Laboration report in Computational Statistics

Laboration 1

732A90

Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

06 november 2023

Contents

1 Question 1: Maximization of a likelihood function in one variable 1
1.1 1 a) 1
1.2 1 b) 4
1.3 1 c) 5
1.4 1 d) 6

2 Question 2: Computer arithmetics (variance) 7
2.1 2 a) 7
2.2 2 b) 7
2.3 2 c) 7
2.4 2 d) 10

3 Statement of Contribution 12
3.1 Question 1 12
3.2 Question 2 12

4 Appendix 12

1 Question 1: Maximization of a likelihood function in one variable

We have independent data $x_1, ..., x_n$ from a Cauchy-distribution with unknown location parameter θ and known scale parameter 1. The log likelihood function is

$$-nlog(\pi) - \sum_{i=1}^{n} log(1 = (x_i - \theta)^2),$$

and it's derivative with respect to θ is

$$\sum_{i=1}^{n} \frac{2(x_i - \theta)}{1 = (x_i - \theta)^2}$$

Data of size $n = 5$ is given: $x = (-2.8, 3.4, 1.2, -0.3, -2.6)$

1.1 1 a)

Question

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.

Solution

```
library(ggplot2)
theta <- seq(-4,4,0.001)
x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
n <- 5

# Calculates the likelihood values from data
likelihood_data <- 0
for(i in 1:length(theta)){
  likelihood_data[i] <- -n * log(pi) - sum(log(1 + (x - theta[i])^2))
}

# Calculates the value of the derivative
derivative_data <- 0
for(i in 1:length(theta)){
  derivative_data[i] <- sum( (2 * (x - theta[i])) / (1 + (x - theta[i])^2) )
}

# Creates data frames to plot
plot_likelihood <- data.frame(theta, value=likelihood_data)
plot_derivative <- data.frame(theta, value=derivative_data)

# Plot for the likelihood function
ggplot(plot_likelihood, aes(x=theta, y=value)) +
  geom_line() +
  theme_bw() +
  labs(y="value")
```

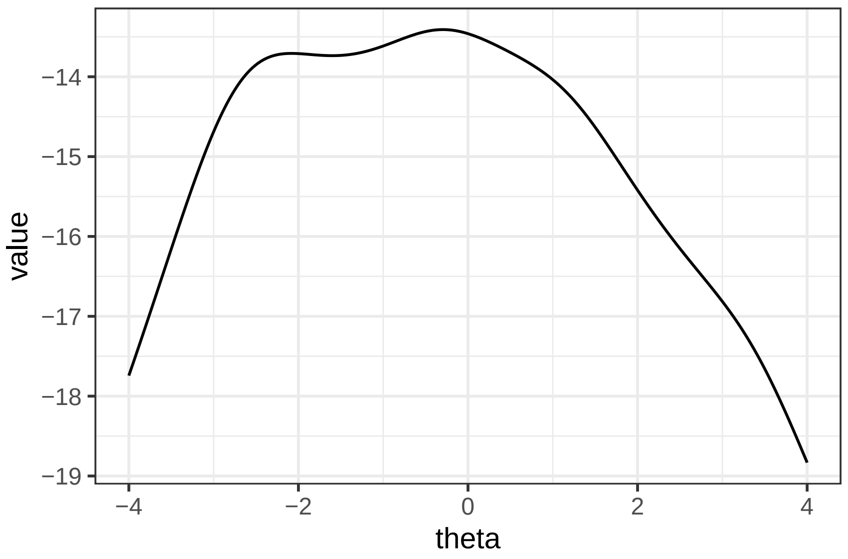

Figure 1: Log likelihood function for the given data in the range from -4 to 4.

Question

Plot the derivative in the same range and check visually how often the derivative is equal to 0.

Solution

```r
# Plot for the derivative of the likelihood function
ggplot(plot_derivative, aes(x=theta, y=value)) +
  geom_line() +
  theme_bw() +
  labs(y="value") +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)
```

From figure 2, the derivative is 0 three times, around theta = -2.1, -1.7, and -0.2.

