Computational statistics Lab1

YiHung Chen, Jonathan Dorairaj

2022-11-14

Question 1: Be careful when comparing

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

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

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

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

Q1. Check the results of the snippets. Comment what is going on.

Answer 1 For x1=1/3 and x2=1/4. The comparison with 1/12 is not the same, because 1/3 cannot be stored exactly in floating point format (underflow).
For x1 = 1, x2 = 1/2. They can be stored correctly, therefore the calculation result is exactly the same with 1/2.

Q2. If there are any problems, suggest improvements.

Answer 2 It would be better to use `all.equal(x1-x2, 1/12)`. This takes the machine tolerance into account, by default the tolerance value is close to `1.5e-8`.

```
x1 <- 1/3
x2 <- 1/4
print(all.equal(x1-x2, 1/12))
```

```
## [1] TRUE
```

Question 2: Derivative

```
derivative <- function(x){

  epsilon <- 10^(-15)
  derivative <- ((x+epsilon)-x)/epsilon

  return(derivative)
}
```

Q1. Write your own R function to calculate the derivative of f(x) = x in this way with ϵ = 10^-15.

```
derivative(1)
```

Q2.Evaluate your derivative function at x = 1 and x = 100000.

```
## [1] 1.110223024625156540424
```

```
derivative(100000)
```

```
## [1] 0
```

Q3. What values did you obtain? What are the true values? Explain the reasons behind the discovered differences.

Answer: The values we obtain are 1.110223... for `x`=1 and 0 for `x`=100000
Both of them should be result in 1, since it is ϵ/ϵ in the case of f(x) = x.

In the first case, x is magnitudes greater than epsilon. Therefore, when adding epsilon to x, there is a rounding error due to underflow.That results in a number close to 1 but not exactly that.

In the second case, x is so large (compare to ϵ), that adding ϵ to `x` leads to underflow and the value stored is x itself. Which means that (x-x)=0.

Question 3: Variance

```
myvar <- function(x)
{
  n <- length(x)
term1 <- sum(x^2)
term2 <- (sum(x)^2) * (1/n)
var1 <- (1/(n-1)) * (term1 - term2)
return(var1)
}
```

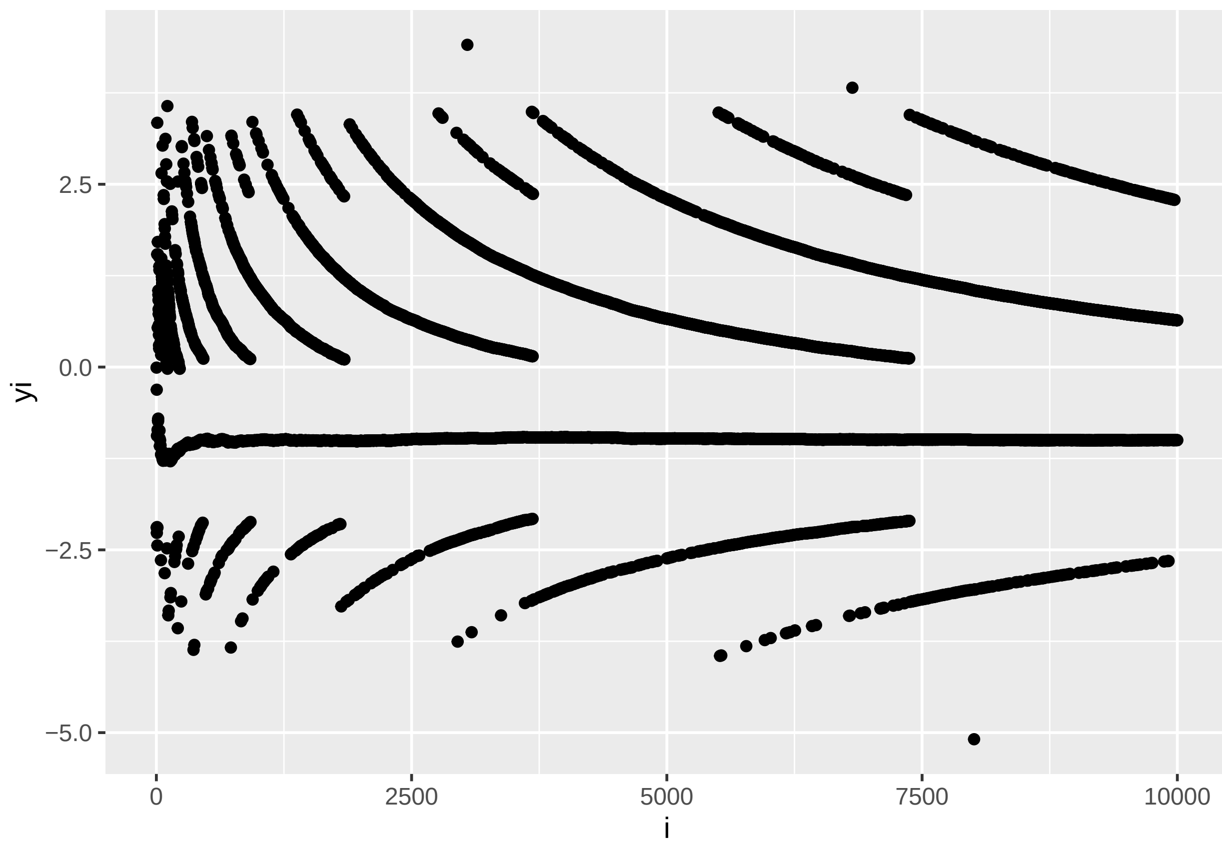
Q1. Write your own R function, myvar, to estimate the variance in this way.

```
set.seed(12345)
x <- rnorm(10000, mean = 10^8,sd = 1)
```

