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.

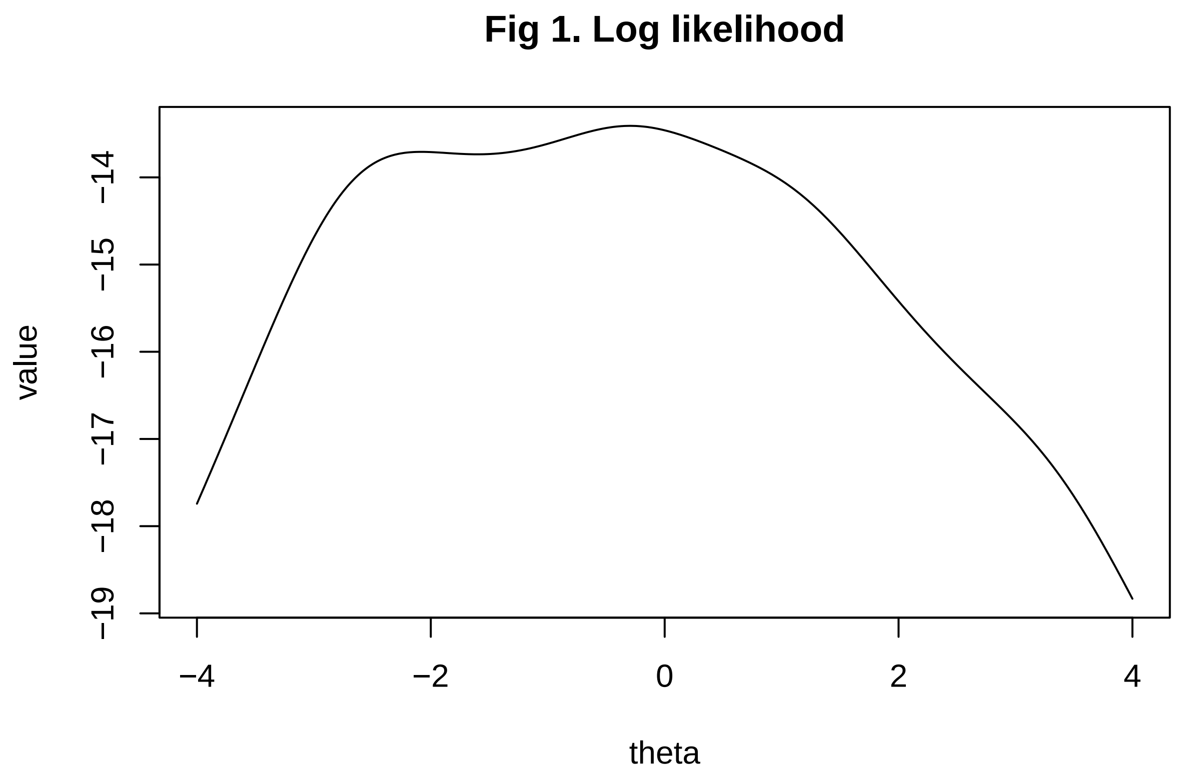
```r
# Problem setup

## 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))}
```

Subquestion 1.a

> Plot the log likelihood function for the given data in the range from -4 to 4. Plot the derivative in the same range and check visually how often the derivative is equal to 0.

