Computational Statistics Lab 1

Simge Cinar & Ronald Yamashita

2023-11-01

QUESTION 1:

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,

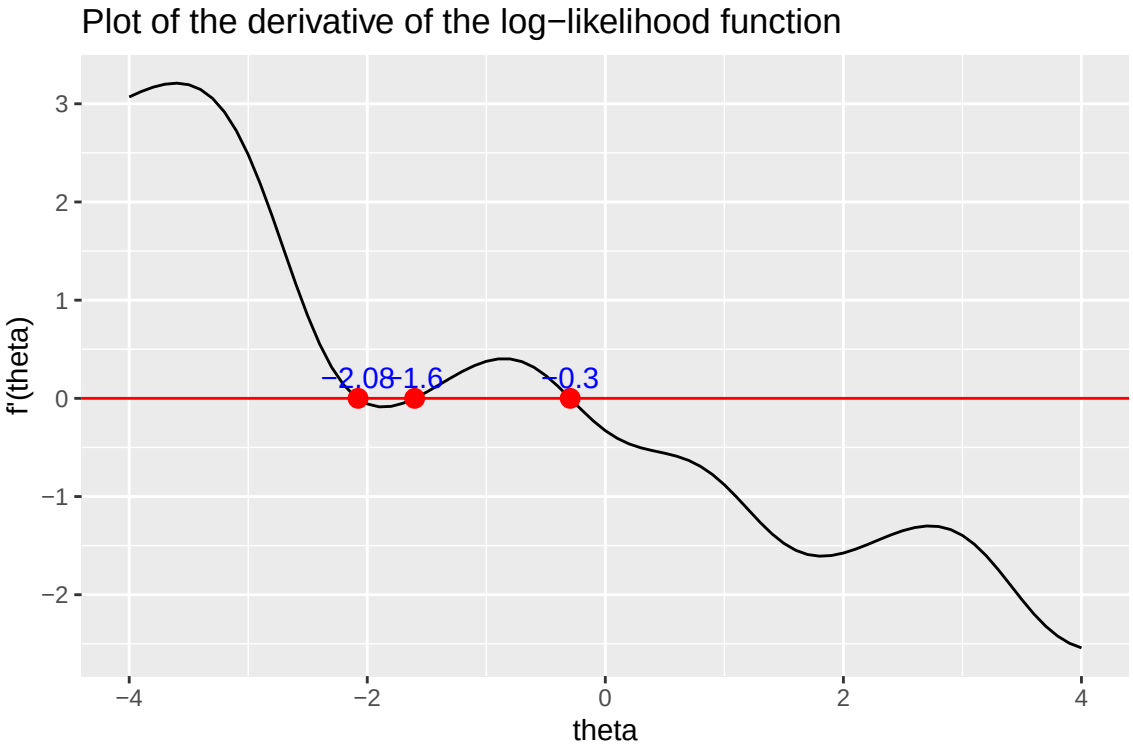
$$-n \cdot log(\pi) - \sum_{i=1}^{n} log(1 + (x_i - \theta)^2)$$

