Lab1

Cui Qingxuan, Nisal Amashan

2025-01-28

Contents

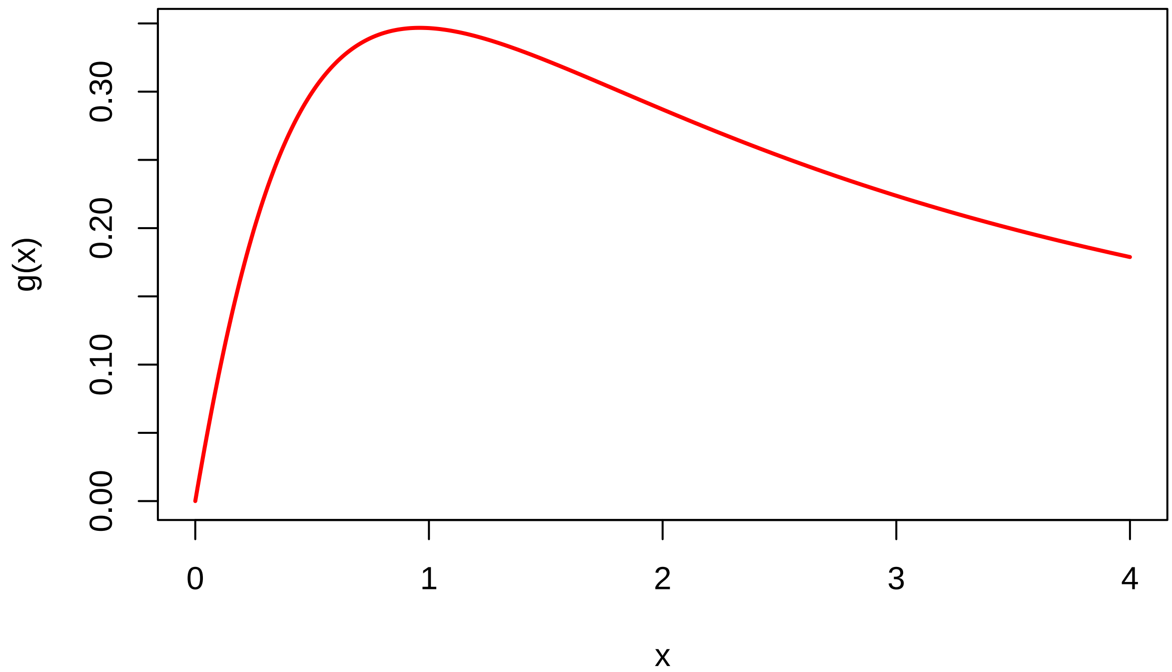
Collaborations:

Cui Qingxuan: Responsible for the question 1.

Nisal Amashan: Responsible for the question 2.

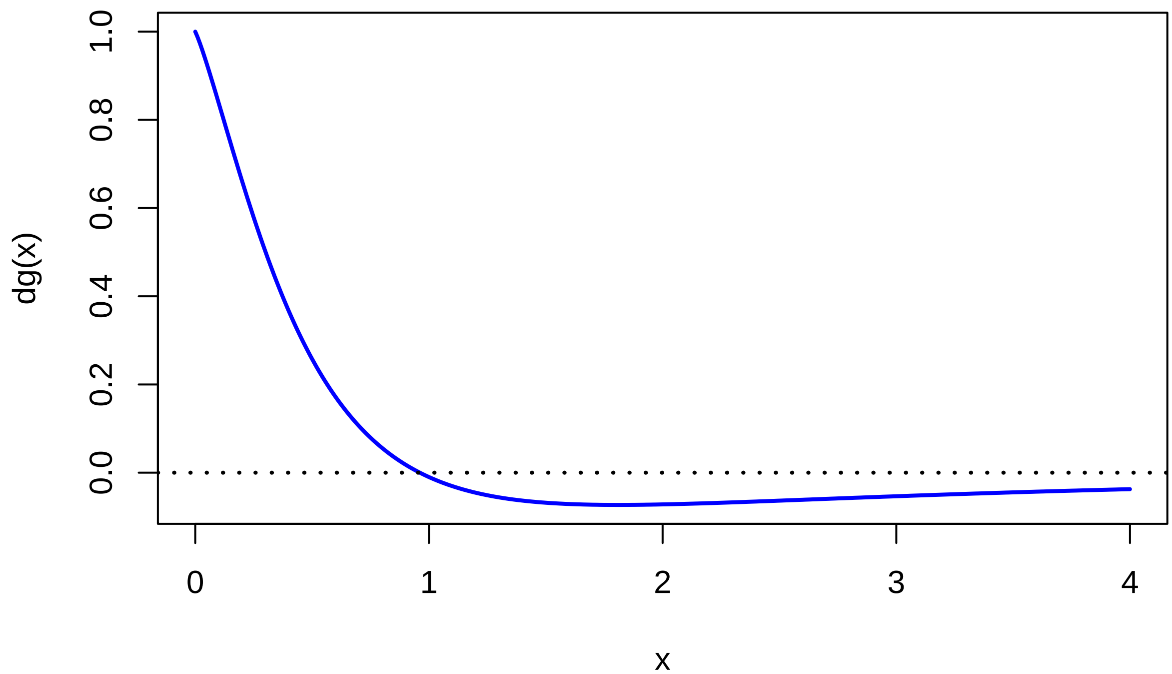
Question 1

Plot the function and guess maximum



I guess the maximum point would be 0.9 approximately.

