Lab1

Andreas C Charitos (andch552), Omkar Bhutra (omkbh878)

29 February 2019

Question 1:

Be careful when comparing

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

Due to underflow the subtraction is displaying the same number although when the digits are increased using options we can see that the number is actually different. Underflow is the loss of significant digits.

On using the function all.equal we can get the solution as required

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

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

Evaluating the results of the 2 snippets we see that in the first occasion we get the wrong print of the if-else statement.The problem lies to the fact that float numbers that have infinite numbers of decimals can't be represented exactly in the binary system in computers due to memory storage limitation.Using print(x1-x2,digits=16) and print(1/12,digits=16) we will see that the resulting floats are (0.08333333333333331,0.08333333333333333) respectfully and they are not the same causing the condition of unerflow which leads to the failure of the if statement and evaluation of else. We can adress this problem using the "all_equals()" in the if statement instead of "==" to compare the numbers and we will see that the if statement will be executed and the correctly print message will be outputed. The second statement is evaluated correctly and we get the correct print output because 1/2 has finite numbers of decimals so we don't have the occurence of underflow here.

Question 2:

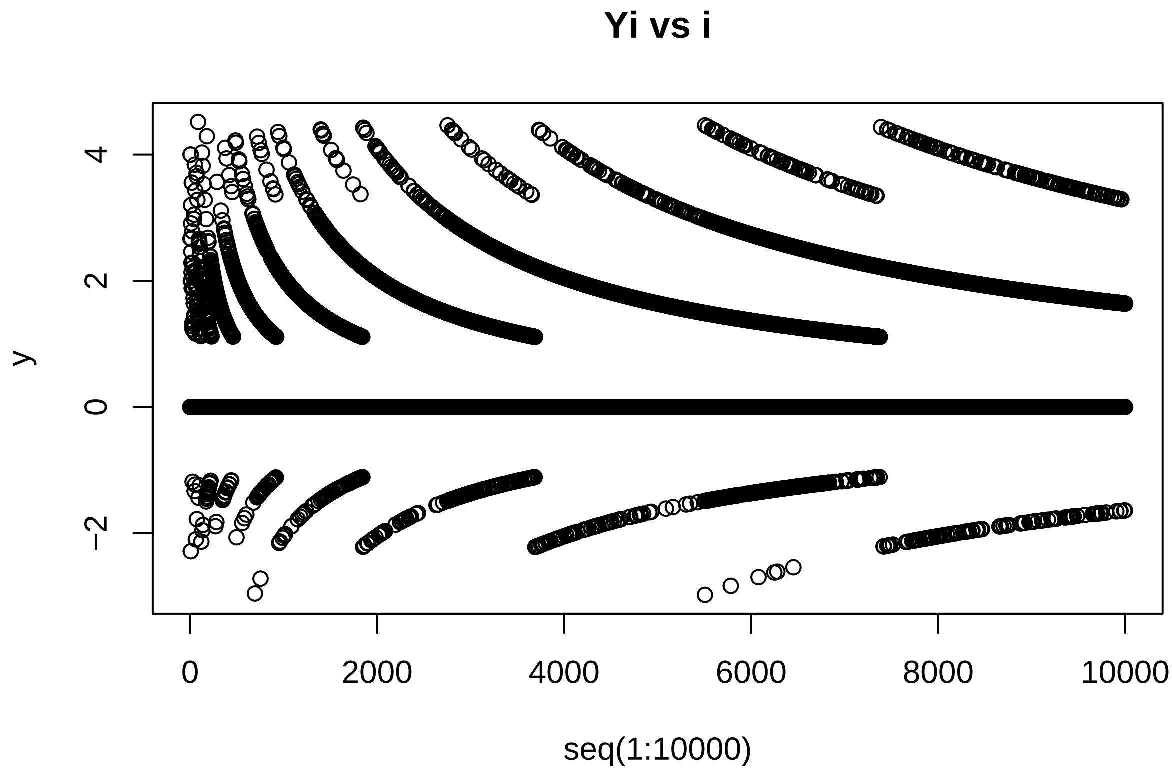
Derivative

```
## ==============================================
##  The derivative for x=1 is : 1.1102230246251565
##  The derivative for x=10000 is : 0
```

