GU4206-GR5206

Name and UNI

3/02/2018

Part 1 (CDC Cancer Data - Subsetting and Plotting)

Consider the following dataset **BYSITE.TXT** taken directly from the Center of Disease Control's website. This dataset describes incidence and mortality crude rates of several types of cancer over time and also includes demographic variables such as **RACE** and **SEX**. The variables of interest in this exercise are: **YEAR**, **RACE**, **SITE**, **EVENT_TYPE**, and **CRUDE_RATE**.

Load in the **BYSITE.TXT** dataset. Also look at the levels of the variable **RACE**.

```
cancer <- read.table("BYSITE.TXT",sep = "|",header=T,
                     na.strings=c("~","."))
dim(cancer)
```

```
## [1] 44982    13
```

```
levels(cancer$RACE)
```

```
## [1] "All Races"                      "American Indian/Alaska Native"
## [3] "Asian/Pacific Islander"         "Black"
## [5] "Hispanic"                       "White"
```

Problem 1.1

Create a new dataframe named **Prostate** that includes only the rows for prostate cancer. Check that the **Prostate** dataframe has 408 rows.

```
#levels(cancer$SITE)
Prostate <- cancer[cancer$SITE=="Prostate",]
dim(Prostate)
```

```
## [1] 408  13
```

Problem 1.2

Using the **Prostate** dataframe from Problem 1.1, compute the average incidence crude rate for each level of **RACE**. To accomplish this task, use the appropriate function from the **apply** family. **Note:** first extract the rows that correspond to **EVENT_TYPE** equals **Incidence**. Then use the appropriate function from the **apply** family with continuous variable **CRUDE_RATE**.

```
#levels(cancer$EVENT_TYPE)
Prostate.I <- Prostate[Prostate$EVENT_TYPE=="Incidence",]
tapply(Prostate.I$CRUDE_RATE,Prostate.I$RACE,mean)
```

```
##                     All Races American Indian/Alaska Native
##                     140.92941                      41.68824
##        Asian/Pacific Islander                         Black
##                      51.67647                     152.40000
##                      Hispanic                         White
##                        53.65882                      141.75882
```

Problem 1.3

Refine the **Prostate** dataframe by removing rows corresponding to **YEAR** level **2010-2014** and removing rows corresponding to **RACE** level **All Races**. After removing the rows, convert **YEAR** into a numeric variable. Check that the new **Prostate** dataframe has 320 rows.

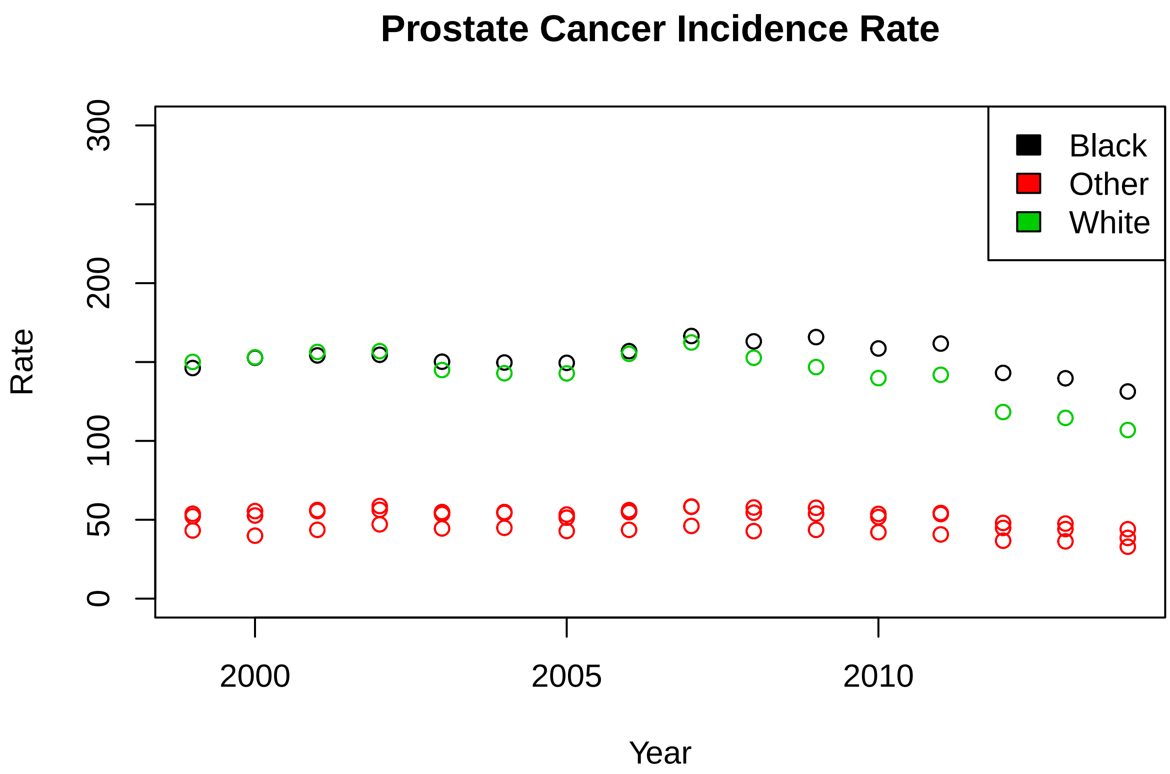
```
#levels(cancer$YEAR)
#levels(cancer$YEAR)
Prostate <- Prostate[Prostate$YEAR!="2010-2014"&Prostate$RACE!="All Races",]
Prostate$YEAR <- as.numeric(as.character(Prostate$YEAR))
dim(Prostate)
```

```
## [1] 320  13
```

