732A90 Lab 1

2023-11-06

Computational Statistics - Fall 23

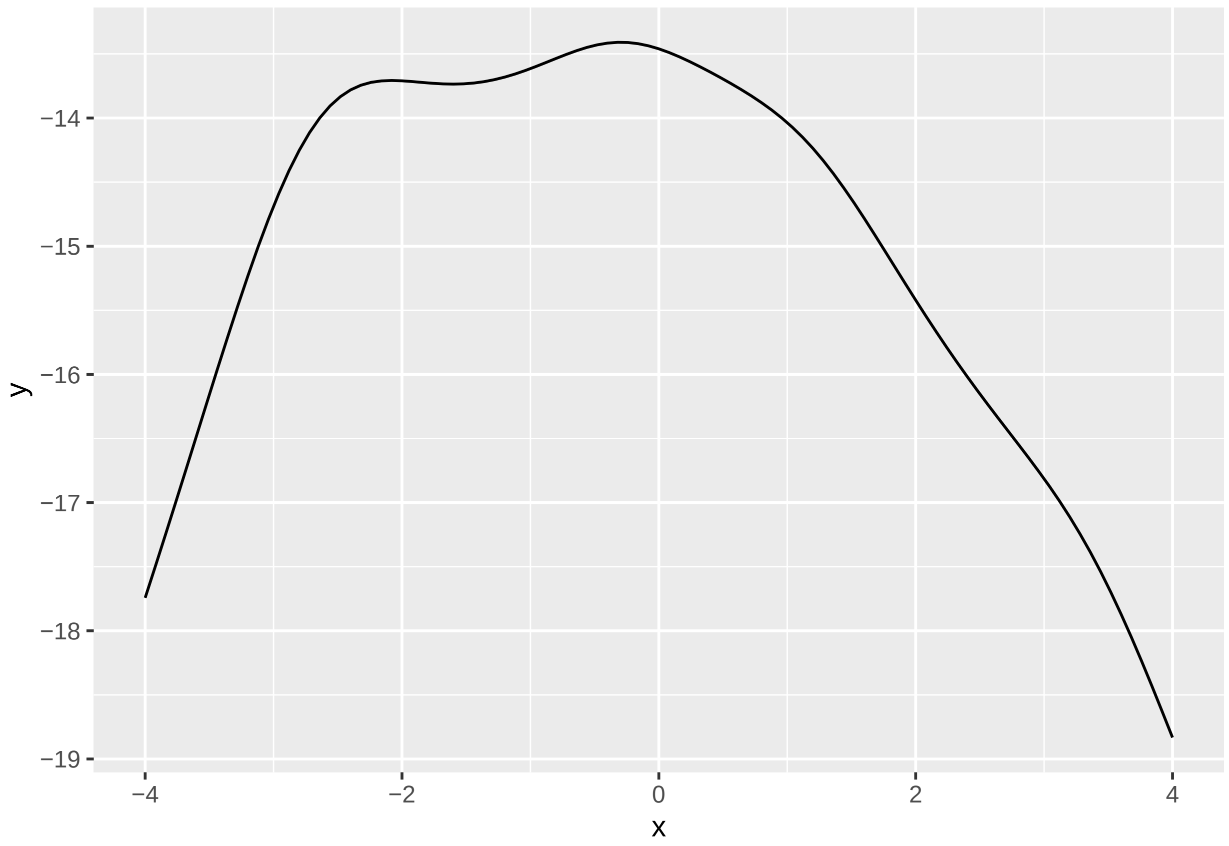
Computer Lab 1

Hugo Morvan, Daniele Bozzoli

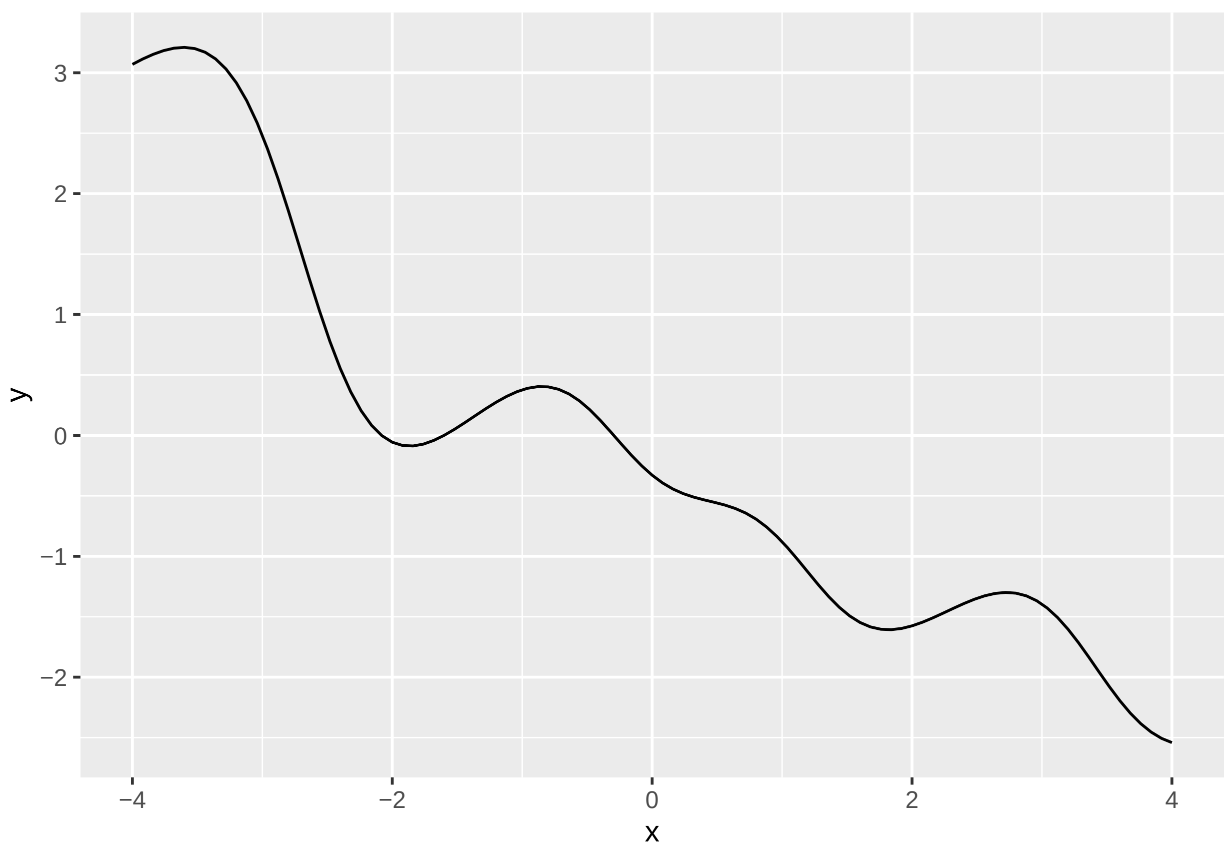
Question 1

a)

Plot of the Likelihood function:



Plot of the first derivative of the likelihood function:



From this plot, we can see that the first derivative is equal to 0 in three locations.

b)

See Appendix Question 1 b)

c)

First optimum:

```
bisection(-3, -2, first_derivative, 9)
```

```
## [1] -2.082124
```

