Practicum_1

Minxin Cheng

0. Load packages

```
#install.packages("DMwR")

library(dplyr)
library(ggplot2)
library(ggpubr)
library(reshape2)
require(zoo)
library(class)
library(gmodels)
library(DMwR)
```

Problem 1

Question 1

Download the data set (glass data) along with its explanation. Note that the data file doew not contain header names, you may wish to add those. The description of each column can be found in the data set explanation.

```
# read csv
glassData <- read.csv("glass.data",
                      header = FALSE)
# assign column names
colnames(glassData) <- c("id", "refIdx", "Na", "Mg",
                         "Al", "Si", "K", "Ca", "Ba",
                         "Fe", "type")
```

Question 2

Explore the data set to get a sense of the data and to get comfortable with it.

```
# check if there is any missing values
any(is.na(glassData))
```

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

```
# check if the data is structured
str(glassData)
```

```
## 'data.frame':    214 obs. of  11 variables:
##  $ id    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ refIdx: num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na    : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg    : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al    : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si    : num  71.8 72.7 73 72.6 73.1 ...
##  $ K     : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca    : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe    : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ type  : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
# check each column's type see if anyone needs to be changed
sapply(glassData, class)
```

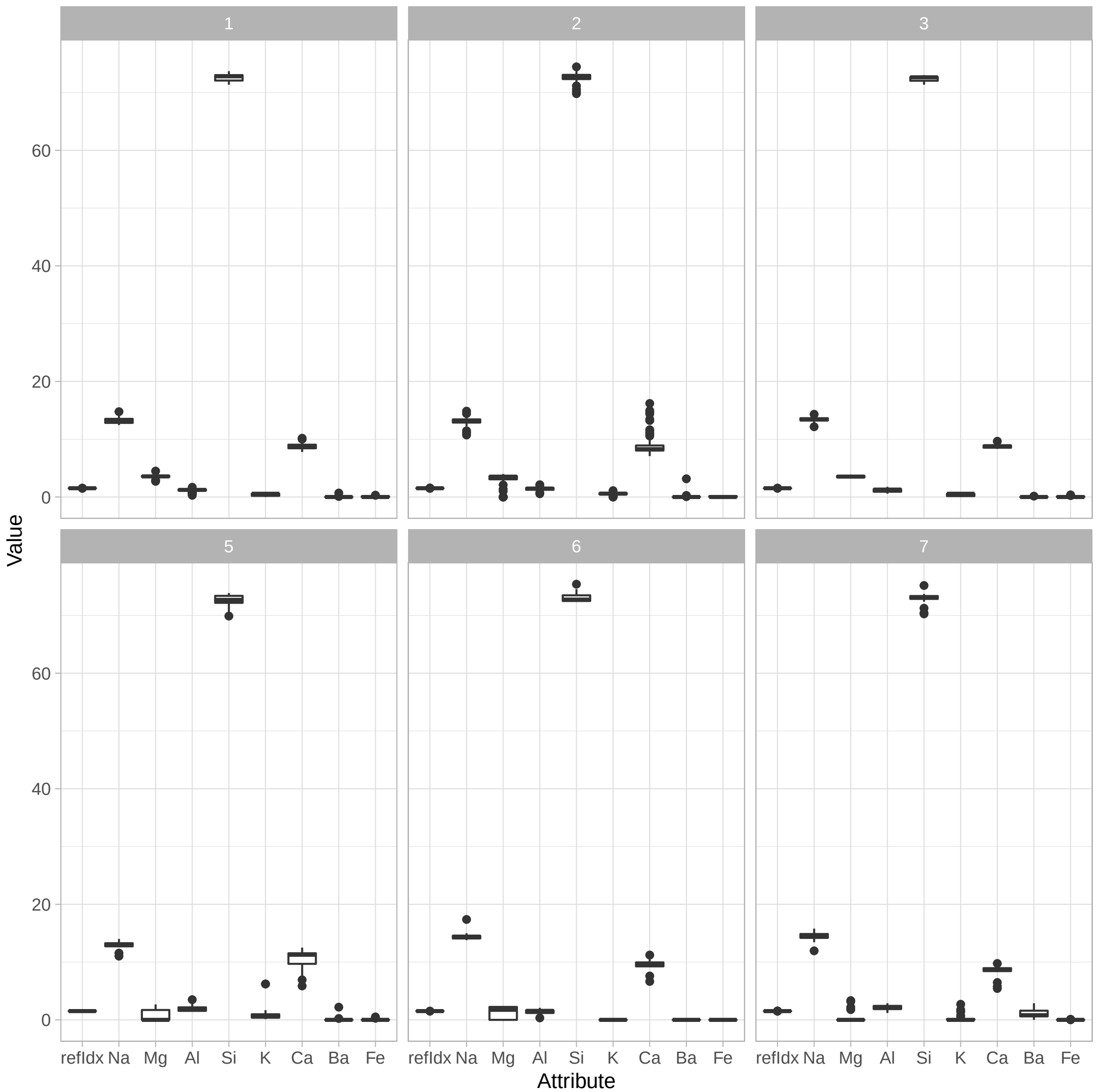
```
##        id    refIdx        Na        Mg        Al        Si         K        Ca
## "integer" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##        Ba        Fe      type
## "numeric" "numeric" "integer"
```

```r
# get an overall information of the data set
summary(glassData)
```

```
##        id             refIdx            Na              Mg       
##  Min.   :  1.00   Min.   :1.511   Min.   :10.73   Min.   :0.000  
##  1st Qu.: 54.25   1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115  
##  Median :107.50   Median :1.518   Median :13.30   Median :3.480  
##  Mean   :107.50   Mean   :1.518   Mean   :13.41   Mean   :2.685  
##  3rd Qu.:160.75   3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600  
##  Max.   :214.00   Max.   :1.534   Max.   :17.38   Max.   :4.490  
##        Al              Si              K                Ca        
##  Min.   :0.290   Min.   :69.81   Min.   :0.0000   Min.   : 5.430  
##  1st Qu.:1.190   1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240  
##  Median :1.360   Median :72.79   Median :0.5550   Median : 8.600  
##  Mean   :1.445   Mean   :72.65   Mean   :0.4971   Mean   : 8.957  
##  3rd Qu.:1.630   3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172  
##  Max.   :3.500   Max.   :75.41   Max.   :6.2100   Max.   :16.190  
##        Ba              Fe               type     
##  Min.   :0.000   Min.   :0.00000   Min.   :1.00  
##  1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:1.00  
##  Median :0.000   Median :0.00000   Median :2.00  
##  Mean   :0.175   Mean   :0.05701   Mean   :2.78  
##  3rd Qu.:0.000   3rd Qu.:0.10000   3rd Qu.:3.00  
##  Max.   :3.150   Max.   :0.51000   Max.   :7.00  
```

```r
# box plots to get an overall sense by glass type
## extract all glass type information
glassType <- glassData %>%
  select(id, type)
## reshape the data
glassOverview <- melt(glassData[, 1:10],
```

