Exercise 6

Question 1

```r
# According to the definition of exercise we have only one vector and we compute the L2 norm
l2norm <- function(x){
  if(!is.numeric(x) | length(x) < 1){
    print('Invalid input: should be a numeric vector of length at least equals to 1')
    return (NaN)
  }
  return (distance <- sqrt( sum(x^2, na.rm = T)))
}


res <- l2norm(c(1,2,3)) # test with some random values

res1 <- l2norm(c(4,3,NA)) # test with NA values in the vector

res2 <- l2norm(c(-1,-2,-3)) # test with negative values

res3 <- l2norm(c('b','b',3)) # test with characters as values
```

```
## [1] "Invalid input: should be a numeric vector of length at least equals to 1"
```

```r
# If we want to compute the euclidian distance between two vectors
# then we have:

euclidian_distance<- function(x, y){
  if(!(is.numeric(x) & is.numeric(y)) | (length(x) | length(y)) < 1){
    print('Invalid input: should be a numeric vector of length at least equals to 1')
    return (NaN)
  }
  return (distance <- sqrt( sum( (x - y)^2 ) ) )
}

dist <- euclidian_distance(c(2,2), c(1,1)) # test 1.41 = sqrt(2)

dist1 <- euclidian_distance(c(2,2), c(3,3)) # test 1.41
```

Question 2

```r
# L1 norm
l1norm <- function(x){
  if(!is.numeric(x) | length(x) < 1){
    print('Invalid input: should be a numeric vector of length at least equals to 1')
    return (NaN)
  }
  return (distance <- sum(abs(x), na.rm=TRUE))
}
```

```r
res0 <- l1norm(c(1,2,3)) # test with some random values

res1 <- l1norm(c(4,3,NA)) # test with NA values in the vector

res2 <- l1norm(c(-1,-2,-3)) # test with negative values

res3 <- l1norm(c('b','b',3)) # test with characters as values
```

```
## [1] "Invalid input: should be a numeric vector of length at least equals to 1"
```

```r
# manhatan distance between two vectors x,y

manhatan_distance <- function(x,y){
  if(!(is.numeric(x) & is.numeric(y)) | (length(x) | length(y)) < 1){
    print('Invalid input: should be a numeric vector of length at least equals to 1')
    return (NaN)
  }
  return (distance <- sum(abs(x-y), na.rm=T))
}
```

Question 3

```r
#Loading the 'FNN' library which includes the knn algorithm
library('FNN')

#We will use this function to normalize(scale) values
normalize <- function(x){
  result <- (x - mean(x))/sd(x)
  return (result)
}

carsData <- read.table('Cars2Data.txt', header = T)
carsData <- na.omit(carsData)

carsData$name <- as.integer(carsData$name)

cars_n <- as.data.frame(lapply(carsData, normalize))

cars.cl <- cars_n$mpg

## Applying K-nn regression to cars dataset

cars.knn <- knn.reg(train = cars_n[-1], test = NULL, y = cars.cl, k = 2)

cars.knn
```

```
## PRESS =  66.49033
## R2-Predict =  0.829948
```

```r
library('FNN')

# Loading the datasets and removing 3 columns ERP, vendor and model
computersData <- read.table('ComputerData.txt', header = T)
computersData <- computersData[, -c(1, 2, 10)]

#computersData$vendor <- as.integer(computersData$vendor)

#computersData$model <- as.integer(computersData$model)

computers_n <- as.data.frame(lapply(computersData, normalize))

pc.cl <- computers_n$PRP

pc.knn <- knn.reg(train = computers_n[-7], test = NULL, y = pc.cl, k = 2)

pc.knn
```

```
## PRESS =  35.15308
## R2-Predict =  0.8309948
```

