Lab 1 report

Simon Jorstedt & Siddhesh Sreedar

2023-11-07

Question 1

We will begin by predefining our given data, an appropriate interval of θ values, and two functions for the log likelihood and the derivative of the log likelihood (or the score) of a Cauchy distribution as instructed in an R chunk (see Appendix).

We will now calculate and plot the log-likelihood and the score for θ in the range $[-4, 4]$ in Figure 1 and Figure 2 below. For the code, see the Appendix. In Figure 1, we see that there appears to be two local maximi and one local minimi in the interval $[-4, 4]$. When inspecting Figure 2, we see that the curve appears to cross $y = 0$ three times within the interval, which supports this.

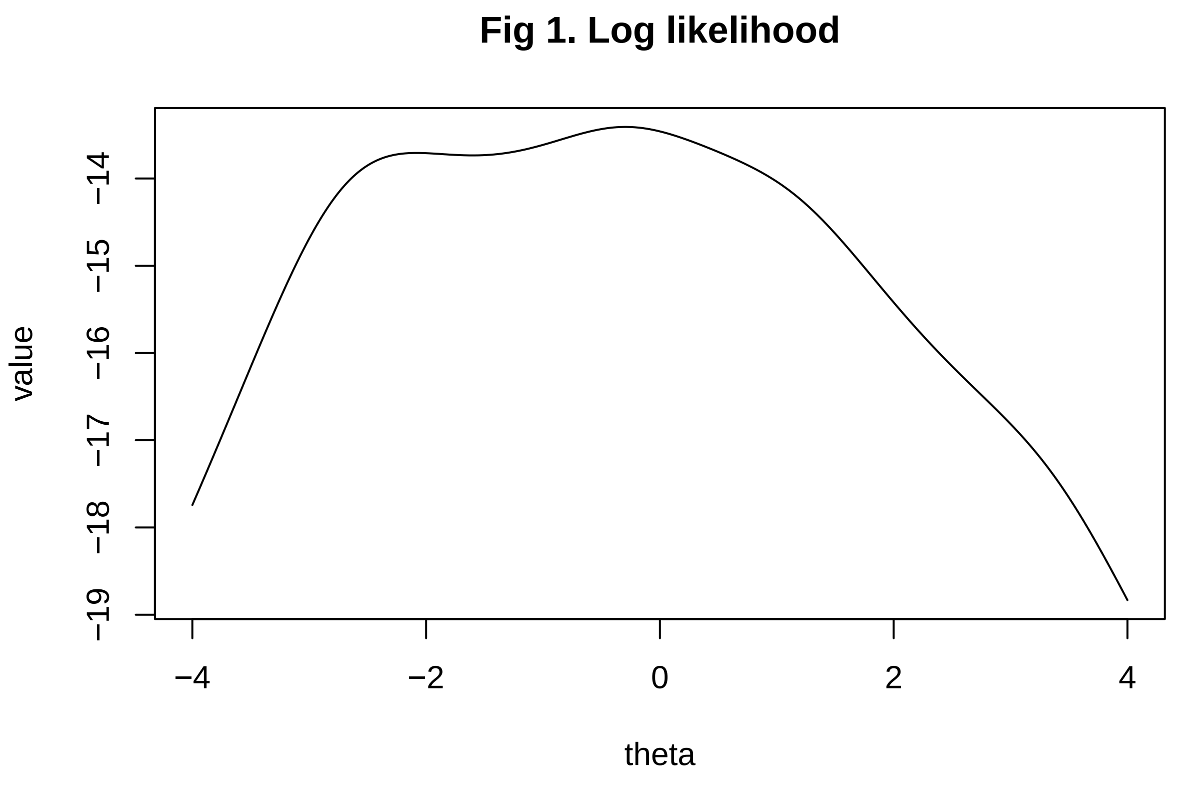
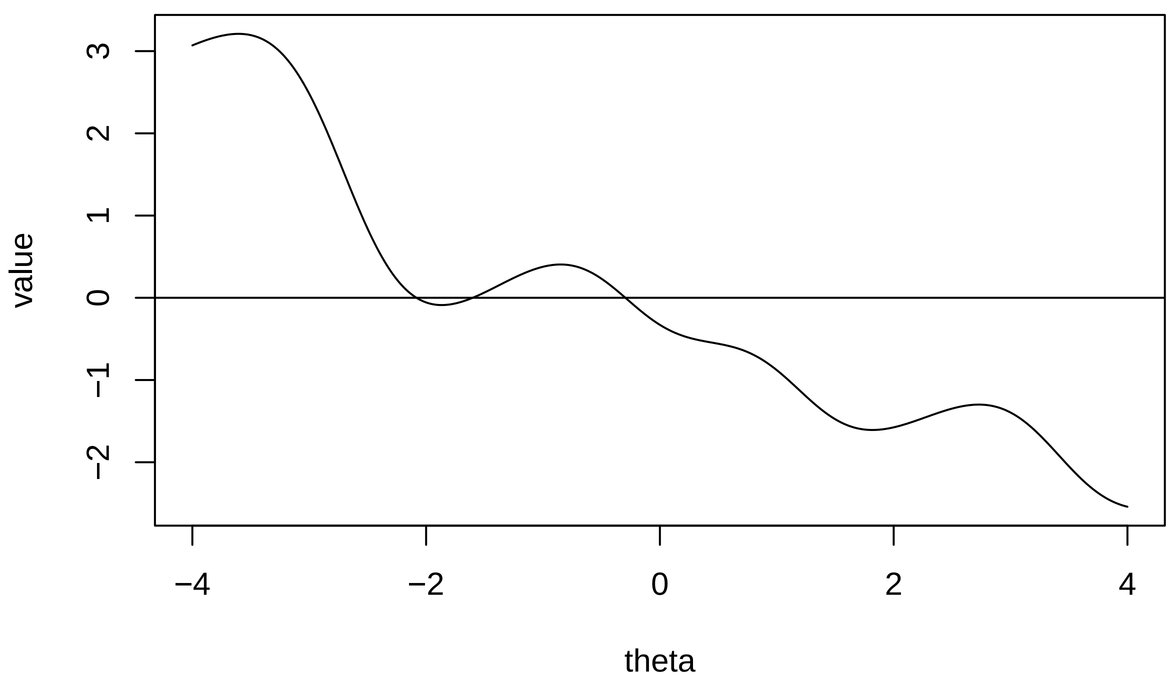