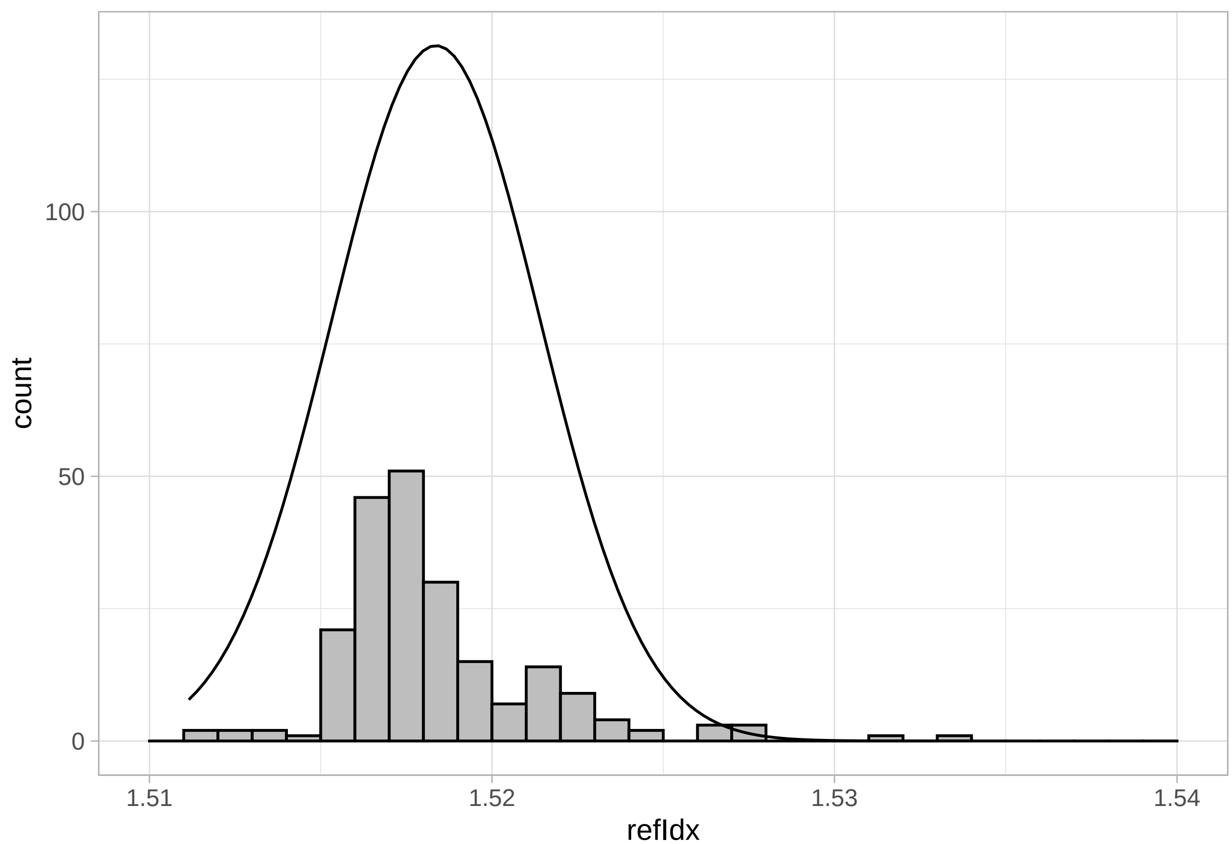
```r
                              id.vars = "id",
                              variable.name = "attribute")
## add type information to the reshaped data
glassOverview <- glassOverview %>%
  left_join(glassType, by = "id")
## box plot
ggplot(glassOverview, aes(x = attribute, y = value)) +
  geom_boxplot() +
  facet_wrap(~type) +
  labs(x = "Attribute", y = "Value") +
  theme_light()
```



Question 3

Create a histogram of column 2 (refractive index) and overlay a normal curve.

```
ggplot(glassData, aes(x = refIdx)) +
  geom_histogram(breaks = seq(1.510, 1.540, by = 0.001),
                 colour = "black",
                 fill = "grey") +
  stat_function(fun = dnorm,
                args = list(mean = mean(glassData$refIdx),
                            sd = sd(glassData$refIdx))) +
  theme_light()
```

