CompStatLab1

Xuan Wang & Priyarani Patil

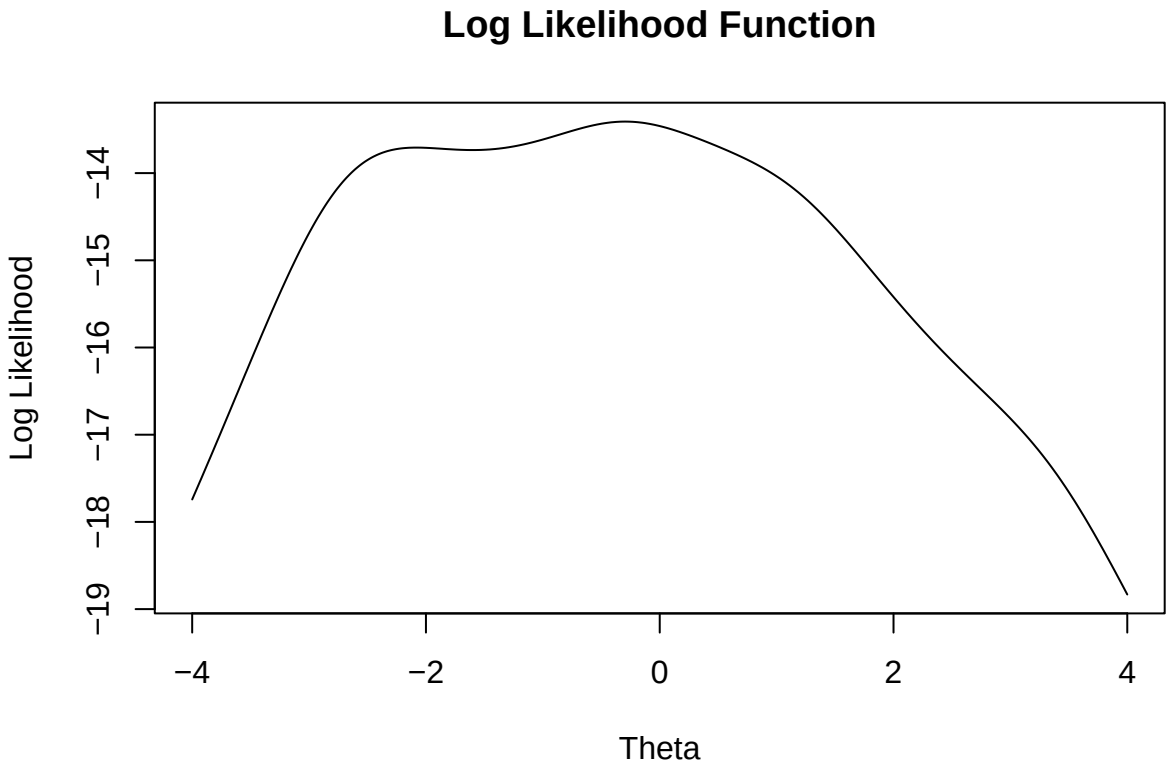
2023-11-01

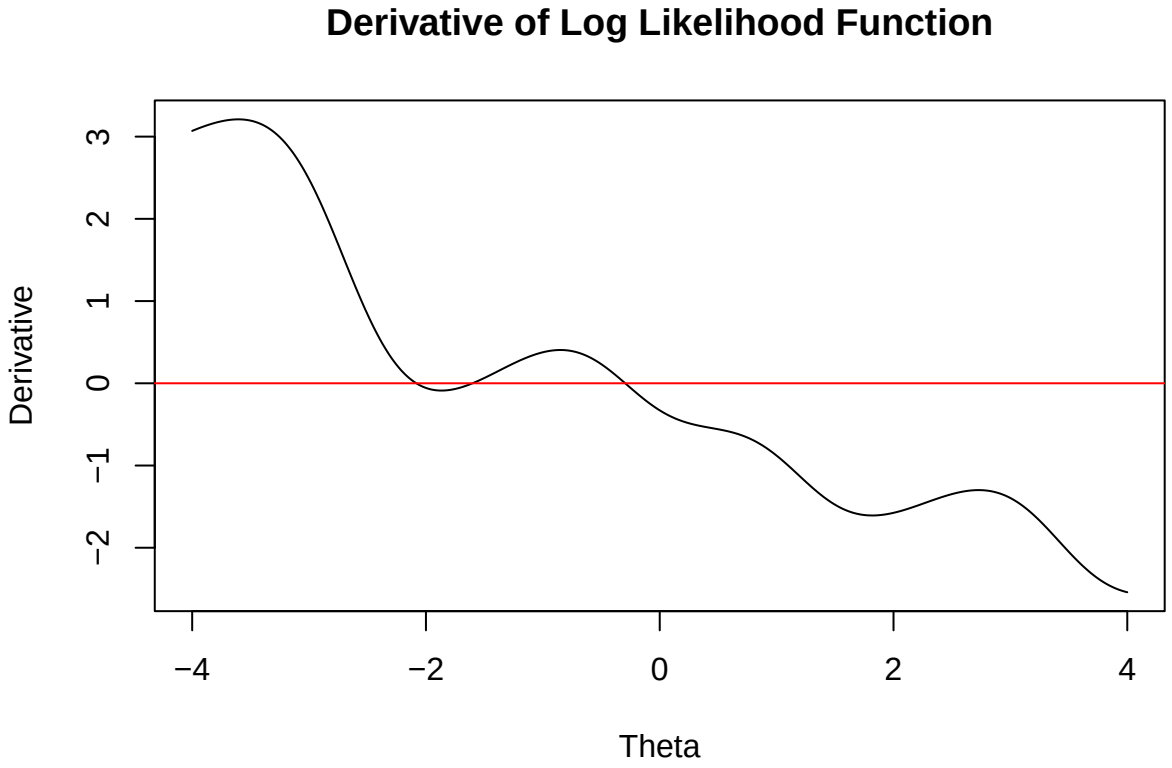
Computational Statistics 732A90 | Computer Lab 1

Question 1: Maximization of a likelihood function in one variable

Answered.

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.





b.Choose one of the following methods: bisection, secant, or Newton-Raphson. Write your own code to optimize with the chosen method. If you have chosen Newton-Raphson, describe how you derived the second derivative.

We chose Newton_Raphson, the second derivative is calculated by using this derivative function:g(x)/f(x) = f (x)g(x)−f(x)g (x)/(g (x))^2, then it can get that the second derivative is:

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

c.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.

By inspecting the log likelihood function plot, it's visually identified the local maxima, therefore, we chose starting_values = (0, -1, -2, -1.5) to do optimization and got the different local maximum. And it's also found that Newton-Raphson method might not converge to a local maximum for some starting values, especially if the starting value is chosen near a point where the derivative of the log-likelihood function is near to zero but the second derivative is positive(indicating a local minimum rather than a maximum). Therefore, it's important to choose starting values near the peaks of of the log-likelihood function when using the Newton-Raphson method.

The global maximum is the highest peak in the log likelihood function plot, according the result, when initial theta is 0, the global maximum is -13.41304.

```
newton_raphson(x, inital_theta = 0)
```

```
## [1] -0.3722976
```

```
newton_raphson(x, inital_theta = -1)
```

```
## [1] -2.07893
```

```
newton_raphson(x, inital_theta = -1.5)
```

```
## [1] -1.595163
```

```
newton_raphson(x, inital_theta = -2)
```

```
## [1] -2.11249
```

```
newton_raphson(x, inital_theta = 2)
```

```
## [1] 6.822643
```

```
log_likelihood(x, newton_raphson(x, inital_theta = 0))
```

```
## [1] -13.41304
```

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

We are using multi-start strategy, it involves running the Newton_raphson method multiple times from different randomly chosen starting points and then select the best solution.

```
## The theta value that gives the global maximun of the log_likelihood function is:  -0.295239
## Global maximun of the log_likelihood function is:  -13.40942
```

Question 2: Computer arithmetics (variance)

Answered.