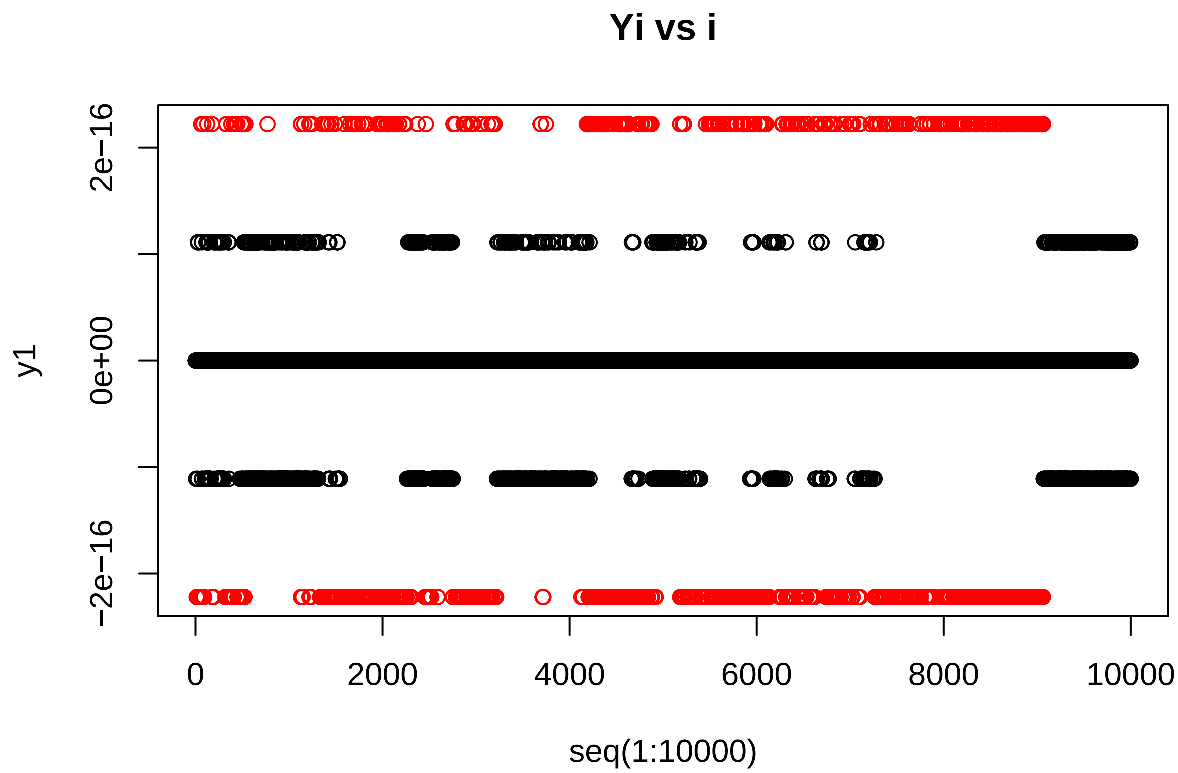
The true value for the function using the function $f(x) = x$ is $f'(x) = \frac{f(x+\epsilon)-f(x)}{\epsilon} = \frac{(x+\epsilon)-x}{\epsilon} = 1$ is always constant with value 1.Regarding the result of the derivative function we see that for $x = 100000$ R doesn't take into account the decimals after a specific number of x and rounds the number to the nearest integer which is 100000 due to underflow occurance so the numeretor of the derivative formula becomes 0 leading finally to 0.When instead $x = 1$ the numerator evaluated is 1.1102230246251565e-15 and the devision with epsilon 10^-15 is just discards the last 15 decimals resulting 1.1102230246251565.

Question 3:

Variance



The plot above shows the dependence Y_i on i with the formula $Var(x) = \frac{1}{n-1}\ (\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2)$ given. As we can see from the plot we got a lot of curves under and over 0 meaning that we have diffrences in the calculations of the variance using the formula given compared with the var() function. This occurs because if we see the formula the term $\sum_{i=1}^{n} x_i^2$ where we square each value of the vector we tend to lose precision because of arithmetic underflow and all the latter calcucations are affected leading to deviations from the true result.