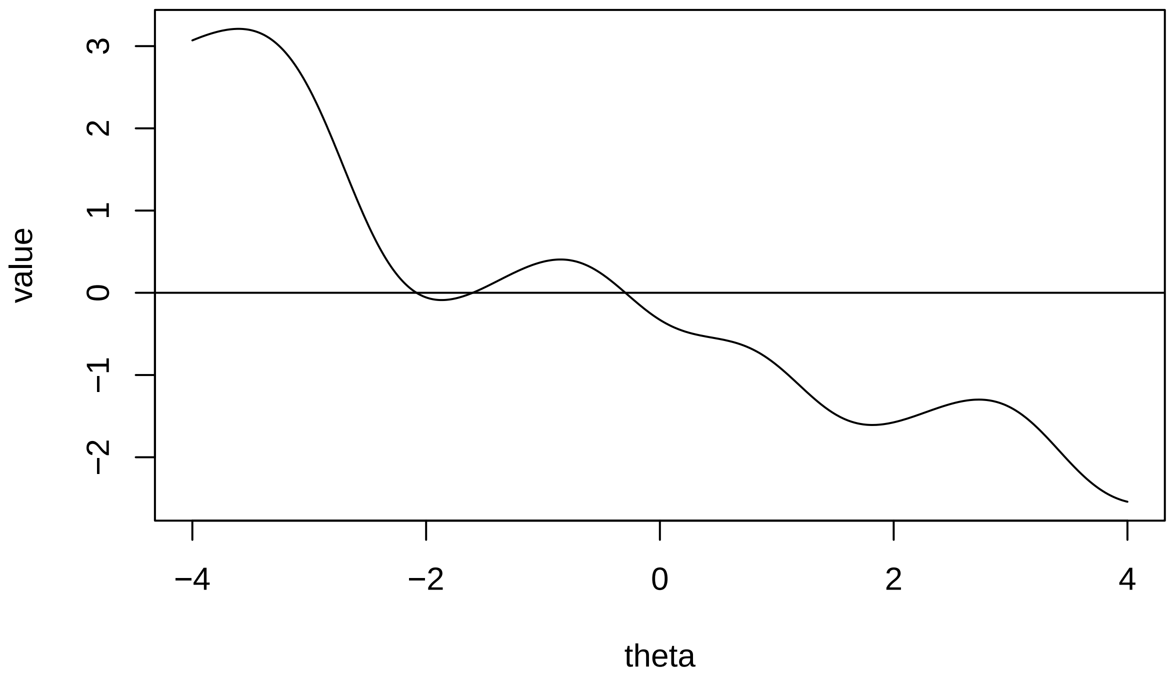
In the chunk below we calculate and plot first the log likelihood values, and then the score for the θ interval $[-4, 4]$.

```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")
```



```r
# 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)
```

Fig 2. Log likelihood derivative



Inspecting the derivative plot, we see that the curve appears to cross $y = 0$ three times within the interval.

Subquestion 1.b

We have chosen the Bisection method, and in the Chunk below we implement a function that performs the Bisection method given left and right starting boundaries.

```
# 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))
}
```

Subquestion 1.c

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, and -0.5 and 0 for the global maximum 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 maximum. 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.

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

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

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

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

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

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

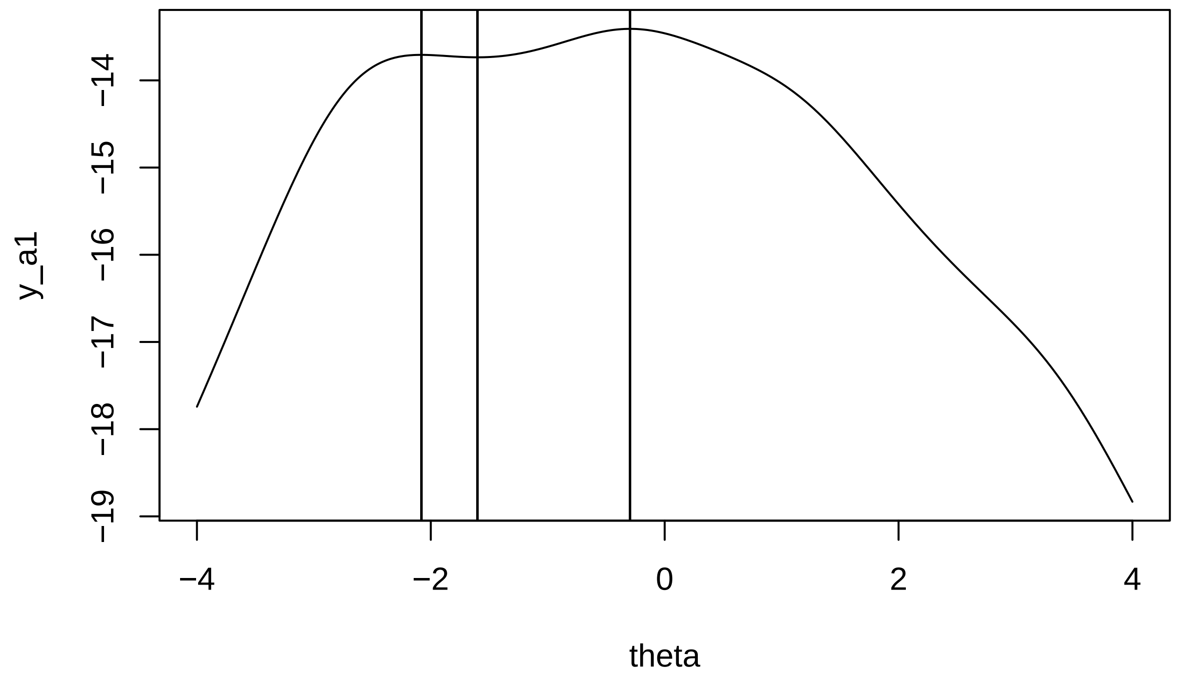
```
cat("loglike value: ", loglike(m2), "\n")
```