Second optimum:

```
bisection(-2, -1, first_derivative, 9)
```

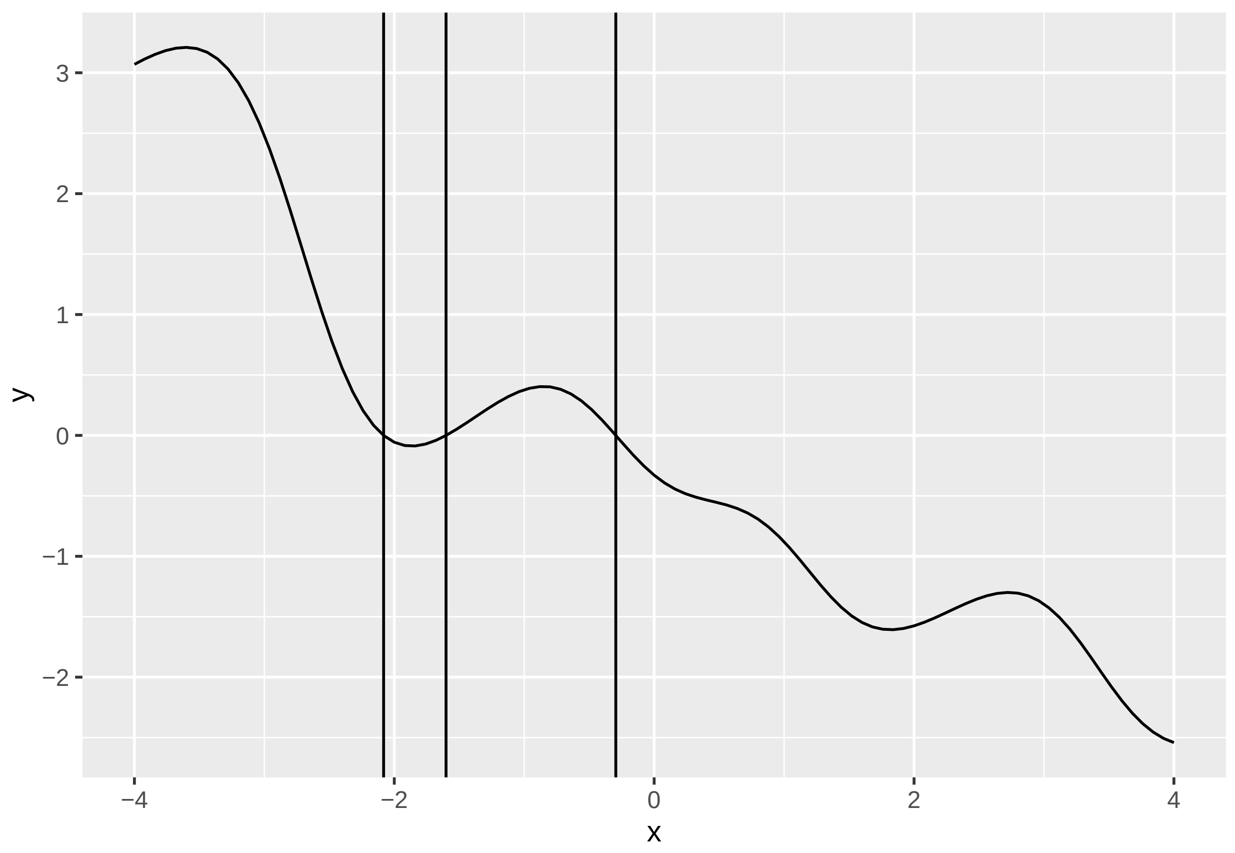
```
## [1] -1.601775
```

Third optimum:

```
bisection(-1, 0, first_derivative, 9)
```

```
## [1] -0.2952455
```

We can check our result by adding the X values on the graph:



To find the global maximum, we plug in the local maxima found into f(x):

```
## first intercept, f(x)= -13.7077
```

```
## second intercept, f(x)= -13.73572
```

```
## third intercept, f(x)= -13.40942
```

Based on the computation, the third local optima is the global maximum:

```
## x =  -2.082124
```

If our starting interval contains several local optima, it is not guaranteed that the bisection method finds the local maximum of that interval, it is instead only finding one of the local optima that the interval contains. Otherwise, our function prevents the user from choosing an interval that does not contain a local optima.

d)

Pseudo-code to find all the local optima given a wide starting range [A,B] containing several optima.

Step 1: set left boundary a to A, the left limit of our range. Set right boundary b to be slightly bigger than A such that the stopping condition of our while loop is not met (i.e. a != b, but very close)

Step 2: Calculate f'(a)*f'(b). If the result is positive, increase the value of b until the result becomes negative. Once the result is negative, run the bisection function to find the optima and store it in a vector.

Step 3: set the left boundary to be b (the previous right boundary)

Step 4: Repeat steps 1,2,3 until the right boundary reaches the value of B, the end of our range. You should have found all of the optima between A and B.

Step 5: Find the global maximum by comparing the values of the candidates in the vector into the original function f(x).