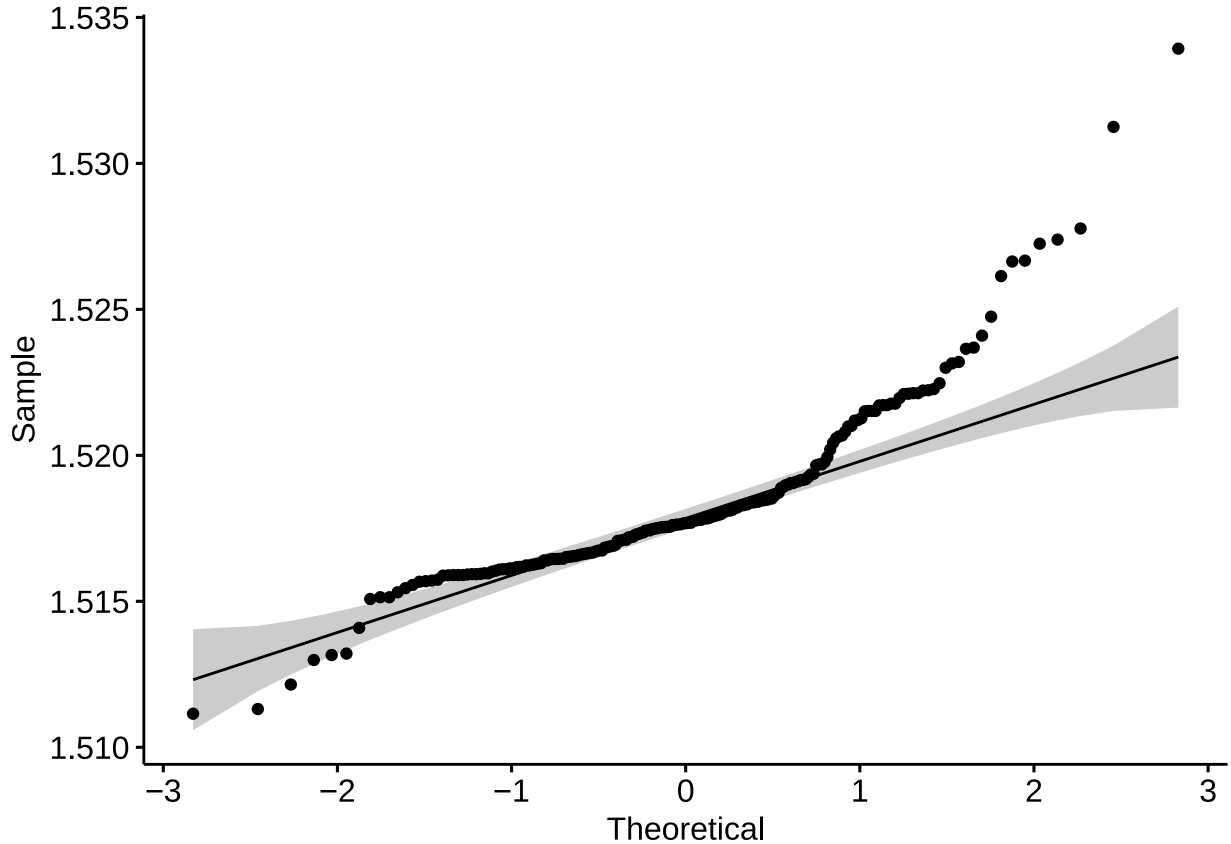


Question 4 Test normality of column 2 by performing either a Shapiro-Wilk or Kolmogorov-Smirnof test. Describe what you found.

```
# run shapiro test
shapiro.test(glassData$refIdx)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glassData$refIdx
## W = 0.86757, p-value = 1.077e-12
```

```r
# do a qq plot to visualize
ggqqplot(glassData$refIdx)
```



As the Shapiro test result, $p < 0.05$, therefore column 2 is not normally distributed. Q-Q plot also supported it.

Question 5

Identify any outliers for the columns using a z-score deviation approach. i.e., consider any values that are more than 2 standard deviations from the mean as outliers. Which are your outliers for each column? What would you do? Summarize potential strategies in your notebook.

```r
# normalize all columns of the dataset except id and type columns
glassDataNorm <- as.data.frame(scale(glassData[, 2:10]))
# summary the normalized dataset to get an overview
summary(glassDataNorm)
```

```
##      refIdx              Na                Mg                Al
##  Min.   :-2.3759   Min.   :-3.2793   Min.   :-1.8611   Min.   :-2.3132
##  1st Qu.:-0.6069   1st Qu.:-0.6127   1st Qu.:-0.3948   1st Qu.:-0.5106
##  Median :-0.2257   Median :-0.1321   Median : 0.5515   Median :-0.1701
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2608   3rd Qu.: 0.5108   3rd Qu.: 0.6347   3rd Qu.: 0.3707
##  Max.   : 5.1252   Max.   : 4.8642   Max.   : 1.2517   Max.   : 4.1162
##        Si                K                 Ca                Ba
```

```
##  Min.   :-3.6679   Min.   :-0.76213   Min.   :-2.4783   Min.   :-0.3521
##  1st Qu.:-0.4789   1st Qu.:-0.57430   1st Qu.:-0.5038   1st Qu.:-0.3521
##  Median : 0.1795   Median : 0.08884   Median :-0.2508   Median :-0.3521
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5636   3rd Qu.: 0.17318   3rd Qu.: 0.1515   3rd Qu.:-0.3521
##  Max.   : 3.5622   Max.   : 8.75961   Max.   : 5.0824   Max.   : 5.9832
##        Fe
##  Min.   :-0.5851
##  1st Qu.:-0.5851
##  Median :-0.5851
##  Mean   : 0.0000
##  3rd Qu.: 0.4412
##  Max.   : 4.6490
```

```r
sapply(glassDataNorm, class)
```