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

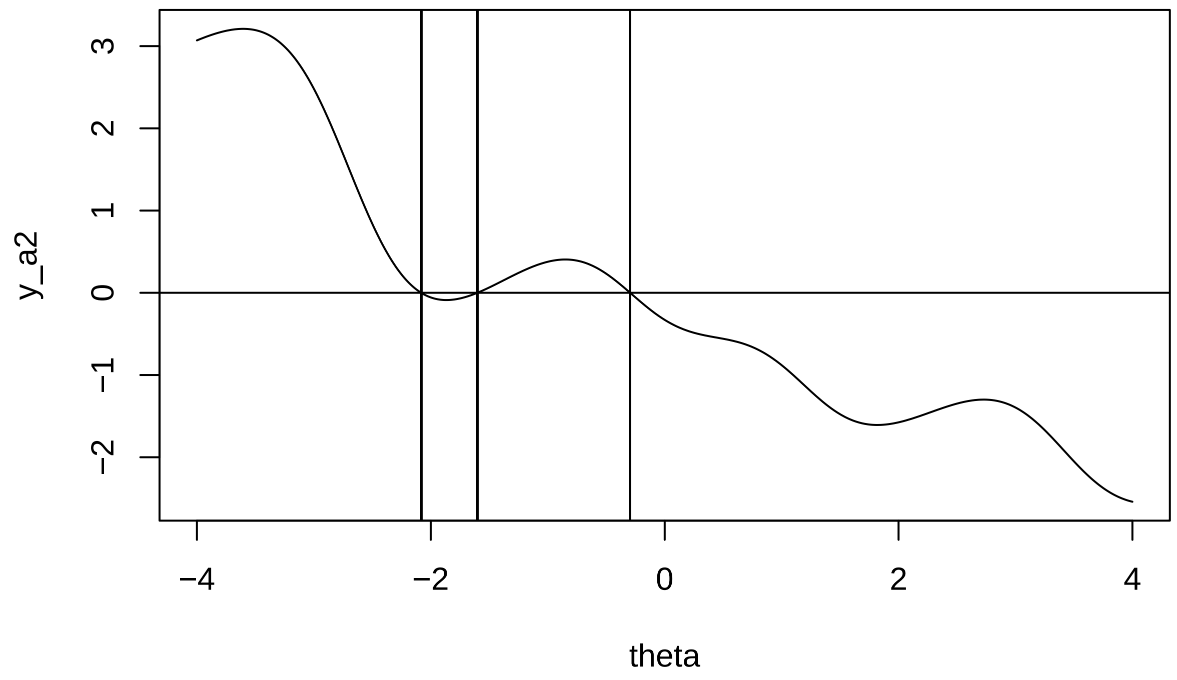
The global maximum is thus achieved at about x = -0.2952452.

Subquestion 1.d

```r
values<-seq(-4,4,0.1)

first_der_values<-data.frame("value"= NA,"derivative" =NA)

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

}

pos<-subset(first_der_values,derivative>0)
neg<-subset(first_der_values,derivative<0)

final<-c()

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

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)
```



```
final2<-as.data.frame(final)


#-1.6017752,-0.2952455,-2.0821239

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
```

