Nayun Li HW2 MLE problem

Nayun Li

2018/2/5

Abstract

This is the Homework 2

Question 1

(a) Show the fomula as follows

The Cauchy (x,theta) has probability density:

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

Let $x_1, x_2 ... x_n$be an i.i.d smple, and $l(\theta)$ the log-likelihood function is:

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

$$l^{'}(\theta) = 0-(\sum_{i=1}^{n}\ln(1+(x_i-\theta)^2))^{'} = -\sum_{i=1}^{n}\frac{1}{1+(x_i-\theta)^2} * (1+x_i^2-2x_i\theta+\theta^2)^{'} = -\sum_{i=1}^{n}\frac{2(\theta-x_i)}{1+(\theta-x_i)^2} = -2\sum_{i=1}^{n}\frac{\theta-x_i}{1+(\theta-x_i)^2}$$

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

$$P(x) = \frac{1}{\pi(1+x^2)}$$

We have:

$$P^{'}(x) = -\frac{2x}{\pi(1+x^2)^2}$$

the fisher scoring function is equal to:

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

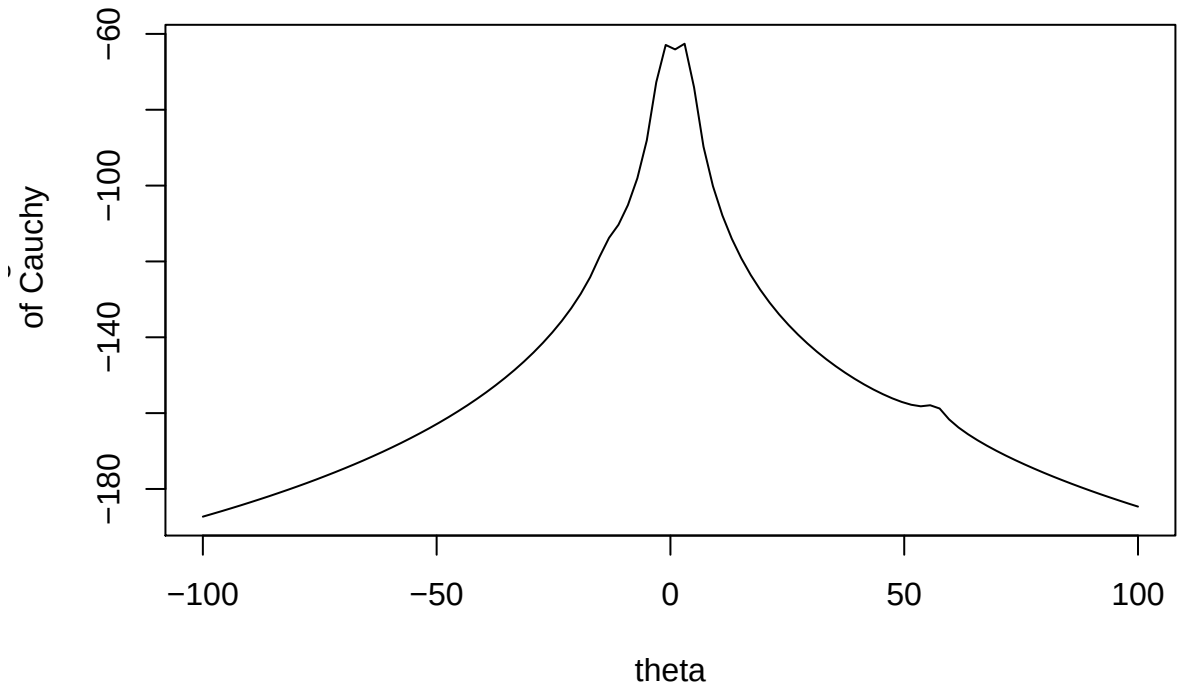
Set $x = \tan(\alpha); \alpha \in (-\frac{\pi}{2}, \frac{\pi}{2})$ So,

$$I(\theta) = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{\cos^{-2}(\alpha)-1}{(cos^{-2}(\alpha))^3} = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{\tan^2(\alpha)}{(1+tan^2(\alpha))^3}\frac{1}{\cos^2(\alpha)}d\alpha = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} sin^2(\alpha) * cos^2(\alpha)d\alpha = \frac{4n}{\pi} * \frac{\pi}{8} = \frac{n}{2}$$

(b) Graph the log-likelihood function. Find the MLE for θ using the Newton-Raphsonmethod.