Fig 2. Log likelihood derivative



We will now use the Bisection method to locate the maximi. We define a function that does this using two start interval boundaries. For this code, see the Appendix.

The bisection method requires two initial boundaries. It is also required that the product of the score at these boundaries is negative. The point of this is to ensure that one of the boundaries is to the left of a maximum, and that one is to the right.

With this in mind, we carefully examine Figure 1 and 2, and determine that starting values -2.5 and -2 for the local maximum to the left, and -0.5 and 0 for the local maximum to the right are appropriate.

There are many possible starting intervals that do not result in a local maximum, for example $[1, 4]$, and $[-4, -3]$. This is because they do not fulfil the requirement that the interval must contain a local optimum. Indeed the Bisection method does not even find the local maximum within the intervals $[1, 4]$ and $[-4, -3]$, because the method fundamentally relies on the the signs of the interval boundaries to refine the interval.

When the log-likelihood is computed for the two local maximi x-value estimates, we get the following output. For the code that produces this output, please see the Appendix. The global maximum is apparently achieved at about x = -0.2952452.

```
## Estimate of x_value:  -2.082124

## loglike value:  -13.7077

## Estimate of x_value:  -0.2952452

## loglike value:  -13.40942
```

In a situation where one cannot visually choose starting values, one could direct a program to try out many combinations of starting boundaries. In the end, the resulting local optimi can be compared and the global maximum or global minimum can be selected as requried. For a chunk of code that implements this, please see the Appendix.

Question 2

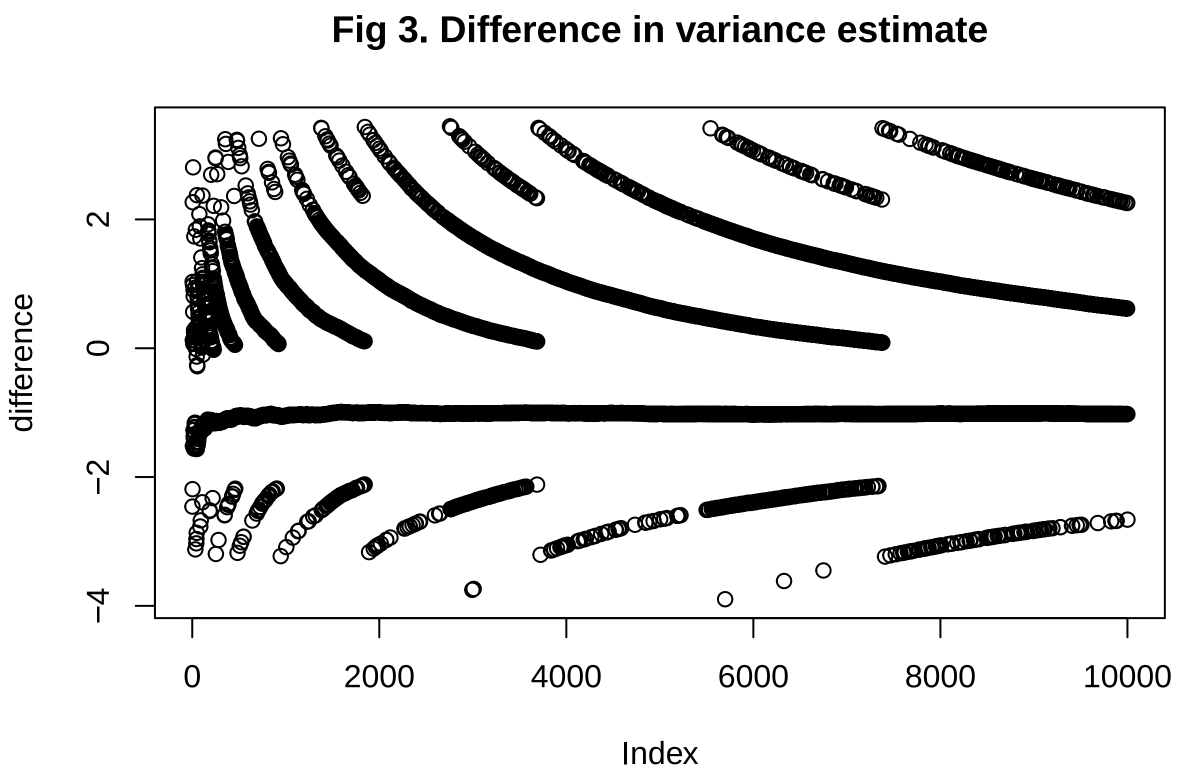
We will begin by writing a function `myvar()` which estimates variance as instructed. For this code, please see the Appendix.