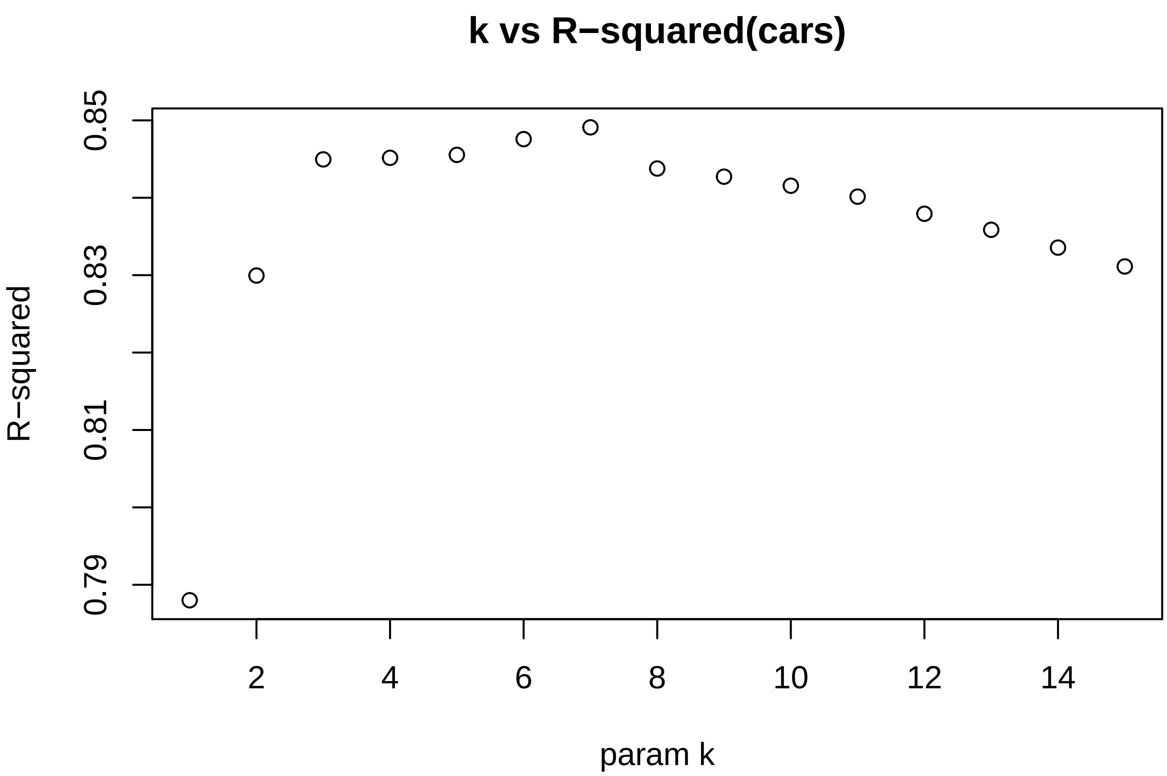
We can see from $R^2 = 0.829$(respectively 0.83 in computers dataset) which indicates that our models explains a relatively large portion of variance in the response variable and PRESS $= 66.49$(respectively 35.15) (*the sums of squares of the predicted residuals*) which is much higher than 1 so it means that our model is a good one.