Graph the log-likelihood function.

```
log_like <- function(theta, x)
  sum(-log(pi) - log(1 + (x - theta) ^ 2))
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)
theta <- seq(from = -100, to = 100, , length.out = 100)
plot(theta, sapply(theta, log_like, x), type = "l", ylab = "LogL
     of Cauchy", xlim = c(-100, 100))
```



Find the MLE for θ using the Newton-Raphson method

```
# b
inits <- c(-11, -1, 0, 1.5, 4, 4.7, 7, 8, 38)
newton <- function(inits){
  g <- function(theta){
    sum(log(pi) + log(1 + (theta - x) ^ 2))
  }
  gr_g <- function(theta){
    (sum(2 * (theta - x) / (1 + (theta - x) ^ 2)))
  }
  hess_g <- function(theta){
    y <- sum(2 * (theta - x) / (1 + (theta - x) ^ 2)) /
             (sum(2 * (1 - (theta - x) ^ 2) / ((1 + (theta - x) ^ 2) ^ 2)))
    return(matrix(y, nrow = 1))
```

```
  }
  z <- nlminb(inits, g, gr_g, hess_g)
}
#print results
results_1b <- matrix(0, nrow = 1, ncol = 9, byrow = FALSE, dimnames = list('newton',
                     c('-11', '-1', '0', '1.5', '4', '4.7', '7', '8', '38')))
for (i in (1 : length(inits))){
  results_1b[1, i] <- as.numeric(newton(-11)[1])
}
knitr::kable(head(results_1b), booktabs = TRUE, longtable = T,
             caption = '(results_1b)')
```

Table 1: (results_1b)

	-11	-1	0	1.5	4	4.7	7	8	38
newton	-0.591414	-0.591414	-0.591414	-0.591414	-0.591414	-0.591414	-0.591414	-0.591414	-0.591414

(c) Apply fixed-point iterations

```
alphas <- c(1, 0.64, 0.25)
inits <- c(-11, -1, 0, 1.5, 4, 4.7, 7, 8, 38)
gr_g <- function(theta){
  -2 * (sum((theta - x) / (1 + (theta - x) ^ 2)))
}
fix_point <- function(init, alpha, eps = 1e-06, itmax = 100){
  count <- 0
  flag <- TRUE
  theta <- init
  while(flag){
    g <- gr_g(theta)
    count <- count + 1
    if(abs(g) < eps | count == itmax){
      flag <- FALSE
      return(theta)
    }
    else{
      theta <- alpha * (gr_g(theta)) + theta
    }
  }
}
results_1c <- matrix(0, nrow = 3, ncol = 9, byrow = F,
                     dimnames = list(c('1', '0.64', '0.25'),
                   c('-11', '-1', '0', '1.5', '4', '4.7', '7', '8', '38')))
for (i in (1 : length(alphas))){
  for (j in (1 : length(inits))){
    results_1c[i, j] <- as.numeric(fix_point(inits[j], alphas[i],
                           eps = 1e-06, itmax = 100))
  }
}
knitr::kable(head(results_1c), booktabs = TRUE,
             caption = '(results_1c)')
```

Table 2: (results_1c)

	-11	-1	0	1.5	4	4.7	7	8	38
1	-0.5907336	0.1035079	-1.1063091	0.1035079	-1.1063091	-1.1713919	-1.171392	0.2417269	0.2417269
0.64	-0.5914734	-0.5914734	-0.5914731	3.2398379	-0.5914731	-0.5914733	2.591507	-0.5914733	2.5916269
0.25	-0.5914740	-0.5914738	-0.5914730	3.0213454	3.0213454	3.0213454	3.021345	3.0213454	3.0213454

