Group15_Lab1

Naveen Gabriel, Sridhar Adhirkala

31 January 2019

Contents

1. Subtracting Fractions

1 Snippet 1 -

```
#Q1
x1 <- 1/3
x2 <- 1/4
if(x1-x2 == 1/12){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
```

```
[1] "Subtraction is wrong"
```

The way 1/3 is represented in r is makes the difference in this case. Since the representation of 1/3 is different, the subtraction (1/3 - 1/4) results in a different answer than 1/12. The difference occurs in the 17th decimal place. The difference is very small but this is what results in the inequality in the first case.

2 Snippet 2 -

```
x1 <- 1
x2 <- 1/2
if(x1-x2 == 1/2){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
```

```
[1] "Subtraction is correct"
```

1/2 is not such a complex fraction so the representation is easy for it and results in x2 being equal to (0.5). This is the reason the equality is true in this case. 1 - 1/2 is 0.5 and 1/2 is also 0.5, resulting in the equality.

3 Improvement -

```
x1 <- 1/3
x2 <- 1/4
a = formatC(x1-x2, digits = 10, format = 'f')
b = formatC(1/12, digits = 10, format = 'f')
if(a == b){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
```

```
[1] "Subtraction is correct"
```

The way we can improve this is comparing only a few of the decimal places, like the first 10 decimal places of a number.

2. Derivative

1 Creating the Derivative Function

```
#Q2
f =function(x){
  return(x)
}
derivative = function(f, x){
  eps = 10^-15
  d = (f(x+eps) - f(x))/(eps)
  return(d)
}
```

We created a function f that is an identity function, and we also created a derivative function that accepts as input a function that you want to find the derivative of and the value of x at which you want the derivative of the function and returns the derivative value.

The derivative function is an approximation of the actual derivative, and for the function f the actual derivative for any value of x is equal to 1. We shold get an answer close to 1.

2 Derivative of f(X) at X=1

```
derivative(f, 1)
```

```
[1] 1.110223
```

For this case, the computer is trying to match exponent of 10^{-15} to the exponent of 1 while doing the sum. As a result, the mantissa of the resulting sum looses some of its significant bit and computer rounds off the right most bits. When it is converted to real number we get the value as 0.00000000000000111022302462515 65. This value when divided by 10^{-15} gives 1.110223 .

3 Derivative of f(X) at X=100000

```
derivative(f, 100000)
```

```
[1] 0
```

For x=100000 the derivative turns out to be zero. Since the function we are finding the derivative for is an identity function, the numerator for the derivative turns out to be-

{epsilon(which is a very small number) + x - x}

Internally the computer is trying to match exponent of 10^{-15} to the exponent of 10000 while doing a sum. As a result the mantissa of 10^{-15} is very small(becomes zero). As a result the sum of both is 100000 which once it is subtracted from 100000 yields 0. Basically an overflow is happening.

3. Variance

1. Function to calculate variance

```
#function accepts a vector whose variance needs to be calculated.
myvar <- function(x){
    len <- length(x)
    var <- (sum(x^2)-(1/len)*(sum(x))^2)*1/(len-1)

    return(var)
}
```

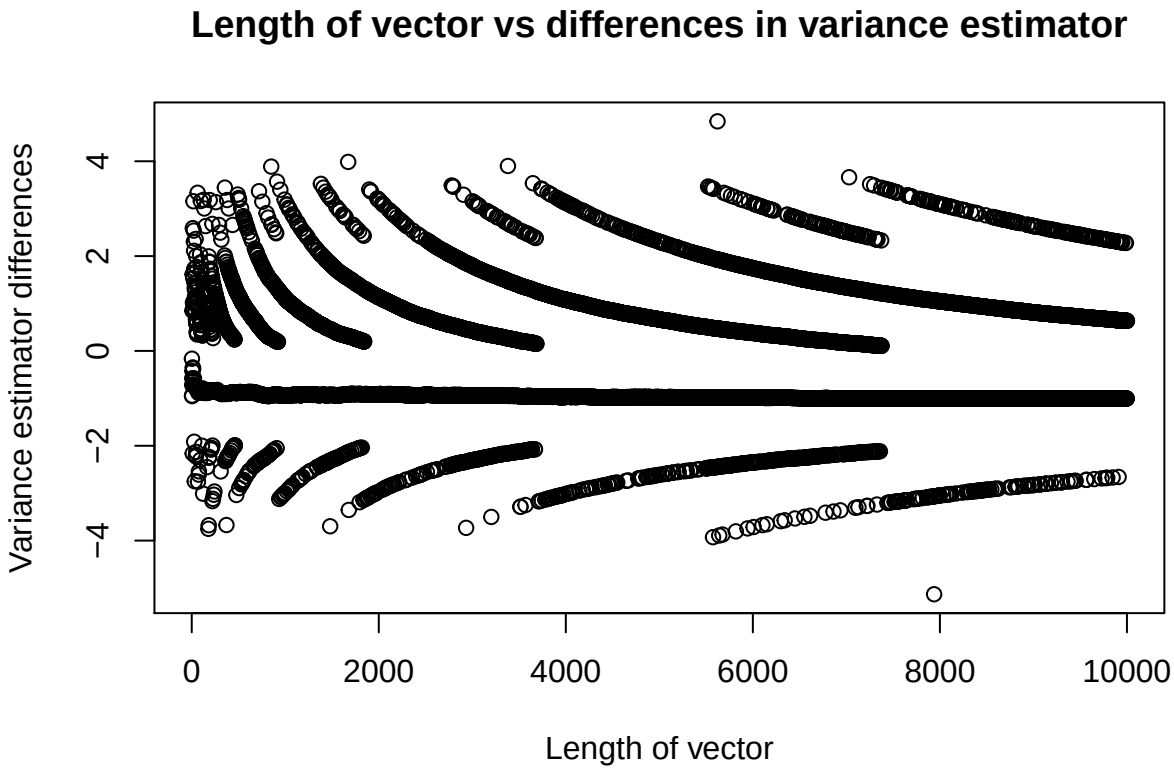
2. Generating vectors and analysing variance

