Computer Lab 1

Thomas Zhang

2016 M02 1

Assignment 1

OK, we are supposed to see if the code below produces teacher true or teacher lied.

```
x1 <- 1/3
x2 <- 1/4
if( (x1 - x2) == 1/12 ){
  print("Teacher was true")
} else{
  print("Teacher was lied")
}
```

```
## [1] "Teacher was lied"
```

OK, the code produced teacher lied. I imagine this is due to the finite nature of the significand of floating point numbers. Therefore, I resolve to make this code produce teacher true by rounding the floats and then comparing them using `==`. It turns out the outcome is true if we round the compared floating numbers to $<$ 16 decimal places. An example is below.

```
x1 <- 1/3
x2 <- 1/4
if( round((x1 - x2),10) == round(1/12,10) ){
  print("Teacher was true")
} else{
  print("Teacher was lied")
}
```

```
## [1] "Teacher was true"
```

Assignment 2

We are to calculate, for `x = 100000` and `epsilon = 10^-15` the derivative of $f(x) = x$ using what british mathematics education call first principles. We implement this in R below.

```
derivative <- function(x,epsilon){
  res <- ((x + epsilon) - x) / epsilon
  return(res)
}

derivative(x = 100000, epsilon = 10^-15)
```

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

The obtained value is zero, while the real value is one. The reason for the difference is cancellation in the numerator between `x + epsilon` and `x` due to the large difference in magnitude between `x` and `epsilon`. When we try a larger epsilon (a smaller difference in magnitude) the obtained value will be one. The numerical breakdown happens when order difference is around 16.

```
derivative(x = 100000, epsilon = 10^-5)
```