(d) First use Fisher scoring to find the MLE for θ, then refine the estimate by running Newton-Raphson method

```
n <- length(inits)
newton_fc <- function(inits){
  g <- function(theta){
    sum(log(pi) + log(1 + (theta - x) ^ 2))
  }
  gr_g <- function(theta){
    (sum(2 * (theta - x) / (1 + (theta - x) ^ 2)))
  }
  hess_g <- function(theta){
    y <- sum(2 * (theta - x) / (1 + (theta - x) ^ 2)) /
      (n / 2)
    return(matrix(y, nrow = 1))
  }
  z <- nlminb(inits, g, gr_g, hess_g)
}
#print results
results_1d <- matrix(0, nrow = 2, ncol = 9, byrow = FALSE,
                     dimnames = list(c('fish', 'refine'),
    c('-11', '-1', '0', '1.5', '4', '4.7', '7', '8', '38')))
for (i in (1 : length(inits))){
  results_1d[1, i] <- as.numeric(newton_fc(inits[i])[1])
}

fc_refine <- array()
for (i in (1:length(inits))){
  fc_refine[i] <- newton(results_1d[i])[1]
  results_1d[2, i] <- as.numeric(fc_refine[i])
}
knitr::kable(head(results_1d), booktabs = TRUE,
             caption = '(results_1d)')
```

Table 3: (results_1d)

	-11	-1	0	1.5	4	4.7	7	8	38
fish	-0.5914722	-0.5914735	-0.5914724	3.0213454	3.0213454	3.0213454	3.021345	-0.5914736	3.021345
refine	-0.5914722	-0.5914722	-0.5914735	-0.5914722	-0.5914735	-0.5914735	3.021345	-0.5914722	3.021345

(e) Comment on the results from different methods

```
final_results <- rbind(results_1b, results_1c, results_1d)
knitr::kable(head(final_results), booktabs = TRUE,
             caption = '(final_results)')
```

Table 4: (final_results)

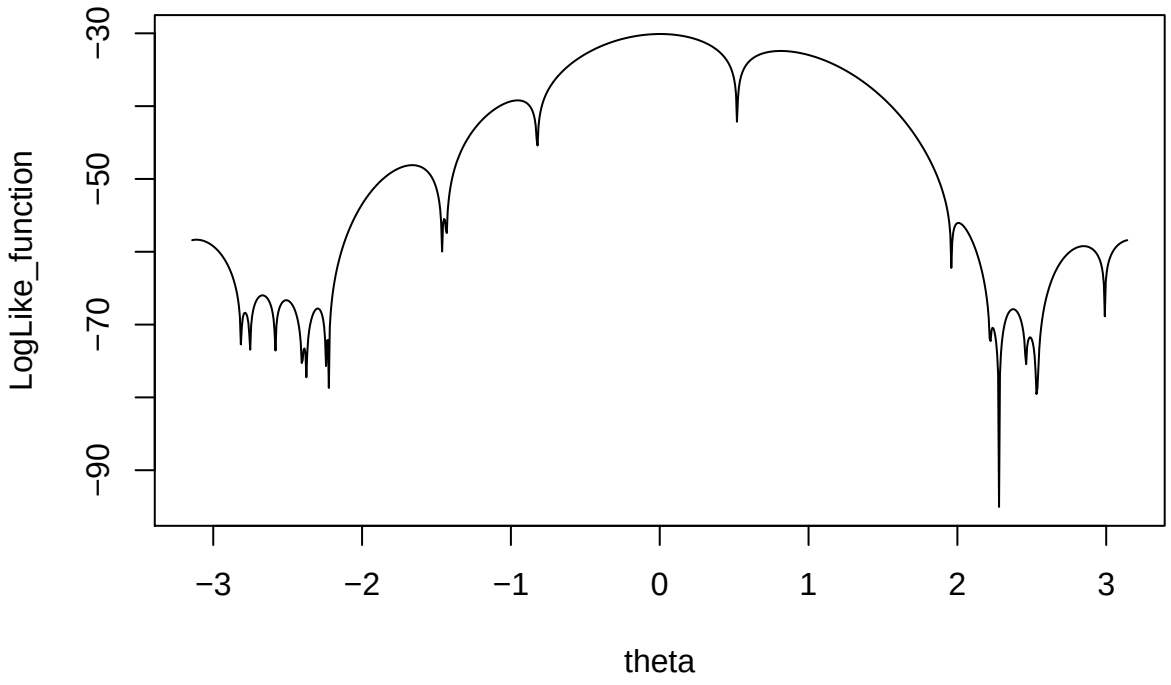
	-11	-1	0	1.5	4	4.7	7	8	3
newton	-0.5914140	-0.5914140	-0.5914140	-0.5914140	-0.5914140	-0.5914140	-0.591414	-0.5914140	-0.591414
1	-0.5907336	0.1035079	-1.1063091	0.1035079	-1.1063091	-1.1713919	-1.171392	0.2417269	0.241726
0.64	-0.5914734	-0.5914734	-0.5914731	3.2398379	-0.5914731	-0.5914733	2.591507	-0.5914733	2.591626
0.25	-0.5914740	-0.5914738	-0.5914730	3.0213454	3.0213454	3.0213454	3.021345	3.0213454	3.021345
fish	-0.5914722	-0.5914735	-0.5914724	3.0213454	3.0213454	3.0213454	3.021345	-0.5914736	3.021345
refine	-0.5914722	-0.5914722	-0.5914735	-0.5914722	-0.5914735	-0.5914735	3.021345	-0.5914722	3.021345

