Code Sample

Wayne Monical

3/3/2021

Part 1: Data Cleaning and Correcting

```
library(dplyr)
```

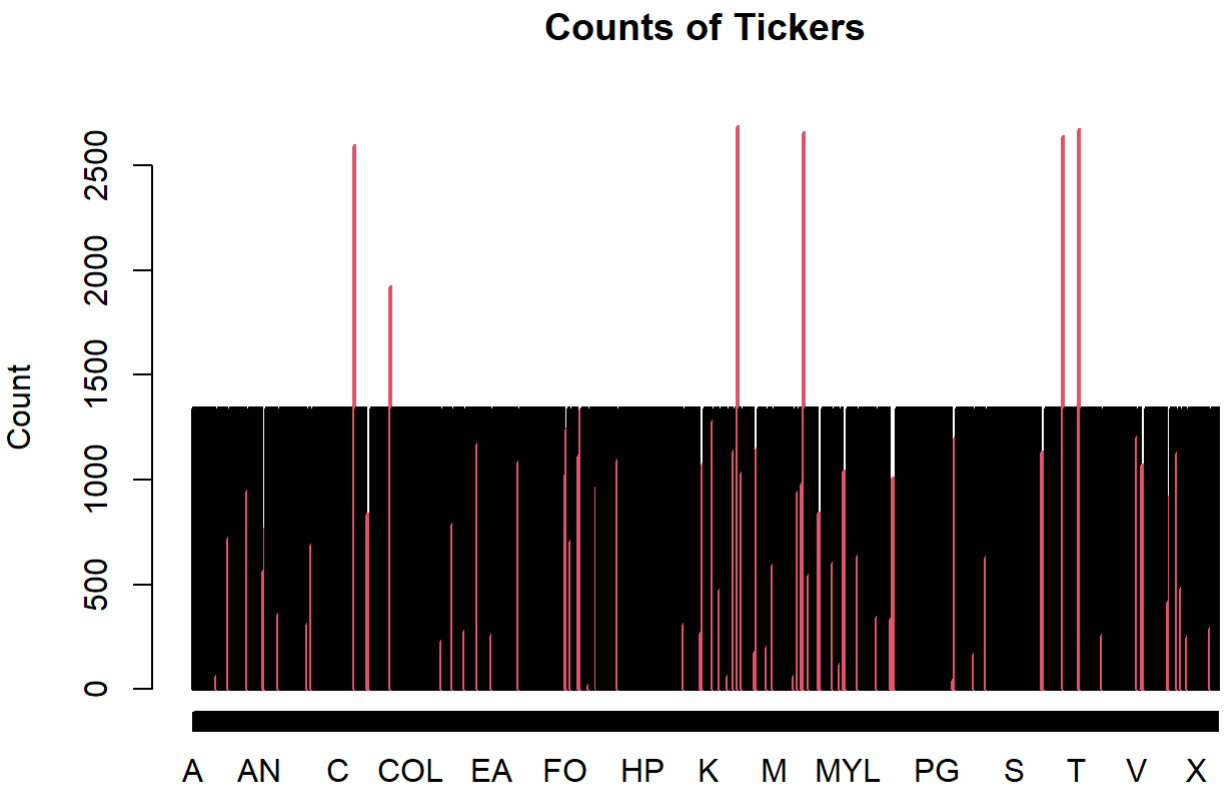
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# data load
stock_rawdata <- read.csv("stock_data.csv", header = TRUE)

# removing unnecessary columns, with pacakage dplyr
stock <- select(stock_rawdata, date, TICKER, PRC)

stock <- na.omit(stock)

