Homework 4

Xiaokang Liu

28 September 2018

Contents

1 Many local maxima

Consider the probability density function with parameter θ:

$$f(x;\theta) = \frac{1 - cos(x-\theta)}{2\pi},\ 0 \le x \le 2\pi,\ \theta \in (-\pi, \pi).$$

A random sample from the distribution is

```
samp <- 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)
```

1.1 Part 1

The log-likelihood function of θ based on the sample is

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

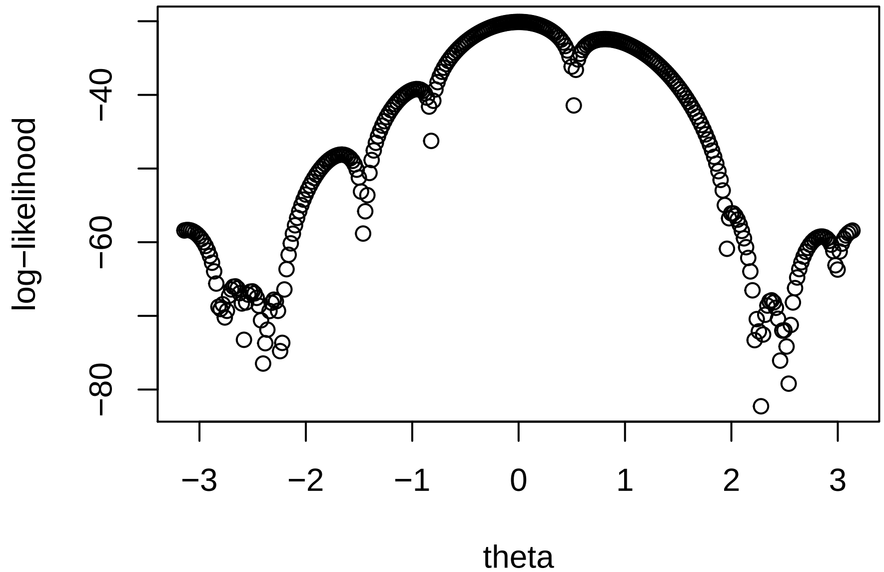
And the plot of it between $-\pi$ and π is:

```
logf <- function(dat,para){
  f <- sum(log(1-cos(dat-para))-log(2*pi))
  return(f)
}

int <- seq(-pi,pi,0.02)
```

```
val <- vector()
for (i in 1:length(int)){
  val <- c(val,logf(samp,int[i]))
}

plot(int,val,xlab = "theta",ylab = "log-likelihood")
```



1.2 Part 2

The expectation of X can be derived by

$$\begin{aligned} E(x|\theta) &= \int_0^{2\pi} x\frac{1 - cos(x-\theta)}{2\pi}dx \\ &= \pi - \frac{1}{2\pi}\int_{-\theta}^{2\pi-\theta} xcos(x)dx \\ &= \pi + sin(\theta). \end{aligned}$$

Thus, the moment estimator of θ is $\tilde{\theta}_n = \arcsin(\bar{X}_n - \pi)$. Since we have defined an interval for θ, thus the only possible moment estimator from the sample is 0.095394.

1.3 Part 3

```
init <- asin(mean(samp)-pi) # it's okay to +/- 2*k*pi
l1 <- function(para,dat){
  f <- -sum(sin(samp-para)/(1-cos(samp-para)))
  return(f)
}
l2 <- function(para,dat){
  f <- sum((cos(samp-para)-1)/(1-cos(samp-para))^2)
```

```r
}

NR <- function(ini,x,tol,max_ite){
  err <- 100
  iter <- 0
  conver <- 0
  while ((err > tol) & (iter < max_ite)) {
    ini1 <- ini-l1(ini,x)/l2(ini,x)
    err <- abs(ini1-ini)
    ini <- ini1
    iter <- iter+1
  }
  if (iter >= max_ite) conver <- 1
  return(list(ini1=ini1,iter=iter,err=err,conver=conver))
}

NR(init,samp,tol = .Machine$double.eps^0.5,max_ite = 200)
```

```
## $ini1
## [1] 0.003118157
##
## $iter
## [1] 4
##
## $err
## [1] 1.525437e-10
##
## $conver
## [1] 0
```

1.4 Part 4

```r
NR(-2.7,samp,tol = .Machine$double.eps^0.5,max_ite = 200)
```

```
## $ini1
## [1] -2.668857
##
## $iter
## [1] 4
##
## $err
## [1] 8.772369e-09
##
## $conver
## [1] 0
```

```r
NR(2.7,samp,tol = .Machine$double.eps^0.5,max_ite = 200)
```

```
## $ini1
## [1] 2.848415
##
## $iter
## [1] 5
##
## $err
## [1] 1.682339e-10
##
## $conver
## [1] 0
```

