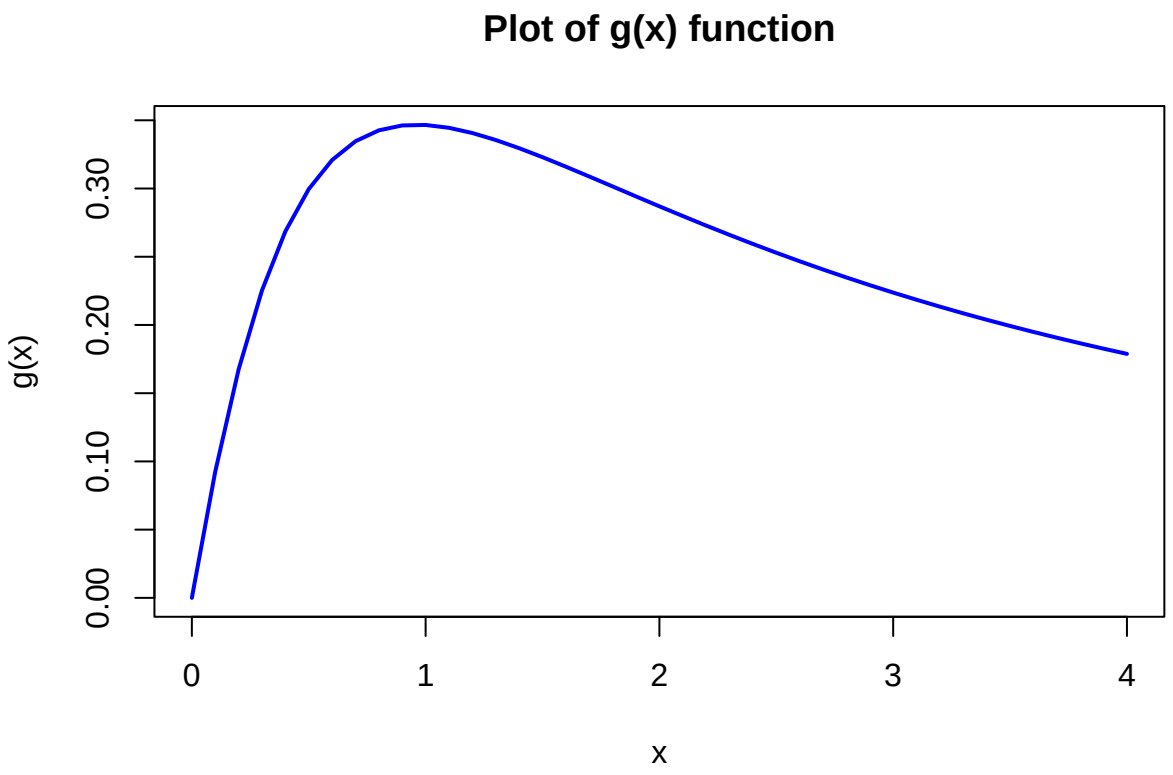
Lab 1 Group 8

Liuxi Mei Xiaochen Liu

2025-01-25

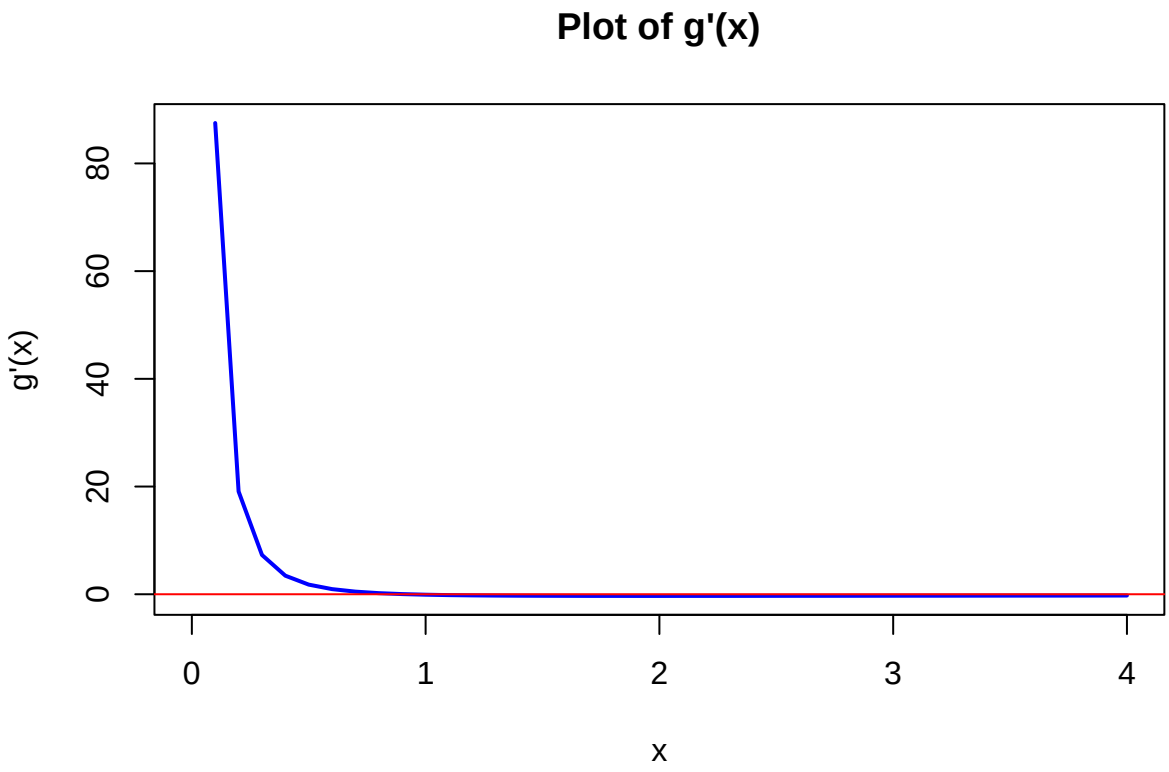
Question 1: Maximization of a function in one variable