Problem 1.4

Create a new variable in the refined **Prostate** dataframe named **RaceNew** that defines three race levels: (1) white, (2) black, and (3) other. Construct a base-R plot that shows the incidence crude rate (not mortality) as a function of time (**YEAR**). Split the scatterplot by **RaceNew**. Make sure to include a legend and label the graphic appropriately.

```
O_W <- ifelse(Prostate$RACE=="White","White","Other")
Prostate$RaceNew <- factor(ifelse(Prostate$RACE=="Black","Black",O_W))
Prostate.I <- Prostate[Prostate$EVENT_TYPE=="Incidence",]
plot(Prostate.I$YEAR,Prostate.I$CRUDE_RATE,
     col=Prostate.I$RaceNew,
     ylim=c(0,300),
     main="Prostate Cancer Incidence Rate",xlab="Year",ylab="Rate")
legend("topright",legend=levels(factor(Prostate$RaceNew)),
       fill=1:length(levels(factor(Prostate$RaceNew))))
```



Part 2 (Basic Web Scraping)

Problem 2.1

Open up the **SP500.html** file to get an idea of what the data table looks like. This website shows the SP500 monthly average closing price for every year from 1871 to 2018. Use regular expressions and the appropriate character-data functions to scrape a "nice" dataset out of the html code. Your final dataframe should have two variables: (1) the variable **Time**, which ranges from 1871 to 2018; (2) the variable **Price** which are the corresponding SP500 price values for each year. Name the final dataframe **SP500.df** and display both the head and the tail of this scrapped dataset.

```
SP500 <- readLines("SP500.html")
head(SP500)
```

```
## [1] "<!DOCTYPE html>"
## [2] "<head>"
## [3] "<style>"
## [4] "#table{width:780px;margin:0 auto}html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,bl
## [5] ""
## [6] "@media only screen and (max-device-width: 779px){html{width:100%}#head{width:100%}#container #a
```

```
date_exp <- "<td class=\"left\">[A-Z][a-z]+ [0-9]{1,2}, [0-9]{4}</td>"
dates.rows <- grep(date_exp ,SP500)

refined_date_exp <- "[0-9]{4}"
date_matches <- gregexpr(refined_date_exp,SP500[dates.rows])
time <- as.numeric(unlist(regmatches(SP500[dates.rows],date_matches)))
#length(time)
```

```r
price_exp  <- "<td class=\"right\">[0-9]+"
price.rows <- grep(price_exp,SP500)
refined_price_exp <- "(>[0-9],[0-9]+\\.[0-9]{2})|(>[0-9]+\\.[0-9]{2})"
price_matches <- gregexpr(refined_price_exp,SP500[price.rows])

price.char <- unlist(regmatches(SP500[price.rows],price_matches))
price.char <- substr(price.char,start=2,stop=nchar(unlist(price.char)))
price.char <- gsub(",", "", price.char)
price <- as.numeric(price.char)
#length(price)
```

```r
SP500.df <- data.frame(Time=time,Price=price)
head(SP500.df)
```

```
##   Time   Price
## 1 2018 2738.60
## 2 2018 2683.73
## 3 2017 2275.12
## 4 2016 1918.60
## 5 2015 2028.18
## 6 2014 1822.36
```