1.5 Part 5

```r
init <- seq(-pi, pi, length.out = 200)
n <- length(init)
eps0 <- .Machine$double.eps^0.5
max_ite0 <- 200

val_NR <- rep(0,n)
conv_NR <- rep(0,n)
iter_NR <- rep(0,n)


for (i in 1:n){
  res <- NR(init[i],samp,eps0,max_ite0)
  val_NR[i] <- res$ini1
  iter_NR[i] <- res$iter
  conv_NR[i] <- res$conver
}

val_NR1 <- round(val_NR,9)
uni <- unique(val_NR1)
gr <- matrix(nrow = 18, ncol = 200)
for (j in 1:18){
  gr[j,] <- init*(val_NR1==uni[j])
}
# group 1 to group 18
table(val_NR1)
```

```
## val_NR1
## -3.112470507 -2.786556852 -2.668857459 -2.509356033 -2.388266628
##           11            2            5            6            1
## -2.297925969 -2.232191899 -1.662712395 -1.447502553 -0.954405837
```

```
##           4           1          24           1          19
##  0.003118157  0.812637417  2.007223238  2.237012923  2.374711666
##          42          46           8           2           6
##  2.488449651  2.848415325   3.1707148
##           2          15           5
```

```
for (i in 1:18){
  x <- gr[i,]
  z <- x[ min( which ( x != 0 )) : max( which( x != 0 )) ]
  cat("At group", i, ",values are:",z,"\n")
}
```

```
## At group 1 ,values are: -3.141593 -3.110019 -3.078445 -3.046871 -3.015297 -2.983724 -2.95215
## At group 2 ,values are: -2.794281 -2.762707
## At group 3 ,values are: -2.731133 -2.69956 -2.667986 -2.636412 -2.604838
## At group 4 ,values are: -2.573264 -2.541691 -2.510117 -2.478543 -2.446969 -2.415395
## At group 5 ,values are: -2.383822
## At group 6 ,values are: -2.352248 -2.320674 -2.2891 -2.257526
## At group 7 ,values are: -2.225953
## At group 8 ,values are: -2.194379 -2.162805 -2.131231 -2.099657 -2.068084 -2.03651 -2.00493
## At group 9 ,values are: -1.436608
## At group 10 ,values are: -1.405034 -1.37346 -1.341886 -1.310313 -1.278739 -1.247165 -1.2155
## At group 11 ,values are: -0.8051318 -0.773558 -0.7419842 -0.7104104 -0.6788366 -0.6472628 -0
## At group 12 ,values are: 0.5209676 0.5525414 0.5841152 0.615689 0.6472628 0.6788366 0.71041
## At group 13 ,values are: 1.973362 2.004936 2.03651 2.068084 2.099657 2.131231 2.162805 2.19
## At group 14 ,values are: 2.225953 2.257526
## At group 15 ,values are: 2.2891 2.320674 2.352248 2.383822 2.415395 2.446969
## At group 16 ,values are: 2.478543 2.510117
## At group 17 ,values are: 2.541691 2.573264 2.604838 2.636412 2.667986 2.69956 2.731133 2.76
## At group 18 ,values are: 3.015297 3.046871 3.078445 3.110019 3.141593
```

2 Modeling beetle data

The counts of a floor beetle at various time points (in days) are given in a dataset as below:

```
beetles1 <- 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))
```

2.1 Fit the population growth model

```
lk <- function(t,a,b){
  f <- (4-4*exp(-b*t))/(2+(a-2)*exp(-b*t))^2
  return(f)
}
```

```r
lr <- function(t,a,b){
  f <- (2*a*(a-2)*t*exp(-b*t))/(2+(a-2)*exp(-b*t))^2
  return(f)
}

GN <- function(t,y,ini,tol,max_ite){
  err <- 100000000
  iter <- 0
  conver <- 0
  while ((err > tol)&(iter < max_ite)) {
     f1 <- lk(t,ini[1],ini[2])
     f2 <- lr(t,ini[1],ini[2])
     A <- cbind(f1,f2)
     ini1 <- ini+solve(t(A)%*%A+0.00001*diag(nrow = 2))%*%t(A)%*%(y-2*ini[1]/(2+(ini[1]-2)*exp
     err <- sum((ini-ini1)^2)
     ini <- ini1
     iter <- iter+1
  }
  ss <- sum((y-2*ini[1]/(2+(ini[1]-2)*exp(-ini[2]*t)))^2)
  if (iter >= max_ite) conver <- 1
  return(list(est=ini,ss=ss,iter=iter,err=err,conver=conver))
}
GN(beetles1$days,beetles1$beetles,c(1100,0.1),.Machine$double.eps^0.5,200)
```

