Understanding Stock Price Beahavior using an R Based Analytical Framework

Francis Kurian

10/7/2021

Introduction

Stock market behavior is a well researched area due to the free availability of historic information. Several studies link the stock price movement to the sentiments of the market participants. Sentiments are usually formed by relevant economic events specific to the companies. IBM Watson APIs enables tracking of company specific news and social media platforms through extensive text mining and help come out with a Sentiment Index (Negative and positive emotions expressed through text) for a business. Business Sentiment Index is one such index calculated by TRaiCE Fintech that quantifies market sentiment for a company . In addition, to understand sensitivity of stock prices to Global events, Foreign Exchanges Rate is also introduced as an additional measure. Idea here is to demonstrate how to extract various relevant data elements from diverse sources, transform and load them into an analytic framework to visualize the relationship.

Purpose

Primary objective is to develop an analysis system using various R libraries. Extracting and processing data from multiple sources, data cleaning, simplifying repetitive tasks using control structures and functions, use of data visualization techniques and application of statistical methods are the focus areas while building the framework in R. In other words, learn to write reproducible R code while analyzing stock price movement with respect to Business Sentiment Index(BSI) and Foreign Exchange Rate is the purpose of this project.

List of Libraries

For this project the following Libraries are used for data validation, graphics and statistical analysis

```
library(ggplot2)
library(tidyr)
library(plyr)
library(lubridate)
library(scales)
library(reshape2)
library(summarytools) #dataframe summaries
library(ggfortify) #autoplot
library(sjPlot) #tabmodel
```

Data sources, Extraction and Cleaning

Data files(.csv)used, Description and source

1.List of companies to Analyze: Internally created CSV

2.Foreign Exchange daily data for various currencies: https://www.federalreserve.gov

3.Daily Stock Prices: https://finance.yahoo.com

4.Business Sentiments Index(BSI,IBM Watson API based): https://www.traice.io

This project uses 6 different companies as test cases. However as long as the data files are available, companies can be added to the list and no code changes are necessary. Stock Price and BSI data will have 6 files each. Foreign Exchange is macro economic data so only one file with three different exchange rates are downloaded from the Federal Reserve. Daily data for two years(01-SEP-2019 to 31-AUG-2021) are used for the analysis.

Reading CSV files(sections 1,2) and formatting

```
CompanyNames <- './data/CompanyNames.csv'
df_companies <- read.csv(CompanyNames, header = T)
head(df_companies) #list records
```

```
##   ticker                            name
## 1     GE        General Electric Company
## 2     BA               The Boeing Company
## 3      F               Ford Motor Company
## 4     FE                First Energy Corp
## 5    KHC          The Kraft Heinz Company
## 6    OXY Occidental Petroleum Corporation
```

```
ForexFile <- './data/FederalReserve_CurrencyXchangeRate.csv'
headers <- read.csv(ForexFile, skip = 5 , header = F, nrows = 1)#header info at row #6
df_forex <- read.csv(ForexFile, skip = 6, header = F) #data begins at row #7
colnames(df_forex) <- headers #apply headers
df_forex$period <- as.Date(df_forex$`Time Period`) #format as date

df_forex$USDxINR <- as.numeric(gsub("ND","",df_forex$RXI_N.B.IN))#clean & format as number
df_forex$USDxEUR <- as.numeric(gsub("ND","",df_forex$`RXI$US_N.B.EU`))
df_forex$USDxYEN <- as.numeric(gsub("ND","",df_forex$RXI_N.B.JA))

df_forex <- subset(df_forex,select = c(period, USDxINR, USDxEUR, USDxYEN))
df_forex <- df_forex[!duplicated(df_forex$period),] # remove duplicates
df_forex2 <- melt(df_forex, id.vars="period") # for all in one graphs

sapply(df_forex, class) #check data structure
```

```
##    period   USDxINR   USDxEUR   USDxYEN
##    "Date" "numeric" "numeric" "numeric"
```

```
head(df_forex)
```

```
##       period USDxINR USDxEUR USDxYEN
## 1 2019-01-01      NA      NA      NA
## 2 2019-01-02   70.02  1.1357  109.22
## 3 2019-01-03   70.12  1.1399  108.07
## 4 2019-01-04   69.63  1.1410  108.29
## 5 2019-01-07   69.78  1.1468  108.62
```

```
## 6 2019-01-08   70.11  1.1444  108.57
```

```
summary(df_forex) #check distribution, missing values
```

```
##      period              USDxINR         USDxEUR         USDxYEN
##  Min.   :2019-01-01   Min.   :68.40   Min.   :1.068   Min.   :102.5
##  1st Qu.:2019-09-05   1st Qu.:71.04   1st Qu.:1.112   1st Qu.:106.5
##  Median :2020-05-11   Median :72.97   Median :1.134   Median :108.5
##  Mean   :2020-05-10   Mean   :72.62   Mean   :1.148   Mean   :108.1
##  3rd Qu.:2021-01-13   3rd Qu.:74.26   3rd Qu.:1.184   3rd Qu.:109.7
##  Max.   :2021-09-17   Max.   :76.95   Max.   :1.230   Max.   :112.0
##                       NA's   :31      NA's   :31      NA's   :31
```