# Removing duplicates.
stock <- stock[!duplicated(stock), ]
```

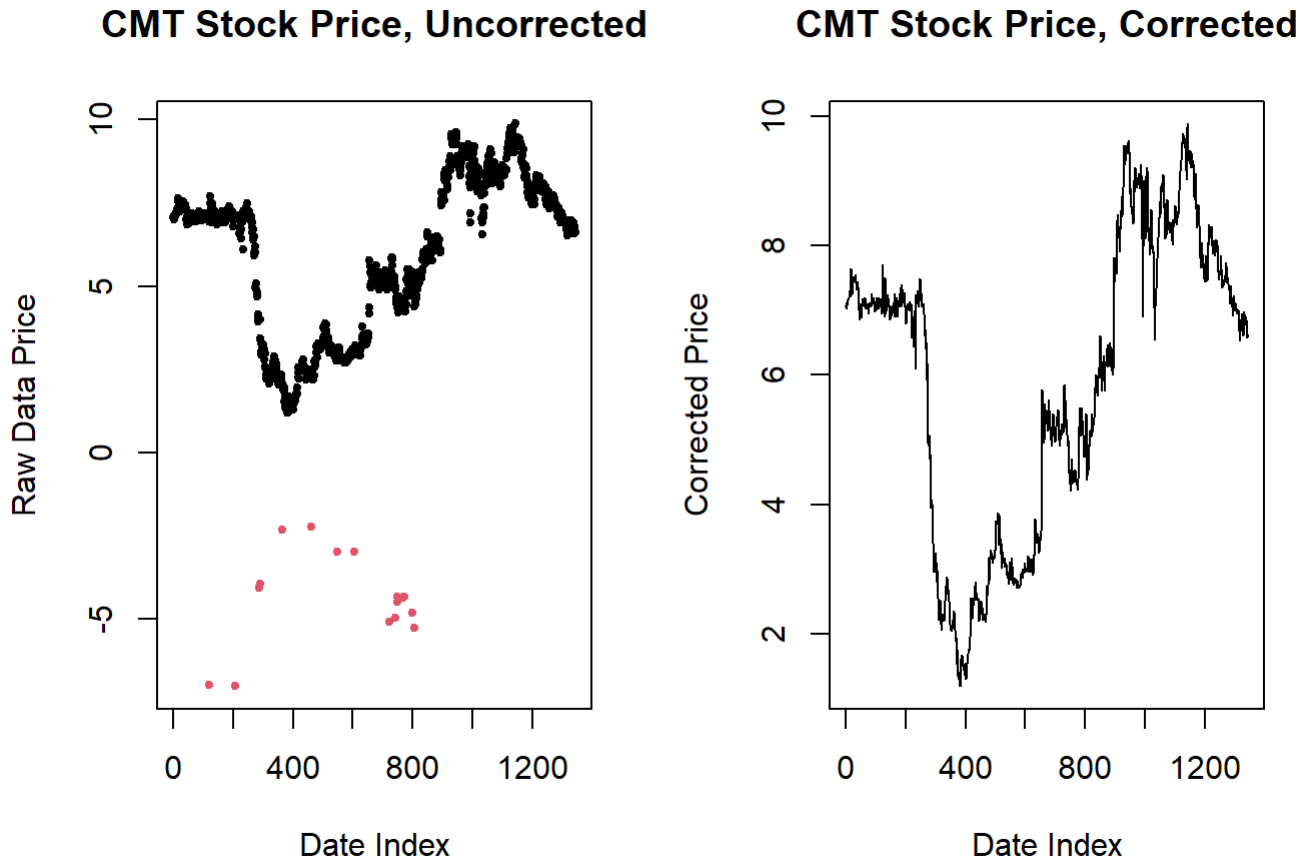
```
tickertable <- table(stock$TICKER)

# boolean of the date stocks that I want to include
tickertable <- table(stock$TICKER)
stock_boolean <- tickertable == 1342

# boolean mask to include only these stocks
ticker_include <- tickertable[stock_boolean]

# peeling the names of the stocks with 1342 entries off the table
# stock_included has the tickers of all these stocks
stock_included <- names(ticker_include)

## Plotting the occurences of each ticker
# base R

plot(table(stock$TICKER),
     main = "Counts of Tickers",
     ylab = "Count",
     col = factor(!stock_boolean))
```



```
## Comparing the dates of the included stocks with exactly 1342 entries in the dataframe.