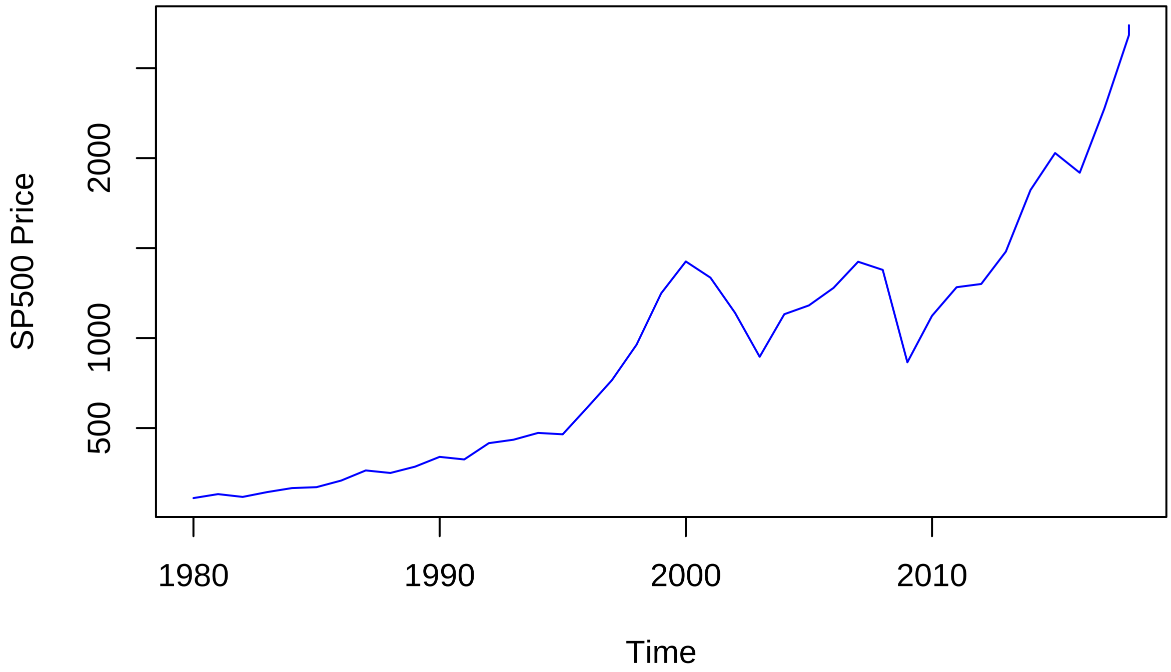
```r
tail(SP500.df)
```

```
##     Time Price
## 144 1876  4.46
## 145 1875  4.54
## 146 1874  4.66
## 147 1873  5.11
## 148 1872  4.86
## 149 1871  4.44
```

Problem 2.2

Create a time series plot of the monthly average SP500 closing price values over the years 1980 to 2018, i.e., use the first 40 lines of **SP500.df**.

```r
plot(SP500.df$Time[1:40],SP500.df$Price[1:40],type="l",xlab="Time",ylab="SP500 Price",col="blue")
```



Part 3 (Knn Regression)