and it's derivative with respect to θ is,

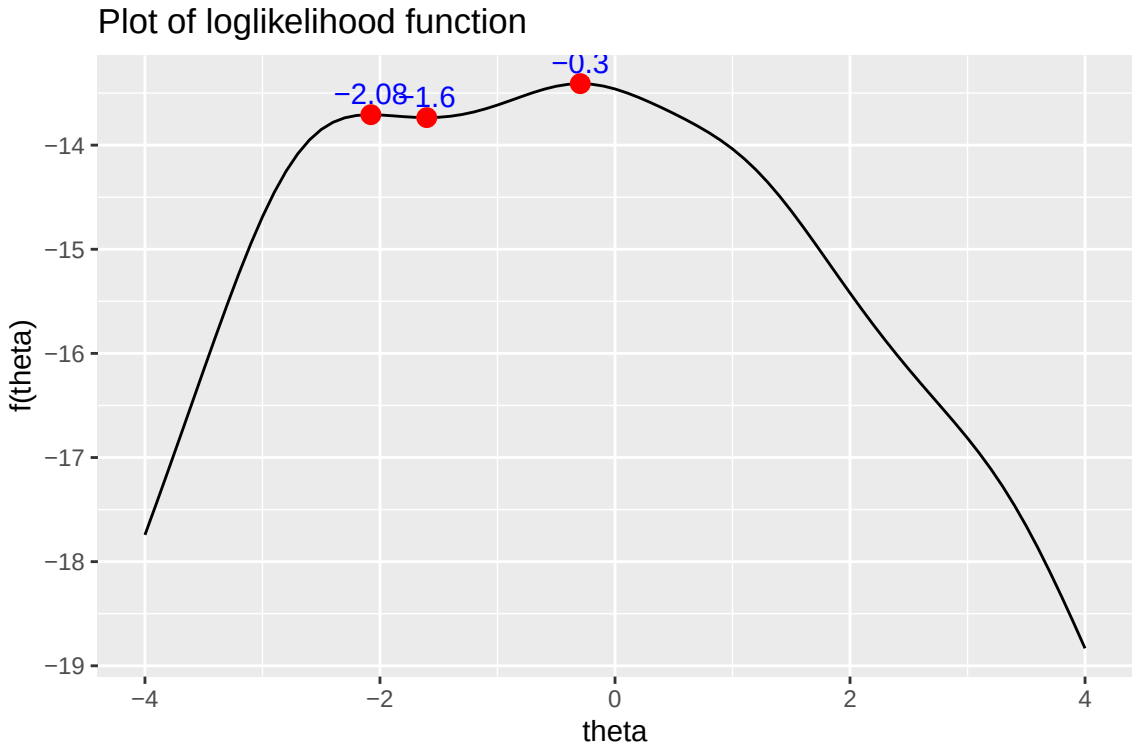
$$\sum_{i=1}^{n} \frac{2 \cdot (x_i - \theta)}{1 + (x_i - \theta)^2}$$

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

Plot of the loglikelihood function and its derivative can be seen below.



Derivative becomes zero 3 times where $\theta = -2.08$, $\theta = -1.6$ and $\theta = -0.3$



The derivative reached to global maximum in the range [-4, 4] when $\theta = -0.3$ as it can be observed from the loglikehood function graph.

Secant method will used for that part. Since the global maximum in the range [-4,4] is -0.3, $t_0 = -0.3$ and $t_1 = -0.2$ was chosen as the starting values. The function found the maximum at the third iteration.

```
## Iteration: 3
```

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

I would modify my program to identify the global maximum even in the presence of other local optima with the method below apart from looking at the plo:
I would implement the bisection method. First I checked whether the given interval, which is [-4,4], satisfies the condition $g'(-4) \cdot g'(4) < 0$ and if the condition is satisfied I divide the interval [-4,4] into two where midpoint is zero. Now I have two intervals which are [-4,0] and [0,4]. I would again check if one of the intervals $g'(-4) \cdot g'(0) < 0$ or $g'(0) \cdot g'(4) < 0$ satisfies the conditon. If both of them satisfies I would just take the first interval I checked, therefore this method doesn't guarantee optimality. I would set a limit for the number of iterations and when iterations reached that maximum number of iteration before reaching the maximum I would break the loop.
Another way to optimize the software is using optimize function R as it can be seen below.

```
# Using optimize function
my_function <- function(theta){
  x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
  n <- length(x)
  y <- -n*log(pi) - sum(log(1+(x-theta)^2))
  return(y)
}

optimize(my_function, c(-4, 4), maximum = TRUE)
```

```
## $maximum
```

```
## [1] -0.2952289
##
## $objective
## [1] -13.40942
```