Question 2

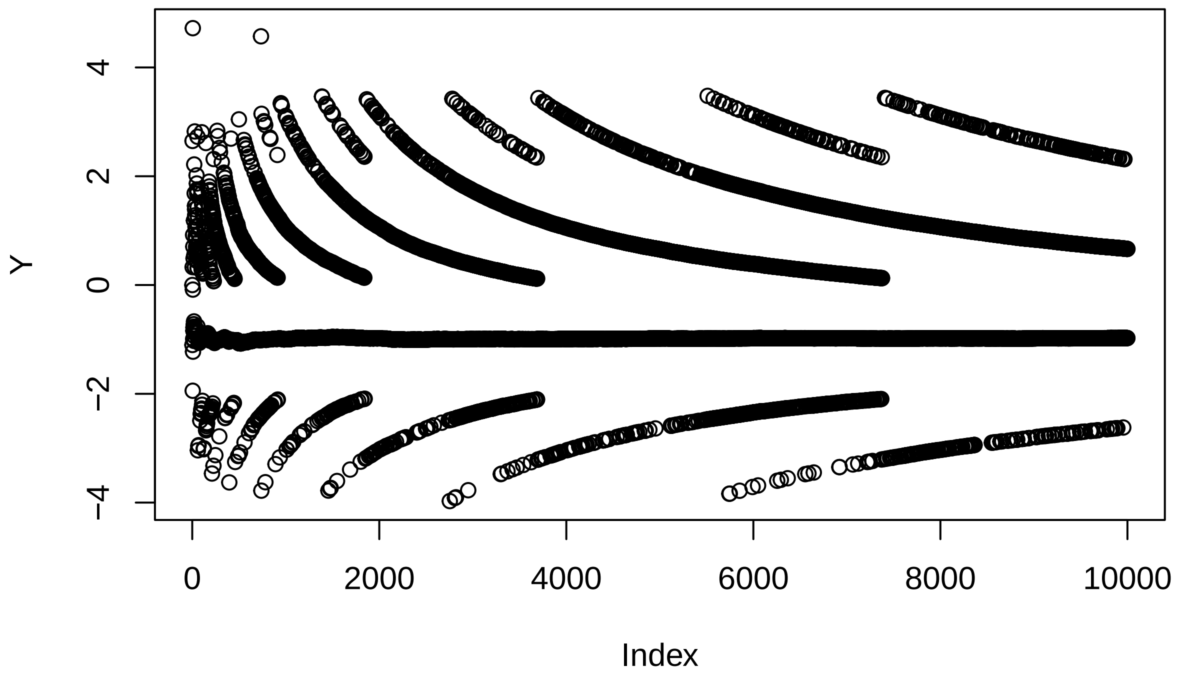
a)

See Appendix Question 2 a)

b)

See Appendix Question 2 b)

c)