Recall the **kNN.decision** function from class. In the **kNN.decision** function, we classified the market direction using a non-parametric classification method known as "k-nearest neighbors."

```
library(ISLR)
head(Smarket, 3)
```

```
##   Year  Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
## 1 2001 0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
## 2 2001 0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
## 3 2001 1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
```

```
KNN.decision <- function(Lag1.new, Lag2.new, K = 5,
                         Lag1 = Smarket$Lag1,
                         Lag2 = Smarket$Lag2,
                         Dir = Smarket$Direction) {
  n <- length(Lag1)
  stopifnot(length(Lag2) == n, length(Lag1.new) == 1,
            length(Lag2.new) == 1, K <= n)

  dists <- sqrt((Lag1-Lag1.new)^2 + (Lag2-Lag2.new)^2)

  neighbors  <- order(dists)[1:K]
  neighb.dir <- Dir[neighbors]
  choice     <- names(which.max(table(neighb.dir)))
  return(choice)
}
KNN.decision(Lag1.new=2,Lag2.new=4.25)
```

```
## [1] "Down"
```

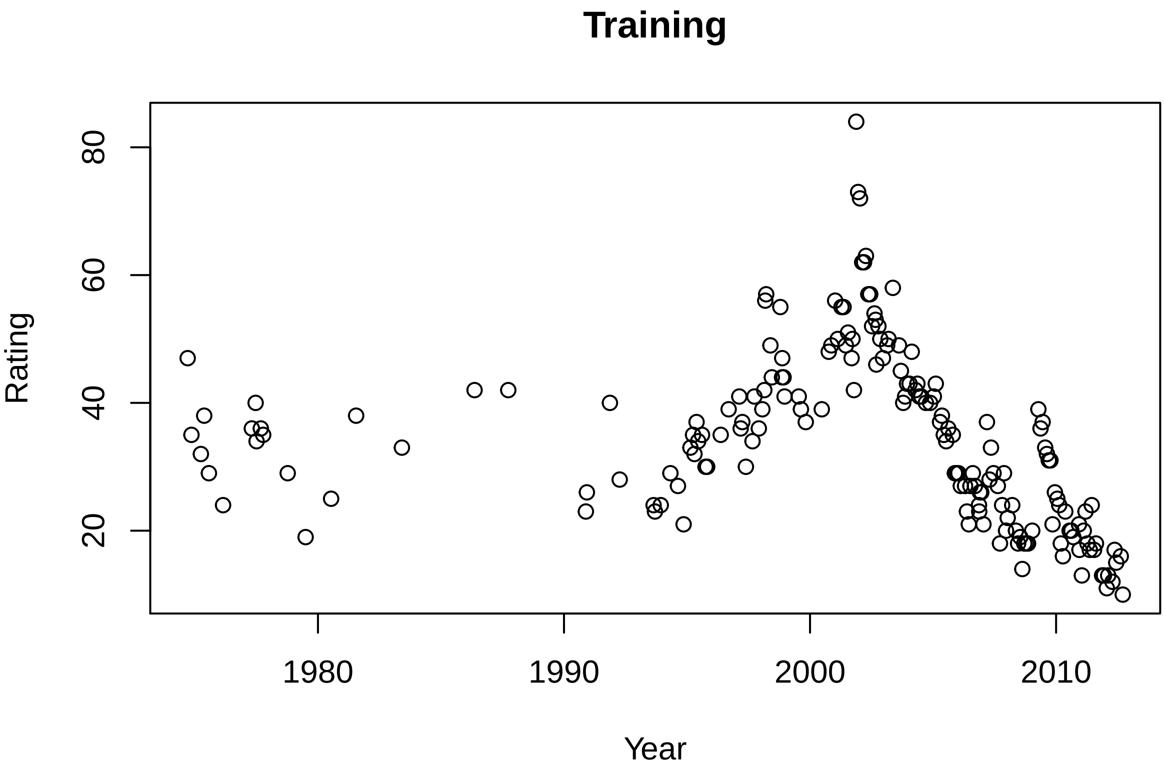
Problem 3.1

In our setting, we consider two datasets that describe yearly US congressional approval ratings over the years 1974 to 2012. The first file **Congress_train.csv** is the training (or model building) dataset and the second file **"Congress_test.csv"** is the test (or validation) dataset. The code below reads in the data and plots each set on separate graphs.

```
Congress_train <- read.csv("Congress_train.csv")
n_train <- nrow(Congress_train)
n_train
```

```
## [1] 181
```