a) Plot the function g(x) in the interval [0, 4]. What is your guess for the maximum point?



The maximum point of g(x) is estimated to be 0.35 where variable x is around 1 according to the plot.

b)Compute the derivative $g'(x)$ of g(x); recall the quotient rule that for g(x) = $\frac{u(x)}{v(x)}$, the derivative is $g'(x) = \frac{u'(x)v(x)-u(x)v'(x)}{(v(x))^2}$. Plot $g'(x)$ in [0,4], and add a horizontal reference line at 0 to the plot.



c) Write your own R function applying the bisection method to g' to find a local maximum of g for a user-selected starting interval.

```
##  [1] 2.5000000 1.2500000 0.6250000 0.9375000 0.7812500 0.8593750 0.8984375
##  [8] 0.9179688 0.9277344 0.9228516 0.9252930 0.9240723 0.9234619
```

```
## [1] 13
```

The local maximum is calculated to be 0.92346 for the range [0,5] by the application of bisection function, convergence reaches at the 13th iteration.

d) Write your own R function applying the secant method to g' to find a local maximum of g for a user-selected pair of starting values.

```
##   x0 x1        x2        x3        x4        x5        x6       x7
## 1  2  1 0.6331424 0.9643919 0.9451736 0.9216776 0.9236817 0.923594
```

```
## [1] 6
```

The local maximum is calculated to be 0.923594 for the range [2,1] by the application of secant function, convergence reaches at the 6th iteration

e) Run the functions in c. and d. for different starting intervals/pairs of starting values and check when they converge to the true maximum and when not. Discuss why. Compare the two methods also in terms of number of iterations used and programming effort required.

```
## [1] 0.9233398
## bisection gives result 0.9233398 at the  12 iteration for a= 0 b= 2
## [1] 0.9233398
## bisection gives result 0.9233398 at the  12 iteration for a= 2 b= 0
## [1] 0.9234619
## bisection gives result 0.9234619 at the  13 iteration for a= 0 b= 5
## [1] 0.9234619
## bisection gives result 0.9234619 at the  13 iteration for a= 5 b= 0
## [1] 25.90015
## bisection gives result 25.90015 at the  12 iteration for a= 3 b= 100
```

Bisection method gives correct values while g'(a) and g'(b) have g'(a)*g'(b)<0. But when both of them have positive or negative value, the method doesn't work even if the true root is in this range, as can be see for range [0,100]. And for example in range [3,100], the given range doesn't include the true root, the function can also convergence at a wrong root, where g'(x)=0 but a local minimum. The iteration number is around 12.

```
## [1] 25.8202
## secant gives result 25.8202 at the  6 iteration for a= 0.1 b= 4
## [1] 0.9199369
## secant gives result 0.9199369 at the  6 iteration for a= 0.1 b= 0.4
## [1] 0.9216776
## secant gives result 0.9216776 at the  6 iteration for a= 2 b= 1
```

Secant method requires the intersection of extensions of points g'(x) and g'(x+1) with x axis. If there is no intersection, it can not find the root.And the position of the inition values values, as the longing of the line is towards x+1 's direction.Even if the range including the true root is given, i.g.[0.1,4], the function can also give a wrong result. Secant method need fewer iteration times but it needs more effort in programming. The convergence speed is fast comparing with bisection method.

f) When you just should program one of them: Would you use bisection or secant, here? In general, for another function g(x) to be maximized: When would you switch and use the other algorithm?

Bisection method is preferred here as it is easier to program and understand in this simple situation. However, for more complicated question, secant method is better than bisection method as it converges quicker.