Reading CSV files(sections 3,4) in a loop and formatting

Checks the csv files are existing for the specified list. Breaks the DO LOOP otherwise

```
for (i in df_companies$ticker) {

  if (!file.exists(paste0("./data/",i, ".csv"))) {
    print("Please remove the Company. Files Missing for:")
    print(i)
    break
  }

  #process stock price/volume files from data directory
  #pick the csv file of every company in the input list and create a dataframe in a loop

  df_c <- read.csv(paste0("./data/",i, ".csv"), header = T)
  df_c$period <- as.Date(df_c$Date) # add a date field
  df_c$ticker <- i # attach ticker
  df_companies_subset <- subset(df_companies, ticker==i)
  df_c <- merge(x=df_c, y=df_companies_subset, by="ticker",all.x=TRUE )#to get name
  assign(paste0("df_",i), df_c) #create a new data frame for later use

  # Appends company data frames to one for graphs.

  if (exists("df_c_all")) {
    df_c_all <- rbind(df_c_all, df_c)

  }else {
    df_c_all=df_c}

  # Process Sentiments Index from data directory

  df_bsi <- read.csv(paste0("./data/",i, "_BSI.csv"), header = T)
  df_bsi$period <- as.Date(parse_date_time(df_bsi$created_date, c('%Y-%m-%d', '%m/%d/%y')))
  df_bsi$ticker <- i # attach ticker
  df_bsi <- merge(x=df_bsi, y=df_companies_subset, by="ticker",all.x=TRUE )##to get name
  df_bsi <- df_bsi[c('ticker','period','name','bsi_score')]

  assign(paste0("df_",i,"_BSI"), df_bsi) #create a new data frame forlater use

  # create a data frame to append  every data frame for graphs
```

```
  if (exists("df_bsi_all")) {
    df_bsi_all <- rbind(df_bsi_all, df_bsi)

  }else {
    df_bsi_all <- df_bsi}

}
```

```
## [1] "Please remove the Company. Files Missing for:"
## [1] "TEST"
```

```
summary(df_c_all) # Stock price series
```