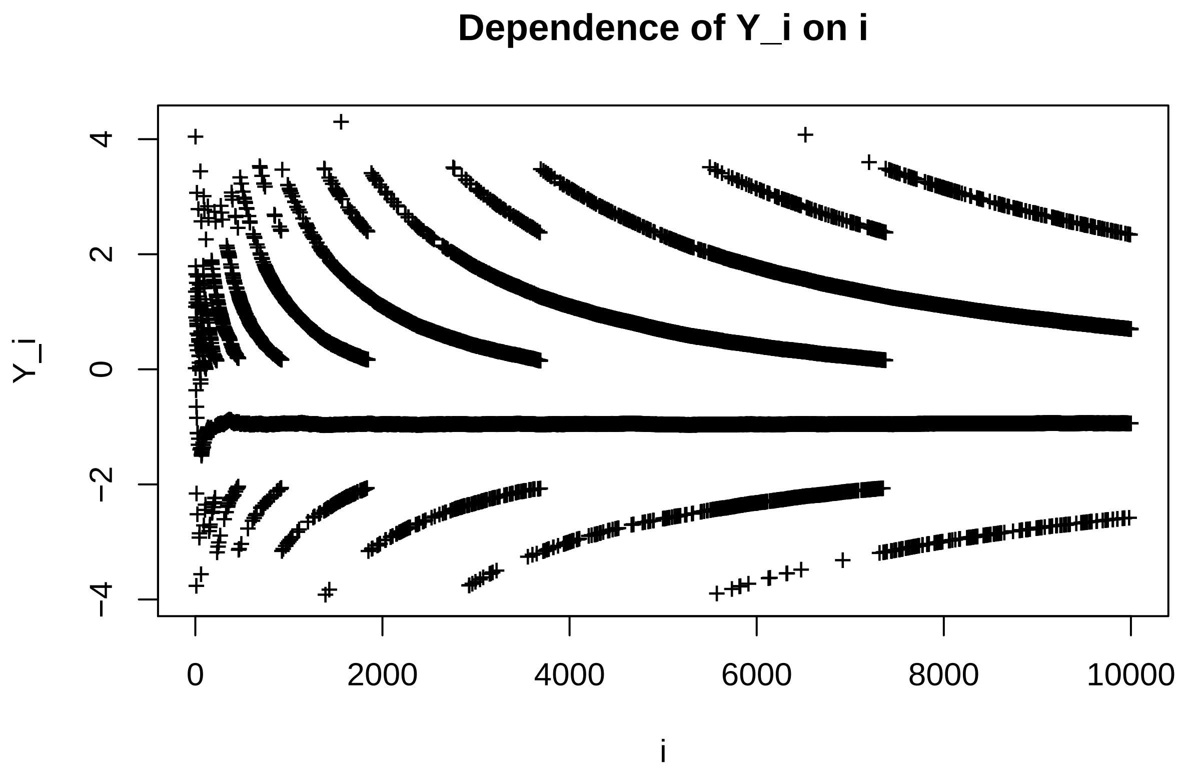
```
## [1] 1
```

Assignment 3

Alright, now we are to create a function called `myvar` which calculates the unbiased estimate of the variance of a vector `x`. Then we are, for `1...length(x)` to calculate the difference between `myvar` and the built-in function `var` and plot the difference versus the number of elements used in the calculation.

```
myvar <- function(x){
  n <- length(x)
  sumsqrd <- sum(x^2)
  sqrdsum <- sum(x)^2
  res <- 1/(n - 1) * (sumsqrd - 1/n * sqrdsum)
  return(res)
}

testvec <- rnorm(10000,10^8,1)
diff <- rep(0,10000)
for(i in 1:10000){
  diff[i] <- myvar(testvec[1:i]) - var(testvec[1:i])
}
plot(1:10000,diff,pch="+",main="Dependence of Y_i on i",xlab="i",ylab="Y_i")
```



We see that `myvar` is not really agreeing on variance with `var`. One reason is possibly the fact that there is cancellation between two terms in `myvar`,(`sumsqrd` and `1/n * sqrdsum`), with large magnitudes but very small differences. This probably leads to roundoff error in floating point arithmetics, and causes the clustering of results seen in the plot.

Assignment 4

We want to solve, in the sense of least squares the linear regression system `A\beta=b` where `A=X'X` are derived from predictors from the `tecator.xls` data sheet and `b=X'Y` is derived from one column chosen as response. We will try to use the built-in `solve` function.

```
X <- as.matrix(data[,-c(1,102)])
#head(X)
Y <- as.matrix(data[,102,drop=FALSE])
A <- t(X) %*% X
b <- t(X) %*% Y
solveres <- solve(A,b) # can not solve the system is ill-conditioned
rcond(A) # rcond returns 7.25 * 10^(-17)
```

Since the `solve` function seems not to work because the condition number of `A` is too large, we will try to scale the data in hope it will reduce condition number and make `solve` work.

```
datascaled <- scale(data)

#head(data)
```

```r
Xscaled <- as.matrix(datascaled[,-c(1,102)])
#head(X)
Yscaled<- as.matrix(datascaled[,102,drop=FALSE])
A <- t(Xscaled) %*% Xscaled
b <- t(Xscaled) %*% Yscaled
solveres <- solve(A,b) #now it works. scaling appears to improve rcond a little
rcond(A) # 7.0 * 10^(-14)
solveres
```

The condition number is smaller now and we are lucky that it works now. Generally we should probably decompose the matrix `A` using QR or SVD or Cholesky decomposition/factorization.

Appendix

R code

```r
library(XLConnect)
wb = loadWorkbook("tecator.xls")
data = readWorksheet(wb,sheet = "data" ,header = TRUE)

#Assignments 1 through 4
x1 <- 1/3
x2 <- 1/4
if( (x1 - x2) == 1/12 ){
  print("Teacher was true")
} else{
  print("Teacher was lied")
}
x1 <- 1/3
x2 <- 1/4
if( round((x1 - x2),10) == round(1/12,10) ){
  print("Teacher was true")
} else{
  print("Teacher was lied")
}
derivative <- function(x,epsilon){
  res <- ((x + epsilon) - x) / epsilon
  return(res)
}

derivative(x = 100000, epsilon = 10^-15)
derivative(x = 100000, epsilon = 10^-5)
myvar <- function(x){
  n <- length(x)
  sumsqrd <- sum(x^2)
  sqrdsum <- sum(x)^2
  res <- 1/(n - 1) * (sumsqrd - 1/n * sqrdsum)
  return(res)
}

testvec <- rnorm(10000,10^8,1)
```

```r
diff <- rep(0,10000)
for(i in 1:10000){
  diff[i] <- myvar(testvec[1:i]) - var(testvec[1:i])
}
plot(1:10000,diff,pch="+",main="Dependence of Y_i on i",xlab="i",ylab="Y_i")
## X <- as.matrix(data[,-c(1,102)])
## #head(X)
## Y <- as.matrix(data[,102,drop=FALSE])
## A <- t(X) %*% X
## b <- t(X) %*% Y
## solveres <- solve(A,b) # can not solve the system is ill-conditioned
## rcond(A) # rcond returns 7.25 * 10^(-17)
## datascaled <- scale(data)
##
## #head(data)
## Xscaled <- as.matrix(datascaled[,-c(1,102)])
## #head(X)
## Yscaled<- as.matrix(datascaled[,102,drop=FALSE])
## A <- t(Xscaled) %*% Xscaled
## b <- t(Xscaled) %*% Yscaled
## solveres <- solve(A,b) #now it works. scaling appears to improve rcond a little
## rcond(A) # 7.0 * 10^(-14)
## solveres
## NA
```