The plot above shows the dependence Y_i on i with the formula $Var(x) = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n-1}$ where μ is the mean. Using the new formula where we center the points arround the mean we see that we have an improvement in the range of the errors and the deviation of the errors is steady and we can see an upper and a lower band with few errors lie beyond these linear bands represented with red in the plot.Also we can observe that the range of the errors is much smaller with means the formula used almost as good as the var() basic function in R.

Question 4:

Linear Algebra

Using the unscaled data first

```
## The result of solve in the unscaled data is :
##  Error in solve.default(A, b) :
##   system is computationally singular: reciprocal condition number = 7.78822e-17
```

When we used the unscaled data solve returns an error that the system is computationally singular and we can't solve the linear equation and the function exits.

```
## The value of condition number is : 1346742158714896.8
```

Printing the number of kappa for the value of A matrix we see that is very big and that implies that the matrix is said to be ill-conditioned a very small change in matrix A will cause a large error in b and makes the solution unstable.

This happens because the tolerence returned is larger than the default threshold set by the function solve (argument tolerence) so an error returned and we cannot get a solution.The torrelance is related to conditon number by the function $tolerance = \frac{1}{conditionnumber}$ so in our case $tolerance = \frac{1}{kappa(A)} = 7.425326e - 16$ and it is bigger that the threshold of $7.425326e - 17$ that is set by solve function as we see in the printed error resulting the end of execution of the function.

Using the scaled data now

```
## The result of solve in the scaled data is :
```

	coefficient
Channel1	-333.7723581641874375
Channel2	-667.2610807294202004
Channel3	1140.3280966267136591
Channel4	-391.2759141564009155
Channel5	1247.1529844508411315
Channel6	-240.5868396857954963
Channel7	-612.4056658136556734
Channel8	249.7665333996565664
Channel9	-399.3523344198769109
Channel10	771.7389786701435241
Channel11	-991.3091492657798653
Channel12	-917.5249334110951622
Channel13	1882.7953033718983988
Channel14	-901.6849924118358786
Channel15	122.9100231856377690
Channel16	-776.9054554395269179
Channel17	510.5409264281486230
Channel18	894.9393744216870346
Channel19	-980.6225625906838559
Channel20	-9.0734249805687046
Channel21	1672.9355298381658486
Channel22	-4120.8801663207659658
Channel23	5612.1140253206194757
Channel24	-4272.0280105506881227
Channel25	1906.0627997120004693
Channel26	-338.0015978357982931
Channel27	51.3341219028812930
Channel28	-690.6121719101871577
Channel29	1340.2310719309848537
Channel30	-1802.1103981019855382
Channel31	1321.7569264046346689
Channel32	950.3754371034500537
Channel33	-1055.2806443510628469
Channel34	-862.5040736009890452
Channel35	1262.7282229681475201
Channel36	-238.6405656807493756
Channel37	-922.9294067309713228
Channel38	857.5065288358730413
Channel39	-1313.9658542000493071
Channel40	2472.9254237011073201
Channel41	-2669.7905230282244702

	coefficient
Channel42	979.1845701106919932
Channel43	1582.5228129695881307
Channel44	-1760.0618630714689061
Channel45	-422.8520254057773968
Channel46	1741.3411474152576375
Channel47	-887.7159427490217922
Channel48	-205.3613848560372901
Channel49	-272.9869413138711138
Channel50	1219.1760041973689113
Channel51	-2108.6616923778556156
Channel52	3797.6577264061820642
Channel53	-5046.1061855032085077
Channel54	4483.4911881428915876
Channel55	-2450.6402972934142781
Channel56	580.6986990963471271
Channel57	-99.2853530209052337
Channel58	22.2488353288514098
Channel59	-267.5521678797088612
Channel60	1040.3858344078735172
Channel61	-1370.6375719483937701
Channel62	1350.3312504164287020
Channel63	-595.5504743115803876
Channel64	670.7214856865929278
Channel65	-1204.4300790776399026
Channel66	1100.6883860869274940
Channel67	-1107.6114575544106629
Channel68	735.8366337001825741
Channel69	-230.1576686547575719
Channel70	-959.8846419397033287
Channel71	988.4639138026977889
Channel72	-538.5485583568369066
Channel73	359.5458612166420380
Channel74	1342.7728566354412578
Channel75	-60.3721512072025348
Channel76	-1938.9788868264736266
Channel77	1114.6085558497597958
Channel78	-225.9533179299097014
Channel79	-70.8482141140722206
Channel80	-2041.8797130989651123
Channel81	3057.2733992743474118
Channel82	-2684.1348453563809926
Channel83	1215.7448495032235769
Channel84	1279.3314054665283948
Channel85	-2416.6551256092452604
Channel86	1975.9707929666433301
Channel87	1988.5490527422625746
Channel88	-6488.3897048616190659
Channel89	5043.3249684079501094
Channel90	901.0776114211523691
Channel91	-1002.0614402561074030
Channel92	-1470.2398945740619638
Channel93	840.5327952596707064