Q2.Generate a vector x = (x1, . . . , x10000) with 10000 random numbers with mean 10^8 and variance 1.

```
var_df <- data.frame()
for(i in 1:10000) {
  xi <- x[1:i]
  t1 <- myvar(xi)
  t2 <- var(xi)
  temp <- data.frame(i,t1, t2, (t1-t2))
  var_df<- rbind(var_df,temp)
}
colnames(var_df) <- c("i","myvar","var1","yi")
ggplot(data = var_df,aes(x=i,y = yi)) + geom_point()
```

Q3.For each subset compute the difference y_i = myvar(x_i)-var(x_i). Plot the dependence y_i on i. Draw conclusions from this plot. How well does your function work? Can you explain the behavior?

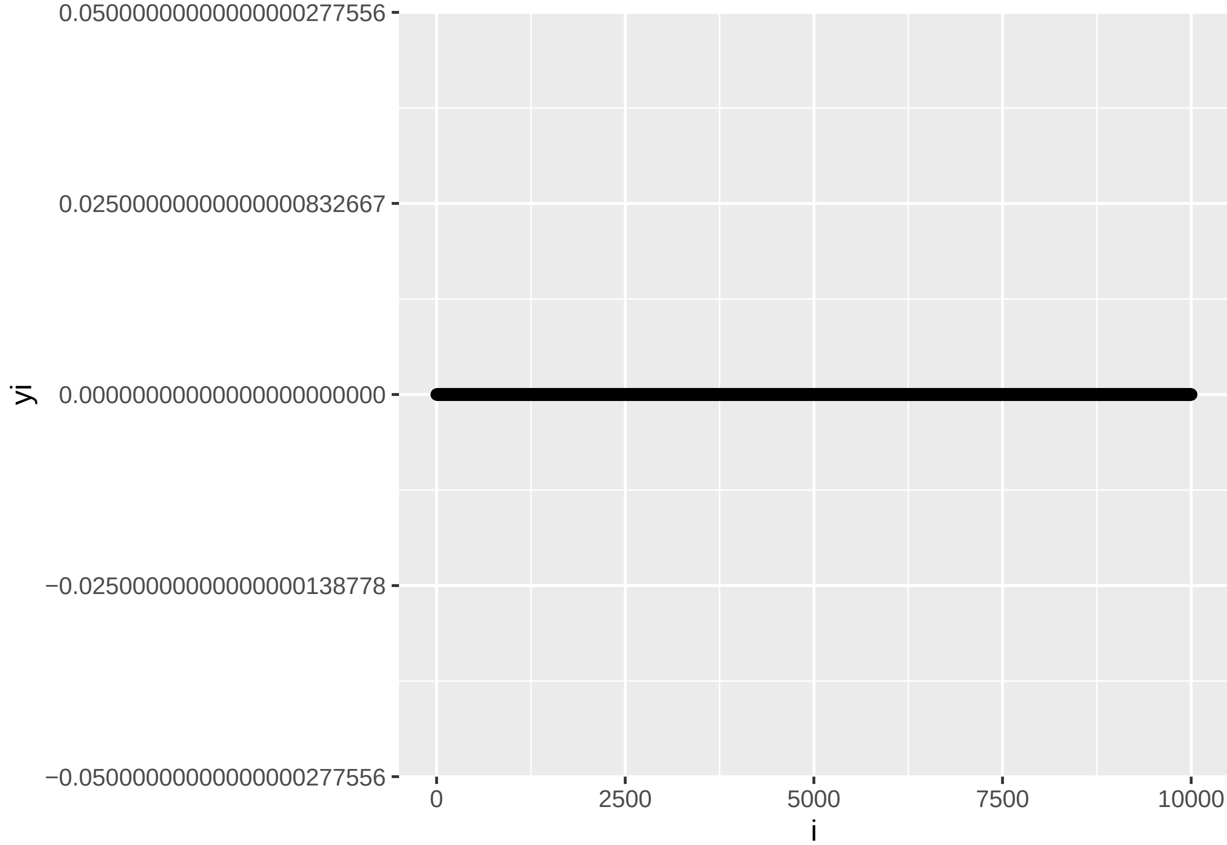


Answer: The function does not seem to work well. Maximum difference between myvar and var is 5.09 and generally the variance is between |2.5| units of variances calculated using var(). The differences are due to the fact that for large subsets of x, the values of $\sum_{i=1}^{n} x_i^2$ is very large and while subtracting the values with $(\sum_{i=1}^{n} x_i)^2$ and dividing by value n, we get underflow.

```
myvar_imp <- function(x){
  n <- length(x)
  m <- mean(x)
  return(
     (1/(n-1) ) * ( sum( (x - m)^2)
  ))
}
var_df2 <- data.frame()
for(i in 2:10000) {
  xi <- x[1:i]
  t1 <- myvar_imp(xi)
  t2 <- var(xi)
  diff <- t1-t2
  temp <- data.frame(i,t1, t2, diff)
  var_df2<- rbind(var_df2,temp)
}
colnames(var_df2) <- c("i","myvar_imp","var1","yi")
```