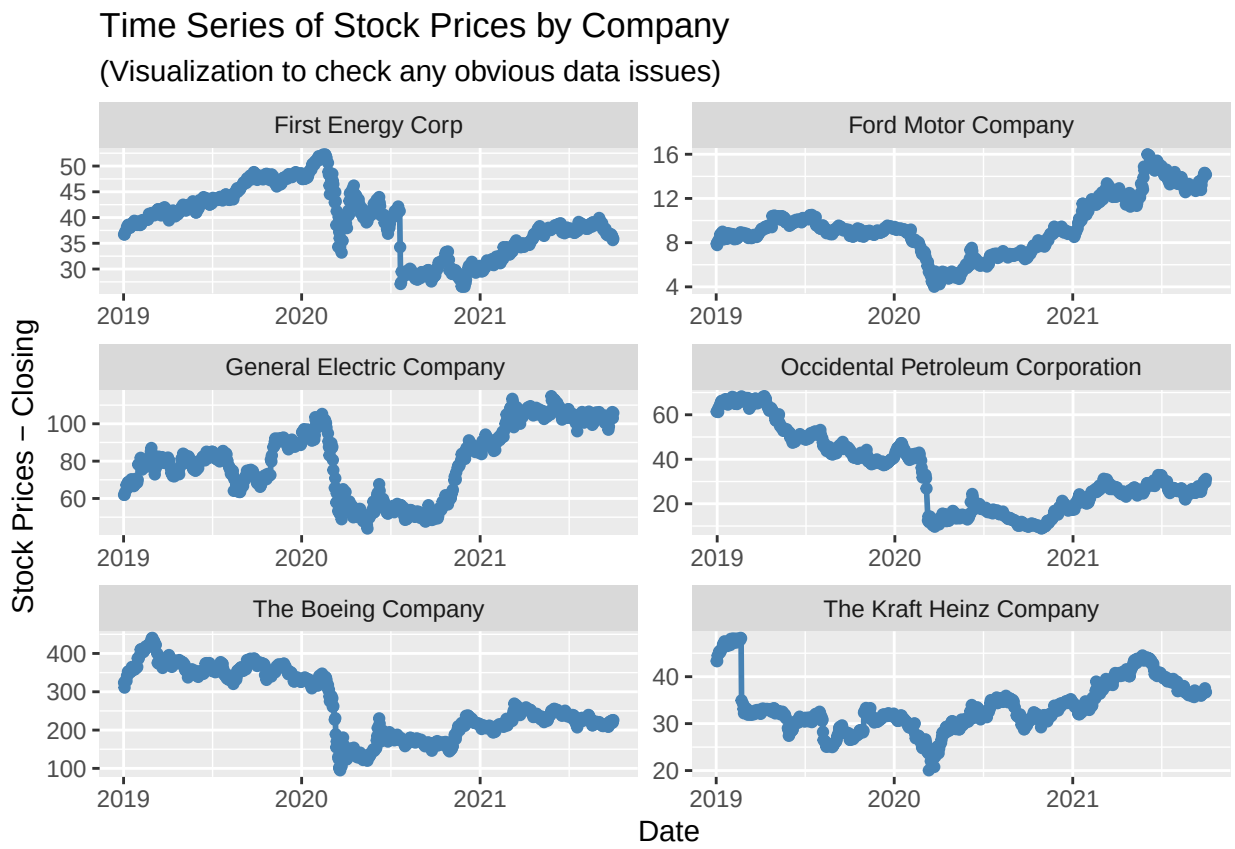
```
##     ticker              Date                Open              High
##  Length:4165        Length:4165        Min.   :  4.27   Min.   :  4.42
##  Class :character   Class :character   1st Qu.: 26.26   1st Qu.: 26.81
##  Mode  :character   Mode  :character   Median : 39.74   Median : 40.36
##                                        Mean   : 77.14   Mean   : 78.23
##                                        3rd Qu.: 81.20   3rd Qu.: 82.80
##                                        Max.   :446.01   Max.   :446.01
##       Low              Close           Adj.Close           Volume
##  Min.   :  3.96   Min.   :  4.01   Min.   :  4.01   Min.   :   958300
##  1st Qu.: 25.73   1st Qu.: 26.26   1st Qu.: 25.28   1st Qu.:  5154700
##  Median : 39.27   Median : 39.85   Median : 37.90   Median :  9205138
##  Mean   : 75.97   Mean   : 77.07   Mean   : 75.60   Mean   : 19959508
##  3rd Qu.: 80.16   3rd Qu.: 81.28   3rd Qu.: 80.53   3rd Qu.: 23073575
##  Max.   :440.19   Max.   :440.62   Max.   :430.30   Max.   :282394100
##      period               name
##  Min.   :2018-12-31   Length:4165
##  1st Qu.:2019-09-10   Class :character
##  Median :2020-05-18   Mode  :character
##  Mean   :2020-05-17
##  3rd Qu.:2021-01-26
##  Max.   :2021-10-01
```

```
summary(df_bsi_all) #BSI index series
```