# simple filter for tickers that occur 1342 times
stock_new <- filter(stock, TICKER %in% stock_included)



# checking that each of the 1342 corresponds to exactly 1 date.
length(unique(stock_new$date))
```

```
## [1] 1342
```

```
# turning the numerical date vector to a more friendly format.
# note: we have to specify the format that the date vector is in, i.e. '%Y%m%d'
dateform <- as.Date(as.character(stock$date), format = '%Y%m%d')

# adding the corrected dates to the data frame
stock_new <- mutate(stock, dateform =  dateform )

# specify first day
start <- min(stock_new$dateform)

# to make stock data work in days, add a day so that no entry is zero
date_num <- as.numeric((as.POSIXct(stock_new$dateform) + 24*60^2) - as.POSIXct(start))

# adding date_num to stock_new
stock_new <- mutate(stock_new, date_num = date_num)

# filtering for CMT stock
cmt <- filter(stock_new, TICKER == "CMT")

# plotting corrected and uncorrected side by side
par(mfrow = c(1, 2))
plot(cmt$PRC, pch = 20, main = "CMT Stock Price, Uncorrected",
     xlab = "Date Index", ylab = "Raw Data Price",
     col = factor(cmt$PRC < 0), cex = 0.8)
plot(abs(cmt$PRC), type = 'l', main = "CMT Stock Price, Corrected",
     xlab = "Date Index" , ylab = "Corrected Price")
```



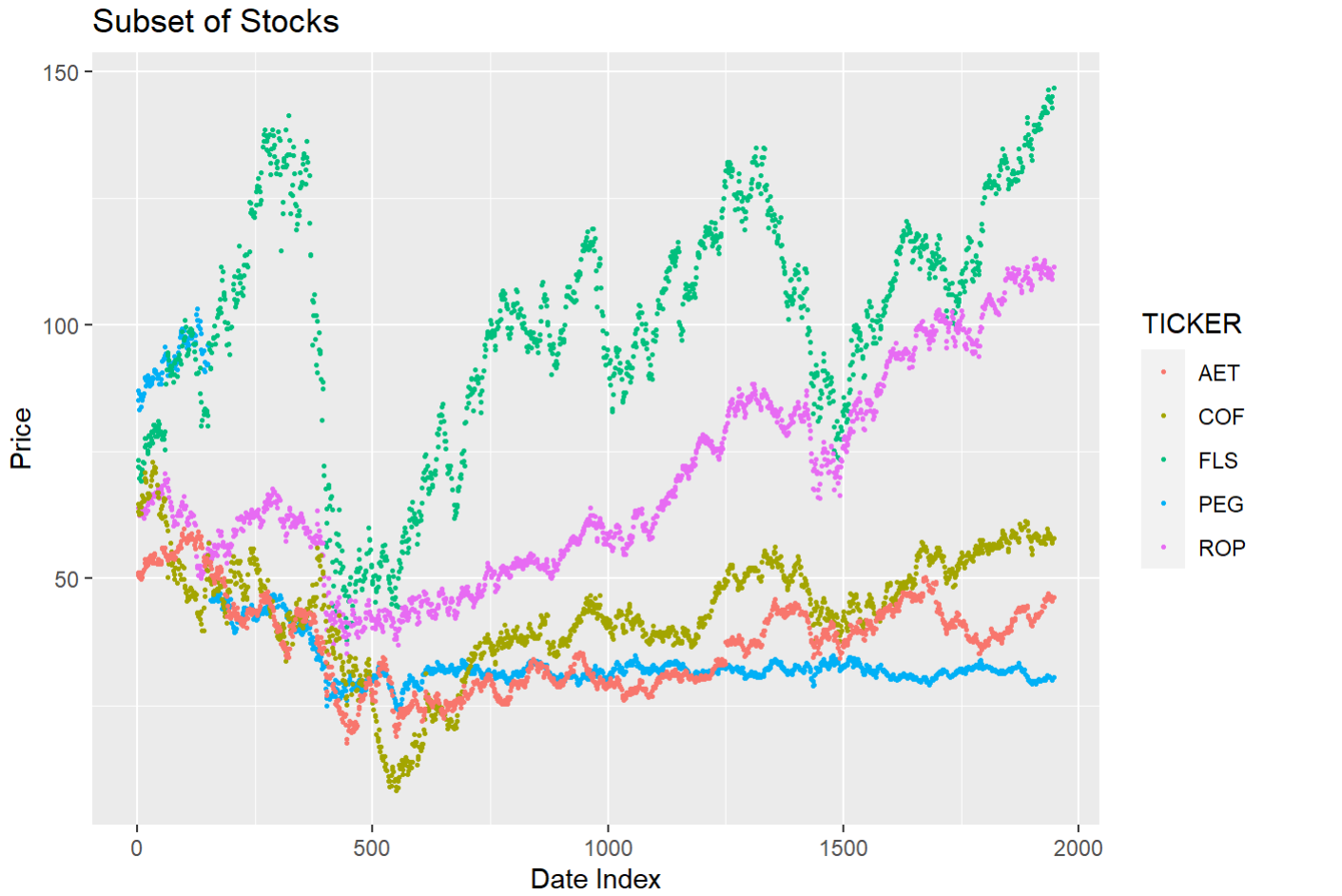