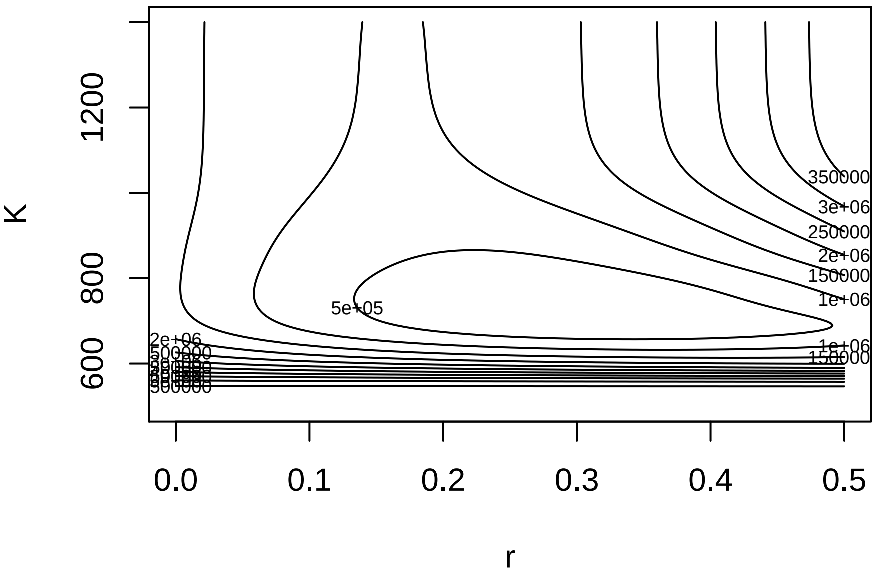
```
## $est
##              [,1]
## f1 1049.4072555
## f2    0.1182684
##
## $ss
## [1] 73419.7
##
## $iter
## [1] 9
##
## $err
## [1] 3.052186e-09
##
## $conver
## [1] 0
```

2.2 Show the contour plot

Show the contour plot of the sum of squared errors.

```r
kseq <- seq(500, 1400, length.out = 200)
rseq <- seq(0, 0.5, length.out = 200)
```

```r
cont <- matrix(nrow = length(kseq), ncol = length(rseq))
y <- beetles1$beetles
t <- beetles1$days
for (i in 1:length(kseq)){
  for (j in 1:length(rseq)){
    cont[i,j] <- sum((y-2*kseq[i]/(2+(kseq[i]-2)*exp(-rseq[j]*t)))^2)
  }
}
# contour plot
contour(rseq,kseq,cont,xlab="r",ylab="K",method = "simple")
```



2.3 Log-Normality

If we assume $\log N_t$ are independent and normally distributed with mean $logf(t)$ and variance σ^2. The the log-likelihood function is

$$l(x, r, K, \sigma^2) = -\frac{n}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\log(x_i) - \log\frac{2k}{2+(k-2)e^{-rt_i}})^2 + C.$$

where C is a constant. Here we use BFGS with linear constraints to solve this problem. Since the commonly used optimization functions in r will do minimization in default, here we use the negative log-likelihood function as the objective function. We provide the function for the objective function and its first order derivative function.

```r
y <- beetles1$beetles
t <- beetles1$days
n <- length(y)
fr <- function(x){
  x1 <- x[1] # sigma^2
  x2 <- x[2] # k
  x3 <- x[3] # r
```

```r
  log(x1)*n/2+sum((log(y)-log(2*x2/(2+(x2-2)*exp(-x3*t))))^2)/2/x1
}
grr <- function(x){
  x1 <- x[1] # sigma^2
  x2 <- x[2] # k
  x3 <- x[3] # r
  c(n/2/x1-sum((log(y)-log(2*x2/(2+(x2-2)*exp(-x3*t))))^2)/2/x1^2,
    sum((log(y)-log(2*x2/(2+(x2-2)*exp(-x3*t))))*(2*exp(-x3*t)-2)/(2+(x2-
                                                2)*exp(-x3*t)))/x2/x1,
    sum((log(y)-log(2*x2/(2+(x2-2)*exp(-x3*t))))*((2-x2)*t*exp(-x3*t))/(2+
                                                (x2-2)*exp(-x3*t)))/x1
  )
}

# add linear constraints on parameters, they are all positive
mod1 <- constrOptim(c(0.01,2200,0.1), fr, grr, ui=diag(nrow = 3),
                    ci=c(0,0,0), hessian = TRUE)
mod1
```

```
## $par
## [1]    0.8559366 2151.1598581    0.1431047
##
## $value
## [1] 4.221507
##
## $counts
## function gradient
##      152      101
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##              [,1]          [,2]         [,3]
## [1,]  6.822987683 -2.915587e-03   -0.0111829
## [2,] -0.002915587  2.755498e-07    0.0175775
## [3,] -0.011182899  1.757750e-02 1186.2494911
##
## $outer.iterations
## [1] 3
##
## $barrier.value
## [1] -1.435492
```

```r
var.est <- diag(solve(-mod1$hessian))
var.est
```

```
## [1]  1.797321e-03  8.125787e+05 -6.646766e-04
```

From the result, we know that the eistimated variances from Hessian matrix are 1.797×10^{-3}, 8.126×10^{5} and 0. Since the last one is negative, thus we set it to be 0.