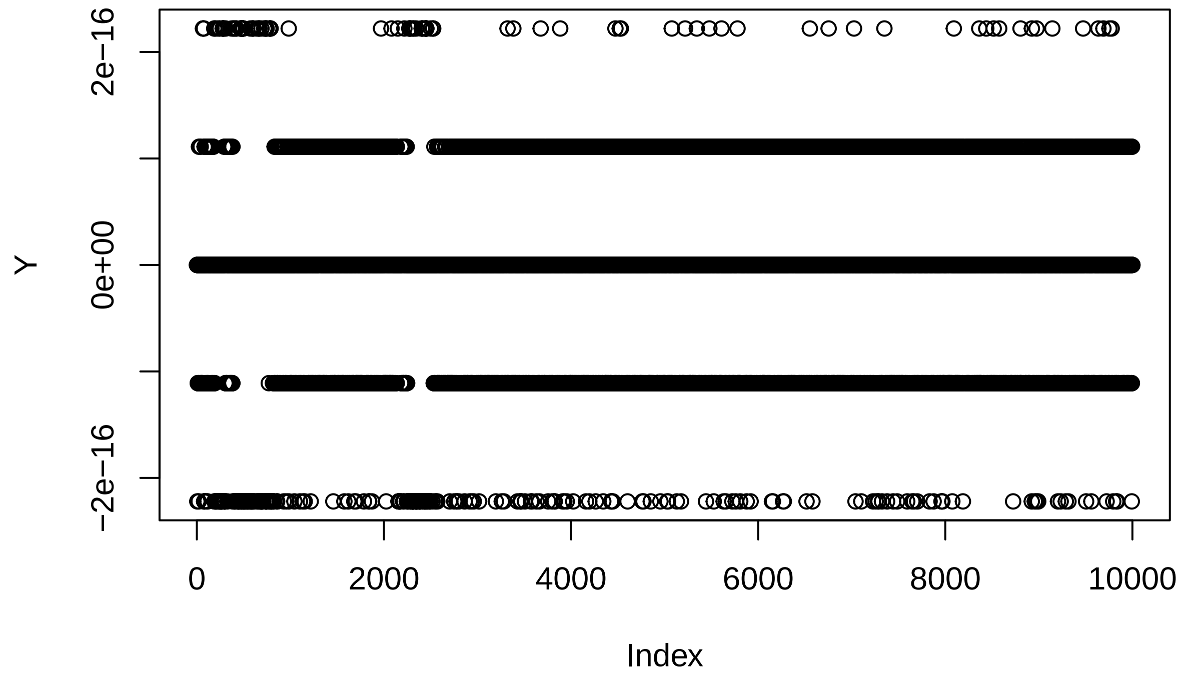


The myvar function does not work well for vectors with large mean (>= 10ˆ7). The myvar functions gives good estimations with vectors of observations coming from a Normal distribution until we get random values from N distributions with mean >= 10ˆ7. The problem comes from the computation of the sums in the myvar function: Since our mean is 10ˆ8 and we have up to 10ˆ4 elements in the vector, the sum can get as big as 10ˆ12 which is greater than the maximum value for integers in R which is ~2*10ˆ9 (32 bits). We therefore run into a stack overflow issue which completely breaks our function. This problem is avoided for smaller means as the overflow limit is not reached when calculating the sums.

d)

By rearranging the formula, we can optimize the computation of the variance to avoid stack overflow:

$$\frac{1}{(n-1)}\sum_{1}^{n}(x_i-\overline{x})^2$$



This graph shows the difference between the myvarBETTER function and the var function. We can see that the difference is negligible. We do not run into the stack overflow issue, even for large means, because we calculate the difference to the mean squared inside of the summation, so we avoid computing a large summation of very large numbers.

