Computational Statistics-Lab1

Fengjuan Chen(fench417) Zhixuan Duan(zhidu838)

1/31/2020

Question 1

1 check two results of the snippets

The result of the first snippet is:

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

The result of the second snippet is :

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

Comment:

Although $1/3 - 1/4$ is equal to $1/12$ in mathematics, we got a feedback of "Subtraction is wrong". While $1 - 1/2$ is equal to $1/2$, we got a result of "Subtraction is correct". There is a discrepancy.

The reason is that the comupter numerical system is different from the mathematichal numerical system. They are not the one-to-one corresponding. The computer numbers can only represent useful subset of the corresponding mathematical entries. Then the operations on computer numerical system can not obtain the exact results with the same operations on mathematical integer and real systems. The computer numerical system can only try its best to make the similiar results with the mathematical numerical system.

To confirm this, we can make the computer system to show the whole parts of each number. If we set the digits to 22 (the maximum digit) by command "options(digit=22)" in R, we will see that

$1/3 - 1 - 4 = 0.083333333333333331482962$ and

$1/12 = 0.083333333333333287074$,

so they are not equal to each other.

Wihle $1 - 1/2 = 0.5$ and $1/2 = 0.5$, exactly equal.

From these two outputs, we see that we should be very careful when coparing with two computer numbers. The operation "==" on two numbers may make some mistakes.

2 improvement of the code

As we have shown before, there is a problem: the result of the first snippet should be "Subtraction is correct" because $1/3 - 1/4$ is equal to $1/12$ on mathematical calculations.

To make sure the computer returns the right result we use command

if(isTRUE(all.equal(x1-x2,1/12)))

instead of if(x1-x2,1/12).

Then we will get the right result.

Question 2

1 function for derivative

Our R function to calculate the derivative of $f(x) = x$ with $\epsilon = 10^{-15}$

```
derivative <- function(x){
  epsilon <- 10^-15
  return((x+epsilon-x)/epsilon)
}
```

2 results of derivatives

The vaules of $f'(x)$ for $x = 1$ and $x = 100000$ are :

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

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

3 true values of derivative and explanations

The ture values should be 1 in both cases of $x = 1$ and $x = 100000$.

When we use command "options(digits=22)", we have the results in computer numbers:

$\epsilon = 10^{-15} = 1.0000000000000000077705e - 15$,

$(1 + \epsilon) - 1 = 1.1102230246251565404 24e - 15 \neq \epsilon$ and

$(1 + \epsilon - 1)/\epsilon = 1.110223024625156540424 \neq 1$.

$100000 + \epsilon - 100000 = 0 \neq \epsilon$

$(100000 + \epsilon - 100000)/\epsilon = 0 \neq 1$

From these numbers, we can find the reason of this phenomenon: $(1 + \epsilon) - 1 \neq \epsilon$ and

$100000 + \epsilon - 100000 = 0$.

Since computer uses finite digits to present real numbers, not all real numbers are presented by the computer flaoting-point system. Real numbers must be rounded when the computer stores and represents them. When adding a small number to a very large one, the overall result in storage system would erase the small value.

Question 3

1 function myvar

Our R function, myvar, to estimate the variance:

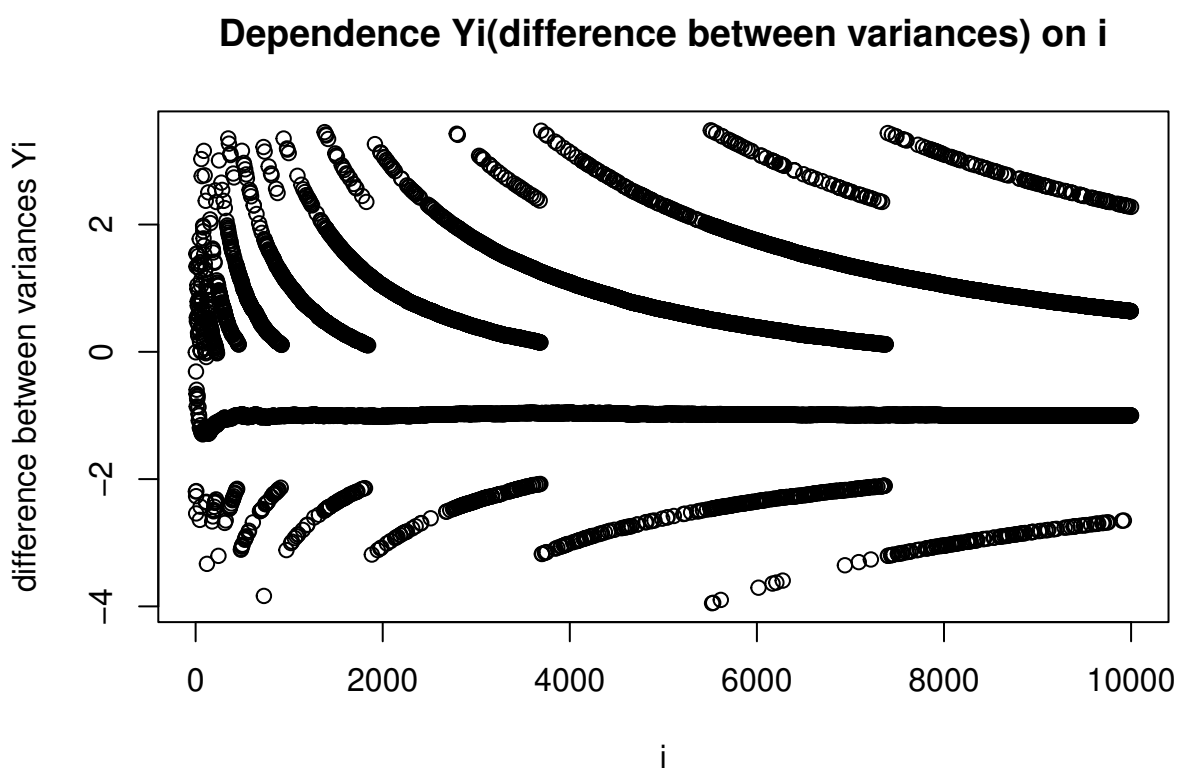
```
myvar <- function(x){
  n <- length(x)
  a <- sum(x^2)-sum(x)^2/n
  return(a/(n-1))
}
```

2 generate a vector

Generate a vector x = $(x_1, \ldots, x_{10000})$ with 10000 random numbers with mean 10^8 and variance 1.

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

3 plot and conlusions



Conlusion: The variance calculated by our function, myvar, has significant difference with the variance obtained by R function var().

In fact, our function, myvar, does not work at all.

This is because the cancellation between the calculation in our function. When we calculate the variance of the vector, the computation of $\sum_{i=1}^{n} x_i^2 - 1/n(\sum_{i=1}^{n} x_i)^2$ in the formula caused the catastrophic cancellation in floating-point system because

$\sum_{i=1}^{n} x_i^2$ and $-1/n(\sum_{i=1}^{n} x_i)^2$

have the approximately equal magnitude and opposite signs.

Not only are their magnitudes are the same but the actual numbers are sometimes very close.

4 better implementation of a variance estimator

There are several methods to solve the catastrophic cancellation in calculating the value of sample variance.

According to the webpage https://angelacorasaniti.wordpress.com/tag/youngs-and-cramer/, compensated two-pass algorithm, Welford's method (one-pass algorithm) and updating Youngs-Cramer algorithm (one-pass algorithm) are solutions for the cancellation in calculating the sample variance.

The updating Youngs-Cramer algorithm is based on the formula

$S_1 = 0$

$S_k = S_{k-1} + \frac{1}{k(k-1)}(k * x_k - t_k)^2, k = 2, ..., n$

where $t_k = \sum_{i=1}^{k} x_i, k = 1, 2, ..., n$ and n is the total number of a sample,

Then sample variance is $S^2 = \frac{1}{n-1} S_n$

R code for the updating Youngs-Cramer algorithm is shown below.

```
newvar <- function(x){
  n=1
  sum=0
  m=x[1]
  for (i in 2:length(x)) {
    n=n+1
    m=m+x[i]
    sum=sum+(1/(n*(n-1)))*(n*x[i]-m)^2
  }
  var=sum/(n-1)
  return(var)
}
```

The updating Youngs-Cramer algorithm gives the same results as R basic function var().