```
var_df2$yi[var_df2$yi< 1e-15]=0
ggplot(data = var_df2,aes(x=i,y = yi)) + geom_point()
```

Q4.How can you better implement a variance estimator? Find and implement a formula that will give the same results as var()?



Answer: We used the formula for unbiased variance and plotted the difference between `myvar_imp()` and `var()` vs `i`. We observed that the differences in the calculations of variance were in of the order of `1e-16`, which is extremely small. Therefore, we included a condition to set all differences variances below the order of `1e-15` to `0`. This ensured that we ignore underflow errors and hence the values are were similar to using `var()` directly.

Question 4: Binomial coefficient

```
approach_A <- function(n,k){
  prod (1:n) / (prod (1:k)*prod(1:(n-k)))
}
approach_B<- function(n,k){
  prod((k+1):n)/prod(1:(n - k))
}
approach_C<- function(n,k){
  prod ((( k+1):n)/(1:(n-k)))
}
```

```
approach_A(2,2)
```

Q1.Even if overflow and underflow would not occur these expressions will not work correctly for all values of n and k. Explain what is the problem in A, B and C respectively

```
## [1] Inf
```

```
approach_B(2,2)
```

```
## [1] Inf
```

```
approach_C(2,2)
```

```
## [1] Inf
```

Answer: For n=k, all the approaches will result in INF, since 0 value is in denominator. This is because, prod(1:0) will gives 0. However, in reality 0!=1