Question 2: Computer arithmetics (variance)

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

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

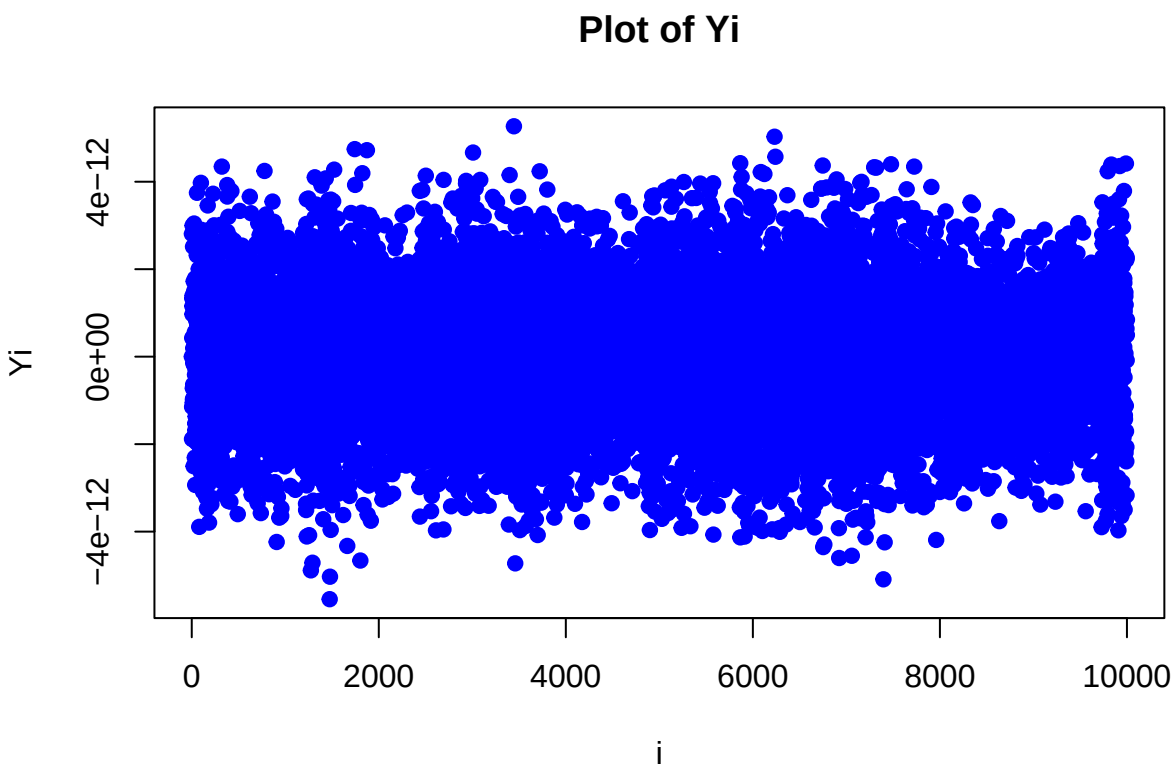
a) Write an R function that calculates the variance in this way

```
myvar = function(x_i) {
    n = length(x_i)
    if (n == 1){
      return(c(0))
    }
    sum_1 = sum(x_i^2)
    sum_2 = (1/n) * (sum(x_i)^2)
    res = (sum_1 - sum_2) / (n - 1)
    return(res)
}
```

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

```
set.seed(1234)
x = rnorm(10000, mean = 108, sd = 1)
```

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?



function works well, the difference between the two variance estimation methods is close to 0. The difference is not exactly 0 because of the floating-point arithmetic in R.

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

Welford's method to calculate the variance is more numerically stable than the standard formula. The formula is as follows:

$$s^2 = s^2_{n-1} + (x_n - \bar{x}_{n-1}) \cdot (x_n - 2 \cdot \bar{x}_{n-1})$$

```
variance_Welford = function(x_i) {
    n = length(x_i)
    if (n == 1){
      return(c(0))
    }
    M = 0
    S = 0
    for (i in 1:n) {
        M = M + (x_i[i] - M) / i
```

```
        S = S + (x_i[i] - M) * (x_i[i] - M)
    }
    res = S / (n - 1)
    return(res)
}

print(variance_Welford(x))
```

```
## [1] 0.9743149
```

```
print(var(x))
```

```
## [1] 0.9752143
```