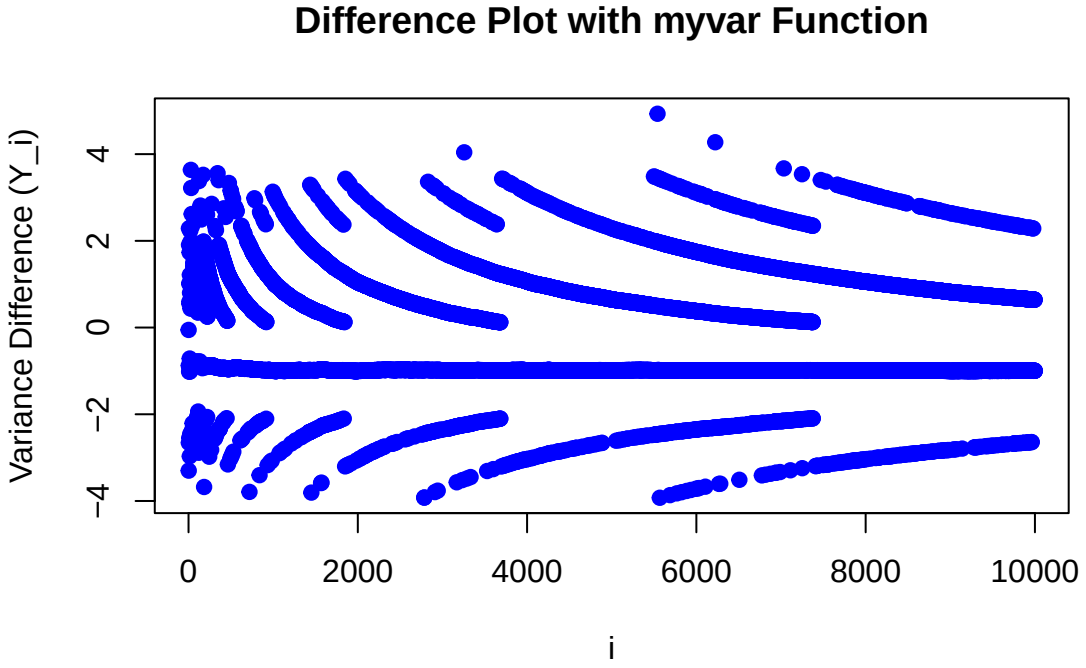
QUESTION 2:

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

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

myvar function is defined and 10,000 random numbers generated with mean 10^8 and variance 1.

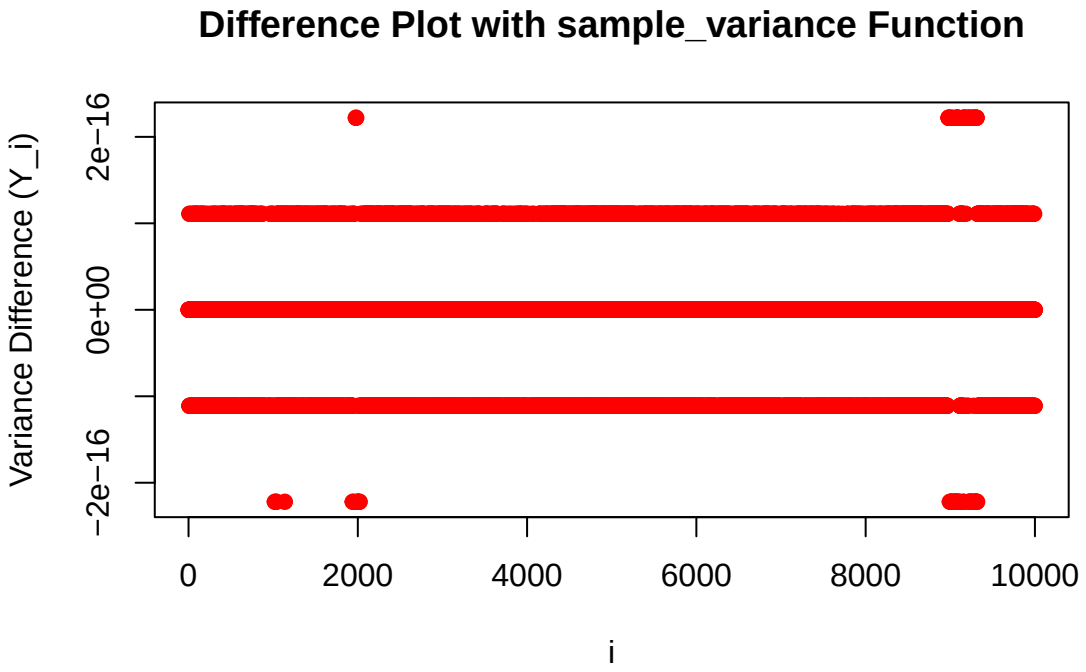
The graph for the $Y_i = myvar(X_i) - var(X_i)$ where $var(X_i)$ is the standard variance estimation function in R can be seen below.