	coefficient
Channel94	608.3534648545289656
Channel95	-1838.6956020550285302
Channel96	1705.2887250944133939
Channel97	-402.2474027076328866
Channel98	-1110.1673074114487463
Channel99	718.5797667625781742
Channel100	74.3366702227628053
Fat	-5.0277328856756220
Moisture	-2.8179177866411296
intercept	17.6827906976687679

Using the scaled data we where able to solve the linear system and get coefficients for every feature value.

```
## The value of condition number is : 490471518993.2923
```

Printing the number of kappa again we can see that is still high but much less that the previous used with the unscaled data and we where able to solve the linear system and get coefficient values.

When we scale the data we see that the linear system did not get any better or worse the linear dependences of the column features are still present but we manage to make the value of condition number smaller with scaling.This is happening because If we look at the definition of the condition number $k(A) = ||A|| * ||A^-1||$ and just by making the range of the columns smaller the magnitude got smaller leading to a smaller value of condition number which is below threshold value of solve function and we manage to get the solution.The tolerence now is $tolerance = \frac{1}{kappa(A1)} = \frac{1}{490471518993} = 2.038854e - 12$ which is smaller than the default $7.425326e - 17$ set by solve so now we are able to get a solution.

Apendix

```r
knitr::opts_chunk$set(echo = TRUE)
options(digits=22)
x1<-1/3;x2<-1/4
if(x1-x2==1/12){
  print("subtraction is correct")
}else{
  print("subtraction is wrong")
}
x1<-1/3;x2<-1/4
if(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")
}
derivative <-function(f,epsilon){
```

```r
  d<-((f+epsilon)-f)/epsilon
  return(d)
}

cat("=============================================\n",
    "The derivative for x=1 is :",derivative(1,10^-15),"\n",
    "The derivative for x=10000 is :",derivative(100000,10^-15))

set.seed(123456)
myvar<-function(vec){

  n<-length(vec)
  variance<-(sum(vec^2)-(sum(vec)^2)/n)/(n-1)
  return(variance)

}

myvec<-rnorm(10000,10^8,1)


y<-double(10000)

for (i in 1:length(myvec)){
  x<-myvec[1:i]
  y[i]<-myvar(x-var(x,na.rm = T))
}

plot(seq(1:10000),y,main="Yi vs i" )

set.seed(12345)
myvar1<-function(v){
  n<-length(v)
  variance<-sum((v-mean(v))^2)/(n-1)
  return(variance)
}


y1<-double(10000)

for (i in 1:length(myvec)){
  x1<-myvec[1:i]
  y1[i]<-myvar1(x1)-var(x1)
}

plot(seq(1:10000),y1,col=ifelse(y1>1.2e-16 | y1< -1.2e-16, "red","black"),
     main="Yi vs i" )



tecator<-readxl::read_excel("tecator.xls")

tecator<-as.data.frame(tecator)
```

```r
X<-tecator[,!names(tecator)%in%c("Sample","Protein")]
X$intercept<-1
X<-as.matrix(X)
y<-tecator$Protein

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

try(solve(A,b))
cat("The result of solve in the unscaled data is : \n","Error in solve.default(A, b) :
  system is computationally singular: reciprocal condition number = 7.78822e-17")

cat("The value of condition number is :",kappa(A))

library(knitr)

X1<-as.data.frame(scale(tecator[,!names(tecator)%in%c("Sample","Protein")]))
X1$intercept<-1
X1<-as.matrix(X1)

y1<-tecator$Protein


A1<-t(X1)%*%X1
b1<-t(X1)%*%y1

cat("The result of solve in the scaled data is : \n")
a<-solve(A1,b1)

kable(a,col.names = c("coefficient") ,top.label="Output solve scaled")


cat("The value of condition number is :",kappa(A1))
```