The in built function variance converges towards one value after some subsets but the variance calculated by myvar function does not converges to one. The variance calculated by myvar oscillates between +ve and -ve value. This is because of the loss in bits and rounding off error due to overflow.

```
vec <- rnorm(n=10000,mean=10^8,sd=1)
diff <- c()

for(i in 1:length(vec)) {
 diff[i]<-myvar(vec[1:i]) - var(vec[1:i])
}


{plot(x=1:length(vec),y=diff,xlab="Length of vector",
     ylab ="Variance estimator differences" )
title(main="Length of vector vs differences in variance estimator")}
```



3. Implementing a better variance estimator

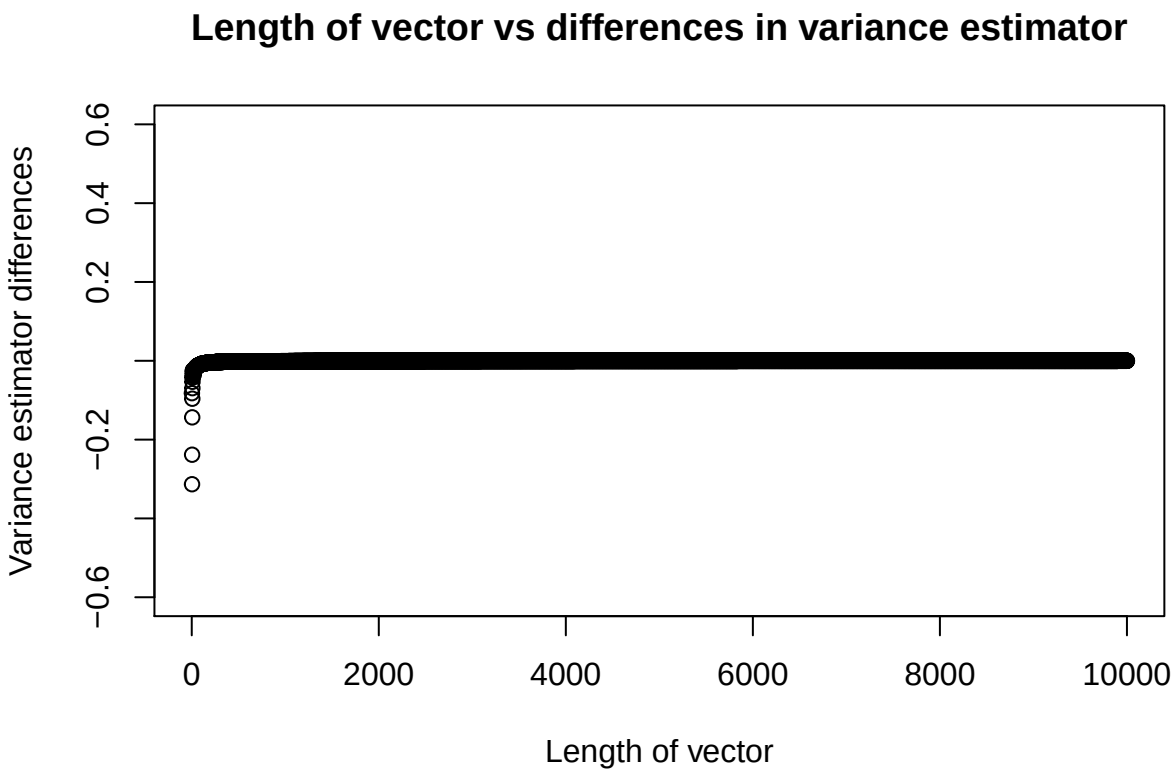
Reimplementing the variance estimator shows that the difference in error between new var estimator and in built R has reduced and it converges to zero.

```
myvar2 <- function(x){
    len <- length(x)
    mn <- sum(x)/len
    sum <- 0

    sum <- sum((x - mn)^2)/len
    return(sum)
}

for(i in 1:length(vec)) {
 diff[i] <- myvar2(vec[1:i]) - var(vec[1:i])
}

{plot(x=1:length(vec),y=diff,ylim =c(-0.6,0.6),xlab="Length of vector",
     ylab ="Variance estimator differences" )
title(main="Length of vector vs differences in variance estimator")}
```