According to the results above, the speed of convergence of Newton method is the fastest mothods. The speed of Fisher scoring method is slow. However, the stability of newton method is not very good. In fixed-point iterations, it depends strongly on the value of alpha. Smaller alpha is better.

Question 2

(a) Graph the function

```
log_like <- function(theta, x)
  sum(log((1 - cos(x - theta)) / (2 * pi)))
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)
theta <- seq(from = -pi, to = pi, , length.out = 1000)
plot(theta, sapply(theta, log_like, x), type = "l", ylab = "LogLike_function")
```



(b) Find the method-of-moments estimator of θ

```
moment <- function(theta, y)
  pi + sin(theta) - y

root1 <- uniroot(moment, c(0, pi / 2), y = mean(x)) $ root
root2 <- uniroot(moment, c(pi / 2, pi), y = mean(x)) $ root
print(root1)
```

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

```
print(root2)
```

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

(c) Find the MLE for ?? using the Newton-Raphson method

```
newton_theta_moment <- function(inti){
  g <- function(theta){
    n <- length(x)
    n * log(2 * pi) - sum(log(1 - cos(x - theta)))
  }
  gr_g <- function(theta){

    sum((sin(x - theta)) / (1 - cos(x - theta)))
  }
  hess_g <- function(theta){
    y <- -sum(1 / (cos(x - theta) - 1))
    return(matrix(y, nrow = 1))
  }
```

```
  z <- nlminb(inti, g, gr_g, hess_g)
}
#print results
results_2c <- matrix(0, nrow = 1, ncol = 2, byrow = FALSE, dimnames = list('theta',
                     c('root_1', 'root_2')))
for (i in (1 : length(inits <- c(root1, root2)))){
  results_2c[1, i] <-  as.numeric(newton_theta_moment(inits[i])[1])
}
knitr::kable(head(results_2c), booktabs = TRUE, longtable = T,
             caption = '(results_2c)')
```

Table 5: (results_2c)

	root_1	root_2
theta	0.0031182	3.170715

(d) start at $\theta_0 = ???2.7$ and $\theta_0 = 2.7$

```
newton <- function(inti){
  g <- function(theta){
    n <- length(x)
    n * log(2 * pi) - sum(log(1 - cos(x - theta)))
  }
  gr_g <- function(theta){

    sum((sin(x - theta)) / (1 - cos(x - theta)))
  }
  hess_g <- function(theta){
    y <- -sum(1 / (cos(x - theta) - 1))
    return(matrix(y, nrow = 1))
  }
  z <- nlminb(inti, g, gr_g, hess_g)
}
#print results
#print results
results_2d <- matrix(0, nrow = 1, ncol = 2, byrow = FALSE, dimnames = list('theta',
                     c('-2.7', '2.7')))
for (i in (1 : length(inits <- c(-2.7, 2.7)))){
  results_2d[1, i] <-  as.numeric(newton_theta_moment(inits[i])[1])
}
knitr::kable(head(results_2d), booktabs = TRUE, longtable = T,
             caption = '(results_2d)')
```