Plot g'(x)



Implement bisection method based on user input

```
## User input:
##  a:  0.8
##  b:  1
##  Criterion:  0.001

##
## The estimated maximum point using bisection method is: 0.96133
```

Implement secant method based on user input

```
## User input:
##  start:  0.8
##  Criterion:  0.001

##
## The estimated maximum point using secant method is: 0.96028
```

Run the functions for different starting intervals/pairs of starting values

Table 1: Bisection Method Results

a	b	Estimated_Maximum
0.5	0.9	Not Found
0.6	1.0	0.96133
0.7	1.1	0.96133
0.8	1.2	0.96133
0.9	1.3	0.96133
1.0	1.4	Not Found

```
## When xt = -0.8190639 xt-1 = 1.6 , the second order derivative is Nan.
## When xt = -3.036705 xt-1 = 1.7 , the second order derivative is Nan.
## When xt = -13.31529 xt-1 = 1.8 , the second order derivative is Nan.
```

Table 2: Secant Method Results

Starting_Point	Estimated_Maximum
0.5	0.96087
0.6	0.96103
0.7	0.96085
0.8	0.96028
0.9	0.96104
1.0	0.96084
1.1	0.96109
1.2	0.96096
1.3	0.96126
1.4	0.96103
1.5	0.96141
1.6	Not Found
1.7	Not Found
1.8	Not Found
1.9	Not Found
2.0	Not Found

Question 2

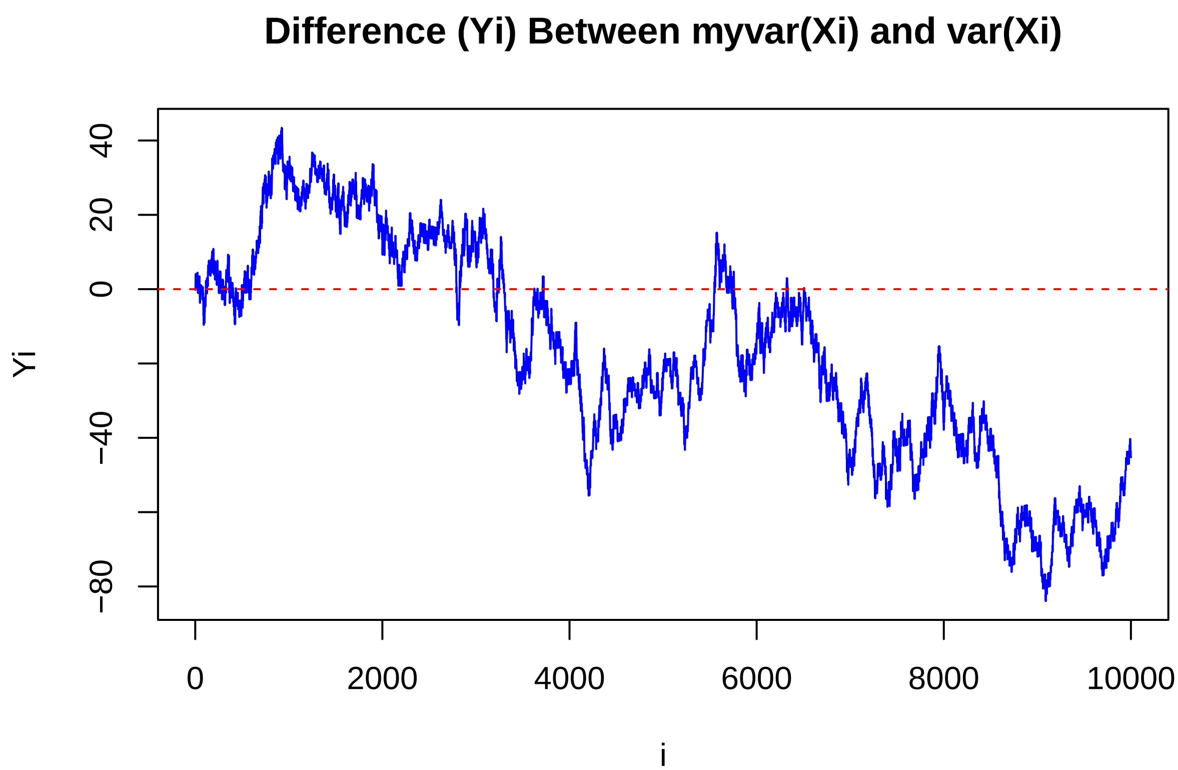
Custom myvar function to estimate the variance

```
myvar = function(data) {
  n = length(data)
  sq_sum = sum(data^2)
  sum_sq = sum(data)^2
  var = (1/(n-1))*(sq_sum - (sum_sq/n))
  return(var)
}
```