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

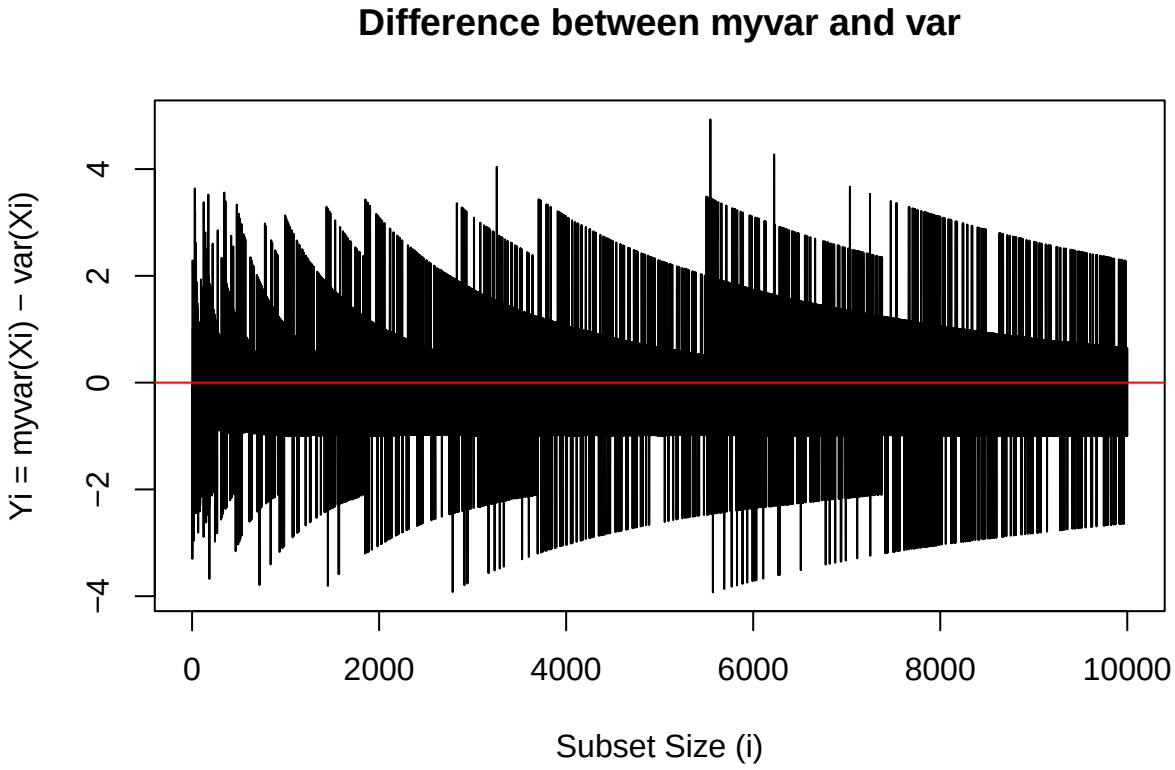
Please see the Appendix code file estimate_variance.R.

b.Generate a vector x = (x1, . . . , x10000) with 10000 random numbers with mean 10^8 and variance 1.

Please see the Appendix code file estimate_variance.R.

c.For each subset Xi = {x1, . . . , xi}, i = 1, . . . , 10000 compute the difference Yi = myvar(Xi) − var(Xi), where var(Xi) is the standard variance estimation function in R. Plot the dependence Yi on i. Draw conclusions from this plot. How well does your function work? Can you explain the behaviour?

The plot below clearly shows significant fluctuations in Yi, ranging from -4 to 4. This variation can be attributed to “loos of precision”, a phenomenon tha occurs when cumulative sums of large numbers are added to smaller ones or when subtracting numbers of similar magnitudes. Consequently, it is understood that myvar() is not a reliable function based on these obaservations.