We will now test `myvar()` by simulating a vector `x` of 10^4 observations from the normal distribution with mean 10^8, and variance 1. For this code, please see the Appendix.

We can now take the vector `x`, and estimate the variance using both our function `myvar()` and the inbuilt R function `var()` for each subset

$$X_i = \{x_1, \cdots, x_i\}$$

where we let $i = 1, \cdots, 10^4$. We then calculate the difference between these variance estimations for each of the subsets, and plot the difference in Figure 3 below. For the code that produces Figure 3, see the Appendix.



As we can see from Figure 3, if the `myvar()` and `var()` functions outputted the same values, then we would have see all the points at 0 on the horizontal axis but it is not the case. The difference range between -4 to +4 and this could be due to rounding errors that R produces.

Since the vector of theta values are very small numbers, the calculation with the var() function may handle these values better than the myvar function.

A better variance estimator may be achieved that avoids these computation errors. To see this implemented, see the Appendix.

Appendix

```r
# Problem setup

## Data parameter values
theta <- seq(-4, 4, 0.01)
x <- c(-2.8, 3.4, 1.2, -.3, -2.6)

# Log-likelihood function
loglike <- function(theta){-5*log(pi)-sum(log(1+(x-theta)^2))}

# Log-likelihood first derivative function
loglike_derivative <- function(theta){sum(2*(x-theta)/(1+(x-theta)^2))}
```

```r
# Log likelihood plot
y_a1 <- c()
for (theta_i in theta){
  y_a1 <- c(y_a1, loglike(theta_i))
}

plot(theta,
     y_a1,
     type="l",
     main = "Fig 1. Log likelihood",
     xlab = "theta",
     ylab = "value")


# Log likelihood derivative plot
y_a2 <- c()
for (theta_i in theta){
  y_a2 <- c(y_a2, loglike_derivative(theta_i))
}

plot(theta,
     y_a2,
     type="l",
     main = "Fig 2. Log likelihood derivative",
     xlab = "theta",
     ylab = "value")
abline(b=0, a=0)
```

```r
# Bisection function

bisect_estimate <- function(a, b, epsilon = 0.000001){
  # a is the left start guess
  # b is the right start guess
  # epsilon is the desired convergence error

  y1 <- loglike_derivative(a)
  y2<- loglike_derivative(b)
  #stopifnot("Invalid boundaries `a` and `b`" = y1 * y2 < 0)

  x_t <- c()
  # In the beginning, x_t is either empty or only has one value.
  # When x_t has at least two values, evaluate the last two values with
  # respect to the convergence error.
  while(length(x_t) < 2 || abs(tail(x_t,1)-tail(x_t,2)[1]) > epsilon){
    y1 <- loglike_derivative(a) #sum(2*(x-a)/(1+(x-a)^2))
    y2<- loglike_derivative(b) #sum(2*(x-b)/(1+(x-b)^2))

    # Calculate new boundary
    x_t <- c(x_t,(a + b)/2)

    if(loglike_derivative(a)*loglike_derivative(tail(x_t,1)) <= 0){
      b <- tail(x_t,1)
    }
    if(loglike_derivative(b)*loglike_derivative(tail(x_t,1)) < 0){
      a <- tail(x_t,1)
    }
  }
  cat("Estimate of x_value: ", tail(x_t, 1), "\n")
  return(tail(x_t, 1))
}
```

```r
# Local maximum
m1 <- bisect_estimate(-2.5, -2)
```

```
## Estimate of x_value:  -2.082124
```

```r
cat("loglike value: ", loglike(m1), "\n\n")
```

```
## loglike value:  -13.7077
```

```r
# Global maximum
m2 <- bisect_estimate(-0.5, 0)
```

```
## Estimate of x_value:  -0.2952452
```

```r
cat("loglike value: ", loglike(m2), "\n")
```

```
## loglike value:  -13.40942
```

```r