Generate a vector x = (x1, . . . , x10000) with 10000 random numbers with mean 108 and variance 1

```r
n= 10000
data = rnorm(n, mean = 10**8, sd = 1)
```

Plot the differance between variance caculated using standerd variance estimation function and myvar() function



The variance of a dataset x with n elements is given by:

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

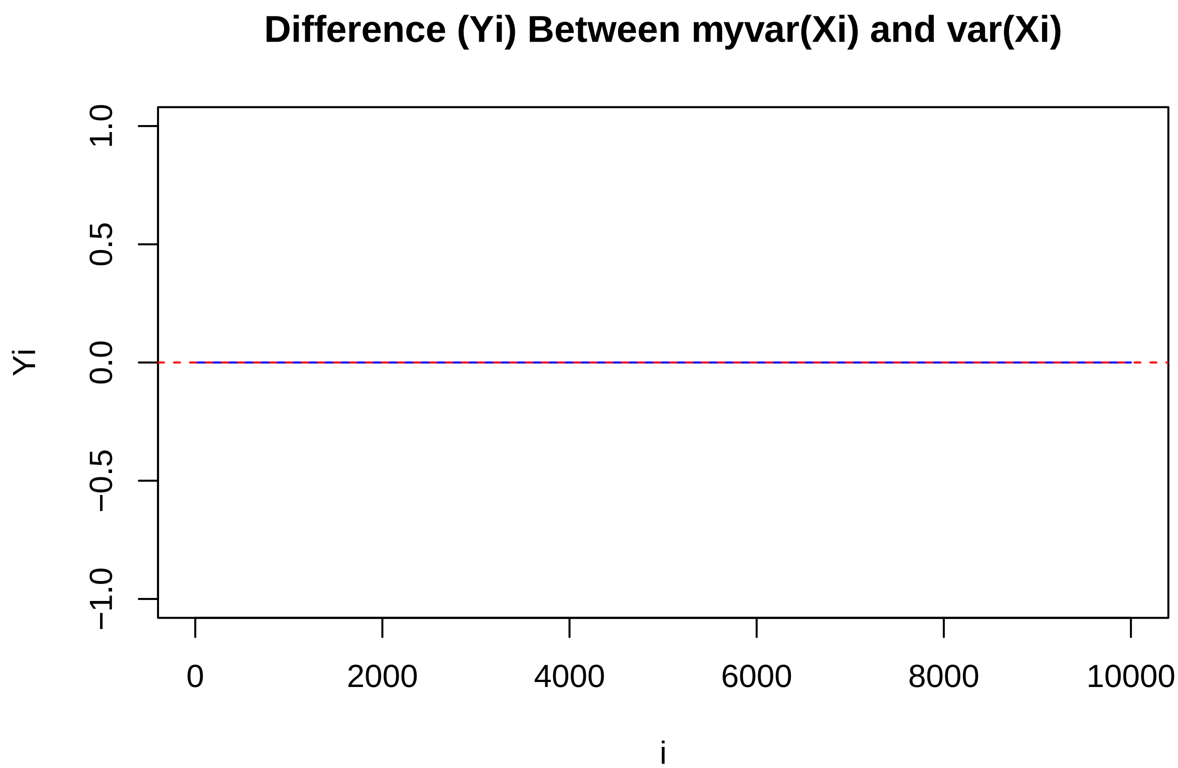
This formula can cause numerical instability when x has very large values.

The function `myvar` suffers from numerical instability due to how floating-point arithmetic handles large numbers. The term $\sum x_i^2$ and $(\sum x_i)^2/n$ can both be very large when x has a high mean. Subtracting these large values leads to a loss of precision. This means small variations in the data can result in significant errors when computing the variance.

Also, the loss of precision grows as more data points are added, which explains why the error fluctuates as shown in the plot. The numerical errors are more noticeable when dealing with large values, making this formula unreliable in such cases.

Improved myvar function to estimate the variance precisely

```
myvar_improved = function(data) {
  n = length(data)
  mu = mean(data)
  var = sum((data-mu)^2)/(n-1)
  return(var)
}
```



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

The improved function produces nearly identical results to R's built-in `var()` function. This is because it computes variance using deviations from the mean, avoiding large intermediate values that lead to precision loss.