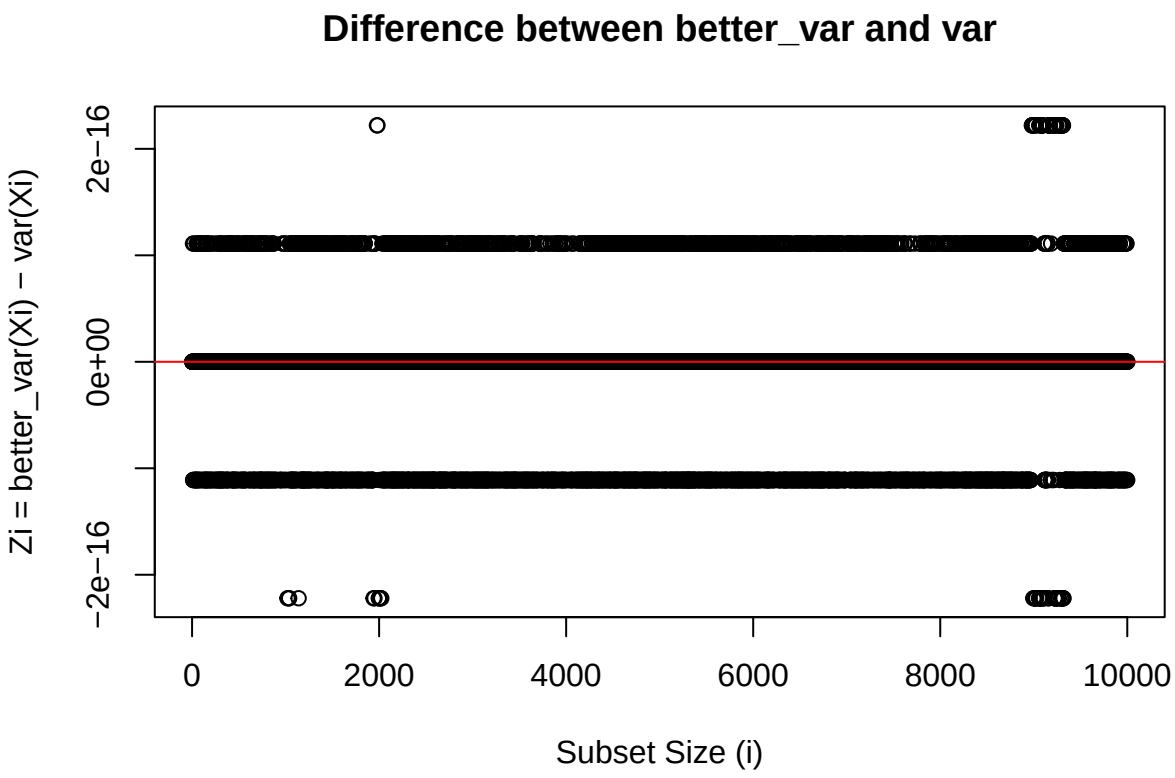


d.How can you better implement a variance estimator? Find and implement a formula that will give the same results as var()?

To better implement a variance estimator, it's better to use the original variance formula:

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

By observing below new plot for the difference Zi between better_var and var, it's clearly that difference Zi is close to zero for most subsets, which means it performs better than myvar



Appendix:

newton_raphson.R

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

# Define the range of theta
theta <- seq(-4, 4, length.out = 1000)

# Define the log likelihood function
log_likelihood <- function(x, theta) {
  n <- length(x)
  return(-n * log(pi) - sum(log(1 + (x - theta)^2)))
}

# First derivative
derivative <- function(x, theta) {
  return(sum(2 * (x - theta) / (1 + (x - theta)^2)))
}

# Second derivative
derivative_second <- function(x, theta) {
  return(sum(2 * ((x - theta)^2 - 1) / (1 + (x - theta)^2)^2))
}

# Calculate the log likelihood and its derivative for each theta
log_likelihood_values <- sapply(theta, log_likelihood, x = x)
derivative_values <- sapply(theta, derivative, x = x)
```

```r
# Plot the log likelihood function
plot(theta, log_likelihood_values, type = "l", main = "Log Likelihood Function", xlab = "Theta", ylab =
abline(h = 0, col = "red")

# Plot the derivative of the log likelihood function
plot(theta, derivative_values, type = "l", main = "Derivative of Log Likelihood Function", xlab = "Theta
abline(h = 0, col = "red")

