Computational Statistics (732A90) Lab1

Anubhav Dikshit(anudi287) and Thijs Quast(thiqu264)

23 January 2019

Contents

Question 1: Be careful when comparing

```
x1 <- 1/3; x2 <- 1/4

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

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

```
x3 <- 1; x4<- 1/2
if(x3-x4==1/2){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
```

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

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

2. If there are any problems, suggest improvements.

Analysis: The numeric precision of the machine creates a barrier in evaluating the floating numbers, although both equation should return "Subtraction is correct", we get the other answer due to this very reason. One way to resolve this is to use *abs(x-y) <= tol* in R, where tol is a very small number that you can manually set, like so

```
tol <- 1e-9
x1 <- 1/3; x2 <- 1/4

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

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

```
x3 <- 1; x4<- 1/2
if(abs(x3-x4-1/2) <= tol){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
```

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

Question 2: Derivative

1. Write your own function to find derivative

```
my_deri <- function(x){
  epi = 10^(-15)
  f_deri <- ((x+epi) - x)/epi
  return(f_deri)
}
```

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

```
my_deri(x=1)
```

```
## [1] 1.110223
```

```
my_deri(x=100000)
```

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

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

Analysis: We should have gotten both values as 1, however for *x=100000* the derivate showed as 0. This is due to the fact that one a small number is added to a large number *x=100000*, due to numerical precision of machine the value is same as the large value, thus the expression *x+epi-x* leads to 0 in this case. In the first case when *x=1*, when a small value is added to *x=1*, the effect of small value is retaianed i.e *x+epi-x* leads to epi, we would believe that epi/epi would lead to 1, but again due to underflow, the value leads to final answer of *1.110223*

Question 3: Variance

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

```
myvar <- function(myvector){
  #vector <- c(1,2,3,4,5)
  n <- length(myvector)
  answer <- (1/(n-1))*(sum(myvector^2) - (1/n) * ((sum(myvector))^2))
  return(answer)
}
```

2. Generate a vector x such that number of terms is 10000 random numbers with mean=10^8, and varience=1

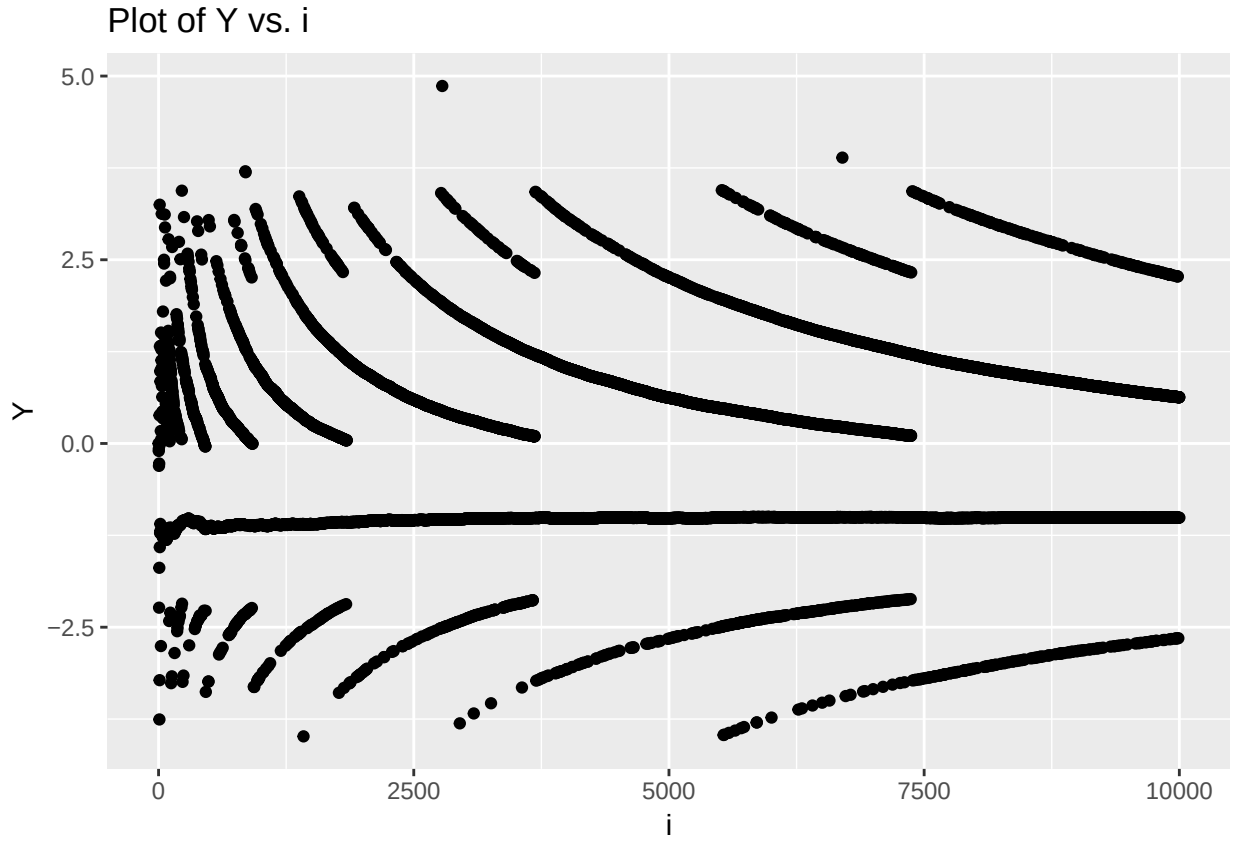
```
my_rand_num <- rnorm(n=10000, mean=10^8, sd=1)
```