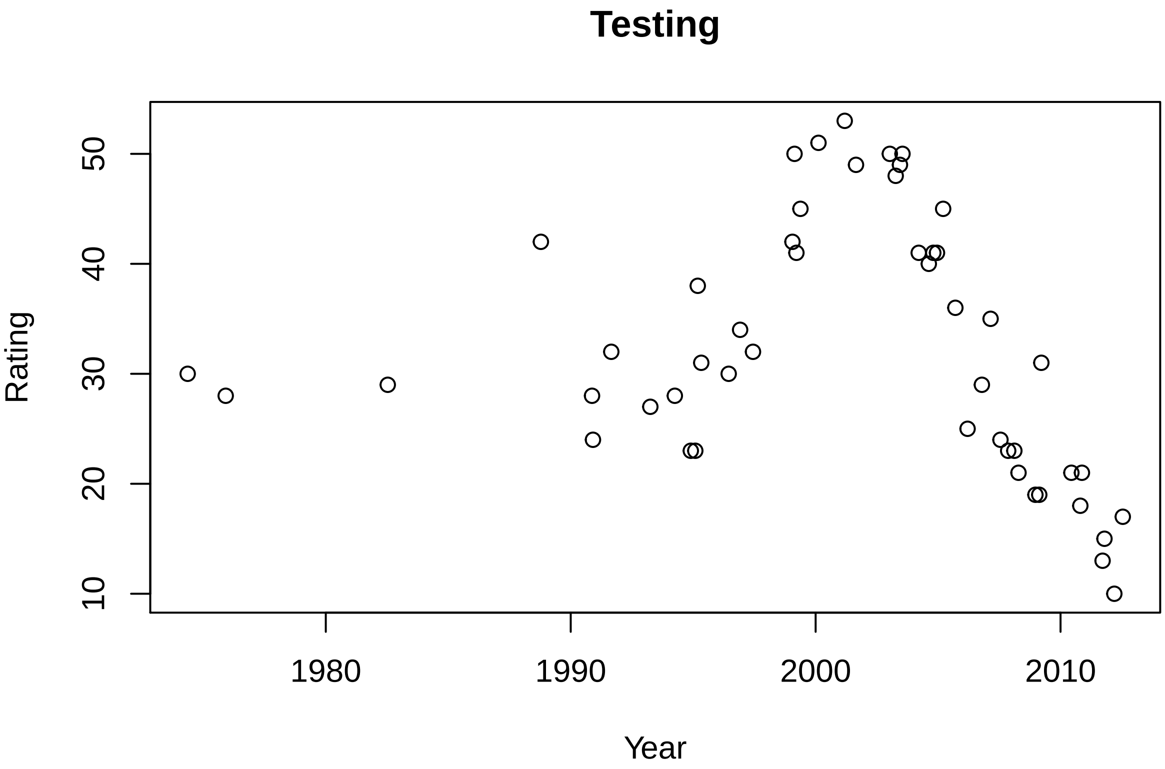
```
plot(Congress_train$Year,Congress_train$Rating,xlab="Year",ylab="Rating",main="Training")
```



```
Congress_test <- read.csv("Congress_test.csv")
n_test <- nrow(Congress_test)
n_test
```

```
## [1] 50
```

```
plot(Congress_test$Year,Congress_test$Rating,xlab="Year",ylab="Rating",main="Testing")
```



Write a function called **kNN.regression** which fits a non-parametric curve to a continuous response. Here you will fit a "moving average" to the yearly congressional approval ratings over the years 1974 to 2012. There is only one feature in this exercise, i.e., **Year** is the only independent variable. Thus for a test time say $t = t_0$, we compute the **arithmetic average rating** of the K closest neighbors of t_0. Using the **Congress_train** dataset, train your model to predict the approval rating when $t = 2000$. Set the tuning parameter to $K = 5$.