Question 4

1 import the data

```
library(readxl)
data <- read_excel("tecator.xls")
```

2 code for regression coefficients

```
X <- as.matrix(cbind(data[,c(-1,-103)]))
y <- data[["Protein"]]
A <- t(X)%*% X
b <- t(X)%*%y
belta <- solve(A)%*% b
```

3 use solve() function

When solving the equation $A\beta = b$ with the default solver function solve(), an error is returned by R.

Error in solve.default(A) :

system is computationally singular: reciprocal condition number = 7.13971e-17

This error happened because the inverse of A does not exist.

4 Condition number of matrix A

Use the function kappa(A) to check the condition number of matrix A. We obtain the condition number of matrix A,

1157834236871692.25.

The condition number is extremely big. This is correnting to the fact that matrix A is singular and has no inverse.

5 Scale the data

For this condition (matrix A has no inverse and extremely condition number), there are two categories methods to solve this problem. One is scaling the original data, the other is using other methods to solve the equation $A\beta = b$, such as QR decomposition, Cholesky decomposition, or SVD decomposition.

Here, we use scale() function to scale the original data.

After scaling, the inverse of matrix A can be calculated by function solve(). And then the correct result is obtained. However, the condition number of the scaled matrix A is still extremely large,490471520662.0501098633,although it is less than the original condition number of the matrix which is not scaled.

Appendix

```
# Question 1

options(digits = 22)
# Sys.setenv("LANGUAGE"="EN")

x1 <- 1/3
x2 <- 1/4
if(x1-x2==1/12){
  # improvement: if(isTRUE(all.equal(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")
}
```

```r
# Question 2

derivative <- function(x){
  epsilon <- 10^-15
  return((x+epsilon-x)/epsilon)
}

derivative(1)
derivative(100000)
```

```r
# Question 3

myvar <- function(x){
  stopifnot(is.vector(x))
  n <- length(x)
  a <- sum(x^2)-sum(x)^2/n
  return(a/(n-1))
}
set.seed(12345)
x <- rnorm(10000,mean=10^8,sd=1)

difference <- vector(length = length(x))
for (i in 1:10000) {
  difference[i] <- myvar(x[1:i])-var(x[1:i])
}

plot(1:10000,difference)

# new formula for variance of vector x

# Youngs-Cramer algorithm
# The result of this algorithm has error at 10^-7
# plot(1:10000,difference,ylim = c(-10^-7,10^-7))
#      can see the difference
#  difference2[i]=isTRUE(all.equal(newvar(x[1:i]),var(x[1:i])))
#  is true for 9999 results of these two methods,
#    but the variance of x[1:2] is false

newvar <- function(x){
  n=1
  sum=0
  m=x[1]
  for (i in 2:length(x)) {
    n=n+1
    m=m+x[i]
    sum=sum+(1/(n*(n-1)))*(n*x[i]-m)^2
  }
  var=sum/(n-1)
  return(var)
}
difference <- vector(length = length(x))
difference2 <- vector(length = length(x))
```

```r
for (i in 1:10000) {
  difference[i] <- newvar(x[1:i])-var(x[1:i])
  difference2[i]=isTRUE(all.equal(newvar(x[1:i]),var(x[1:i])))
}
plot(1:10000,difference,ylim = c(-10^-7,10^-7))



# Question 4


library(readxl)
data <- read_excel("tecator.xls")
X <- as.matrix(cbind(data[,c(-1,-103)]))
y <- data[["Protein"]]
A <- t(X)%*% X
b <- t(X)%*%y
belta <- solve(A)%*% b
kappa(A)

# scale the data and calculate again
X <- scale(X)
y <- scale(y)
A <- t(X)%*% X
b <- t(X)%*%y
belta <- solve(A)%*% b
kappa(A)
```