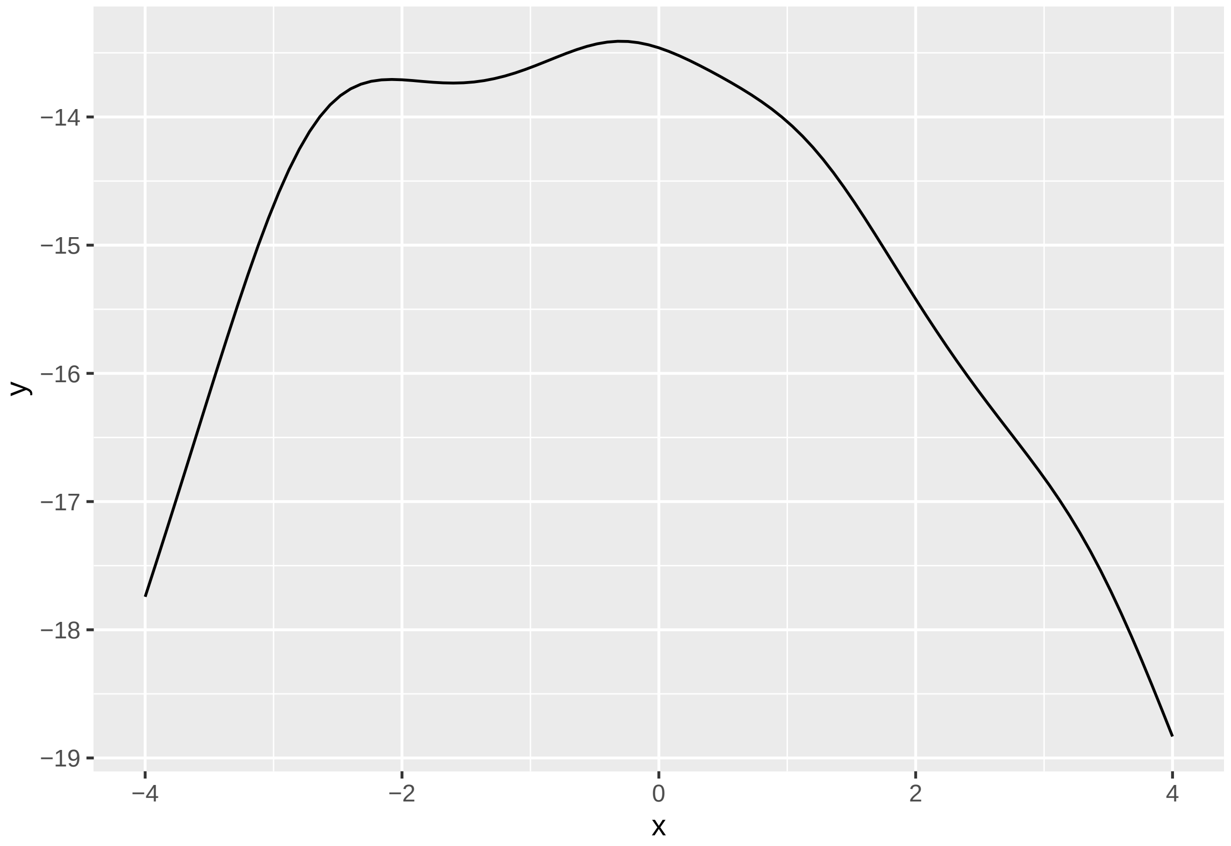
Appendix

Question 1 a)

Plot of the Likelihood function :

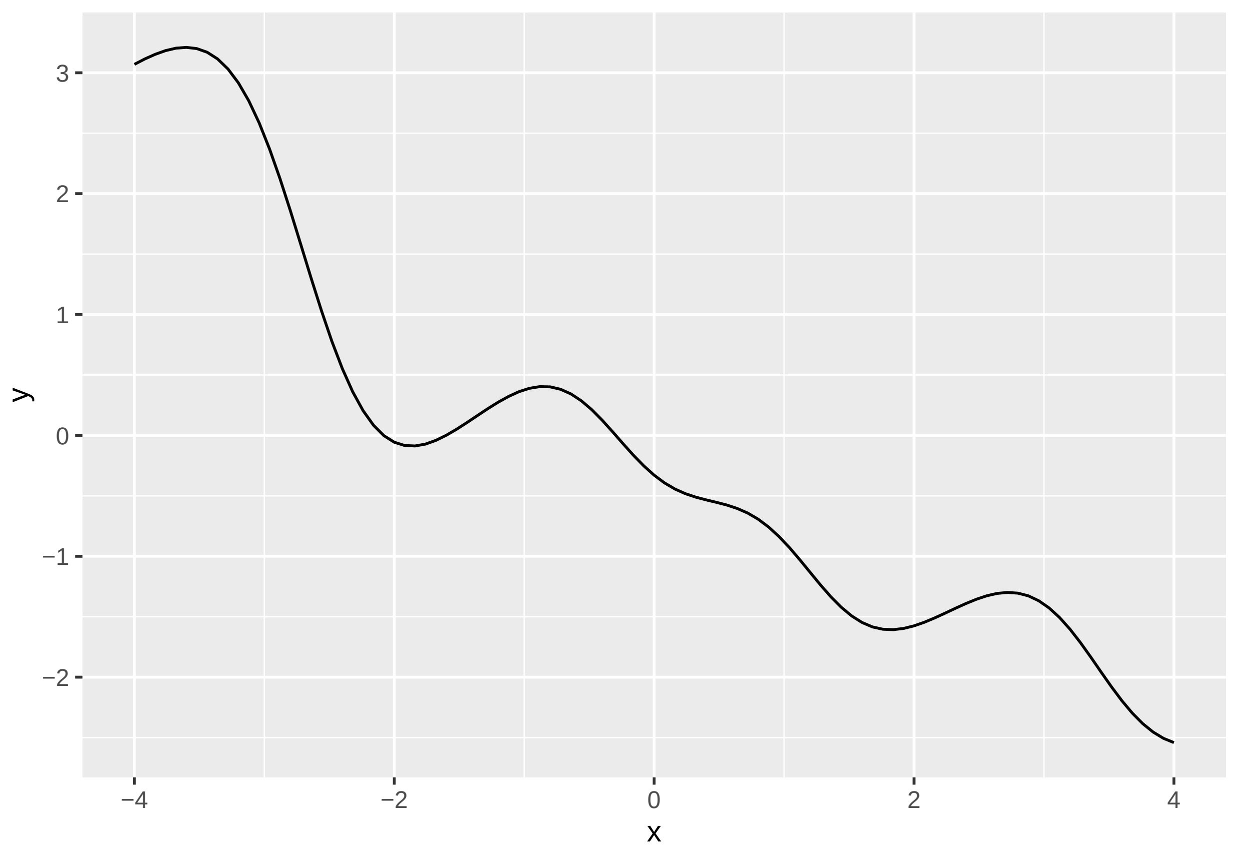
```
library("ggplot2")
log_likelyhood <- function(x){
  vec <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
  n <- 5
  sum_el <- 0
  for(el in vec){
    sum_el <- sum_el + log(1+ (el - x)^2)
  }
  res <- -n*log(pi) - sum_el
  return(res)
}
```

```
ggplot(data.frame(x=c(-4, 4)), aes(x=x)) +
  stat_function(fun=log_likelyhood)
```