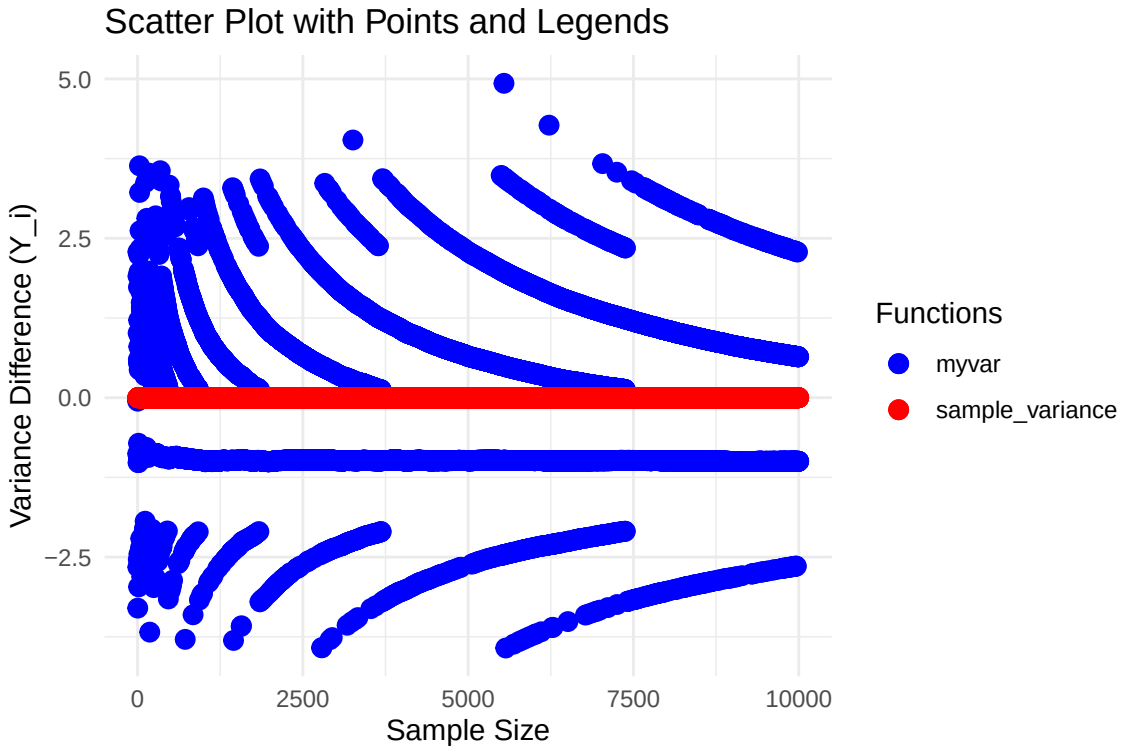
"myvar" function is not working well because difference fluctuates between -4 and 4, it is not stable. The fluctuations in the data are primarily attributed to the inherent limitations of floating-point arithmetic, which arises from the finite precision of floating-point numbers. Overflow occurs since the number we are trying to express in floating point is large in magnitude (Gentle 2009).
When we squared each of the x values and computed the sum of the squared x values and subtract them from each other, we encountered a loss of precision when compared to the sample variance formula in part (d).Let's take the number 10^8 since it is the mean. $\sum_{i=1}^{n} 10^{8^2} - \frac{1}{n}(\sum_{i=1}^{n} 10^8)^2 = 10^4 \cdot 10^{16} - \frac{10^4 \cdot 10^8}{n}$ R, tends to round such values since they are exceedingly large. The subtraction operation itself involves two extremely large numbers. On the other hand, when we use the sample variance formula, the subtractions occurs between smaller numbers, which allows for more accurate calculations and mitigates the adverse effects of limited floating-point precision.

