Data Analysis with R

TRSM R Bootcamp

Introduction

In this segement of the bootcamps, we will be focusing on practical concepts and technqiues that can be used via R.

```
# Importing the needed libraries
library(ggplot2)
library(tidyverse)
library(lubridate)

# Importing the Tesla stock price data from 2008 to 2018
# https://www.kaggle.com/timoboz/tesla-stock-data-from-2010-to-2020
tsla <- read.csv('./TSLA.csv')
tsla$Date <- ymd(tsla$Date) # Changing type from character to Date

# Housing dataframe
housing_df <- read_csv("https://www.dropbox.com/s/tvvtf9dwjufo7os/housing_train.csv?dl=1")
```

Exploratory Data Analysis

- Going over the features of the data
- Examining missing values
- Trying to see if there are any correlations between different columns within a dataframe

Exercise: Answers the questions for tsla dataset.

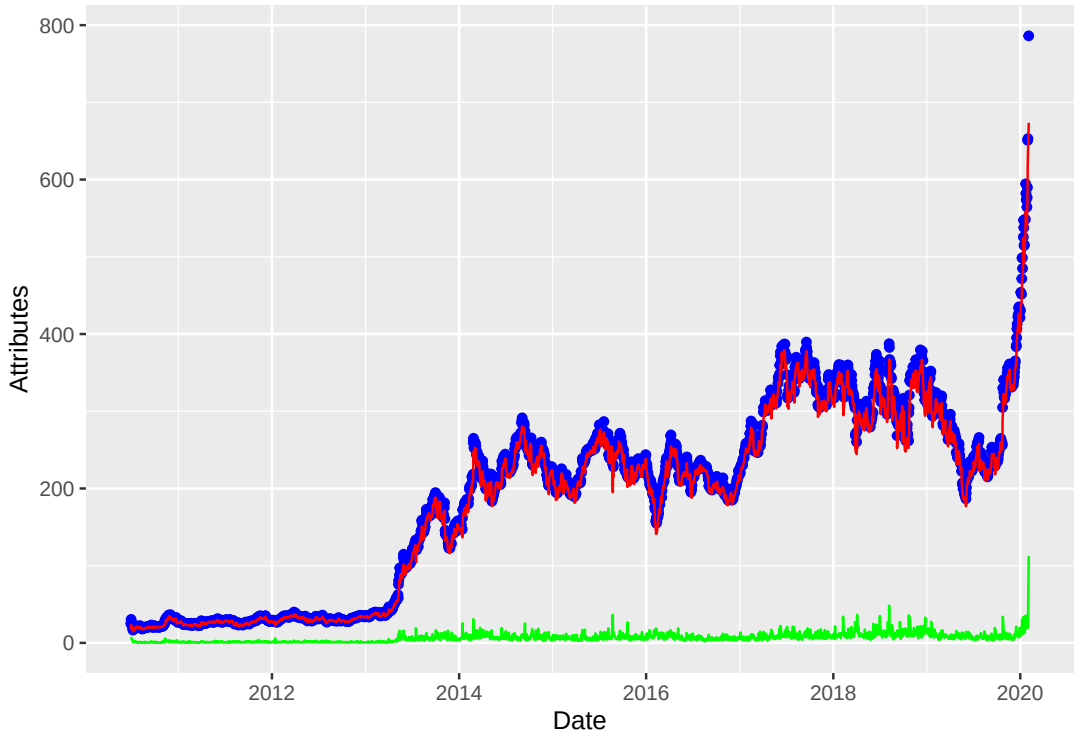
```
1. What are the column names in the tsla?
2. How many columns are there?
3. Are there any missing values within the tsla dataframe?
4. Name 2 columns that are numerical.
5. Name 2 columns that are categorical.
6. Maximum value in Volume column
7. Minimum value in Low column
8. Median of High column
```