Note: to receive full credit, you must extend off of the **kNN.decision** function. You cannot just look up a moving average function online. The new function should also include euclidean distance and the **order** function.

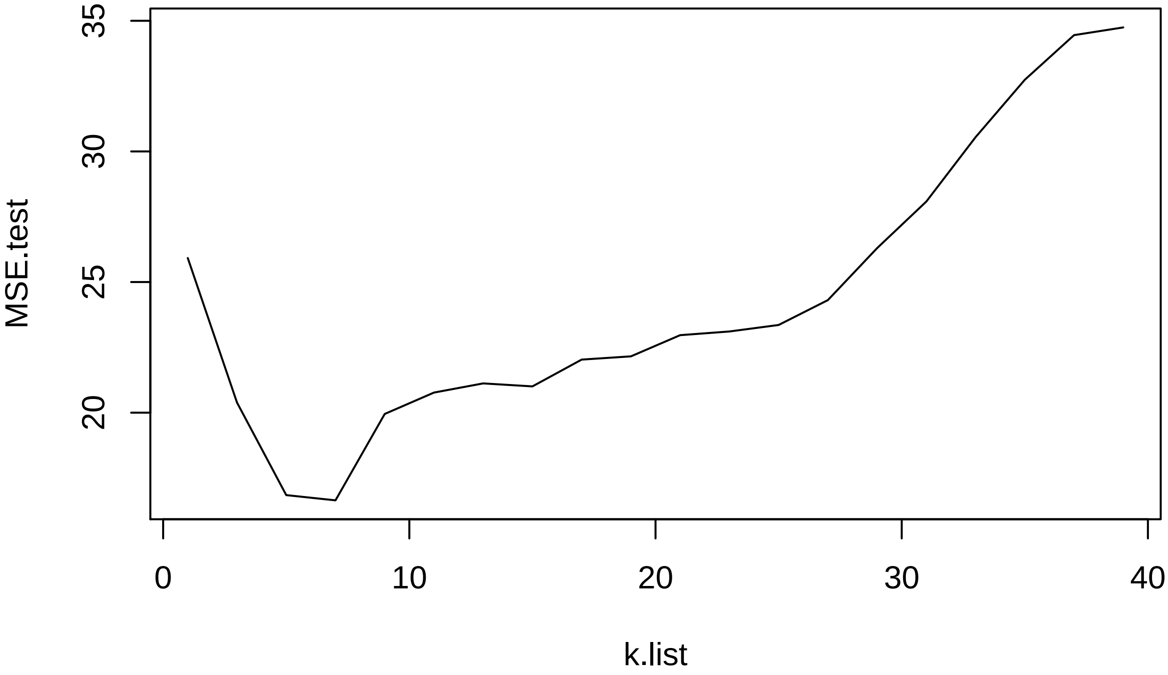
```r
KNN.regression <- function(Year.new,
                           K = 5,
                           Year.train = Congress_train$Year,
                           Congress.train = Congress_train$Rating
                           ) {
  n <- length(Year.train)
  stopifnot(length(Year.train) == n, length(Congress.train) == n,
            length(Year.new) == 1, K <= n)

  dists <- sqrt((Year.train-Year.new)^2)

  neighbors  <- order(dists)[1:K]
  neighb.Congress <- Congress.train[neighbors]
  return(mean(neighb.Congress ))
}
KNN.regression(Year.new=2000)
```

```
## [1] 40.8
```

Problem 3.2

Compute the **test mean squre error** using neighborhood sizes $K = 1, 3, 5, \cdots, 39$. In this exercise you will train the model using **Congress_train** and assess its performance on the test data **Congress_test** with the different tuning parameters K. Plot the test mean square error as a function of K and choose the best value of the tuning parameter based on this output.

```
k.list <- seq(1,39,by=2)
MSE.test <- rep(NA,length(k.list))
test.times <- Congress_test$Year
counter <- 0
 for (k in k.list) {
   counter <- 1 + counter
   Rating.test <- sapply(test.times,KNN.regression,
                         K=k,
                         Year.train=Congress_train$Year,
                         Congress.train=Congress_train$Rating)
   MSE.test[counter] <- mean((Congress_test$Rating-Rating.test)^2)
 }
plot(k.list,MSE.test,type = "l")
```



```
k.list[which.min(MSE.test)]
```

```
## [1] 7
```