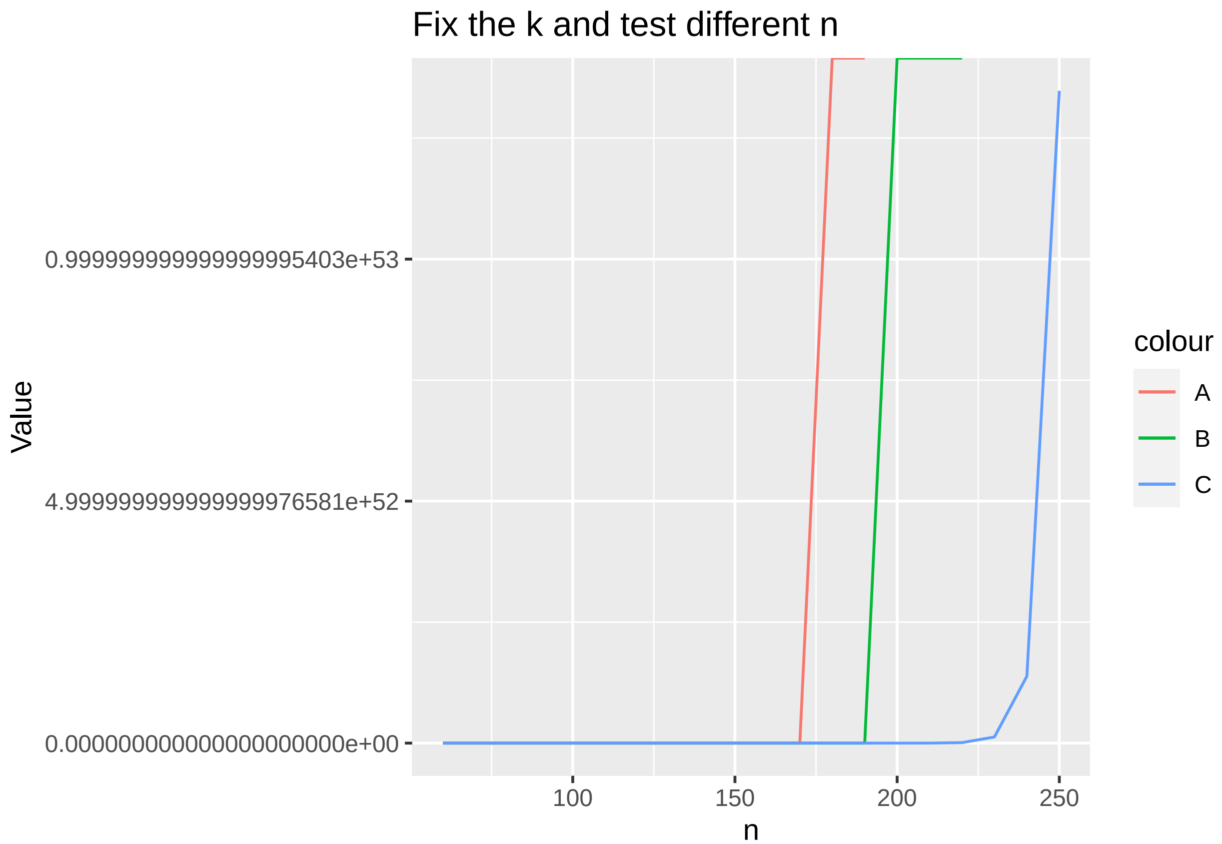
```
max_a_fix_k <- approach_A(170,50)
max_b_fix_k <- approach_B(200,50)
max_c_fix_k <- approach_C(28494182,50)

n<- seq(60,250,10)
result <- data.frame()
for(i in n)
{
  t1 <- approach_A(i,50)
  t2 <- approach_B(i,50)
  t3 <- approach_C(i,50)
  result <- rbind(result,c(i,t1,t2,t3))

}
colnames(result) <- c("n","A","B","C")
ggplot(data = result,aes(x = n, y = C)) +
  geom_line(aes(x = n,y = A, color="A")) +
  geom_line(aes(x = n,y = B, color="B"))+
  geom_line(aes(x = n,y = C, color="C"))+
  ggtitle("Fix the k and test different n")+
  ylab("Value")
```

Q2.In mathematical formula one should suspect overflow to occur when parameters, here n and k, are large. Experiment numerically with the code of A, B and C, for different values of

n and k to see whether overflow occurs. Graphically present the results of your experiments.