Multiple line Graph

- Given that there might be multiple attributes that we are interested in, being able to graph multiple y for the same x is important.
- In time series analysis, Time is the x value and other features will be the y.

```
tsla %>%
  mutate(diff=abs(High - Low)) %>% # Making the diff column
  ggplot(aes(x=Date)) +
    geom_point(aes(y=High), color='blue') +
    geom_line(aes(y=Low), color='red') +
```

```
    geom_line(aes(y=diff), color='green') +
    xlab("Date") + ylab("Attributes")
```



Exercises:

1. Graph the tsla dataset where the absolute difference between High and Low is bigger than 30.
2. Are there any missing values in the graph? What does this represent?

```
tsla %>%
  mutate(diff=High - Low) %>% # Make the diff column
  filter(abs(diff) > 30) %>% # Filter it
  ggplot(aes(x=Date)) + # Plot
    geom_line(aes(y=diff), color='green') +
    xlab("Date") + ylab("Attributes")
```

Normalizing

- Normalizing is a technique used to make data more understandable
- It is one of the most important preprocessing steps (assuming that is applicable)

```
tsla %>%
  mutate(open_norm=(Open-mean(Open)) / sd(Open)) %>% # Making norm_open column
  ggplot(aes(x=Date, y=open_norm)) + geom_line()
```

Exercise: Using the provided methods, normalize the data.

1. Normalize the High column with formula norm(col) = (col - mean(col)) / sd(col)
2. Normalize the Volume column with formula norm(col) = (col - mean(col)) / sd(col)
3. Make new columns norm_low, norm_high, and norm_vol using columns Low and High with the formula in (1) and Volume with formula in (2). (Hint: use mutate)
4. Graph the the normalized column in one graph using independent geom_lines.

Missing values

- Data is gathered in various ways which surveying is one of them. When the participants don't answer questions there will be blank entries within the dataframe.
- The balnk entries within the dataframe are filled with NA
- They are other reasons into why one entry is filled with NA.

NA (not available/applicable):

```
- We cannot perform any kind of arithmetic or logical operations:
```

```
NA + 1
```

```
## [1] NA
```

```
NA > 5
```

```
## [1] NA
```

```
NA == TRUE
```

```
## [1] NA
```

- From a Logical standpoint, given that a value for a variable is not known, there is no a way for us to decide for its addition.
- Since we dont know anything about the value that is not available, it is safe to assume that it can have any type and value.

Exercise: What is the result of following expressions annd why?

1. **NA | TRUE**: Results in TRUE, since we dont know what the other value is but we it does not matter since it is being ored with TRUE
2. **NA | FALSE**: Results in NA, since the NA is unkown but it is being ored with FALSE
3. **NA != TRUE**: Results in NA, since we dont know if it equal to TRUE or not.

Data Imputation:

```
- replacing NA with a value to be able to process the data
- Mean: Imputing the NA with mean of the column
- Mode: Imputing the NA with mode of the column
- Constant: Impute the NA with a constant value
- Using algorithms to predict the missing values (k-means, regression)
```

- Impute the LotFrontage column with the column's mean:

```
# 1. Get the mean
mean_lot_frontage <- mean(housing_df$LotFrontage, na.rm=TRUE)
# 2. Replace it using the replace_na() function of tidyverse
housing_df$LotFrontage <- replace_na(housing_df$LotFrontage, mean_lot_frontage)
```

Inquiry Question: Why do we have na.rm in the mean function in step1? Since they are NA values within the entry, if we do not remove them while getting the average, we are going to get NA since arithmetics are not defined for NA values

- Impute GarageYrBlt column with the column's mode (most reapeted) value

```
# 1. Get the mode/median
mode_garage_year_built <- median(housing_df$GarageYrBlt, na.rm=1)
# 2. Replace and rewrite the column
housing_df$GarageYrBlt <- replace_na(housing_df$GarageYrBlt, mode_garage_year_built)
```

- Impute the Alley column's NA values with sth more meanningful such as 'none'

```
housing_df$Alley <- replace_na(housing_df$Alley, 'none')
```

Exercises:

1. Use the mean of MasVnrArea column to encode its missing values.
2. Impute the missing vales in BsmtFinType1 with 'none'.