```
# Fixing the data
stock_new <- mutate(stock_new, PRC = abs(stock_new$PRC))

# Dropping other date formats
stock_new <- select(stock_new, -date, -dateform)
```

Part 2: kNN Prediction Function

```
# internal function
# returns matrix descrbing closet k neighbors
closest.k <- function(value, vec, k = 1){

  # convience value
  temp_n <- length(vec)

  # matrix whose columns are an index of vector place, the vector,
  # and how close each entry is to the target
  temp_mat <- matrix(c(1:temp_n, vec, abs(vec - value)), nrow = temp_n)


  temp_mat <- temp_mat[order(temp_mat[,3]),]


  effective_k <- max(k, 3)

  answer <- temp_mat[1:effective_k, ]

  if(is.vector(answer)){
    stop("closest.k is trying to return a vector")
  }

  if(! is.matrix(answer)){
    stop("closest.k is trying to return not a matrix")
  }

  return(answer)

  }


# predicts the response at value, from data
knn.predict <- function(value, response, data, k = 1){
  mat_closest <- closest.k(k = k, vec = data, value = value)

  temp_index <- mat_closest[1:k,1]


  closest_response <- response[temp_index]

  return(sum(closest_response) / k)
}


# apply wrapper
knn.apply <- function(sequence, response, data, k){
  answer <- sapply(X = sequence, FUN = function(x){knn.predict(value = x, response = response, d
ata = data , k = k)})
  return(answer)
}
```

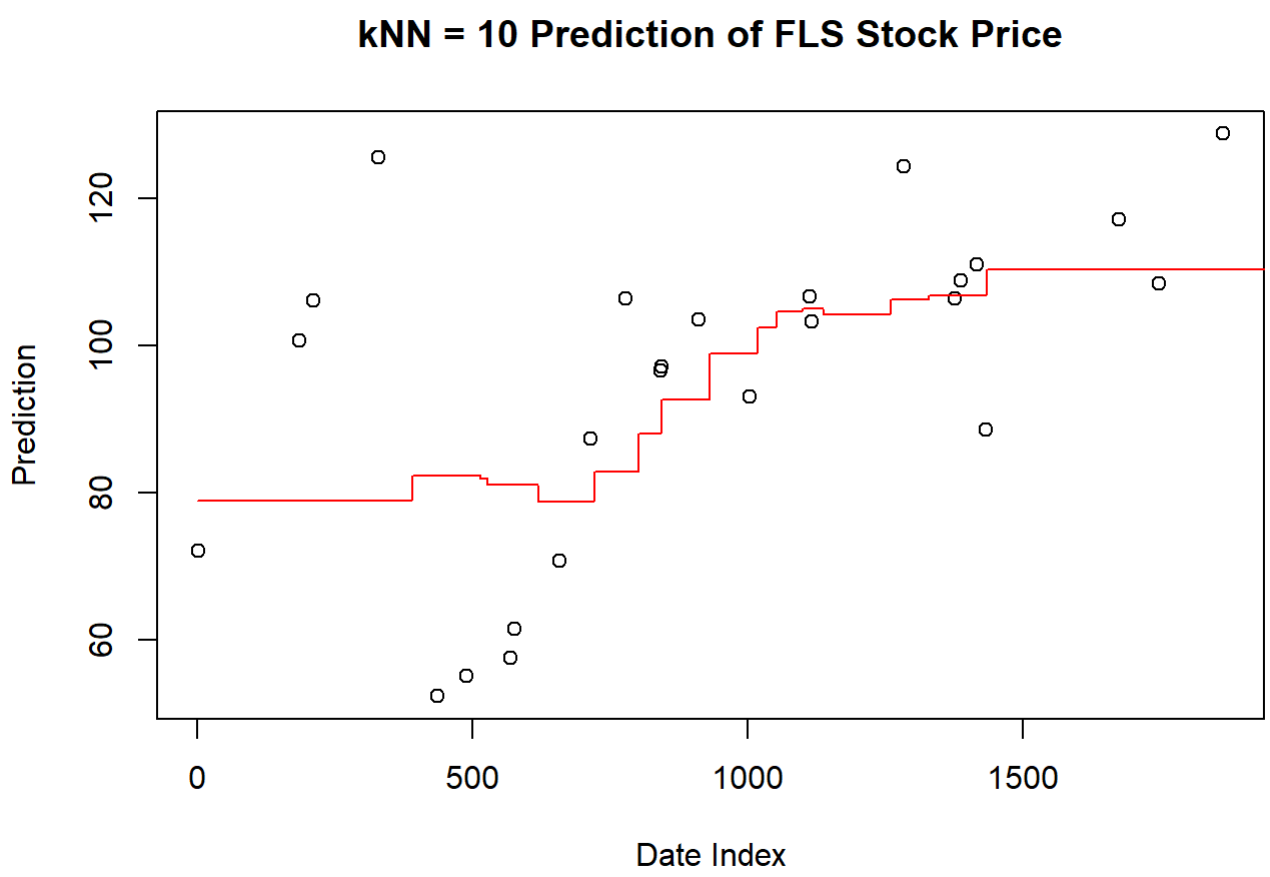
Part 3: Graphics and Analysis

```
library(ggplot2)

# interesting stocks
ticker_subset <- c("COF", "FLS",  "AET", "ROP", "PEG")
stock_subset <- filter(stock_new, TICKER %in% ticker_subset)
stock_subset <- mutate(stock_subset, data_type = "Original Data")