##Appendix

```
##Assignment 1
#a)
numerator<- function(x) {return(log(x+1))}
denominator<-function (x) {return(x^(3/2)+1)}
g_fun<- function(x){
  res<-numerator(x)/denominator(x)
  return(res)
}
# Generate a sequence of input values
x_values <- seq(-0, 4, by = 0.1)

# Compute the function values
y_values <- g_fun(x_values)

# Plot the function
plot(x_values, y_values, type = "l", col = "blue", lwd = 2,
     xlab = "x", ylab = "g(x)", main = "Plot of g(x) function")

#b)
#define the derivation function
g_fun_prime<- function(x){
  res<-(1/(x+1)*denominator(x)-numerator(x)*3/2)/(numerator(x))^2
  return(res)
}
# Compute the function values
y_values<-g_fun_prime(x_values)


# Plot the function
plot(x_values, y_values, type = "l", col = "blue", lwd = 2,
     xlab = "x", ylab = "g'(x)", main = "Plot of g'(x)")
abline(h=0, col="red")

#c)
#pre-define a convergence function
convergence_crit<- function(xt,xt1) {
  return(abs(diff(c(xt,xt1)))/abs(xt1))
}

#bisection
bisection<-function(a,b) {
  if(g_fun_prime(a)*g_fun_prime(b)>=0) {stop('check input')}
#initialization
 var<- c((a+b)/2)
  i=1
  while(length(var)<2 || convergence_crit(var[i-1],var[i])>0.001) {
    if(g_fun_prime(a)*g_fun_prime(tail(var,1))<=0)  {b<-tail(var,1)}
    if(g_fun_prime(tail(var,1))*g_fun_prime(b)<0)  {a<-tail(var,1)}
       var<-c(var,(a+b)/2)
       i<-i+1
  }
  return(var)
```

```
}

# calculate the local maximum of g for self-selected starting interval
print(bisection(0,5))
#iteration number
print(length(bisection(0,5)))

#d)
#define the derivation-derivation function
g_fun_prime_prime<- function(xt,xt_minus1){
  res<-(g_fun_prime(xt)-g_fun_prime(xt_minus1))/(xt-xt_minus1)
  return(res)
}

secant<- function(x0,x1){
  res<- data.frame('x0'=x0,'x1'=x1)
  i=3
  xt1<-as.vector(res)[[i-1]]

  xt<-as.vector(res)[[i-2]]

  while (convergence_crit(xt,xt1)>0.001) {


    res_add<-xt-g_fun_prime(xt)/g_fun_prime_prime(xt1,xt)
    if (is.na(res_add)) {
      stop("NA value encountered, stopping the iteration.")

    }
    res[paste0("x", i-1)]<-res_add
    xt <- xt1
    xt1 <- res_add


    i<-i+1
  }
    return(tail(res,1))
}
# calculate the local maximum of g for self-selected starting interval
print(secant(2,1))
print(dim(secant(2,1))[2]-2)

#e)
values<- matrix (c(0,2,
                   2,0,
                   #0,100,
                   0,5,
                   5,0,
                   3,100
                   #0,100
                   ),
                 ncol = 2,byrow = TRUE)
for (i in 1:nrow(values)) {
```

```
  res_vector<-bisection(values[i,1],values[i,2])
  cat('bisection gives result',print(tail(res_vector,1)), 'at the ',length(res_vector), 'iteration for a

  values<- matrix (c(0.1,4,
                     0.1,
                     0.4,
                     2,1
                     #,1,2
                     #,4,0.1
                    ),
                   ncol = 2,byrow = TRUE)
for (i in 1:nrow(values)) {
  res_df<-secant(values[i,1],values[i,2])
  cat('secant gives result',print(res_df[1,dim(res_df)[2]-2]), 'at the ',dim(secant(2,1))[2]-2, 'iterati
}

###Assignment 2
  #a)
  myvar = function(x_i) {
    n = length(x_i)
    if (n == 1){
      return(c(0))
    }
    sum_1 = sum(x_i^2)
    sum_2 = (1/n) * (sum(x_i)^2)
    res = (sum_1 - sum_2) / (n - 1)
    return(res)

    #b)
    set.seed(1234)
x = rnorm(10000, mean = 108, sd = 1)

#c)
compute_difference = function(Xi) {
    n = length(Xi)
    diff_res = c()
    for (i in 1:n) {
        Xi_i = Xi[1:i]
        if (length(Xi_i) == 1){
          var = 0
        }else{
          var = var(Xi_i)
        }
        Yi = myvar(Xi_i) - var
        diff_res = c(diff_res, Yi)
        #plot(i, Yi, col = "blue", pch = 19, xlab = "i", ylab = "Yi", main = "Plot of Yi")
    }
    browser()
    plot(1:n, diff_res, col = "blue", pch = 19, xlab = "i", ylab = "Yi", main = "Plot of Yi")

}
compute_difference(x)
```

```r
#d)

variance_Welford = function(x_i) {
    n = length(x_i)
    if (n == 1){
      return(c(0))
    }
    M = 0
    S = 0
    for (i in 1:n) {
        M = M + (x_i[i] - M) / i
        S = S + (x_i[i] - M) * (x_i[i] - M)
    }
    res = S / (n - 1)
    return(res)
}

print(variance_Welford(x))

print(var(x))
```