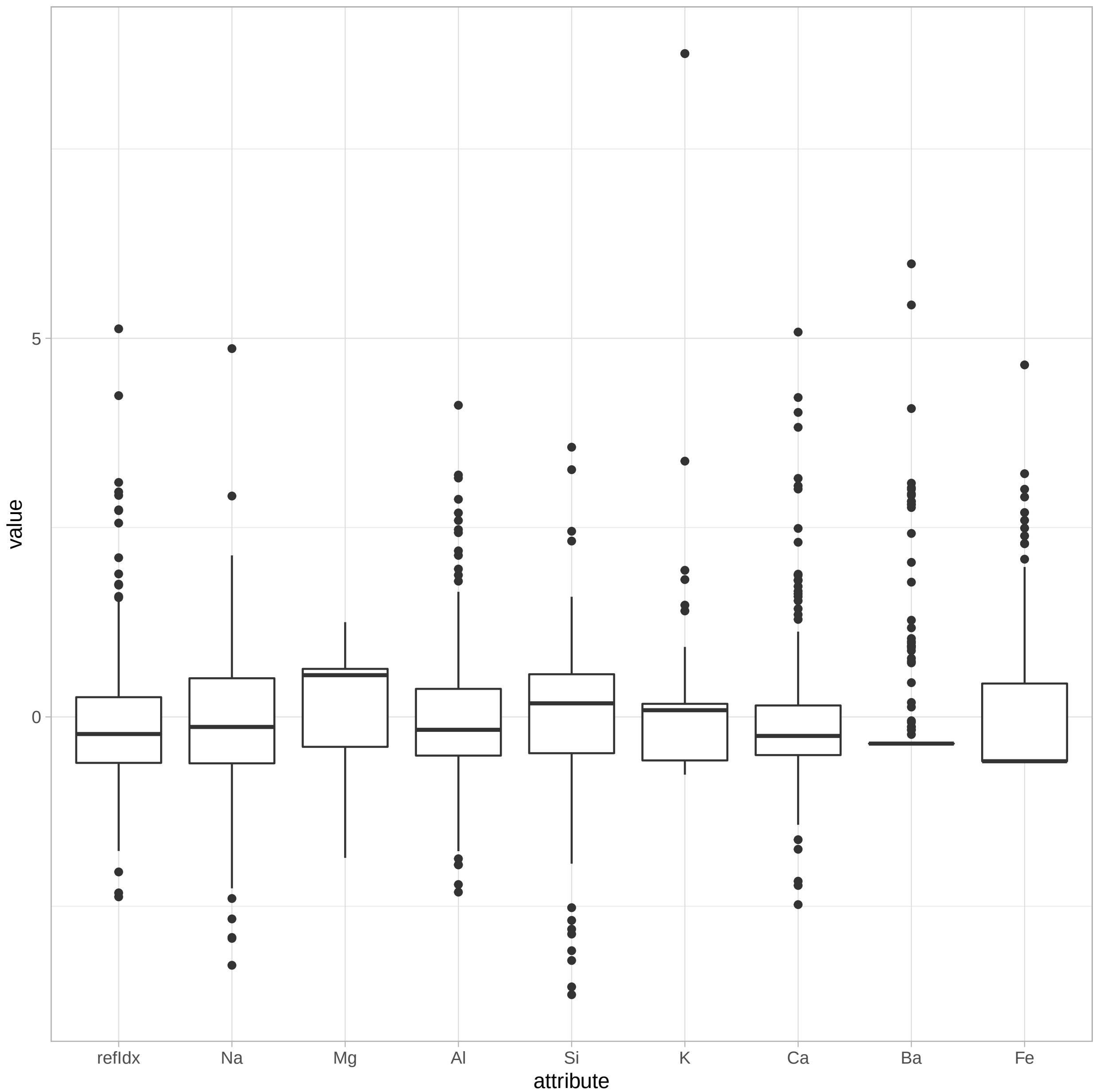
```
##    refIdx        Na        Mg        Al        Si         K        Ca        Ba
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##        Fe
## "numeric"
```

```r
# find out outliers as long as there is one or more columns value is more than
# 2 standard deviations from the mean
outlier <- glassDataNorm %>%
  filter_all(any_vars(abs(.) > 2))
```

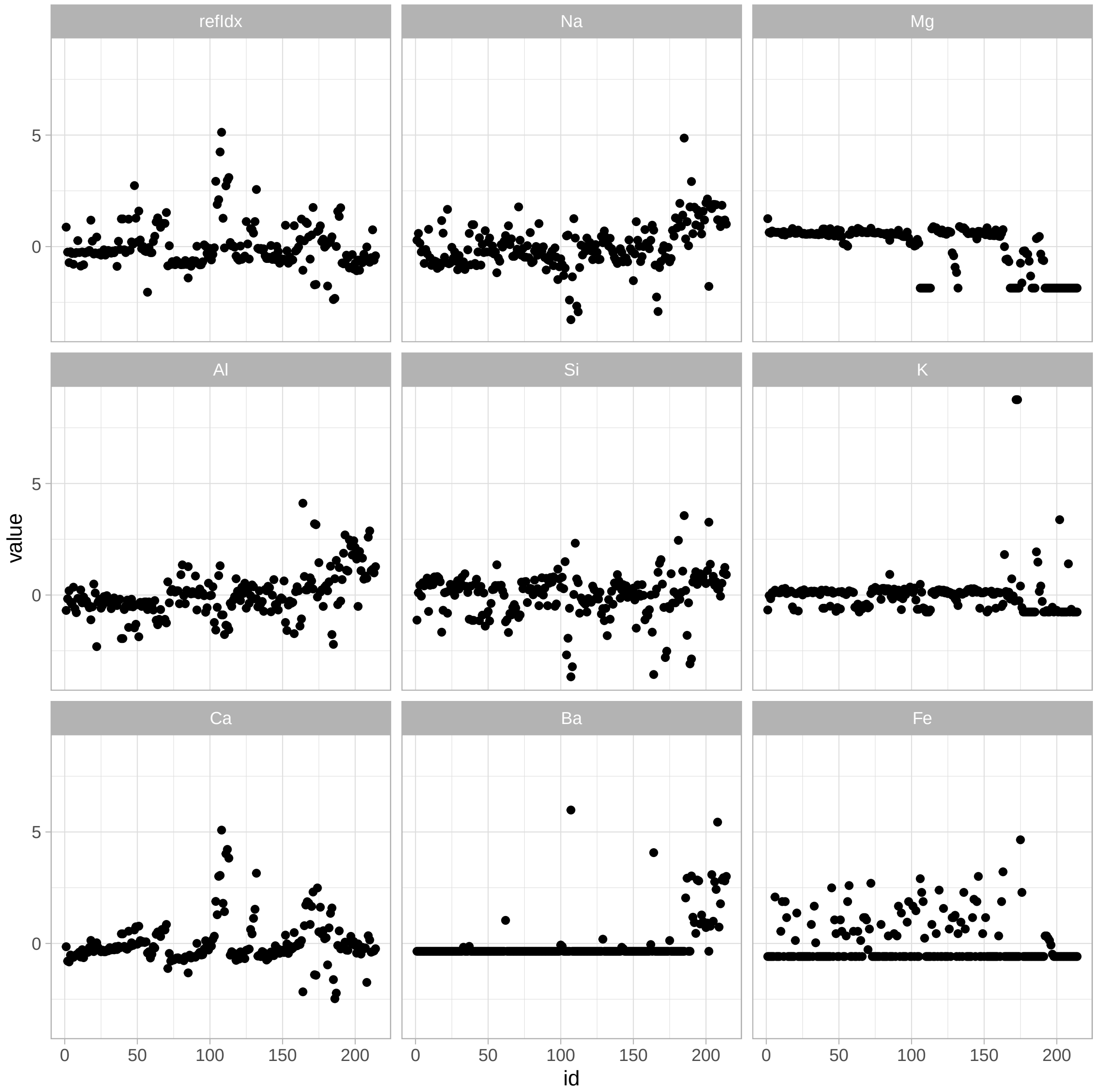
Based on the criteria above, as long as tany of the attribute has a value that is 2 standard deviations from the mean, this id is defined as an outlier. The result returned 53 cases that include outliers, it is about 1/4 of the data, we shouldn't just removed them all.