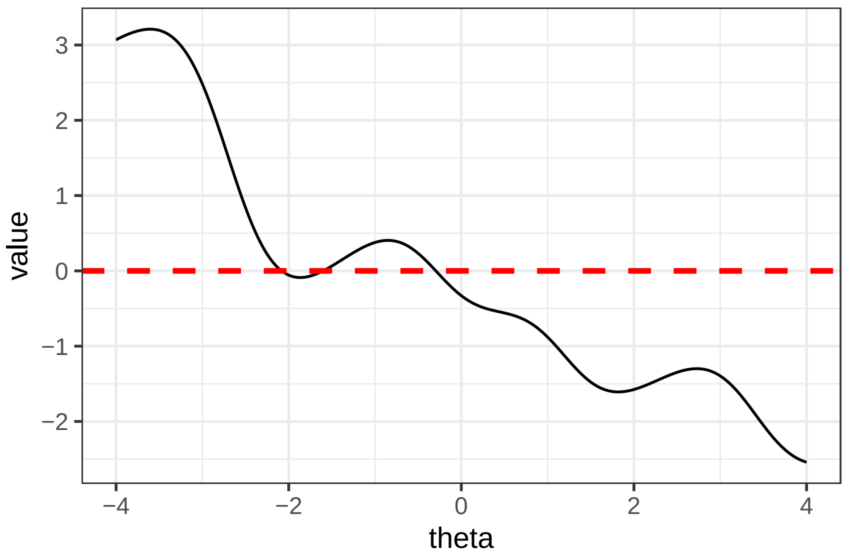

Figure 2: The derivative for the given data in the range from -4 to 4.

1.2 1 b)

Question

Choose one of the following methods: bisection, secant, or Newton-Raphson. Write your own code to optimize with the chosen method.

We have choosen the method: bisection

Solution

```
derivate_theta <- function(theta,x){
  result <- sum( (2*(x-theta)) / (1+(x-theta)^2) )
  return(result)
}

bisection <- function(a, b, x, eps = 0.00005){
  derivate_a <- derivate_theta(a,x)
  derivate_b <- derivate_theta(b,x)
  if(derivate_a * derivate_b >=0){
    stop("Criterion 1 not fulfilled")
  }
  converged = FALSE
  iter <- 1
  while(!converged){
    new_num <- (a+b)/2
    derivate_new_num <- derivate_theta(new_num,x)
    if(derivate_a * derivate_new_num <=0){
      b <- new_num
    } else {
      a <- new_num
    }

    next_num <- (a+b)/2
    if(abs(next_num - new_num)/abs(new_num) < eps){
      return(next_num)
    } else {
      new_num <- next_num
    }

    iter <- iter + 1
    # Max number of iterations is set to 1000 to prevent endless loops.
    if(iter > 1000){
      stop("The function did not converge after 1000 iterations.")
    }
  }
  return(next_num)
}


x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
```

1.3 1 c)

Question

Choose suitable starting values based on your plots to identify all local maxima of the likelihood function. Note that you need pairs of interval boundaries for the bisection or pairs of starting values for secant. Are there any (pairs of) starting values which do not lead to a local maximum? Decide which is the global maximum based on programming results.

Solution

We made three calls to the `bisection()` function with three different interval boundaries:

```
result_vec <- c()

result_vec[1] <- bisection(-3, -1.9, x)
result_vec[2] <- bisection(-1, 0, x)
result_vec[3] <- bisection(-2,-1,x)
```