3. For each subset x_i = (x1,x2,. . . .xi), compute Y_i = myvar(X_i)-var(X_i), plot Y_i vs. i. Draw conclusions and plot.How well does your function work? Can you explain the behaviour?

```
Y <- vector(mode = "numeric", length = length(my_rand_num))
my_mat <- vector(mode = "numeric", length = length(my_rand_num)+1)

for(i in 2:length(my_rand_num)){
  options(digits = 22)
  X_i = my_rand_num[1:i]
  Y = myvar(X_i) - var(X_i)
  temp <- cbind(i, Y)
  my_mat <- rbind(temp, my_mat)
}

my_mat <- as.data.frame(my_mat)
library(ggplot2)
ggplot(my_mat, aes(x=i, y=Y)) + geom_point() + ggtitle("Plot of Y vs. i")
```



Analysis: Our function does oscillate around -1, thus our function is not working the way we would hope for it to work. This is largely to the numeric precision of our function where *myvar(X_i) - var(X_i)* expression is mostly a negative value. This is due to the fact that addition of larger numbers is causing overflow in our functions while the built-in function is performing the same with higher accuracy/precision. We can see from the plot that once we start increasing the value of terms the tendency of oscillation also decrease.

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

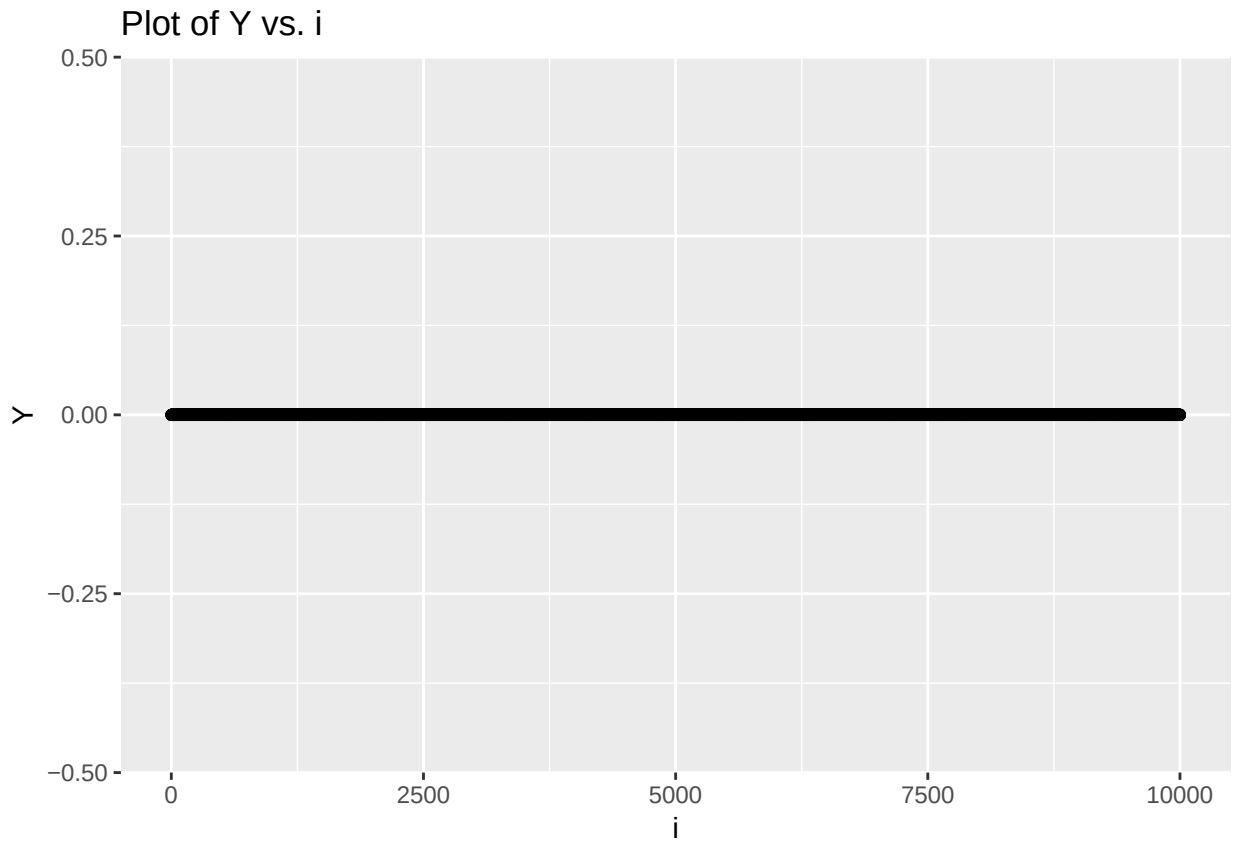
```
my_rand_num <- rnorm(n=10000, mean=10^8, sd=1)

myvar_better <- function(x){
  n <- length(x)
  m <- mean(x)
answer <- (1/(n - 1)) * sum((x - m)^2)
return(answer)
}

Y <- vector(mode = "numeric", length = length(my_rand_num))
my_mat2 <- vector(mode = "numeric", length = length(my_rand_num)+1)

for(i in 2:length(my_rand_num)){
  options(digits = 22)
  X_i = my_rand_num[1:i]
  X_i = na.omit(X_i)
  if((myvar_better(X_i) - var(X_i)) <= 1e-15){
    Y = 0
  }else{
      Y = myvar_better(X_i) - var(X_i)
      }
  temp <- cbind(i, Y)
  my_mat2 <- rbind(temp, my_mat2)
}

my_mat2 <- as.data.frame(my_mat2)
library(ggplot2)
ggplot(my_mat2, aes(x=i, y=Y)) + geom_point() + ggtitle("Plot of Y vs. i")
```