Question 2

Subquestion 2.a

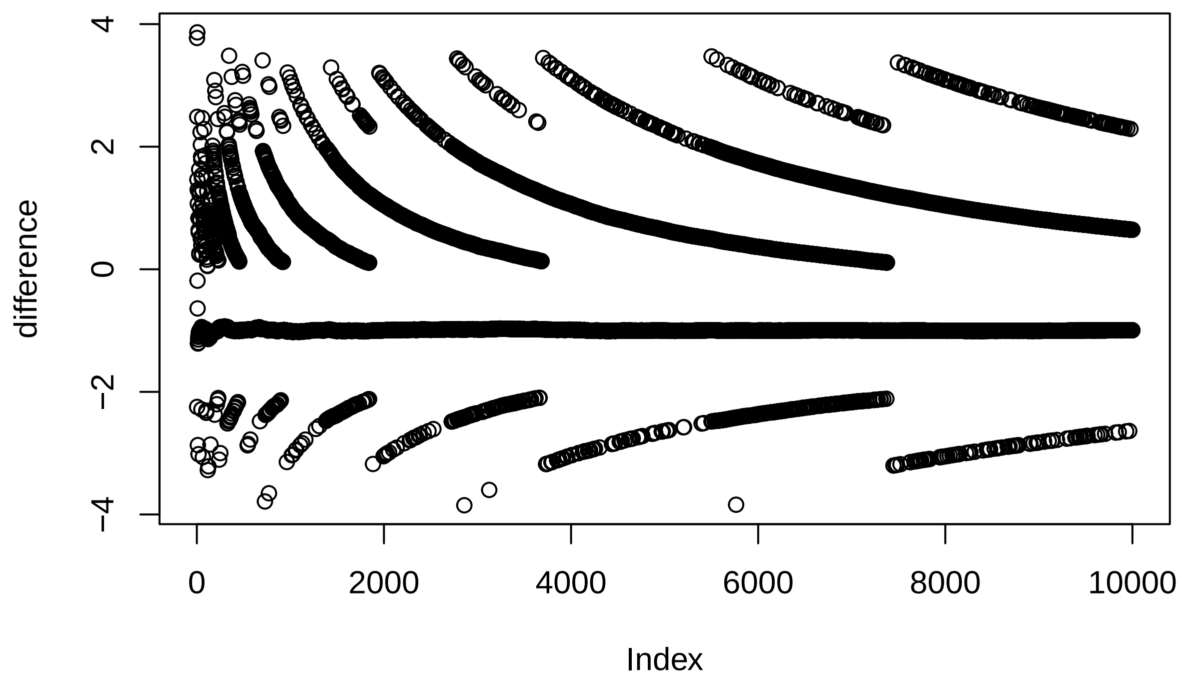
```
# Variance estimation function
myvar <- function(x){
  output <- (sum(x^2) - sum(x)^2 / length(x)) / (length(x)-1)

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

Subquestion 2.b

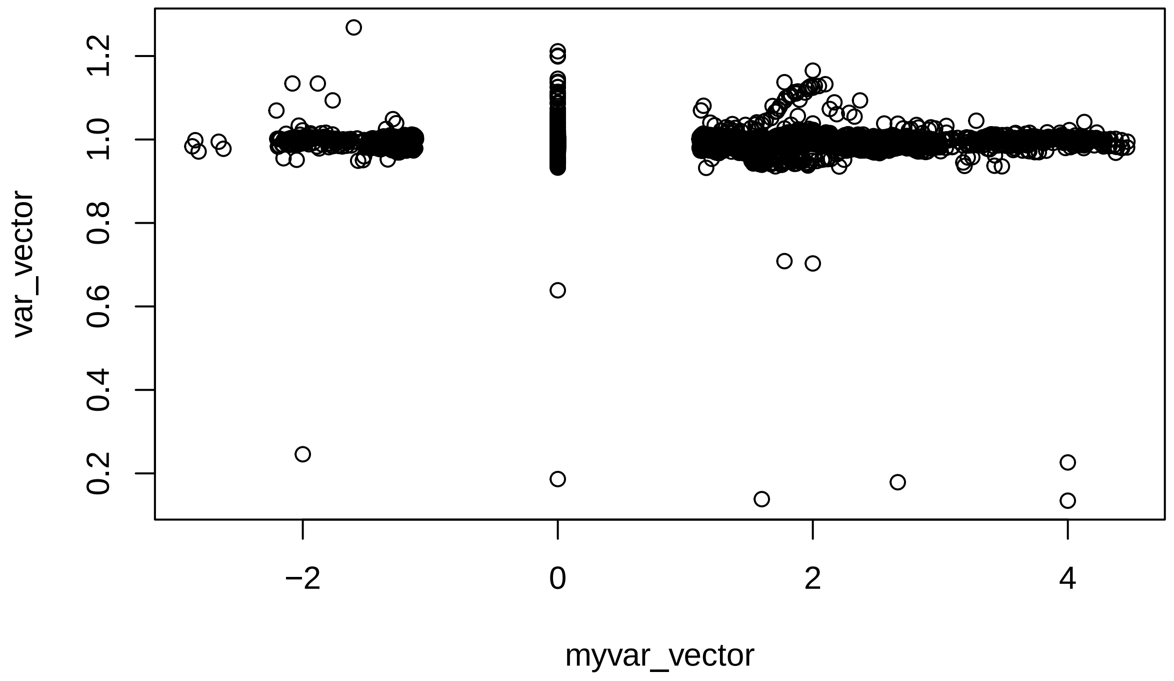
```
x <- rnorm(n=10^4, mean=10^8, sd=1)
```

Subquestion 2.c

```
# 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")
```



```r
plot(myvar_vector, var_vector)
```



Subquestion 2.d

```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
```

```r
#checking difference with the seconf 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 close to the var() function in R.
```