# initial plot
ggplot(data = stock_subset, aes(x = date_num,  y = PRC)) + geom_point(aes(col = TICKER), size =
0.5) +
  ggtitle("Subset of Stocks") +
  labs(x = "Date Index", y = "Price")
```



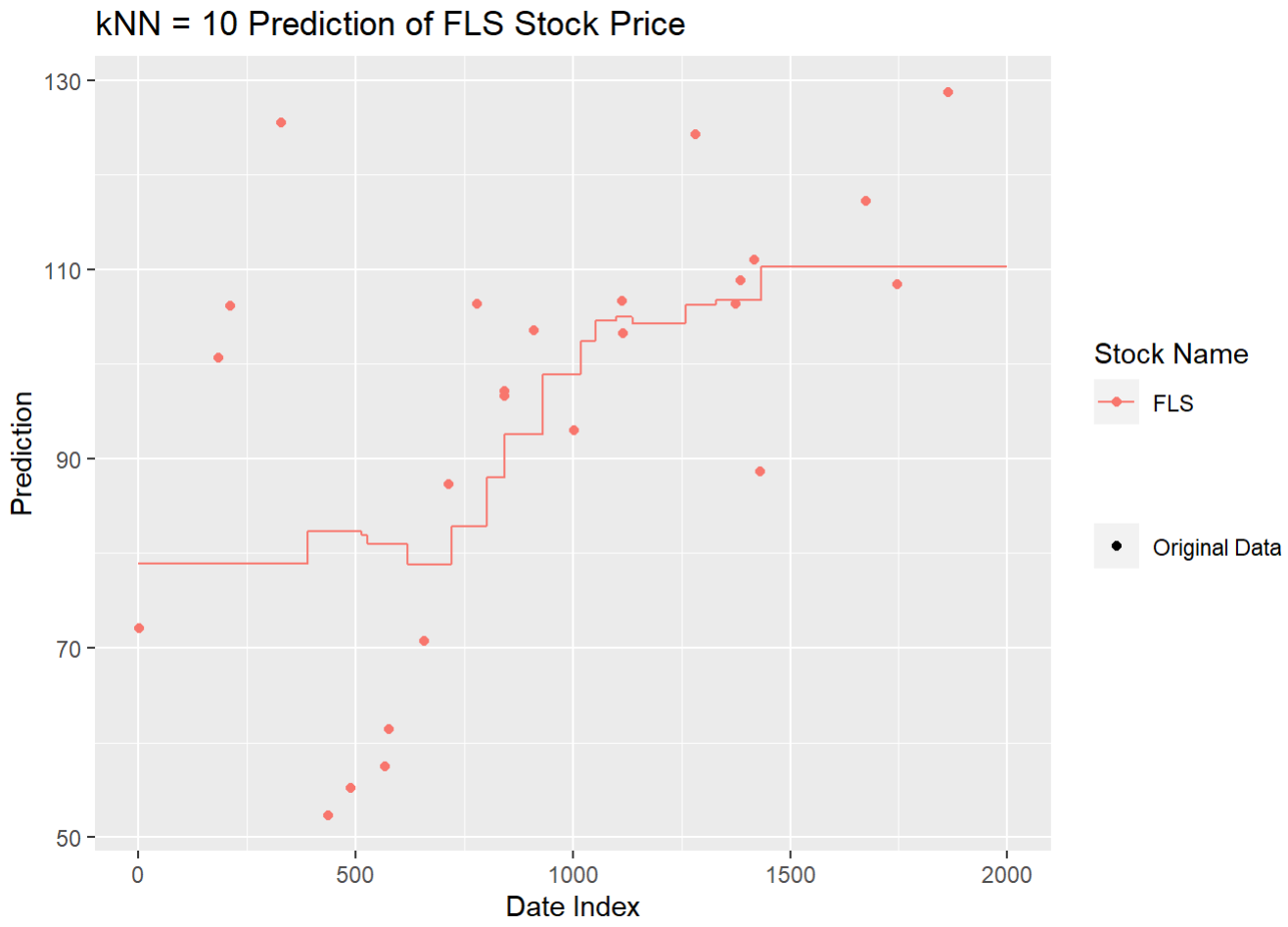

```
# splitting into training and test
sample_index <- sample(1:max(stock_subset$date_num), size = 35, replace = FALSE)
stock_train <- filter(stock_subset, date_num %in% sample_index)
stock_test <- filter(stock_subset, !date_num %in% sample_index)
```