Analysis: We improved our estimator by using the varience using the *(1/(n - 1)) sum((x - m)^2)* expression where *x=mean(vector)* and *m=mean(vector)* and by using a tolerance value, where varience difference below the thershold is rounded to zero, thus this negates the floating point approximation error that machines tend to introduce by the fact that machine are binary data processors with limits of storage.

Question 4: Linear Algebra

1. Import the data set to R

```
library(xlsx)
data <- read.xlsx("tecator.xls", sheetName = "data")
#data <- read.csv("tecator.csv")
data$Sample <- NULL
```

2. Optimal regression coeffcients can be found by solving a system of the type AB = b, where A = X*(t(X))* *and* *b* = *t(X)*y. Compute A and b for the give data set. The matrix X are the observations of the absorbance records, levels of moisture and fat, while ~y are the protein levels.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
X <- data %>% select(c(-Protein)) %>% as.matrix()
Y <- data %>% select(c(Protein)) %>% as.matrix()

A <- t(X) %*% X
b <- t(X) %*% Y
```

3. Try to solve AB = b with default solver solve(). What kind of result did you get? How can this result be explained?

```
B_hat <- solve(A, b, tol=1e-18)

# Error in solve.default(A, b) : system is computationally singular: reciprocal condition number = 7.14
```

Analysis: Running the solve function will return the following error message: "Error in solve.default(A, small_b) : system is computationally singular reciprocal condition number = 7.13971e-17", however this can be negated by adding a tolerance to the kappa function as shown in our function.

This error could be caused by the fact that the matrix is not invertible (singular), and therefor it is impossible to run a regression. Another explanation is that there is high occurance of multicollinearity.

(source: https://stats.stackexchange.com/questions/76488/ error-system-is-computationally-singular-when-running-a-glm)

(source: https://stats.stackexchange.com/questions/90020/ random-effects-model-with-plm-system-is-computationally-singular-error)

4. Check the condition number of the matrix A (function kappa()) and consider how it is related to your conclusion in step 3.