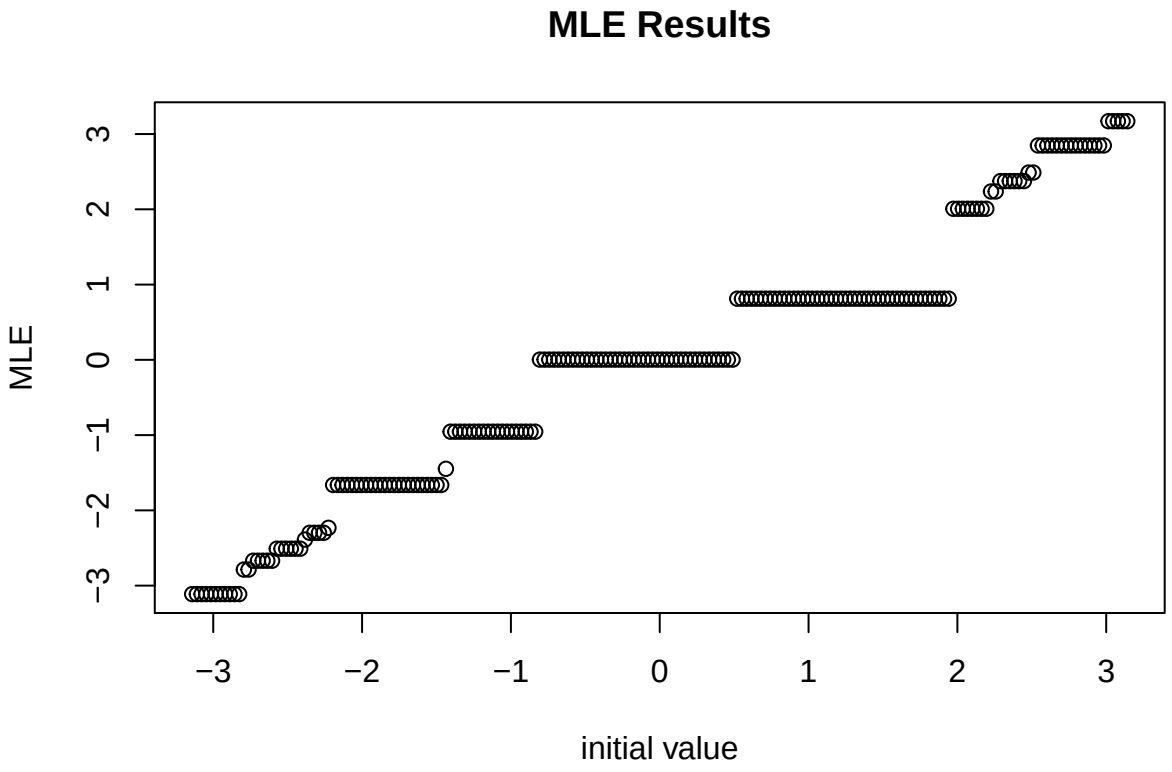
Table 6: (results_2d)

	-2.7	2.7
theta	-2.668858	2.848415

(e) Repeat the above using 200 equally spaced starting values between $-\pi$ and π.

```
init_value <- seq(-pi,pi,length.out = 200)
rep_results <- array()
for (i in 1:length(init_value)){
  rep_results[i] <- newton(init_value[i])
}

plot(init_value, rep_results, main='MLE Results',
     ylab='MLE', xlab='initial value')
```




```
order_list <- matrix(c(1:200), ncol = 1)
rep_results <- matrix(rep_results, ncol = 1, byrow = TRUE)
results <- cbind(order_list, init_value, rep_results)
colnames(results) <- c('order_list','Starting Value','MLE')
results_1 <- subset(results, rep_results <= -3.1124 )
```

Question 3

(a) Fit the population growth model to the beetles data using the Gauss-Newton approach, to minimize the sum of squared errors between model predictions and observed counts.