To implement a better variance estimator, I would use sample variance formula;

$$Var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$



The comparison plot for the two methods can be seen below. The difference is almost zero with sample_variance function and it gives almost the same result with var() function in R whereas myvar FUNCTİON fluctuates because of the floating-point arithmetic.



References

Gentle, James. 2009. *Computational Statistics. International Encyclopedia of Education.* https://doi.org/10.1007/978-0-387-98144-4.

Appendix

QUESTION 1:

part a)

```r
# Define the functions
loglikelihood <- function(x, theta){
  n <- length(x)
  y <- c()
  for (theta_i in theta){
    result <- -n*log(pi) - sum(log(1+(x-theta_i)^2))
    y <- c(y, result)
  }
  return(y)
}

loglikelihood_derivative <- function(x, theta){
  n <- length(x)
  y <- c()
  for (theta_i in theta){
    result <- sum((2*(x-theta_i))/(1+(x-theta_i)^2))
    y <- c(y, result)
  }
  return(y)
}
```

```r
# Derivative graph
x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
theta <- seq(-4, 4, by = 0.1)
y2 <- loglikelihood_derivative(x, theta)

data <- data.frame(theta, y2)

# Create the plot
plot <- ggplot(data, aes(theta, y2)) +
  geom_line() +
  labs(x = "theta", y = "f'(theta)") +
  ggtitle("Plot of the derivative of the log-likelihood function")

# Add a horizontal red line at y = 0
plot <- plot + geom_hline(yintercept = 0, color = "red")

# Find the intersection points by looking for changes in sign of y2
intersection_points <- data.frame()
for (i in 2:length(y2)) {
  if (sign(y2[i]) != sign(y2[i - 1])) {
    x_intersection <- theta[i - 1] - (y2[i - 1] / (y2[i] - y2[i - 1])) * 0.1
    intersection_points <- rbind(intersection_points, data.frame(theta = x_intersection, y2 = 0))
  }
}
plot <- plot + geom_point(data = intersection_points, aes(x = theta, y = y2), color = "red", size = 3)
plot <- plot + geom_text(data = intersection_points, aes(x = theta, y = 0, label = round(theta, 2)), vj

#plot
```

```r
x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
theta <- seq(-4, 4, by = 0.1)
y1 <- loglikelihood(x,theta)

data <- data.frame(theta, y1)
plot <- ggplot(data, aes(theta, y1)) +
  geom_line() +
  labs(x = "theta", y = "f(theta)") +
  ggtitle("Plot of loglikelihood function")


y3 <- loglikelihood(x, intersection_points$theta)
plot <- plot + geom_point(data = intersection_points, aes(x = theta, y = y3), color = "red", size = 3)
plot <- plot + geom_text(data = intersection_points, aes(x = theta, y = y3, label = round(theta, 2)), v
#plot
```

part b,c)

```r
# Define function for the secant method
secant <- function(t0, t1, epsilon, max_iter){
  iter <- 0
  while (iter <= max_iter){
    d_t1 <- loglikelihood_derivative(x, t1)
    d_t0 <- loglikelihood_derivative(x, t0)
    t2 <- t1 - (d_t1*((t1 - t0)/(d_t1 - d_t0)))
    if (abs(t2 - t1) < epsilon){
      cat("Iteration:", iter, "\n")
      return(t2)
    }
    t0 <- t1
    t1 <- t2
    iter <- iter + 1
  }
  stop("Secant method did not converge within the specified number of iterations.")
}
```

```r
t0 <- -0.3
t1 <- -0.2
epsilon <- 0.000000000005
max_iter <- 10000
secant(t0, t1, epsilon, max_iter)
```

```
## Iteration: 3
```

