the related parameter that we have estimated. Therefore, we can easily see the desired variance through the matrix as follow:

```r
var <- solve(-M)
colnames(var) <- row.names(var) <- c("K", "r", "sigma")
knitr::kable(var)
```

	K	r	sigma
K	62464.581865	-9.0598068	0.0000000
r	-9.059807	0.0039784	0.0000000
sigma	0.000000	0.0000000	0.0207422