Question 4

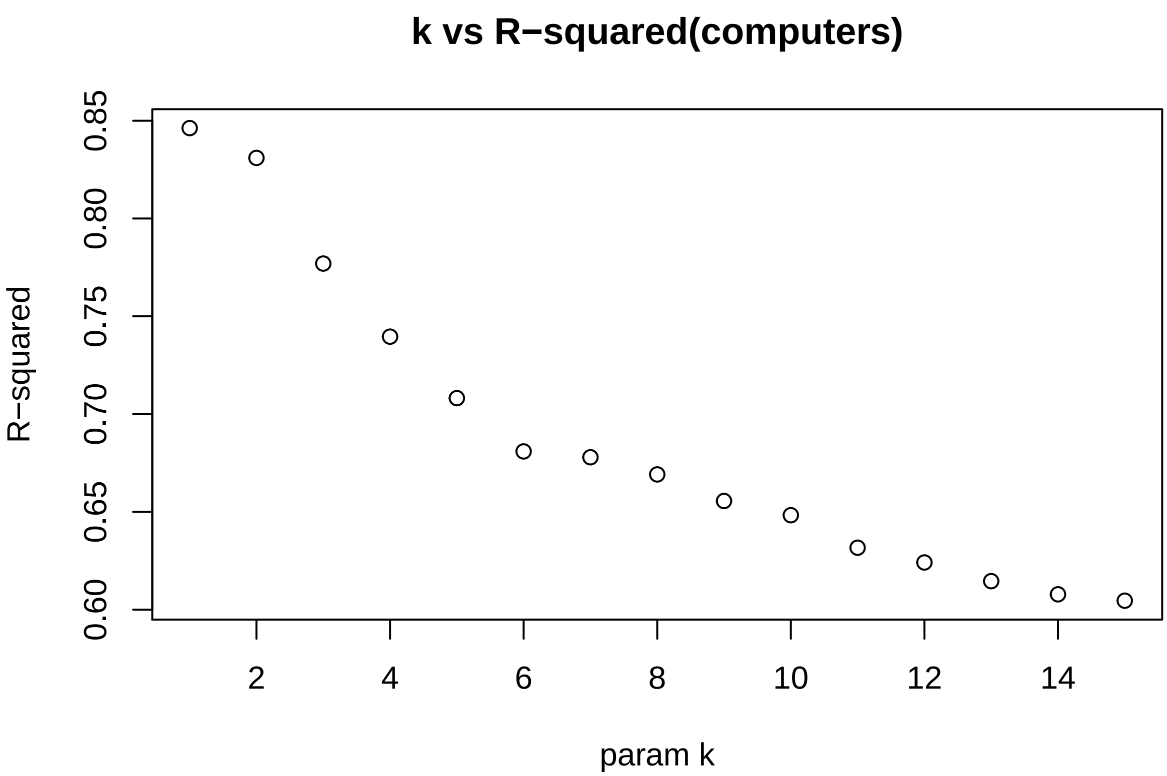
```r
library("FNN")

maxK <- 15

errorRate <- numeric(maxK)
accuracy <- rep(0, maxK)
r2.pc <- rep(0, maxK)
r2.cars <- rep(0, maxK)
for(i in 1:maxK){
  computers_n.knn <- knn.cv(computers_n[-7], computers_n$PRP, prob = F, k = i)
  errorRate[i] <- sum(abs(unclass(computers_n.knn)-computers_n$PRP))
  cat("k-nn:", i, "Error: ", errorRate[i], "\n")

  pc.knn <- knn.reg(train = computers_n[-7], y = pc.cl, k = i)
  r2.pc[i] <- pc.knn$R2Pred
  bestk.pc <- as.integer(which.max(r2.pc))

  cars.knn <- knn.reg(train = cars_n[-1], y = cars.cl, k = i)
  r2.cars[i] <- cars.knn$R2Pred
  bestk.cars <- as.integer(which.max(r2.cars))
}
```

```
## k-nn: 1 Error:  9666
## k-nn: 2 Error:  8296
## k-nn: 3 Error:  6844
## k-nn: 4 Error:  6652
## k-nn: 5 Error:  6204
## k-nn: 6 Error:  5790
## k-nn: 7 Error:  5932.131
## k-nn: 8 Error:  5404.131
## k-nn: 9 Error:  5380.131
## k-nn: 10 Error:  5098.131
## k-nn: 11 Error:  5014.131
## k-nn: 12 Error:  5575.931
## k-nn: 13 Error:  4859.931
## k-nn: 14 Error:  4852.854
## k-nn: 15 Error:  4803
```

```r
plot(1:maxK, r2.cars, main = 'k vs R-squared(cars)', xlab = 'param k', ylab = 'R-squared')
```




```
plot(1:maxK, r2.pc, main = 'k vs R-squared(computers)', xlab = 'param k', ylab = 'R-squared')
```



The best model for *computers dataset* is when $\mathbf{k} = \mathbf{1}$ where we can see also from the plot that we have $R^2 = 0.84$, the higher it is means that our model can better explain the variance of the response value when $\mathbf{k} = \mathbf{1}$.

And for the *cars dataset* we get the best model when **k=7** again based on the R^2 value.

Note that I did not choose the best model based on the error rate like in lectures because that is only related to the training dataset and in a different set the situation might be much more different, so the predicted R squared value was a little bit more trustful for me.