Below I used some basic visualization (scatter plot, box plot) to see if they are helpful to visually define the outlier, then I used local outlier factor to find out outliers based on density following the instruction provided in Module 3's class material.

```r
# box plot to see if we can remove the outliers from there
## reshape the dataset
glassDataCheck <- cbind(glassDataNorm,
                        id = glassData$id)
glassDataCheck <- melt(glassDataCheck,
                       id.vars = "id",
                       variable.name = "attribute")
## box plot
ggplot(glassDataCheck, aes(x = attribute, y = value)) +
  geom_boxplot() +
  theme_light()
```




```
# scatter plot to see if we can remove the outliers from there
ggplot(glassDataCheck, aes(x = id, y = value)) +
  geom_point() +
  facet_wrap(~attribute) +
  theme_light()
```

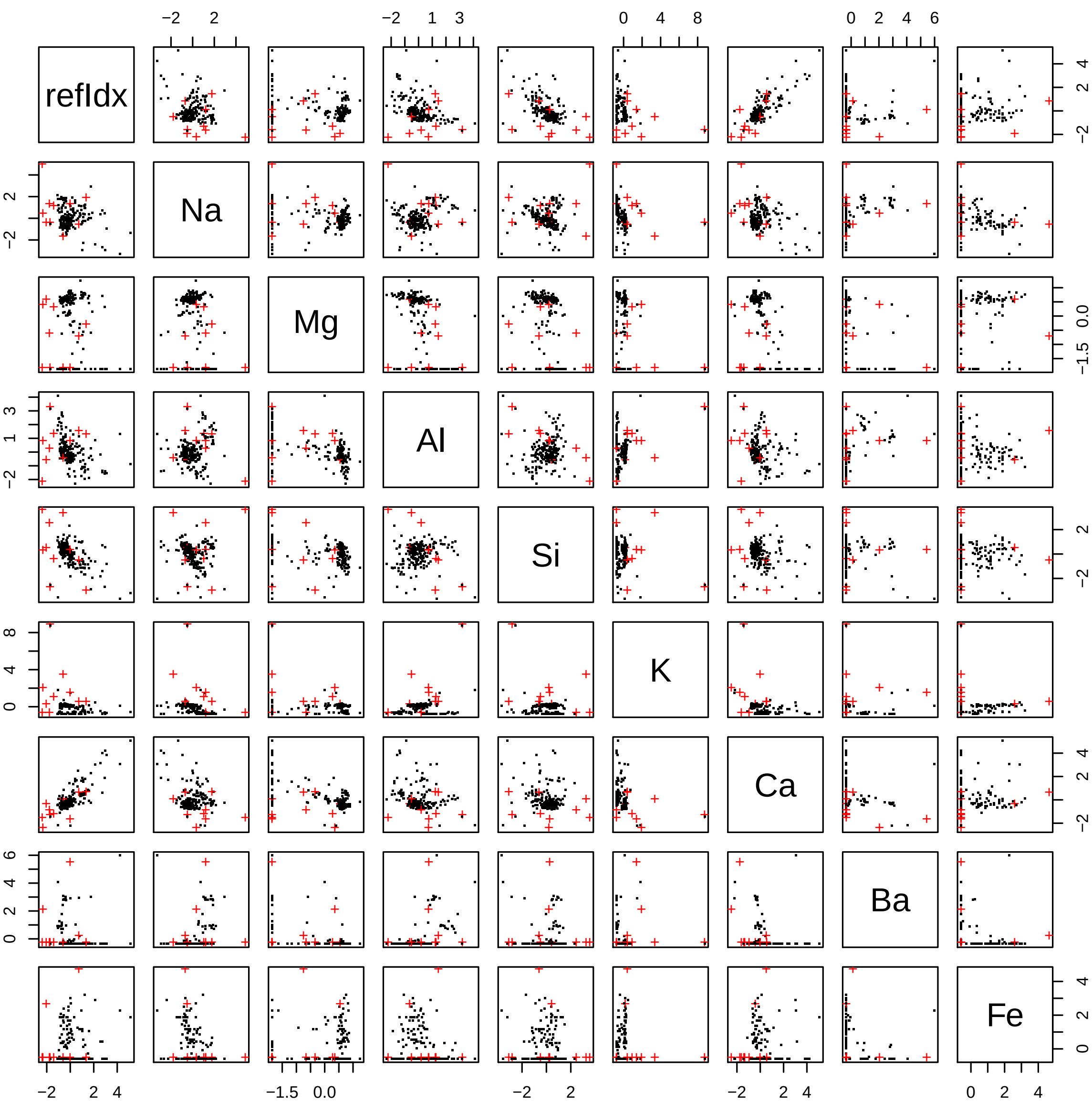


From the box plot, there is still a lot of outliers in each attribute, if we remove them all, it will still be a big portion of the data. The scatter plot doesn't provide a lot of information. I don't think it will be good to remove ids just based on one attribute.