4. Linear Algebra

1. Solving $\mathbf{A}\beta = \mathbf{b}$ using solve()

- On using solve() to compute coeffecients of linear regression, it throws below error which basically means that the matrix A is neary singular. The inverse for the singular matrix does not exist. A small reciprocal condition number suggest that the matrix A is badly conditioned and their is a huge loss of accuracy and the resulting data is meaningless.
- Here, kappa returns the condition number of the matrix A. Large value of condition number suggest that a small perturbation in A or b would result in large change in β which is denotes that A is badly conditioned. Ideally a small perturbation in A or b should have a small perturbation in β.

The value of kappa does supports the above aforementioned conculsion of the result of solve().

```
 Determinant of matrix A :  0
```

```
#z<-solve(a,b)
tryCatch(solve(a,b),
         error = function(e){
           message("\n\nAn error occurred:\n", e)})

cat("\n Condition number for a matrix :",kappa(a))
```

```
 Condition number for a matrix : 4.274694e+15
```

2. Scaling the data and reperforming the computation

- On scaling the data, it is possible to evaluate the coeffecients using solve(). The determinant of matrix A is still computationally zero but solve function is able to calculate the inverse of A. The reasoning would be that, the determinant of scaled matrix A is not as small as the determinant of unscaled matrix A and computer is able to solve the equation.
- Condition value of scaled matrix A is lesser than unscaled one which suggest that the perturbation in scaled matrix A results in output which has less perturbation than unscaled matrix.

```
[1] 664318664630
```

```
Condition number for the scaled matrix A: 664318664630
```

5. Appendix

```
knitr::opts_chunk$set(
    echo = TRUE,
    message = FALSE,
    warning = FALSE,
    comment= NA
)
#Q1
x1 <- 1/3
x2 <- 1/4
if(x1-x2 == 1/12){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
x1 <- 1
x2 <- 1/2
if(x1-x2 == 1/2){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
x1 <- 1/3
x2 <- 1/4
a = formatC(x1-x2, digits = 10, format = 'f')
b = formatC(1/12, digits = 10, format = 'f')
if(a == b){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}

#Q2
f =function(x){
  return(x)
}
```

```
derivative = function(f, x){
  eps = 10^-15
  d = (f(x+eps) - f(x))/(eps)
  return(d)
}
derivative(f, 1)
derivative(f, 100000)
#function accepts a vector whose variance needs to be calculated.
myvar <- function(x){
    len <- length(x)
    var <- (sum(x^2)-(1/len)*(sum(x))^2)*1/(len-1)

    return(var)
}
vec <- rnorm(n=10000,mean=10^8,sd=1)
diff <- c()

for(i in 1:length(vec)) {
 diff[i]<-myvar(vec[1:i]) - var(vec[1:i])
}


{plot(x=1:length(vec),y=diff,xlab="Length of vector",
     ylab ="Variance estimator differences" )
title(main="Length of vector vs differences in variance estimator")}
myvar2 <- function(x){
    len <- length(x)
    mn <- sum(x)/len
    sum <- 0

    sum <- sum((x - mn)^2)/len
    return(sum)
}

for(i in 1:length(vec)) {
 diff[i] <- myvar2(vec[1:i]) - var(vec[1:i])
}

{plot(x=1:length(vec),y=diff,ylim =c(-0.6,0.6),xlab="Length of vector",
     ylab ="Variance estimator differences" )
title(main="Length of vector vs differences in variance estimator")}

tec <- read.csv("tecator.csv")
x <- subset(tec,select = -c(Protein))
y <- as.vector(tec[,c("Protein")])

x <- as.matrix(x)
a <- t(x)%*%x
b <- t(x)%*%y
cat("\n Determinant of matrix A : ", det(a))
#z<-solve(a,b)
tryCatch(solve(a,b),
         error = function(e){
```

```
          message("\n\nAn error occurred:\n", e)})

cat("\n Condition number for a matrix :",kappa(a))
#scaling the data
tec_new <- scale(tec)

x_new <- subset(tec_new,select = -c(Protein))
y_new <- as.vector(tec_new[,c("Protein")])

x_new <- as.matrix(x_new)
a <- t(x_new)%*%x_new
b <- t(x_new)%*%y_new

z<-solve(a,b)
kappa(a)
cat("Condition number for the scaled matrix A:",kappa(a))
```