Plot of the first derivative of the likelihood function

```
first_derivative <- function(theta){
  vec <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
  n <- 5
  sum_res <- 0
  for(i in vec){
    sum_res <- sum_res + 2*(i-theta)/(1+(i-theta)^2)
  }
  res <- sum_res
  return(res)
}
ggplot(data.frame(x=c(-4, 4)), aes(x=x)) +
  stat_function(fun=first_derivative)
```



Question 1 b)

```r
bisection <- function(a,b, fun, acc){
  if( fun(a)*fun(b) >= 0 ){
    stop("Choose better boundaries")
  }
  t <- 0
  while(round(a, acc) != round(b, acc)){
    mid <- (a+b)/2
    t <- t+1
    gp_a <- fun(a)
    gp_mid <- fun(mid)
    gp_b <- fun(b)
    if(gp_a * gp_mid <=0){
      b <- mid
    }
    else{
      a <- mid
    }
  }
  return(a)
}
```

Question 2 a)

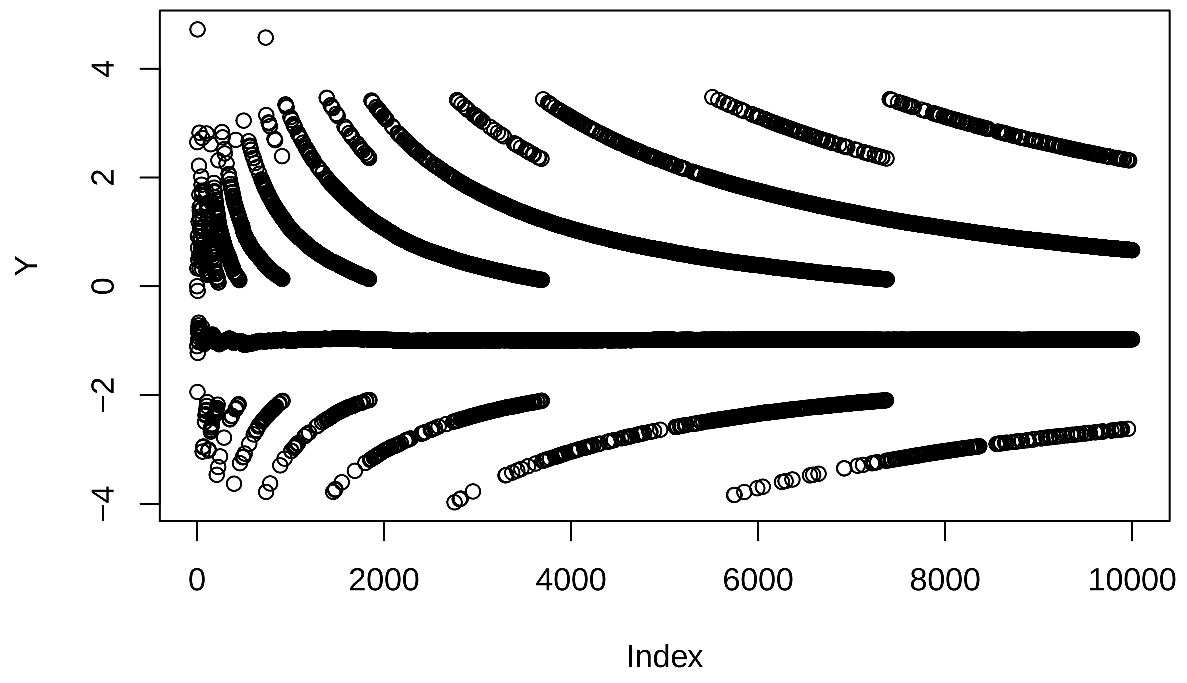
```r
myvar <- function(vector){
  n <- length(vector)
  var1 <- (1/(n-1))*(sum(vector^2)-(1/n)*(sum(vector))^2)
  return(var1)
}
```