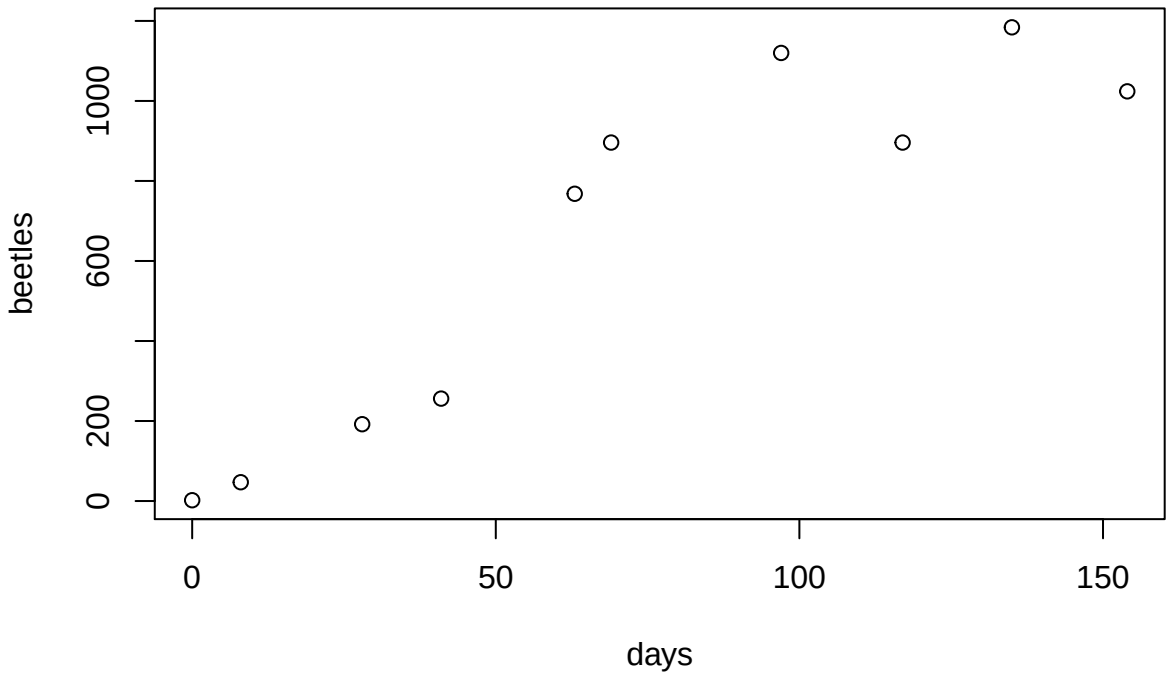
```
library(car)
```

```
## Warning: package 'car' was built under R version 3.4.3
```

```
library(ggplot2)

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))

plot(beetles)
```