```
kappa(A)
```

```
## [1] 1255749130309473.8
```

Analysis: Following the ?kappa function in R, we get a very large number for the condition number. A large value for the condition number indicates that this matrix is close to being singular.

(source: http://www.cse.iitd.ernet.in/~dheerajb/CS210_lect07.pdf

5. Scale the data set and repeat steps 2-4. How has the result changed and why?

```
X_scale <- scale(X)
Y_scale <- scale(Y)

A_scale <- t(X_scale) %*% X_scale
b_scale <- t(X_scale) %*% Y_scale
```

```r
B_hat_scale <- solve(A_scale, b_scale)

kappa(A_scale)
```

```
## [1] 511646543510.82495
```

Analysis: After scaling the variables, the kappa() value has become smaller. We assume this is because the matrix has now become less likely to be singular, which results in a lower value for the condition number of the matrix.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
x1 <- 1/3; x2 <- 1/4

if(x1-x2==1/12){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}

x3 <- 1; x4<- 1/2
if(x3-x4==1/2){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}

tol <- 1e-9
x1 <- 1/3; x2 <- 1/4

if(abs(x1-x2-1/12) <= tol){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}

x3 <- 1; x4<- 1/2
if(abs(x3-x4-1/2) <= tol){
  print("Subtraction is correct")
}else{
  print("Subtraction is wrong")
}
my_deri <- function(x){
  epi = 10^(-15)
  f_deri <- ((x+epi) - x)/epi
  return(f_deri)
}
my_deri(x=1)
my_deri(x=100000)
```

```r
myvar <- function(myvector){
  #vector <- c(1,2,3,4,5)
  n <- length(myvector)
  answer <- (1/(n-1))*(sum(myvector^2) - (1/n) * ((sum(myvector))^2))
  return(answer)
}

my_rand_num <- rnorm(n=10000, mean=10^8, sd=1)

Y <- vector(mode = "numeric", length = length(my_rand_num))
my_mat <- vector(mode = "numeric", length = length(my_rand_num)+1)

for(i in 2:length(my_rand_num)){
  options(digits = 22)
  X_i = my_rand_num[1:i]
  Y = myvar(X_i) - var(X_i)
  temp <- cbind(i, Y)
  my_mat <- rbind(temp, my_mat)
}

my_mat <- as.data.frame(my_mat)
library(ggplot2)
ggplot(my_mat, aes(x=i, y=Y)) + geom_point() + ggtitle("Plot of Y vs. i")

my_rand_num <- rnorm(n=10000, mean=10^8, sd=1)

myvar_better <- function(x){
  n <- length(x)
  m <- mean(x)
answer <- (1/(n - 1)) * sum((x - m)^2)
return(answer)
}

Y <- vector(mode = "numeric", length = length(my_rand_num))
my_mat2 <- vector(mode = "numeric", length = length(my_rand_num)+1)

for(i in 2:length(my_rand_num)){
  options(digits = 22)
  X_i = my_rand_num[1:i]
  X_i = na.omit(X_i)
  if((myvar_better(X_i) - var(X_i)) <= 1e-15){
    Y = 0
  }else{
      Y = myvar_better(X_i) - var(X_i)
      }
  temp <- cbind(i, Y)
  my_mat2 <- rbind(temp, my_mat2)
}

my_mat2 <- as.data.frame(my_mat2)
library(ggplot2)
ggplot(my_mat2, aes(x=i, y=Y)) + geom_point() + ggtitle("Plot of Y vs. i")
```

```r
library(xlsx)
data <- read.xlsx("tecator.xls", sheetName = "data")
#data <- read.csv("tecator.csv")
data$Sample <- NULL
library(dplyr)

X <- data %>% select(c(-Protein)) %>% as.matrix()
Y <- data %>% select(c(Protein)) %>% as.matrix()

A <- t(X) %*% X
b <- t(X) %*% Y

B_hat <- solve(A, b, tol=1e-18)

# Error in solve.default(A, b) : system is computationally singular: reciprocal condition number = 7.14
kappa(A)

X_scale <- scale(X)
Y_scale <- scale(Y)

A_scale <- t(X_scale) %*% X_scale
b_scale <- t(X_scale) %*% Y_scale

B_hat_scale <- solve(A_scale, b_scale)

kappa(A_scale)
```