```
# useful constants
date_sequence <- seq(0, 2000, 1)
k_star <- 10

# work on 1 stock first
train_FLS <- filter(stock_train, TICKER == "FLS")

# applying kNN to FLS on date index
predict_FLS <- knn.apply(sequence = date_sequence, response = train_FLS$PRC, data = train_FLS$da
te_num, k = k_star)
df_predict_FLS <- data.frame(TICKER = "FLS", PRC = predict_FLS, date_num = date_sequence, data_t
ype = "Prediction")

# in order to use ggplot, these need to fit together
train_FLS <- rbind(train_FLS, df_predict_FLS)

# plotting the original data and the kNN prediction
# base R
plot(train_FLS$date_num[train_FLS$data_type == "Original Data"], train_FLS$PRC[train_FLS$data_ty
pe == "Original Data"],
     main = "kNN = 10 Prediction of FLS Stock Price", xlab = "Date Index",
     ylab = "Prediction") +
  points(date_sequence, predict_FLS, type = "l", col = "red")
```

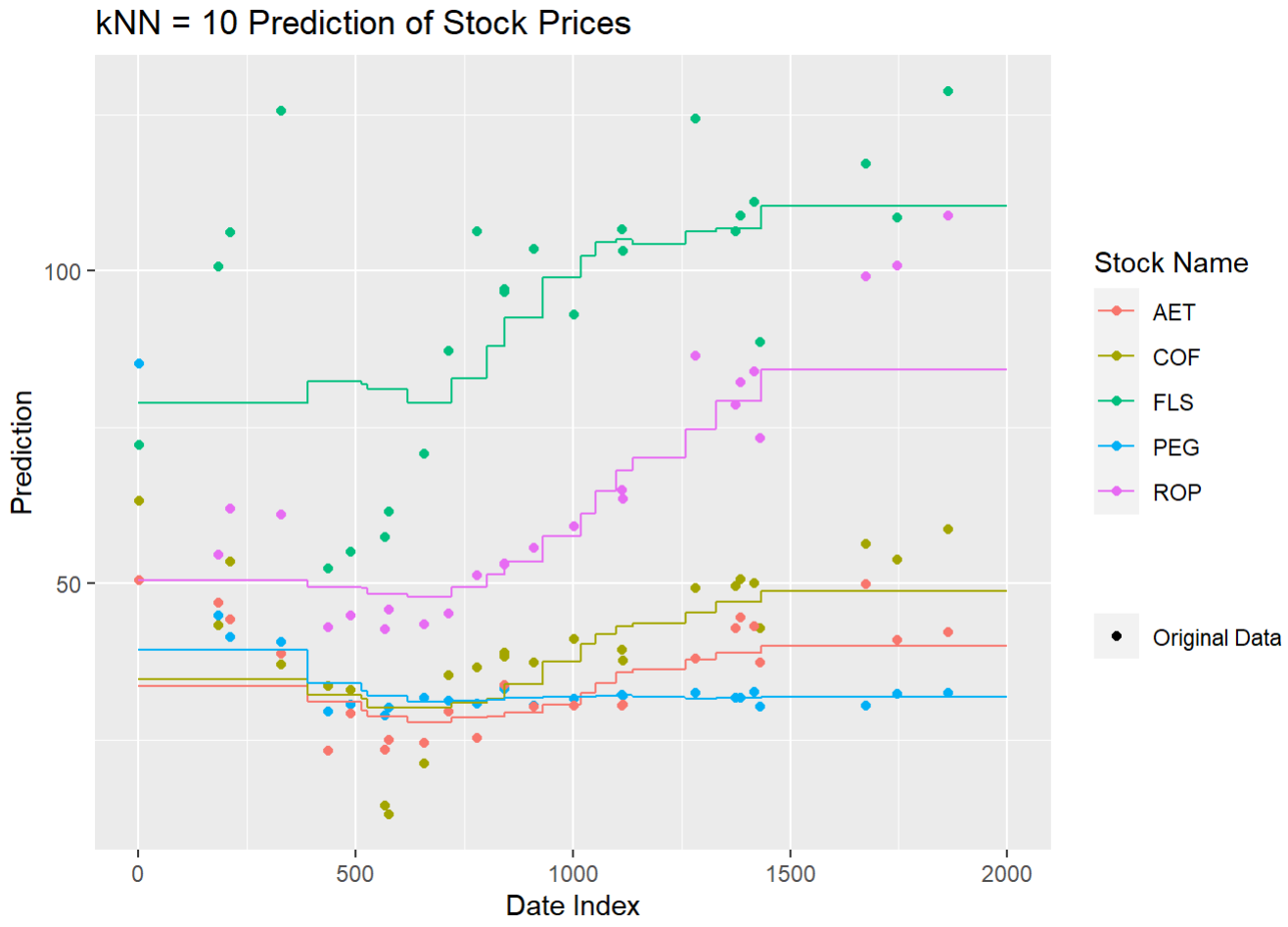