Here I find the outliers using local outlier factor as it detects if a data point has a substantial lower density than their neighbors.

```
# calculate the local outlier factor of the matrix
outlier.scores <- lofactor(glassDataNorm,
                           k = 4)
# find out the top 10 outlier.scores and define them as outliers in the data set
outliersLOF <- order(outlier.scores,
                     decreasing = TRUE)[1:10]
# plot the outliers with a pairs plot
n <- nrow(glassDataNorm)
```

```
labels <- 1:n
labels[-outliersLOF] <- "."
pch <- rep(".", n)
pch[outliersLOF] <- "+"
col <- rep("black", n)
col[outliersLOF] <- "red"
pairs(glassDataNorm,
      pch = pch,
      col = col)
```



Question 6

After removing the ID column (column 1), standardize the scales of the numeric columns, except the last one (the glass type), using z-score.

```r
# the dataset has already been standardized above (in question 5),
# add type column back and make sure it's normalized
glassDataNorm$type <- glassData$type
glassDataNorm$type <- as.factor(glassDataNorm$type)
summary(glassDataNorm)
```

```
##      refIdx              Na                Mg                Al
##  Min.   :-2.3759   Min.   :-3.2793   Min.   :-1.8611   Min.   :-2.3132
##  1st Qu.:-0.6069   1st Qu.:-0.6127   1st Qu.:-0.3948   1st Qu.:-0.5106
##  Median :-0.2257   Median :-0.1321   Median : 0.5515   Median :-0.1701
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2608   3rd Qu.: 0.5108   3rd Qu.: 0.6347   3rd Qu.: 0.3707
##  Max.   : 5.1252   Max.   : 4.8642   Max.   : 1.2517   Max.   : 4.1162
##        Si                K                  Ca                Ba
##  Min.   :-3.6679   Min.   :-0.76213   Min.   :-2.4783   Min.   :-0.3521
##  1st Qu.:-0.4789   1st Qu.:-0.57430   1st Qu.:-0.5038   1st Qu.:-0.3521
##  Median : 0.1795   Median : 0.08884   Median :-0.2508   Median :-0.3521
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5636   3rd Qu.: 0.17318   3rd Qu.: 0.1515   3rd Qu.:-0.3521
##  Max.   : 3.5622   Max.   : 8.75961   Max.   : 5.0824   Max.   : 5.9832
##        Fe          type
##  Min.   :-0.5851   1:70
##  1st Qu.:-0.5851   2:76
##  Median :-0.5851   3:17
##  Mean   : 0.0000   5:13
##  3rd Qu.: 0.4412   6: 9
##  Max.   : 4.6490   7:29
```

Question 7

The data set is sorted, so creating a validation data set requires random selection of elements. Create a stratified sample where you randomly select 15% of each of the cases for each glass type to be part of the validation data set. The remaining cases will form the training data set.

```r
# randomly select 15% from each type
glassDataValid <- glassDataNorm %>%
  group_by(type) %>%
  sample_frac(., 0.15)
# get the remain data as training data
glassDataTraining <- glassDataNorm %>%
  anti_join(glassDataValid)
```

```
## Joining, by = c("refIdx", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "type")
```

Question 8

Implement the k-NN algorithm in R (do not use an implementation of k-NN from a package) and use your algorithm with a k = 4 to predict the glass type for the following two cases:

```r
# create a data frame for the cases need to be predicted
glassDataTest_1 <- data.frame("id" = c(215, 216),
                              "refIdx" = c(1.51621, 1.57930),
                              "Na" = c(12.52, 12.69),
                              "Mg" = c(3.48, 1.86),
                              "Al" = c(1.39, 1.82),
                              "Si" = c(73.39, 72.62),
                              "K" = c(0.60, 0.52),
                              "Ca" = c(8.55, 10.52),
                              "Ba" = c(0.00, 0.00),
                              "Fe" = c(0.07, 0.05),
                              "type" = c(0, 0))
# normalize the dataset
glassDataTest_1 <- rbind(glassData,
                         glassDataTest_1)
glassDataTest_1[, 2:10] <- as.data.frame(scale(glassDataTest_1[,2:10]))
glassDataTest_1 <- glassDataTest_1 %>%
  filter(id == 215 | id == 216) %>%
  select(-id)
# duplicate one for comparing with the class package
glassDataTest_2 <- glassDataTest_1
glassDataTest_2
```

```
##       refIdx         Na         Mg         Al          Si          K         Ca
## 1 -0.4728742 -1.0783026  0.5533436 -0.1133190  0.95238059 0.15767490 -0.2902021
## 2 11.8167501 -0.8700927 -0.5733715  0.7508082 -0.04428423 0.03444459  1.0963553
##           Ba          Fe type
## 1 -0.3502235  0.13365304    0
## 2 -0.3502235 -0.07255451    0
```

Create a knn function named “knn_1” to predict these two cases glass type.

```r
knn_1 <- function(trainData, testData, k){

  for(i in 1:nrow(testData)){
    for(j in 1:(ncol(trainData)-1)){
      # calculate distance between each value in test data and each value in train data
      d <- (trainData[, j] - testData[i, j]) ^ 2
      # get the index of top k nearest neighbors
      knnIdx <- head(sort(d, index.return = TRUE)$ix, k)
      # get the type of glass of these k nearest neighbors
      nn <- sort(table(trainData[knnIdx, 10]), TRUE)
      # add the predicted type to test data's type column
      testData[i, 10] <- as.numeric(names(nn)[1])
    }
  }
  testData
}
# use this function to predict these two cases type of glass
glassDataPred_1 <- knn_1(glassDataTraining,
                         glassDataTest_1,
                         4)
glassDataPred_1
```

```
##       refIdx         Na         Mg         Al          Si          K         Ca
## 1 -0.4728742 -1.0783026  0.5533436 -0.1133190  0.95238059 0.15767490 -0.2902021
## 2 11.8167501 -0.8700927 -0.5733715  0.7508082 -0.04428423 0.03444459  1.0963553
##           Ba          Fe type
## 1 -0.3502235  0.13365304    1
## 2 -0.3502235 -0.07255451    1
```

Question 9

Apply the knn function from the class package with k = 4 and redo the cases from Question(8). Compare your answers.

```r
set.seed(123)
# run knn function from class package
glassDataPred_2 <- knn(train = glassDataTraining,
                       test = glassDataTest_2,
                       cl = glassDataTraining$type,
                       k = 4)
glassDataPred_2
```

```
## [1] 1 2
## Levels: 1 2 3 5 6 7
```

```r
# add the predicted results to the data frame
glassDataTest_2$type <- glassDataPred_2
glassDataTest_2
```

```
##       refIdx         Na         Mg         Al          Si          K         Ca
## 1 -0.4728742 -1.0783026  0.5533436 -0.1133190  0.95238059 0.15767490 -0.2902021
## 2 11.8167501 -0.8700927 -0.5733715  0.7508082 -0.04428423 0.03444459  1.0963553
##           Ba          Fe type
## 1 -0.3502235  0.13365304    1
## 2 -0.3502235 -0.07255451    2
```

My function predicted both of the cases are glass type 1, the knn function from class package predicted them as 1 and 2.