Question 2 b)

```r
set.seed(1234)
vectorx <- rnorm(1E4, 1E8, 1)
```

Question 2 c)

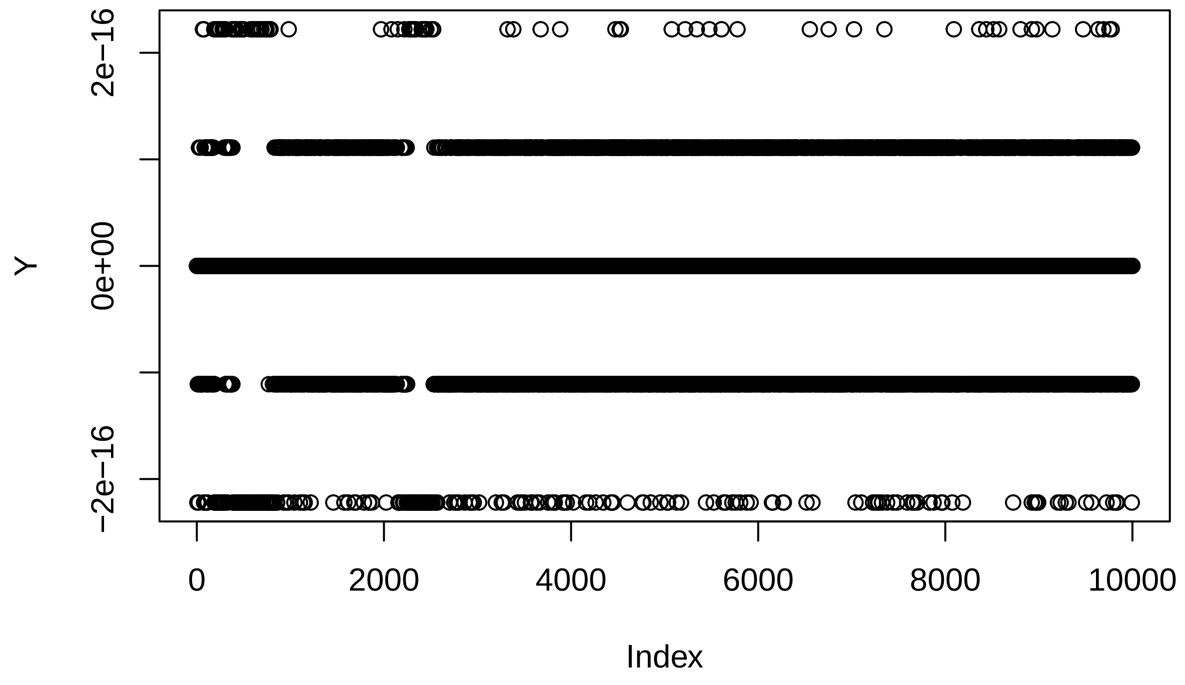
```r
Y <- rep(0,1E4)
for (q in 2:1E4){
  Y[q] <- myvar(vectorx[1:q]) - var(vectorx[1:q])
}
plot(Y)
```



Question 2 d)

```r
myvarBETTER <- function(vector){
  n <- length(vector)
  var1 <- (1/(n-1))*(sum((vector - mean(vector))^2))
  return(var1)
}

Y <- rep(0,1E4)
for (q in 2:1E4){
  Y[q] <- myvarBETTER(vectorx[1:q]) - var(vectorx[1:q])
}
plot(Y)
```


Y
2e−16
0e+00
−2e−16
Index