In figure 3 we plot the bisection optimization to the log likelihood function to know if we identify all local maxima of the likelihood function.

```
# Calculate the log likelihood value for the three bisection optimization
Log_lik_value <- unlist(lapply(result_vec, function(theta) -5 * log(pi) - sum(log(1 + (x - theta)^2))))

# Plot for the likelihood function
ggplot(plot_likelihood, aes(x=theta, y=value)) +
  geom_line() +
  geom_vline(xintercept = result_vec ,colour="red") +
  geom_point(aes(x = result_vec[1], y = Log_lik_value[1]), colour = "red", size = 3) +
  geom_point(aes(x = result_vec[2], y = Log_lik_value[2]), colour = "red", size = 3) +
  geom_point(aes(x = result_vec[3], y = Log_lik_value[3]), colour = "red", size = 3) +
  theme_bw() +
  labs(y="value")
```

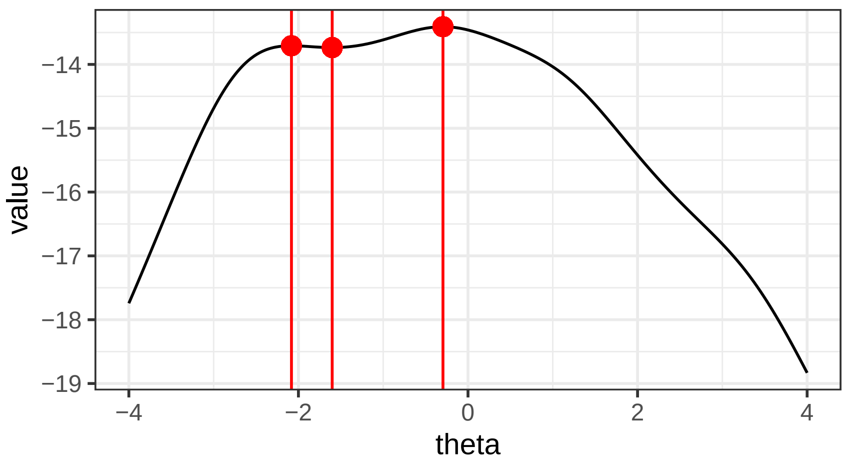

Figure 3: Log likelihood function for the given data in the range from -4 to 4 with thrre different bisection optimization pointed out in red.

We can see that all the local maxima of the likelihood function was identified, but we can also that bisection optimization identified a local minimum.

In the table 1 below we take a closer look on the Interval boundaries and the bisection optimization.

Table 1: Bisection optimization for three different interval boundaries

Interval.boundaries	Bisection.optimization	Log.likelihood.value	Local.max	Global.max
-3 & -1.9	-2.08	-13.71	TRUE	FALSE
-1 & 0	-0.30	-13.41	TRUE	TRUE
-2 & -1	-1.60	-13.74	FALSE	FALSE

The global maxima is given by the interval boundary -1 & 0, because the log likelihood value is the highest in this interval boundary.

Starting values that does not lead to local maximum are for example the interval boundaries -2 & -1 which identifies a local minimum instead of a local maxima. Other values for the interval can be 2 & 4, where there is no local maximum or minimum.

1.4 1 d)

Question

Assume now that you are in a situation where you cannot choose starting values based on a plot since the program should run automatized. How could you modify your program to identify the global maximum even in the presence of other local optima?

Solution

If we can not plot and do not know how many local maximum or minimum the function has over the interval, we can not assure that we found a global maximum. To find local optimas, we can randomize values for the intervals and apply it to our function to find the local maximums and then decide which of these has the highest value of the log-likelihood and that is our approximation for the global maximum.

When we have intervals, we thought that this would be a better way than randomizing intervals. We could implement this by testing different interval boundaries. The more interval boundaries we test, the more secure the results become, but it also increases computational effort.

Then we check if the first assumption is met $g'(a_t) \cdot g'(b_t) < 0$, and if it is met, the bisection optimization is saved in a vector.

After all different intervals have been processed, the log-likelihood value of the bisection optimization is calculated, and then the bisection optimization with the highest log-likelihood value is indexed out. This bisection optimization is then assumed to be the global maximum.

2 Question 2: Computer arithmetics (variance)

A known formula for estimating the variance based on a vector of n observations is:

$$Var(\vec{x}) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

2.1 2 a)

Question: Write your own R function, myvar, to estimate the variance in this way.