Question 10

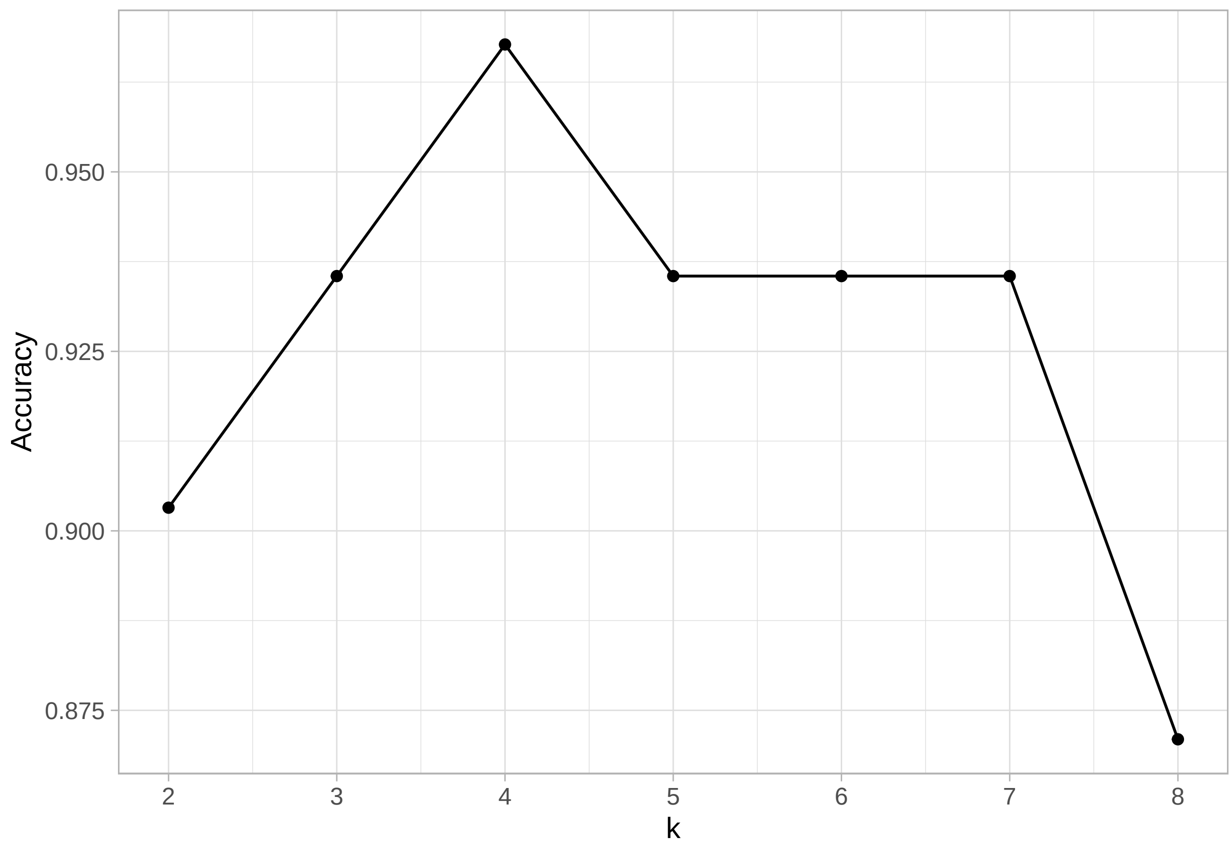
Using k-NN from the class package, create a plot of k (x-axis) from 2 to 8 versus accuracy (percentage of correct classifications) using ggplot.

```r
# create a data frame called accurach with one column for all k values,
# and each one column for each k's accuracy
accuracy <- data.frame("k" = c(2, 3, 4, 5, 6, 7, 8),
                       "accuracy" = c(0, 0, 0, 0, 0, 0, 0))
# create a loop to calculate accuracy for each k
for (k in 2:8){

  set.seed(123)
```

```r
  # get the predict type
  glassDataPred_3 <- knn(train = glassDataTraining,
                         test = glassDataValid,
                         cl = glassDataTraining$type,
                         k)
  # create a matrix with type in validation data and type predicted by knn
  mtrix <- as.matrix(table(glassDataValid$type,
                           glassDataPred_3))
  # calculate accuracy and add it to the accuracy data frame
  accuracy[which(k == accuracy$k), 2] <- sum(diag(mtrix)) / nrow(glassDataValid)

}
# plot the k-accuracy table
ggplot(accuracy, aes(x = k, y = accuracy)) +
  geom_point() +
  geom_line()+
  scale_x_continuous(breaks = seq(2, 8, by = 1)) +
  labs(x = "k", y = "Accuracy") +
  theme_light()
```