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

part d)

```r
# Using optimize function
my_function <- function(theta){
  x <- c(-2.8, 3.4, 1.2, -0.3, -2.6)
  n <- length(x)
  y <- -n*log(pi) - sum(log(1+(x-theta)^2))
  return(y)
```

```
}

optimize(my_function, c(-4, 4), maximum = TRUE)
```

```
## $maximum
## [1] -0.2952289
##
## $objective
## [1] -13.40942
```

QUESTION 2:

part a)

```
myvar <- function(x) {
  n <- length(x)
  sum_sq <- sum(x^2) # Define the sum of squares of x
  sum_x <- sum(x)  # Define the sum of x
  variance_x <- (sum_sq - (1/n) * sum_x^2) / (n - 1)
  return(variance_x)
}
```

part b)

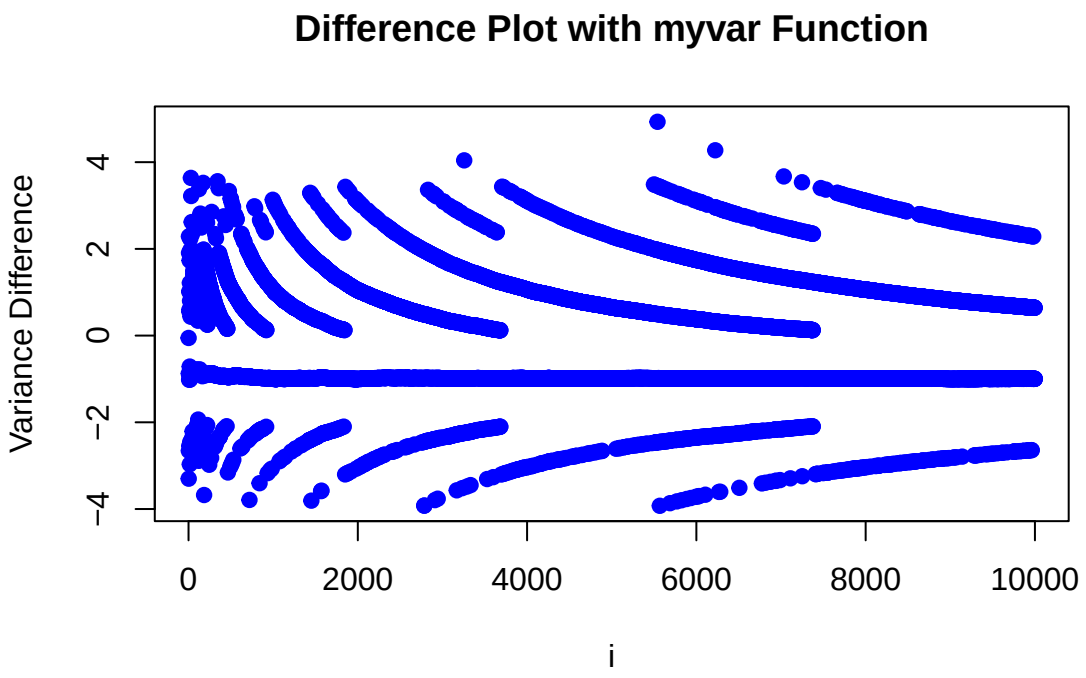
```
# Define the given mean and variance
given_mean <- 10^8
given_variance <- 1

# Calculate the standard deviation
std_dev <- sqrt(given_variance)

# Generate the vector x with 10,000 random numbers
set.seed(123)
n <- 10000
x <- rnorm(n, given_mean, std_dev)
```

part c)