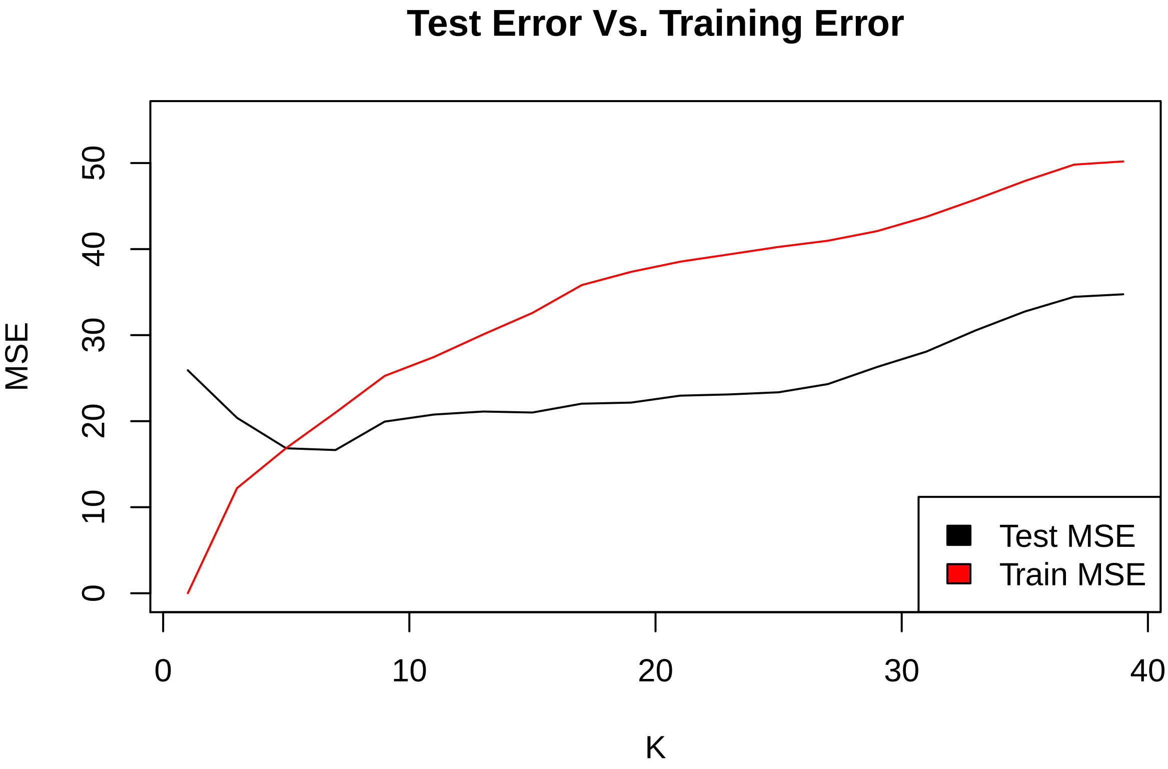
```
MSE.test[which.min(MSE.test)]
```

```
## [1] 16.64571
```

Problem 3.2

Compute the **training mean squre error** using neighborhood sizes $K = 1, 3, 5, \cdots, 39$. In this exercise you will train the model using **Congress_train** and assess its performance on the training data **Congress_train** with the different tuning parameters K. Plot both the test mean square error and the training mean square error on the same graph. Comment on any interesting features/patterns displayed on the graph.

```
k.list <- seq(1,39,by=2)
MSE.train <- rep(NA,length(k.list))
test.times <- Congress_train$Year
counter <- 0
 for (k in k.list) {
   counter <- 1 + counter
   Rating.test <- sapply(test.times,KNN.regression,
                         K=k,
                         Year.train=Congress_train$Year,
                         Congress.train=Congress_train$Rating)
   MSE.train[counter] <- mean((Congress_train$Rating-Rating.test)^2)
 }
plot(k.list,MSE.test,type = "l",ylim=c(0,55),ylab="MSE",xlab="K",main="Test Error Vs. Training Error",c
lines(k.list,MSE.train,col=2)
legend("bottomright",legend=c("Test MSE","Train MSE"),fill=1:2)
```