Question 11

Download this (modified) version of the glass dataset containing missing values in column 4. Identify the missing values. Impute the missing values of this continuous numeric column using your regression version of kNN from Problem 2 below using the other columns are predictor features.

```r
# read in the data file
glassDataModified <- read.csv("da5030.glass.data_with_missing_values.csv",
                              header = FALSE)
# give columns names
colnames(glassDataModified) <- c("id", "refIdx", "Na", "Mg", "Al",
                                 "Si", "K", "Ca", "Ba", "Fe", "type")
# normalize the dataset
glassDataModifiedNorm <- as.data.frame(scale(glassDataModified[,2:11]))
```

```r
# filter out cases that contain missing Mg values
glassDataModifiedImpute <- glassDataModifiedNorm %>%
  filter(is.na(Mg)) %>%
  select(-Mg)
# identify the target data
glassDataModifiedTarget <- glassDataModifiedNorm %>%
  filter(!is.na(Mg)) %>%
  select(Mg)
# identify the training data
glassDataModifiedTrain <- glassDataModifiedNorm %>%
  filter(!is.na(Mg)) %>%
  select(-Mg)
```

```r
# copy the knn.reg function from below
knn.reg <- function(new_data, target_data, train_data, k){
  # create a predict table for results
  pred <- as.data.frame(rep(0, nrow(new_data)))
  colnames(pred) <- "pred"

  for(i in 1:nrow(new_data)){
    for(j in 1:ncol(new_data)){
      # calculate distance between each value in test data and each value in train data
      d <- (train_data[, j] - as.numeric(new_data[i, j])) ^ 2
      # get the index of top k nearest neighbors
      knnIdx <- head(sort(d, index.return = TRUE)$ix, k)
      # get the type of glass of these k nearest neighbors
      nn <- sort(table(target_data[knnIdx, 1]), TRUE)
      # calculate the weighted products
      weighted <- 4 * as.numeric(names(nn)[1]) +
        2 * as.numeric(names(nn)[2]) +
        sum(as.numeric(names(nn)[3:k]))
      # calculate the weighted average
      weightedAve <- weighted / (4 + 2 + (k - 2))
      # assign the weighted average to predict table
      pred[i, 1] <- weightedAve
    }
  }
  pred
}

# run the knn.reg function
set.seed(123)
knn.reg(glassDataModifiedImpute,
        glassDataModifiedTarget,
```

```
        glassDataModifiedTrain,
        k = 4)
```

```
##           pred
## 1  0.685387979
## 2  0.685387979
## 3  0.633846248
## 4  0.577861264
## 5 -0.601377992
## 6 -0.340114735
## 7 -0.005982135
## 8 -0.005982135
## 9 -0.005982135
```

Above is the predicted Mg values for the missing data.

Problem 2

Question 1

Investigate this data set of home prices in King County (USA)

```
houseData <- read.csv("kc_house_data.csv")
# check missing values
any(is.na(houseData))
```

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

```
# check structure
str(houseData)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
```

```
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

Question 2

Save the price column in a separate vactor/dataframe called target_data. Move all of the columns except the ID, data, price, yr_renovated, zipcode, lat, long, sqft_living12, and sqft_lot 15 columns into a new data frame called train_data.

```r
# save the price column to target_data
target_data <- houseData %>% select(price)
# move required columns to train_data
train_data <- houseData %>% select(-c(id, date, price, yr_renovated,
                                      zipcode, lat, long, sqft_living15,
                                      sqft_lot15))
# check column info of train_data
sapply(train_data, class)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##     "integer"     "numeric"     "integer"     "integer"     "numeric"
##    waterfront          view     condition         grade    sqft_above
##     "integer"     "integer"     "integer"     "integer"     "integer"
## sqft_basement      yr_built
##     "integer"     "integer"
```

Question 3

Normalize all of the columns (except the boolean columns waterfront and view) using min-max normalization

```r
# create the min-max function
minMax <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}
# move waterfront and view column to the end of the data set
train_data_norm <- train_data %>%
  relocate(c(waterfront, view),
           .after = last_col())
# normalize the data set using min-max function
train_data_norm[, 1:10] <- as.data.frame(lapply(train_data_norm[, 1:10],
                                                 minMax))
```

Question 4

Build a function called knn.reg that implements a regression version of k-NN that averages the prices of the k nearest neighbors using a weighted average where the weight is 4 for the closest neightbor, 2 for the second closest and 1 for the remaining neighbors (recall that a weighted average requires that you divide the sum product of the weight and values by the sum of the weights).

```r
# create the knn.reg function
knn.reg <- function(new_data, target_data, train_data, k){
  # create a predict table for results
  pred <- as.data.frame(rep(0, nrow(new_data)))
  colnames(pred) <- "pred"

  for(i in 1:nrow(new_data)){
    for(j in 1:ncol(new_data)){
      # calculate distance between each value in test data and each value in train data
      d <- (train_data[, j] - as.numeric(new_data[i, j])) ^ 2
      # get the index of top k nearest neighbors
      knnIdx <- head(sort(d, index.return = TRUE)$ix, k)
      # get the type of glass of these k nearest neighbors
      nn <- sort(table(target_data[knnIdx, 1]), TRUE)
      # calculate the weighted products
      weighted <- 4 * as.numeric(names(nn)[1]) +
        2 * as.numeric(names(nn)[2]) +
        sum(as.numeric(names(nn)[3:k]))
      # calculate the weighted average
      weightedAve <- weighted / (4 + 2 + (k - 2))
      # assign the weighted average to predict table
      pred[i, 1] <- weightedAve
    }
  }
  pred
}
```

Question 5

Forecast the price of this new home using your regression k-NN using k = 4.

```r
# type in the data
new_data <- data_frame("bedrooms" = 4,
                       "bathrooms" = 3,
                       "sqft_living" = 4852,
                       "sqft_lot" = 11245,
                       "floors" = 3,
                       "condition" = 3,
                       "grade" = 11,
                       "sqft_above" = 2270,
                       "sqft_basement" = 820,
                       "yr_built" = 1986,
                       "waterfront" = 1,
                       "view" = 1)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
# normalize the new data
new_data <- rbind(train_data, new_data)
```

```r
new_data <- new_data %>%
  relocate(c(waterfront, view),
           .after = last_col())
new_data[, 1:10] <- as.data.frame(lapply(new_data[, 1:10],
                                         minMax))
new_data <- new_data %>% slice(n())
# run the knn.reg function
set.seed(123)
knn.reg(new_data,
        target_data,
        train_data,
        4)
```

```
##       pred
## 1 584437.5
```

As the result from knn.reg, the predicted price of this new home is 584437.5