values <- seq(-4,4,0.1) #The range of theta values

first_der_values<-data.frame("value"= NA,"derivative" =NA) #creating an empty dataframe to store the de

#We now store the theta values and its corresponding derivative into the dataframe

for (i in 1:length(values)){
  first_der_values[i,1]<-values[i]
  first_der_values[i,2]<-loglike_derivative(values[i])

}

#separating the positive and negtaive derivative value into
pos<-subset(first_der_values,derivative>0)
neg<-subset(first_der_values,derivative<0)

final<-c()

len<-nrow(pos)
len2<-nrow(neg)



#Using a double for-loop to go through all the values and combinations of theta.
for (i in 1:len){
  for( j in 1:len2){

    if(pos[i,1]<neg[j,1]){
      a<-pos[i,1]
      b<-neg[j,1]
    }else{
      b<-pos[i,1]
      a<-neg[j,1]
    }
    x_i <- c()

    while(length(x_i) < 2 || abs(tail(x_i,1)-tail(x_i,2)[1])>0.00000001){
      y1 <- loglike_derivative(a) #sum(2*(x-a)/(1+(x-a)^2))
      y2<- loglike_derivative(b) #sum(2*(x-b)/(1+(x-b)^2))

      if(length(x_i) == 0 && y1 * y2 >= 0){print("Break");break}

      x_i <- c(x_i,(a + b)/2)

      if(loglike_derivative(a)*loglike_derivative(tail(x_i,1)) <= 0){
        b <- tail(x_i,1)
      }
      if(loglike_derivative(b)*loglike_derivative(tail(x_i,1)) < 0){
        a <- tail(x_i,1)
      }
    }
    final<-c(final,(tail(x_i, 1)+tail(x_i, 2)[1])/2)
  }
}

#plot(theta, y_a1, type="l")
#abline(v=final)


#plot(theta, y_a2, type="l")
#abline(b=0, a=0)
#abline(v=final)


#We get all the optimal values:
final2<-as.data.frame(final)
#-1.6017752,-0.2952455,-2.0821239

#Now we compare these values with the function to see which gives the highest value
y_a11 <- c()
optim_val<-c(-1.6017752,-0.2952455,-2.0821239)
for (theta_i in optim_val){
  y_a11 <- c(y_a11, -5*log(pi)-sum(log(1+(x-theta_i)**2)))
}
#[1] -13.73572 -13.40942 -13.70770

#So the highest value is when theta is -0.2952455
```

```r
# Variance estimation function
myvar <- function(x){
  output <- (sum(x^2) - sum(x)^2 / length(x)) / (length(x)-1)

  #cat("Variance estimate:", output, "\n")
  return(output)
}
```

```r
set.seed(18078)
x <- rnorm(n=10^4, mean=10^8, sd=1)
```

```r
# The difference for each subset

myvar_vector <- c()
var_vector <- c()
for (i in 1:10^4){
  myvar_vector <- c(myvar_vector, myvar(x[1:i]))
  var_vector <- c(var_vector, var(x[1:i]))
  #difference <- c(difference, myvar(x[1:i])-var(x[1:i]))
}
difference <- myvar_vector - var_vector

plot(difference,
     type = "p",
     main = "Fig 3. Difference in variance estimate")
#plot(myvar_vector, var_vector)
```

```r
myvar_improved_1<-function(x){
   m <- mean(x)
  sum((x-m)**2)/(length(x)-1)
}

myvar_improved_2<-function(x){
  var(x)*(length(x)-1)/length(x)

}

#checking difference with the first improvement
myvar_improved_1_vector <- c()
var_vector <- c()
for (i in 1:10^4){
  myvar_improved_1_vector <- c(myvar_improved_1_vector, myvar_improved_1(x[1:i]))
  var_vector <- c(var_vector, var(x[1:i]))
}

difference1 <- myvar_improved_1_vector - var_vector


#checking difference with the second improvement
myvar_improved_2_vector <- c()
var_vector <- c()
for (i in 1:10^4){
  myvar_improved_2_vector <- c(myvar_improved_2_vector, myvar_improved_2(x[1:i]))
  var_vector <- c(var_vector, var(x[1:i]))
}

difference2 <- myvar_improved_2_vector - var_vector



#myvar_improved_1 gives better results and the difference to the var() function in R is much smaller.
```