Answer: The function is implemented with the code as follows

```
myvar <- function(x){
  n <- length(x)
  var <- (1/(n-1)) * (sum(x^2) - 1/n * sum(x)^2)
  return(var)
}
```

2.2 2 b)

Question: Generate a vector $x = (x_1, ..., x_{10000})$ with 10000 random numbers with mean 10^8 and variance 1.

Answer: The numbers are generated with the code as follows

```
set.seed(13)
x <- rnorm(mean=1e+08, sd = 1, n = 10000)
```

2.3 2 c)

The whole question for this task is: For each subset $X_i = x_1, ..., x_i, i = 1, ..., 10000$ compute the difference $Y_i = myvar(X_i) - var(X_i)$, where $var(X_i)$ is the standard variance estimation function in R. Plot the dependence Y_i on i. Draw conclusions from this plot. How well does your function work? Can you explain the behaviour?

The answer is divided into two sub-questions.

Sub-question For each subset $X_i = x_1, ..., x_i, i = 1, ..., 10000$ compute the difference $Y_i = myvar(X_i) - var(X_i)$, where $var(X_i)$ is the standard variance estimation function in R. Plot the dependence Y_i on i.

Answer: We have calculated with $i = 2$ instead of $i = 1$ since an estimation of the variance needs 2 numbers. The computation and plot used for this question is as follows

```
# Empty vectors
diff <- c()
myvar_var <- c()
var_var <- c()

# Variance estimation for each subset
for(i in 2:10000){
  data <- x[1:i]

  # Estimation of the variance for myvar and var function
  myvar_var[i] <- myvar(data)
  var_var[i] <- var(data)

  diff[i] <- myvar_var[i] - var_var[i]
}

# Plot over difference between var and myvar function
plot_diff <- data.frame(num = 2:10000, value = diff[2:10000])
ggplot(plot_diff, aes(x=num, y=value)) +
  geom_point() +
  theme_bw() +
  labs(x="Subset",
       y="Difference")
```

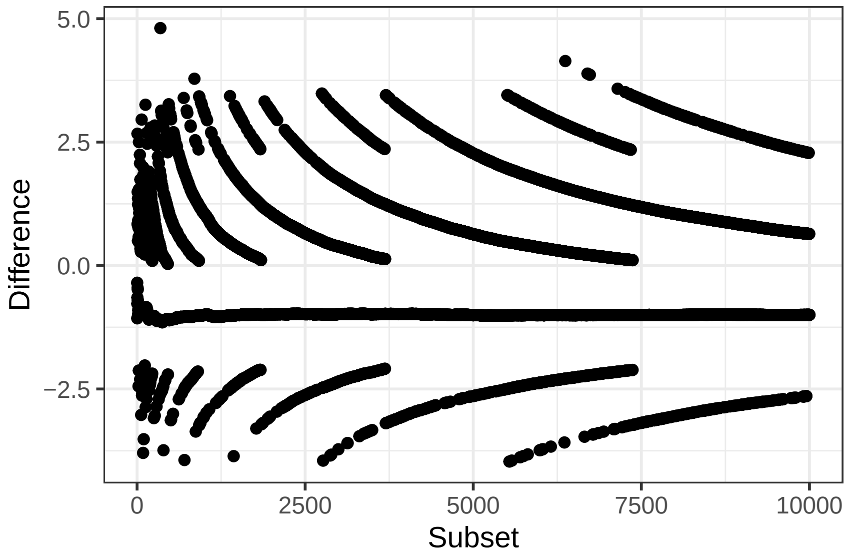

Figure 4: Difference in variance estimation between the implemented function myvar and function var for various subsets.

Sub-question: Draw conclusions from this plot. How well does your function work? Can you explain the behaviour?

