GU4206-GR5206

Name and UNI

3/02/2018

The STAT GU4206/GR5206 Spring 2018 Midterm is open notes, open book(s), open computer and online resources are allowed. Students are **not** allowed to communicate with any other people regarding the exam with the exception of the instructor (Gabriel Young) and TA (Fan Gao). This includes emailing fellow students, using WeChat and other similar forms of communication. If there is any suspicion of one or more students cheating, further investigation will take place. If students do not follow the guidelines, they will receive a zero on the exam and potentially face more severe consequences. The exam will be posted on Canvas at 10:05AM. Students are required to submit both the .pdf and .Rmd files on Canvas (or .html if you must) by 12:40PM. Late exams will not be accepted. If for some reason you are unable to upload the completed exam on Canvas by 12:40PM, then immediately email markdown file to the course TA (fg2425).

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)
# code goes here
```

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**.

```r
#levels(cancer$EVENT_TYPE)
# code goes here
```

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.

```r
#levels(cancer$YEAR)
# code goes here
```

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.

```r
# code goes here
```

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.

```r
setwd("~/Desktop/Data")
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
```

```r
# code goes here
```

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
# code goes here
```

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."

```r
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
```

```r
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"
```

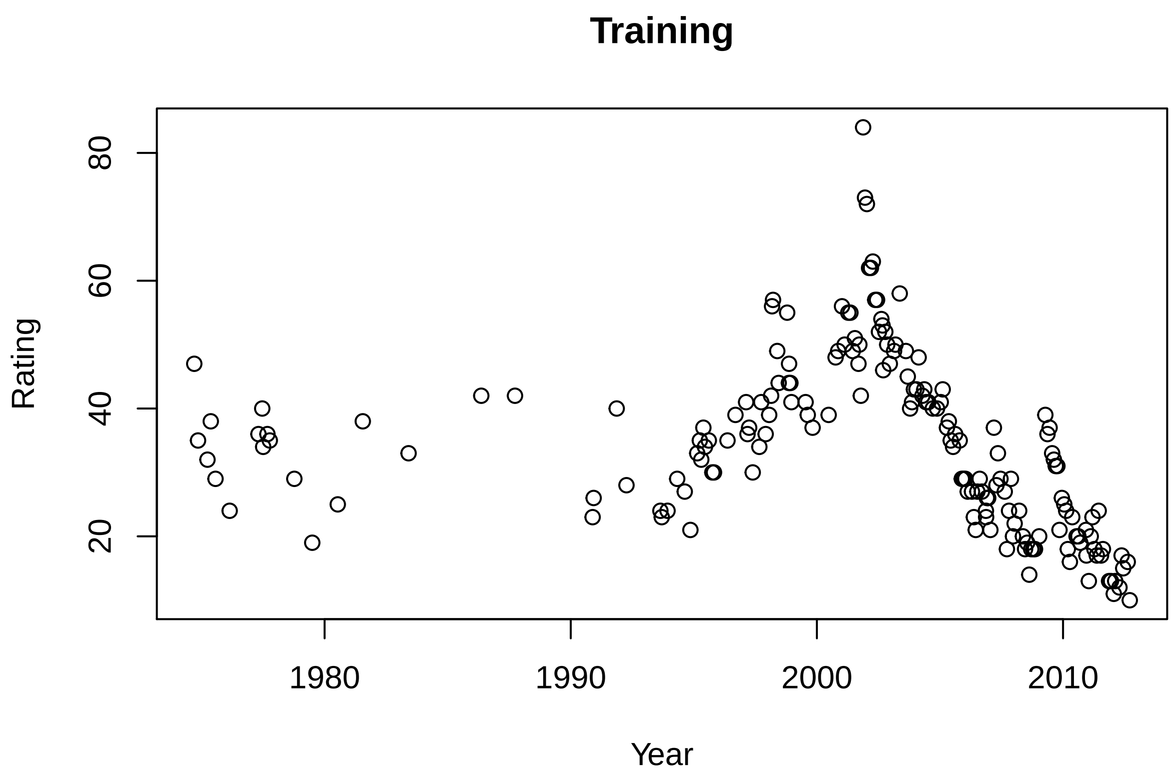
Problem 3.1

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.

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

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

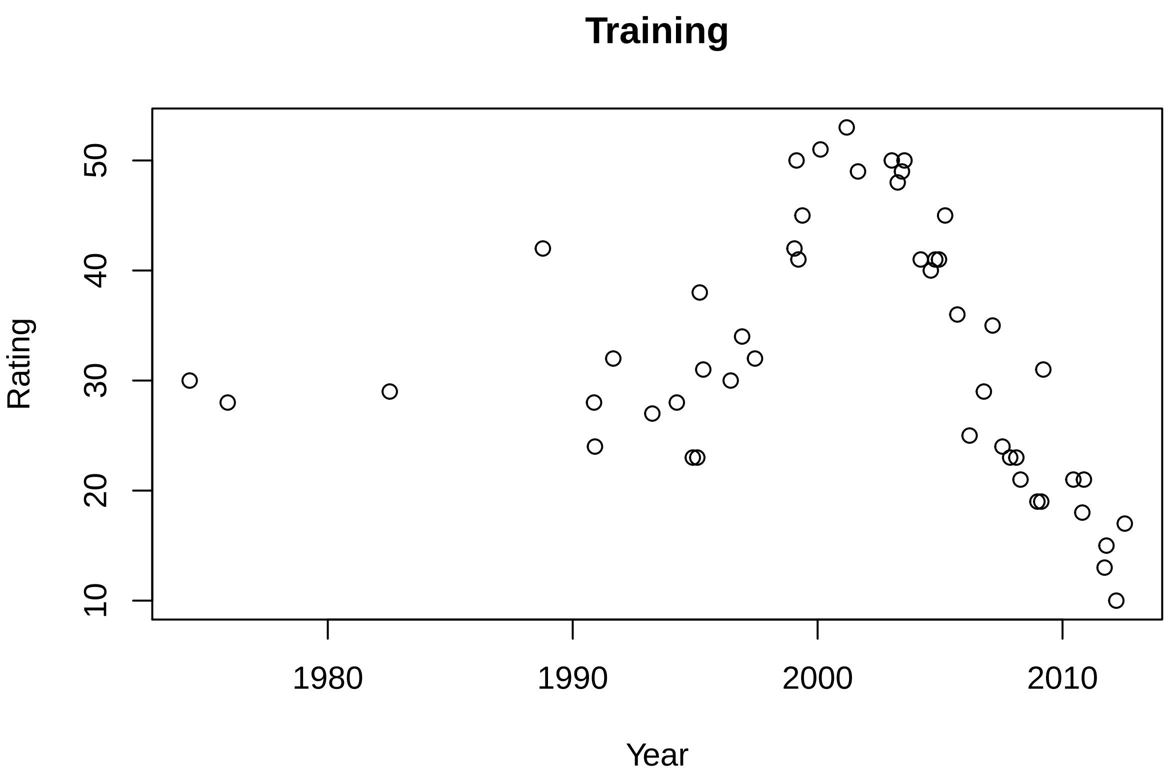
```r
plot(Congress_train$Year,Congress_train$Rating,xlab="Year",ylab="Rating",main="Training")
```




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

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

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



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.

```
# code goes here
```

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.

```
# code goes here
```

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.

```r
# code goes here
```

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.

```r
# code goes here
```