```
## integer(0)
```

```
# the same exact plot in ggplot
ggplot(data = train_FLS, ) +
        geom_point(data = subset(train_FLS, data_type == "Original Data"),
                   aes(x = date_num, y = PRC, color = TICKER, shape = data_type)) +
        geom_line(data = subset(train_FLS, data_type == "Prediction"),
                  aes(x = date_num, y = PRC, color = TICKER)) +
        ggtitle(label = "kNN = 10 Prediction of FLS Stock Price") +
        labs(x = "Date Index", y = "Prediction", col = "Stock Name", shape = "")
```



```
# predicing the other stock prices from kNN
for(g in ticker_subset){

  df_g <- filter(stock_train, TICKER == g)
  predict_g <- knn.apply(sequence = date_sequence, response = df_g$PRC,
                         data = df_g$date_num, k = k_star)

  df_predict_g <- data.frame(TICKER = g, PRC = predict_g,
                             date_num = date_sequence, data_type = "Prediction")


  stock_train <- rbind(stock_train, df_predict_g)


}

# in gg plot, we can add the other data too
ggplot(data = stock_train) +
         geom_point(data = subset(stock_train, data_type == "Original Data"),
                    aes(x = date_num, y = PRC, color = TICKER, shape = data_type)) +
         geom_line(data = subset(stock_train, data_type == "Prediction"),
                   aes(x = date_num, y = PRC, color = TICKER)) +
         ggtitle(label = paste0("kNN = ", k_star," Prediction of Stock Prices")) +
         labs(x = "Date Index", y = "Prediction", col = "Stock Name", shape = "")
```



```
## Computing expected error
empirical_error <- matrix(rep(0, 10), nrow = 2, dimnames = list(c("test", "training"), ticker_su
bset))


# the first step in joining the predictions to the original matrix
stock_train_predict <- filter(stock_train, data_type == "Prediction")
stock_train_predict <- mutate(stock_train_predict,
                              Prediction = stock_train_predict$PRC) %>% select(-PRC, -data_type)


# inner join, since the matrix should be full
stock_merge <- inner_join(stock_subset, stock_train_predict, by = c("TICKER", "date_num"))
stock_merge <- mutate(stock_merge, Error = abs(stock_merge$Prediction - stock_merge$PRC))

stock_merge_train <- filter(stock_merge, date_num %in% sample_index)
stock_merge_test <- filter(stock_merge, !date_num %in% sample_index)

empirical_error[1,] <- aggregate(stock_merge_train[,"Error"], list(stock_merge_train$TICKER), me
an)$x
empirical_error[2,] <- aggregate(stock_merge_test[,"Error"], list(stock_merge_test$TICKER), mea
n)$x


empirical_error
```

```
##               COF      FLS      AET      ROP      PEG
## test     4.915960 7.226360 13.84936 3.203040 6.481320
## training 5.502315 7.897964 16.88168 5.858787 8.248177
```