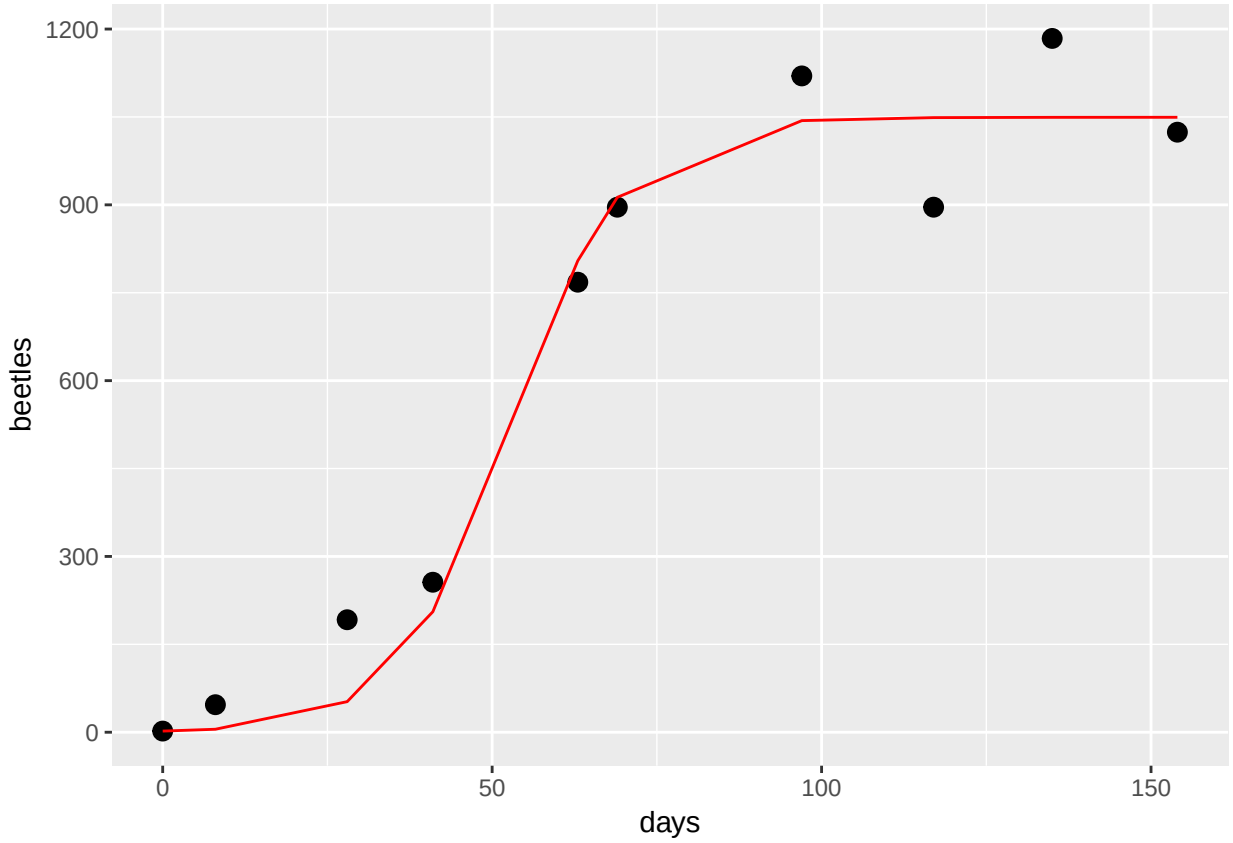



```
## Since K is the parameter that represents the population carrying capacity,and the number of beetles c
K <- 1200
r <- 0.15
beetles_mod <- nls(beetles ~ K * 2 / (2 + (K - 2) * exp((-r) * days)),
           start = list(K = 1200, r = 0.15),
           data = beetles)
summary(beetles_mod)
```

```
##
## Formula: beetles ~ K * 2/(2 + (K - 2) * exp((-r) * days))
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## K 1.049e+03  4.717e+01   22.25 1.76e-08 ***
## r 1.183e-01  6.533e-03   18.10 8.90e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 95.8 on 8 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 2.022e-06
```

```
#plot the results
p <- ggplot(beetles, aes(days, beetles))
```

```
p + geom_point(size = 3) + geom_line(aes(days, fitted(beetles_mod)), col='red')
```