Answer: In figure 4, the difference in estimation of the variance between the function myvar and var is between the range -3 to 5. The numbers are generated with a variance 1, the difference should be close to 0 if both functions estimate the variance correctly. To evaluate how well our function work, the min and max variance estimation of the function myvar and var are presented in table 1.

```r
# Removes first value which is NA for variance estimation.
myvar_var <- myvar_var[-1]
var_var <- var_var[-1]

myvar1 <- c(min(myvar_var), max(myvar_var))
var1 <- c(min(var_var), max(var_var))
table_var <- data.frame(myvar = myvar1, var = var1)
rownames(table_var) <- c("Min", "Max")
knitr::kable(table_var,
             caption = "Min and max value for the variance estimation of the functions.",
             digits=3)
```

Table 2: Min and max value for the variance estimation of the functions.

	myvar	var
Min	-2.962	0.348
Max	5.919	1.158

From table 2, the estimation of the variance for the function myvar is between -2.962 and 5.919, and for the function var the estimation is between 0.348 and 1.158. The function myvar does not work well. It has estimated the variance to be negative, which should not be possible due to the definition and in some cases the variance is estimated to be way larger than it is. To understand why the estimation of the variance is weird for the function myvar we have a closer look at the terms: $\sum_{i=1}^{n} x_i^2$ and $\frac{1}{n}(\sum_{i=1}^{n} x_i)^2$.

The term $\sum_{i=1}^{n} x_i^2$ will be around 2e+16 to 1e+20.
The term $(\sum_{i=1}^{n} x_i)^2$ will be around 4e+16 to 1e+22 and $\frac{1}{n}(\sum_{i=1}^{n} x_i)^2$ will be around 2e+16 and 1e+19.

In R "All real numbers are stored in double precision format." in the format IEEE 754[1].

Further investigation in the IEEE 754 format in a 64 bit system the bits are used as follows [2]:

- Sign bit: 1 bit
- Exponent: 11 bits
- Significand precision: 53 bits (52 bits stored)

What this means is that R can only accurately store integers up to $2^{53} \approx 9e + 15$. Integers larger than this will be rounded according to the formula[2]: Integers between 2^n and 2^{n+1} round to a multiple of 2^{n-52}.

[1] https://stat.ethz.ch/R-manual/R-devel/library/base/html/double.html
[2] https://en.wikipedia.org/wiki/Double-precision_floating-point_format

Two examples that this happens in R are presented as follows

```
options(digits=20)
# 16 ONES
1111111111111111
```

```
## [1] 1111111111111112
```

```
# 17 ONES
11111111111111111
```

```
## [1] 11111111111111104
```

We also see that numbers that are further away from what R can accurately store $(9e + 15)$ will be more inaccurate (which explained by the rounding formula). In our task, the function myvar handles numbers larger than what R can accurately store. Since the two terms $\sum_{i=1}^{n} x_i^2$ and $\frac{1}{n}(\sum_{i=1}^{n} x_i)^2$ are of equal magnitude we will have a termed called "Catastrophic Cancellation"[3]. Which in short will give us that the difference between $\sum_{i=1}^{n} x_i^2$ and $\frac{1}{n}(\sum_{i=1}^{n} x_i)^2$ will not be the actual difference but instead the difference are just the difference between the rounding of the numbers. In figure 4 we can see this happening, for subsets under 1000, there are more changes to the way the numbers are rounded happening compared to subsets above 7500.

2.4 2 d)

Question: How can you better implement a variance estimator? Find and implement a formula that will give the same results as var()?

Answer: A better implementation of a variance estimator is one where there is no rounding error of integers. Our new variance estimator is therefore: $\frac{1}{n-1} * \sum \left(x - \bar{x}\right)^2$

The function is implemented with the code as follows:

```
myvar2 <- function(x){
  n <- length(x)
  (1/(n-1)) * sum((x - mean(x))^2)
}

for(i in 2:10000){
  data <- x[1:i]
  diff[i] <- myvar2(data) - var(data)
}

# Plot over difference between var and myvar2 function
plot_diff2 <- data.frame(num = 2:10000, value = diff[2:10000])
ggplot(plot_diff2, aes(x=num, y=value)) +
  geom_point() +
  theme_bw() +
  labs(x="Subset",
       y="Difference")
```

In figure 5, the variance estimator for myvar2 is similar to var. The difference between the functions now depends on the precision for the number of decimals instead of integers.