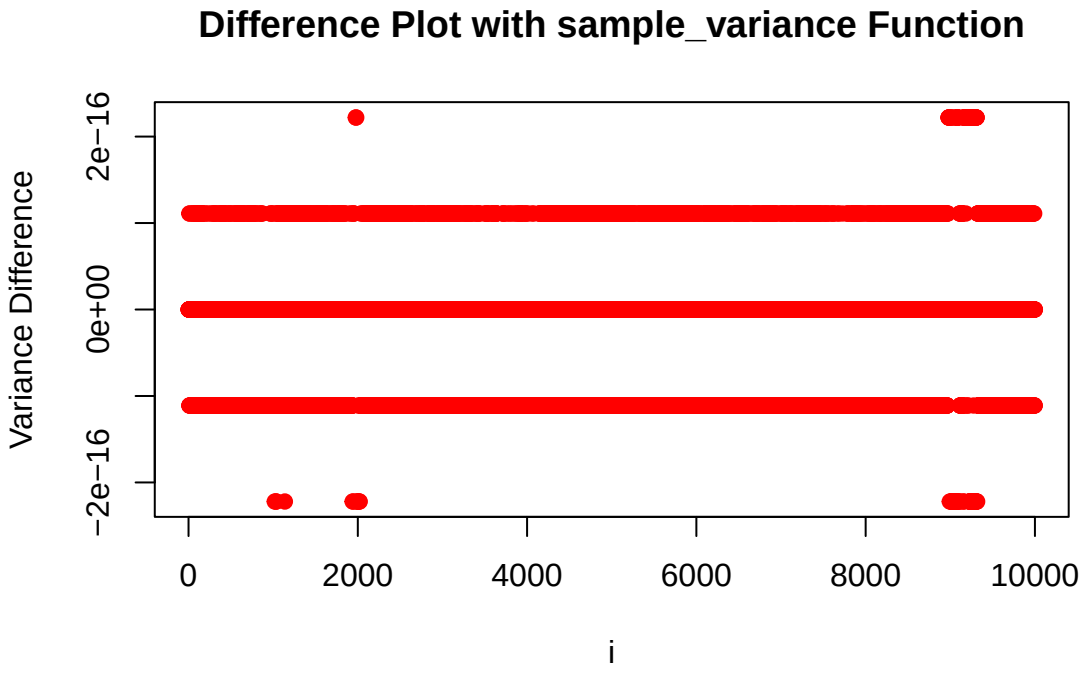
```
y1 <- c()
for (i in 2:length(x)){
  x_i <- x[1:i]
  y_i <- myvar(x_i) - var(x_i)
  y1 <- c(y1, y_i)
}
plot(y1, type = "p", col = "blue", pch = 19, xlab = "i", ylab = "Variance Difference", main = "Differen
```



part d)

```
y2 <- c()
for (i in 2:length(x)){
  x_i <- x[1:i]
  y_i <- sample_variance(x_i) - var(x_i)
  y2 <- c(y2, y_i)
}

plot(y2, type = "p", col = "red", pch = 19, xlab = "i", ylab = "Variance Difference", main = "Differenc
```



```r
x_df <- seq(2, length(x), by = 1)
df <- data.frame(x_df = x_df, 'myvar' = y1, 'sample_variance' = y2)

ggplot(df, aes(x = x_df)) +
  geom_point(aes(y = y1, color = "myvar"), size = 3) +
  geom_point(aes(y = y2, color = "sample_variance"), size = 3) +
  scale_color_manual(values = c("myvar" = "blue", "sample_variance" = "red"), name = "Functions") +
  labs(
    title = "Scatter Plot with Points and Legends",
    x = "Sample Size",
    y = "Variance Difference (Y_i)"
  ) +
  theme_minimal()
```