# Newton Raphson function
newton_raphson <- function (x, inital_theta) {
  tol <- 1e-6
  new_theta <- inital_theta
  #old_theta <- inital_theta + 2
  #while ( abs(new_theta - old_theta) > tol) {
  #  old_theta <- new_theta
  #  new_theta <- old_theta - derivative(x, old_theta) / (derivative_second(x, old_theta) + 1e-6)
  #}
  ## Only perform 10000 loop to avoid the infinite loop when using while function
  for (i in 1:10000) {
    old_theta <- new_theta
    new_theta <- old_theta - derivative(x, old_theta) / (derivative_second(x, old_theta) + 1e-8)
    if ( abs(new_theta - old_theta) > tol ) {
      break
    }
  }
  #cat("The estimated theta is:", new_theta, "\n")
  return(new_theta)
}
newton_raphson(x, inital_theta = 0)

newton_raphson(x, inital_theta = -1)

newton_raphson(x, inital_theta = -1.5)

newton_raphson(x, inital_theta = -2)

newton_raphson(x, inital_theta = 2)

log_likelihood(x, newton_raphson(x, inital_theta = 0))
multi_start_strategy <- function(x, num_starts, lower_bound, upper_bound) {
  set.seed(123)
  # Generate random starting values
  theta_initial_values <- seq(lower_bound, upper_bound, length.out = num_starts)
  # Apply newton_raphson method to each starting values
  theta_opt_values <- sapply(theta_initial_values, newton_raphson, x = x)

  # Evaluate the log_likelihood function at each optimized theta value
  log_likelihood_opt_values <- sapply(theta_opt_values, log_likelihood, x = x)

  # Find the theta value that gives the maximum log-likelihood
  theta_global_max <- theta_opt_values[which.max(log_likelihood_opt_values)]
  log_likelihood_max <- log_likelihood(x, theta_global_max)
  cat("The theta value that gives the global maximun of the log_likelihood function is: ", theta_global
```

```
  cat("Global maximun of the log_likelihood function is: ", log_likelihood_max,"\n")
}
# Call the multi_start_strategy function
multi_start_strategy(x, num_starts = 100, lower_bound = -4, upper_bound = 4)
```

estimate_variance.R

```
myvar <- function(X) {
  if ( !is.vector(X) ) {
    stop("Argument x should be a vector of n observations")
  }
  n <- length(X)
  if ( n < 1) {
    stop("Argument X's size must be greater than 1 for variance estimation.")
  }

  sum_x_squared <- sum(X^2)
  sum_x <- sum(X)

  estimate_variance <- (1 / (n-1) ) * (sum_x_squared - ( 1 / n ) * sum_x^2)

  return(estimate_variance)
}
# Generate a vector x = (x1, . . . , x10000)
set.seed(123)
X <- rnorm(n = 10000, mean = 100000000, sd = sqrt(1))

# Compute the difference between myvar() and var()
Y <- numeric(10000)
for ( i in 1:length(X)) {
  diff <- myvar(X[1:i]) - var(X[1:i])
  Y[i] <- diff
}

# Plot the dependence Y on i
n <- 1:10000
plot(n, Y, type = "l", main = "Difference between myvar and var", xlab = "Subset Size (i)", ylab = "Yi =
abline(h = 0, col = "red")
# better implement a variance estimater using original formula
better_var <- function(X) {
  if ( !is.vector(X) ) {
    stop("Argument x should be a vector of n observations")
  }
  n <- length(X)
  if ( n < 1) {
    stop("Argument X's size must be greater than 1 for variance estimation.")
  }

  mean_x <- mean(X)
  estimate_variance <- sum((X - mean_x)^2) / (n-1)

  return(estimate_variance)
}
```

```r
# Compute the difference between myvar() and var()
Z <- numeric(10000)
for ( i in 1:length(X)) {
  Z[i] <- better_var(X[1:i]) - var(X[1:i])
}

# Plot the Difference Z on i
plot(1:10000, Z, pch = 1, main = "Difference between better_var and var", xlab = "Subset Size (i)", ylab
abline(h = 0, col = "red")
```