[3]J. E. Gentle, *Computational Statistics*, 100-101

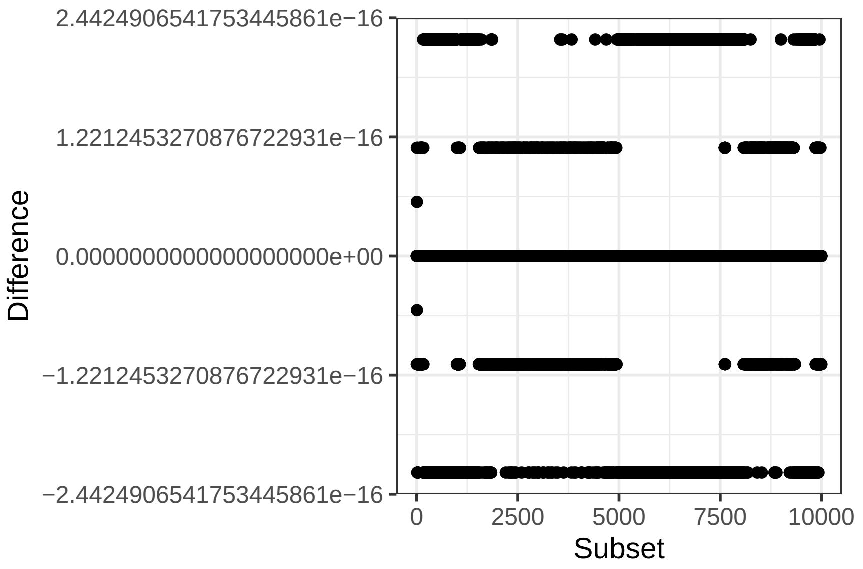

Figure 5: Difference in variance estimation between the implemented function myvar2 and function var for various subsets.

3 Statement of Contribution

We both worked on the tasks individually and helped each other when needed. We later compared and discussed our solutions before dividing the task of writing the laboration report.

3.1 Question 1

Text written by William.

3.2 Question 2

Text written by Duc.