Appendix

```
# Question 1

g = function(x){
```

```r
  gx = log(x + 1) / (x ^ 1.5 + 1)
  return(gx)
}

dg = function(x){
  dux = 1 / (x+1)
  ux = log(x+1)
  dvx = 1.5 * sqrt(x)
  vx = x ^ 1.5 + 1
  dg = (dux*vx - dvx*ux) / (vx ^ 2)
  return(dg)
}

estimate_d2g = function(xt, xt_1){
  return((dg(xt) - dg(xt_1)) / (xt - xt_1))
}


bisection = function(a, b, threshold = 0.001) {
  est_list = vector("list", length(a))

  for (i in seq_along(a)) {
    left = a[i]
    right = b[i]

    while (abs(left - right) > threshold) {
      mid = (left + right) / 2
      dg_mid = dg(mid)
      dg_left = dg(left)
      dg_right = dg(right)


      if (dg_left * dg_mid <= 0) {
        right = mid
      } else if (dg_right * dg_mid < 0) {
        left = mid
      }
      else{
        answer = 'Not Found'
        break
      }


      if (abs(left - right) <= threshold) {
        answer = round((left + right) / 2, 5)
        break
      }
    }

    est_list[[i]] = answer
  }

  return(unlist(est_list))
```

```r
}


secant = function(start, threshold) {

  xt1_list = vector("list", length(start))
  for (x0 in start) {
    xt = x0
    xt_1 = x0 - 0.1
    while (TRUE) {
      d2g = tryCatch({
        estimate_d2g(xt, xt_1)
      }, warning = function(w) {
        cat("When xt =",xt,"xt-1 =",xt_1,", the second order derivative is Nan.\n")
        return(1)
      })

      if (is.nan(d2g) || d2g == 0) {
        xt = 'Not Found'
        break
      }


      if(d2g == 1){
        xt = 'Not Found'
        break
      }
      xt1 = xt - dg(xt) / d2g

      if (abs(xt1 - xt) < threshold) {
        break
      }
      xt_1 = xt
      xt = xt1
    }
    if (is.numeric(xt)) {
      xt1_list[[which(start == x0)]] = round(xt, 5)
    } else {
      xt1_list[[which(start == x0)]] = xt
    }

  }
  return(unlist(xt1_list))
}


x = seq(from = 0, to = 4, by = 0.01)
gx = g(x)
plot(x=x, y=gx, type="l", col = "red",lwd = 2, ylab = "g(x)")


plot(x=x, y=dg(x), type = "l", col = "blue",lwd = 1)
abline(a = 0, b = 0, lwd = 1, lty = 3)
```

```r
a = 0.8
b = 1.0
threshold = 0.001
cat("User input: \n a: ", a, "\n b: ",b, "\n Criterion: ", threshold)
b_op = bisection(a, b, threshold)
cat("\nThe estimated maximum point using bisection method is:", b_op)
```

```r
start = 0.8
cat("User input: \n start: ", start, "\n Criterion: ", threshold)
s_op = secant(start, threshold)
cat("\nThe estimated maximum point using secant method is:", s_op)
```

```r
a_vec = seq(from = 0.5, to = 1.0, by = 0.1)
b_vec = seq(from = 0.9, to = 1.4, by = 0.1)
start_vec = seq(from = 0.5, to = 2, by = 0.1)
cat("Bisection Method \n")
bisection(a = a_vec, b = b_vec, threshold = threshold)
cat("Secant Method \n")
secant(start = start_vec, threshold = threshold)
```

```r
# Question 2

set.seed(12345)
myvar = function(data) {
  n = length(data)
  sq_sum = sum(data^2)
  sum_sq = sum(data)^2
  var = (1/(n-1))*(sq_sum - (sum_sq/n))
  return(var)
}

n= 10000
data = rnorm(10000, mean = 10**8, sd = 1)

Y = numeric(10000)

for (i in 2:10000) {
  custom_var = myvar(data[1:i])
  actual_var = var(data[1:i])
  diff = custom_var - actual_var
  Y[i] = diff
}

Y
```

```r
plot(2:n, Y[2:n], type = "l", col = "blue", xlab = "i", ylab = "Yi",
     main = "Difference (Yi) Between myvar(Xi) and var(Xi)")
abline(h = 0, col = "red", lty = 2)

myvar_improved = function(data) {
  n = length(data)
  mu = mean(data)
  var = sum((data-mu)^2)/(n-1)
  return(var)
}


for (i in 2:10000) {
  custom_var = myvar_improved(data[1:i])
  actual_var = var(data[1:i])
  diff = custom_var - actual_var
  Y[i] = diff
}

plot(2:n, Y[2:n], type = "l", col = "blue", xlab = "i", ylab = "Yi",
     main = "Difference (Yi) Between myvar(Xi) and var(Xi)")
abline(h = 0, col = "red", lty = 2)
```