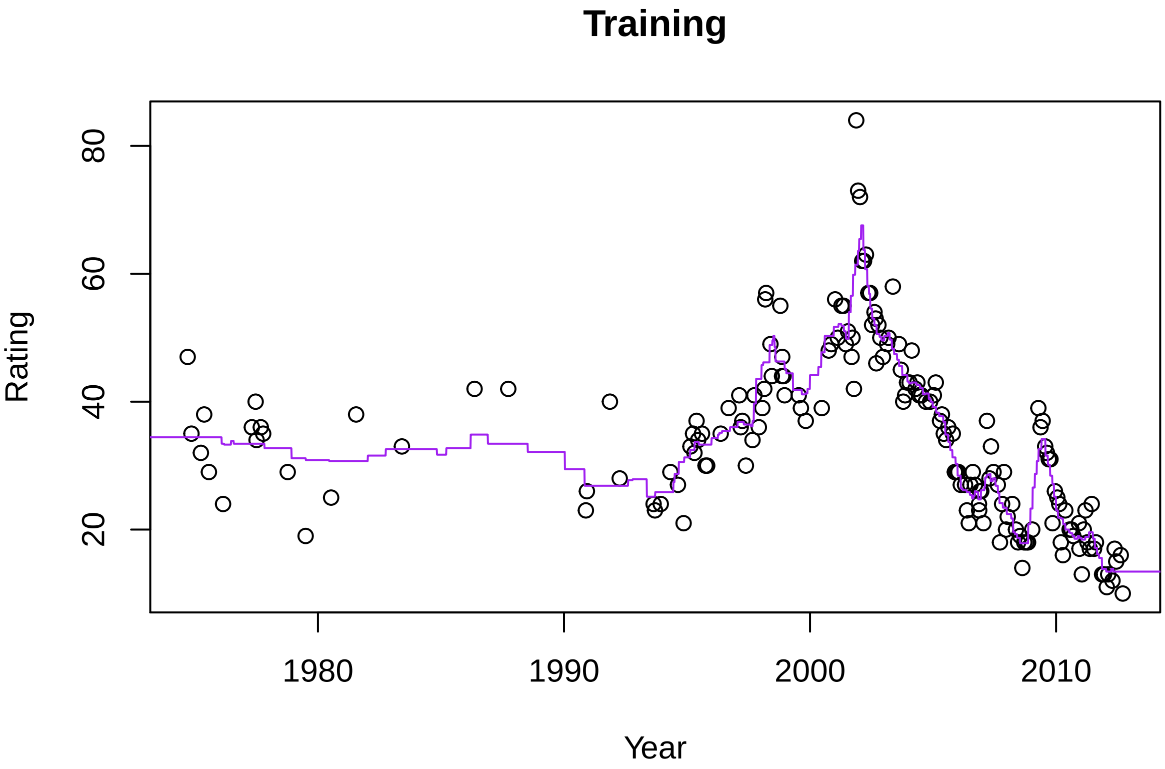


Problem 3.3 (Extra Credit)

Plot the kNN-regression over the training data set **Congress_train** using optimal tuning parameter K. In this plot, the years must be refined so that the smoother shows predictions for all years from 1973 to 2015.

```
test.times <- seq(1973,2015,by=.01)
Rating.test <- sapply(test.times,KNN.regression,
                      K=7,
                      Year.train=Congress_train$Year,
                      Congress.train=Congress_train$Rating
                      )
```

```
plot(Congress_train$Year,Congress_train$Rating,xlab="Year",ylab="Rating",main="Training")
lines(test.times,Rating.test,col="purple")
```


Training
Rating
Year