4 Appendix

The code used in this laboration report are summarised in the code as follows:

```r
library(ggplot2)
knitr::opts_chunk$set(
  echo = TRUE,
  fig.width = 4.5,
  fig.height = 3)

library(ggplot2)
theta <- seq(-4,4,0.001)
x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
n <- 5

# Calculates the likelihood values from data
likelihood_data <- 0
for(i in 1:length(theta)){
  likelihood_data[i] <- -n * log(pi) - sum(log(1 + (x - theta[i])^2))
}

# Calculates the value of the derivative
derivative_data <- 0
for(i in 1:length(theta)){
  derivative_data[i] <- sum( (2 * (x - theta[i])) / (1 + (x - theta[i])^2) )
}

# Creates data frames to plot
plot_likelihood <- data.frame(theta, value=likelihood_data)
plot_derivative <- data.frame(theta, value=derivative_data)

# Plot for the likelihood function
ggplot(plot_likelihood, aes(x=theta, y=value)) +
  geom_line() +
  theme_bw() +
  labs(y="value")
```

```r
# Plot for the derivative of the likelihood function
ggplot(plot_derivative, aes(x=theta, y=value)) +
  geom_line() +
  theme_bw() +
  labs(y="value") +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)



derivate_theta <- function(theta,x){
  result <- sum( (2*(x-theta)) / (1+(x-theta)^2) )
  return(result)
}

bisection <- function(a, b, x, eps = 0.00005){
  derivate_a <- derivate_theta(a,x)
  derivate_b <- derivate_theta(b,x)
  if(derivate_a * derivate_b >=0){
    stop("Criterion 1 not fulfilled")
  }
  converged = FALSE
  iter <- 1
  while(!converged){
    new_num <- (a+b)/2
    derivate_new_num <- derivate_theta(new_num,x)
    if(derivate_a * derivate_new_num <=0){
      b <- new_num
    } else {
      a <- new_num
    }

    next_num <- (a+b)/2
    if(abs(next_num - new_num)/abs(new_num) < eps){
      return(next_num)
    } else {
      new_num <- next_num
    }

    iter <- iter + 1
    # Max number of iterations is set to 1000 to prevent endless loops.
    if(iter > 1000){
      stop("The function did not converge after 1000 iterations.")
    }
  }
  return(next_num)
}


x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)

result_vec <- c()

result_vec[1] <- bisection(-3, -1.9, x)
result_vec[2] <- bisection(-1, 0, x)
result_vec[3] <- bisection(-2,-1,x)



# Calculate the log likelihood value for the three bisection optimization
Log_lik_value <- unlist(lapply(result_vec, function(theta) -5 * log(pi) - sum(log(1 + (x - theta)^2))))

# Plot for the likelihood function
ggplot(plot_likelihood, aes(x=theta, y=value)) +
  geom_line() +
  geom_vline(xintercept = result_vec ,colour="red") +
  geom_point(aes(x = result_vec[1], y = Log_lik_value[1]), colour = "red", size = 3) +
  geom_point(aes(x = result_vec[2], y = Log_lik_value[2]), colour = "red", size = 3) +
  geom_point(aes(x = result_vec[3], y = Log_lik_value[3]), colour = "red", size = 3) +
  theme_bw() +
  labs(y="value")



df <-data.frame("Interval boundaries" = c("-3 & -1.9",
                                          "-1 & 0",
                                          "-2 & -1"),
                "Bisection optimization" = round(result_vec,2),
                "Log likelihood value" = round(Log_lik_value,2),
                "Local max" = c(TRUE, TRUE, FALSE),
                "Global max" = (Log_lik_value == max(Log_lik_value)))


knitr::kable(df, caption = "Bisection optimization for three different interval boundaries")


myvar <- function(x){
  n <- length(x)
  var <- (1/(n-1)) * (sum(x^2) - 1/n * sum(x)^2)
  return(var)
}
set.seed(13)
x <- rnorm(mean=1e+08, sd = 1, n = 10000)
# Empty vectors
diff <- c()
```

```
myvar_var <- c()
var_var <- c()

# Variance estimation for each subset
for(i in 2:10000){
  data <- x[1:i]

  # Estimation of the variance for myvar and var function
  myvar_var[i] <- myvar(data)
  var_var[i] <- var(data)

  diff[i] <- myvar_var[i] - var_var[i]
}

# Plot over difference between var and myvar function
plot_diff <- data.frame(num = 2:10000, value = diff[2:10000])
ggplot(plot_diff, aes(x=num, y=value)) +
  geom_point() +
  theme_bw() +
  labs(x="Subset",
       y="Difference")
# Removes first value which is NA for variance estimation.
myvar_var <- myvar_var[-1]
var_var <- var_var[-1]

myvar1 <- c(min(myvar_var), max(myvar_var))
var1 <- c(min(var_var), max(var_var))
table_var <- data.frame(myvar = myvar1, var = var1)
rownames(table_var) <- c("Min", "Max")
knitr::kable(table_var,
             caption = "Min and max value for the variance estimation of the functions.",
             digits=3)
options(digits=20)
# 16 ONES
1111111111111111

# 17 ONES
11111111111111111
myvar2 <- function(x){
  n <- length(x)
  (1/(n-1)) * sum((x - mean(x))^2)
}

for(i in 2:10000){
  data <- x[1:i]
  diff[i] <- myvar2(data) - var(data)
}

# Plot over difference between var and myvar2 function
```

```r
plot_diff2 <- data.frame(num = 2:10000, value = diff[2:10000])
ggplot(plot_diff2, aes(x=num, y=value)) +
  geom_point() +
  theme_bw() +
  labs(x="Subset",
       y="Difference")
```