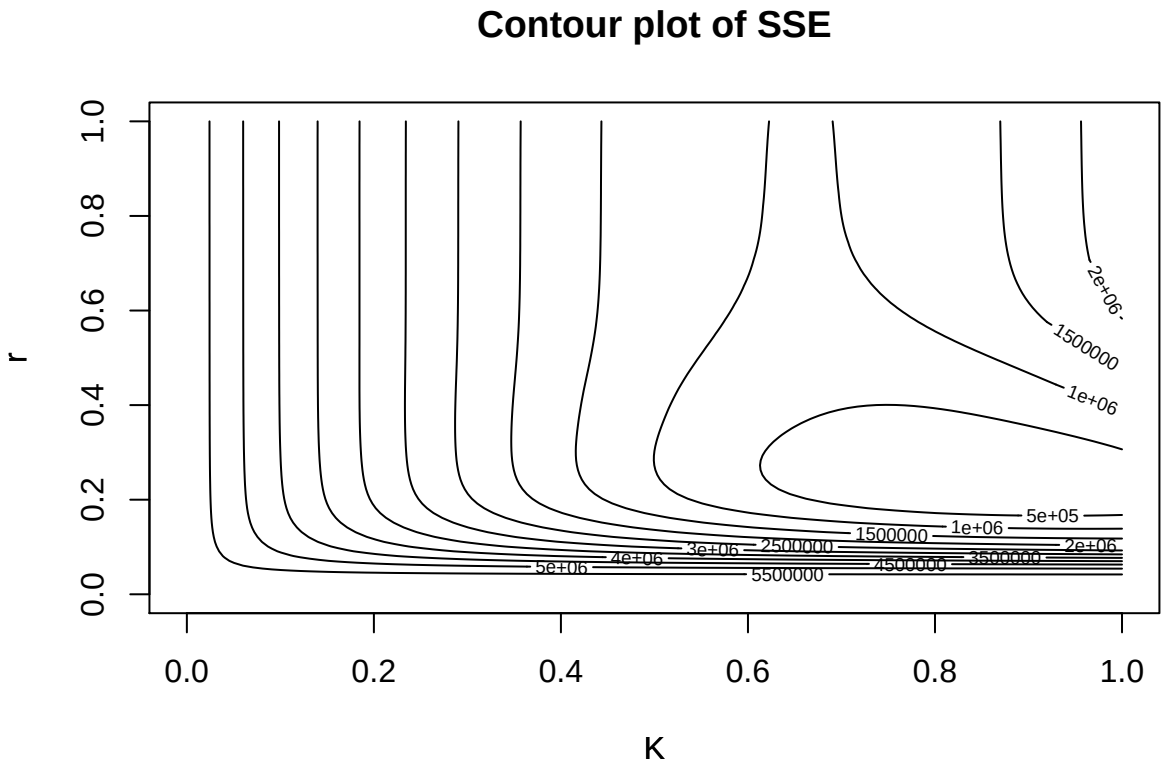
(b)Show the contour plot of the sum of squared errors.

```
days    = c(0,  8,  28,  41,  63,  69,   97, 117,  135,  154)
beetles = c(2, 47, 192, 256, 768, 896, 1120, 896, 1184, 1024)

error <- function(K, r){
  sum((beetles - 2 * K / (2 + (K - 2) * exp(-r * days))) ^ 2)
}

results_3b <- matrix(0, 100, 100, byrow=TRUE)
for (i in (1 : 100)){
  for (j in (1 : 100)){
    K <- 12 * i
    r <- 0.005 * j
    results_3b[i,j] <- error(K, r)
  }
}

contour(results_3b, xlab = 'K', ylab = 'r',
              plot.title = title ( main = "Contour plot of SSE",
                                   xlab = "K", ylab = "r"))
```



(c) Estimate the variance your parameter estimates.

```
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))
log_function <- function(estimate_values, beetles, days) {
  K <- estimate_values[1]
  r <- estimate_values[2]
  sigma <- estimate_values[3]
  value <- log((K * 2) / (2 + (K - 2) * exp(-r * days)))
           - sum(dnorm(log(beetles), value, sigma, log = TRUE))
}
estimate_values <- c(1200, 0.17,2.03)
sigma_value <- sqrt(var(log(beetles$beetles)))
mod <- nlm(log_function, estimate_values,
           beetles = beetles$beetles,
           days = beetles$days,
           hessian = TRUE)

print(mod)
```

```
## $minimum
## [1] 9.790127
##
## $estimate
## [1] 820.3815695   0.1926394   0.6440836
##
## $gradient
```

```
## [1]  1.138721e-08  2.900435e-05 -2.714273e-06
##
## $hessian
##               [,1]         [,2]          [,3]
## [1,]  2.389705e-05   0.05442387 -3.048933e-06
## [2,]  5.442387e-02 373.52578204 -5.767475e-02
## [3,] -3.048933e-06  -0.05767475  4.817365e+01
##
## $code
## [1] 2
##
## $iterations
## [1] 38
```

```
estimates <- mod$estimate
print(estimates)
```

```
## [1] 820.3815695   0.1926394   0.6440836
```

```
hh <- mod$hessian
print(hh)
```

```
##               [,1]         [,2]          [,3]
## [1,]  2.389705e-05   0.05442387 -3.048933e-06
## [2,]  5.442387e-02 373.52578204 -5.767475e-02
## [3,] -3.048933e-06  -0.05767475  4.817365e+01
```

```
var_matrix <- solve(hh)
diag(var_matrix)
```

```
## [1] 6.262790e+04 4.006745e-03 2.075824e-02
```