Q3. Which of the three expressions have the overflow problem? Explain why.

Answer: According to the graph, approach A and B will suffer from overflow while fixing `k` value (In this case `k=50`). However, approach C can take much bigger `n` than others before overflow occur. We calculated that if the `n >= 28494182`, approach C will also suffer from overflow. Interestingly, for fixing value `n`, all approach will not suffer overflow, unless the `n >= 170` (for approach A and B).

Appendix

```
library(ggplot2)
options(digits=22)

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


  #=====2=====
  x1 <- 1
  x2 <- 1/2
  if(x1-x2 == 1/2){
    print("Subtraction is correct")
  }else{
    print("Subtraction is wrong")
  }
  #=====3=====
  x1 <- 1/3
  x2 <- 1/4
  print(all.equal(x1-x2, 1/12))

#Question 2:

  derivative <- function(x){

  epsilon <- 10^(-15)
  derivative <- ((x+epsilon)-x)/epsilon

  return(derivative)
  }


  derivative(1)
  derivative(100000)


#Question 3:
  #=====1=====
  myvar <- function(x)
  {
   n <- length(x)
  term1 <- sum(x^2)
  term2 <- (sum(x)^2) * (1/n)
  var1 <- (1/(n-1)) * (term1 - term2)
  return(var1)
```

```r
  }
  #=====2=====
  set.seed(12345)
  x <- rnorm(10000, mean = 10^8,sd = 1)


  #=====3=====
  var_df <- data.frame()
  for(i in 1:10000) {
    xi <- x[1:i]
    t1 <- myvar(xi)
    t2 <- var(xi)
    temp <- data.frame(i,t1, t2, (t1-t2))
    var_df<- rbind(var_df,temp)
  }
  colnames(var_df) <- c("i","myvar","var1","yi")
  ggplot(data = var_df,aes(x=i,y = yi)) + geom_point()

  #=====4=====
  myvar_imp <- function(x){
    n <- length(x)
    m <- mean(x)
    return(
       (1/(n-1)) * ( sum((x-m)^2))
       )
  }
  var_df2 <- data.frame()

  for(i in 2:10000) {
  xi <- x[1:i]
  t1 <- myvar_imp(xi)
  t2 <- var(xi)
  diff <- t1-t2
  temp <- data.frame(i,t1, t2, diff)
  var_df2<- rbind(var_df2,temp)
  }

  colnames(var_df2) <- c("i","myvar_imp","var1","yi")
  ggplot(data = var_df2,aes(x=i,y = yi)) + geom_point()
  var_df2$yi[var_df2$yi< 1e-15]=0
  ggplot(data = var_df2,aes(x=i,y = yi)) + geom_point()

#Question 4:
  approach_A <- function(n,k){
    prod (1:n) / (prod (1:k)*prod(1:(n-k)))
  }
  approach_B<- function(n,k){
    prod((k+1):n)/prod(1:(n - k))
  }
  approach_C<- function(n,k){
    prod ((( k+1):n)/(1:(n-k)))
  }
  #=====1=====
```

```r
approach_A(2,2)
approach_B(2,2)
approach_C(2,2)



#=====2=====

# hit fix k=50 limit
max_a_fix_k <- approach_A(170,50)
max_b_fix_k <- approach_B(200,50)
max_c_fix_k <- approach_C(28494182,50)

n<- seq(60,250,10)
result <- data.frame()
for(i in n)
{
 t1 <- approach_A(i,50)
 t2 <- approach_B(i,50)
 t3 <- approach_C(i,50)
 result <- rbind(result,c(i,t1,t2,t3))

}
colnames(result) <- c("n","A","B","C")

ggplot(data = result,aes(x = n, y = C)) +
geom_line(aes(x = n,y = A, color="A")) +
geom_line(aes(x = n,y = B, color="B"))+
geom_line(aes(x = n,y = C, color="C"))+
ggtitle("Fix the k and test different n")+
ylab("Value")

# Test fix n limit
max_a_fix_n <- approach_A(170,85)
max_b_fix_n <- approach_B(170,85)
max_c_fix_n <- approach_C(170,85)
#won't hit limit unless n > 170
```