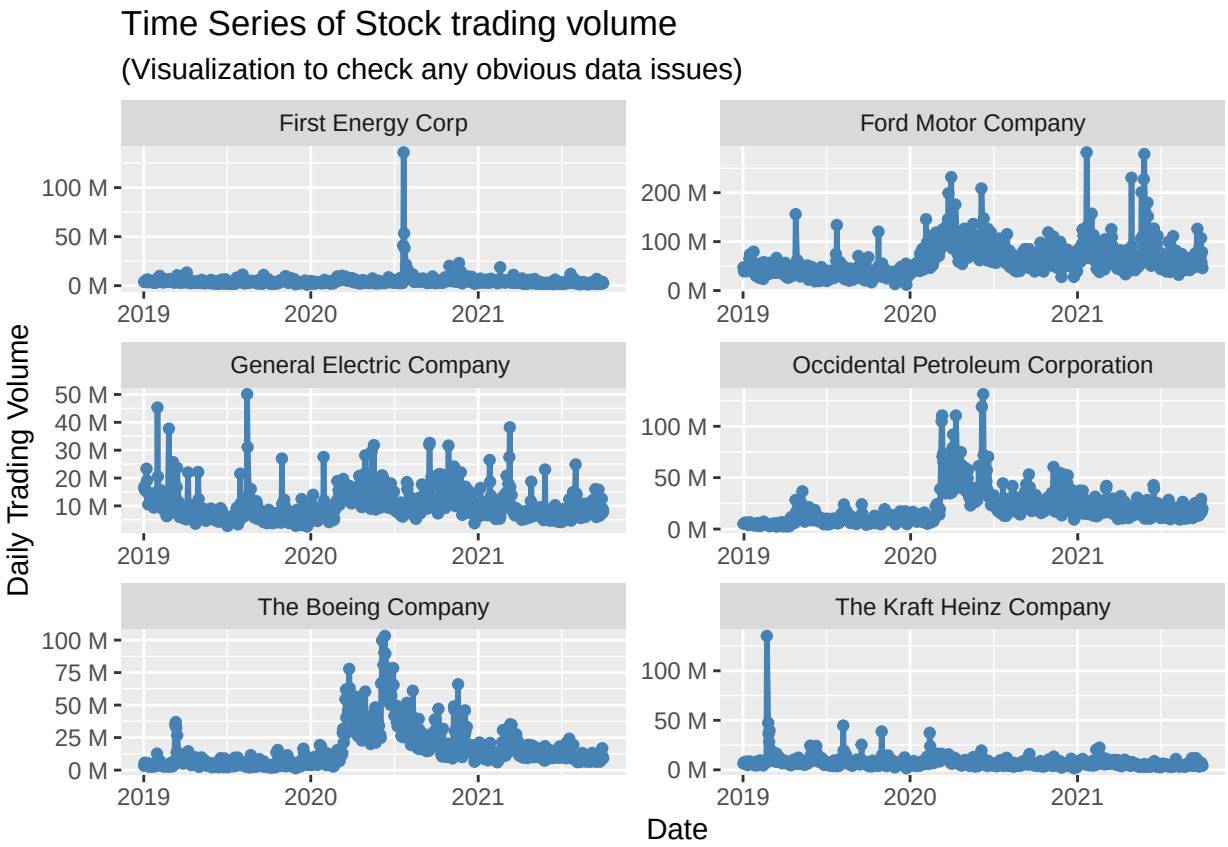
```
##     ticker              period               name             bsi_score
##  Length:3012        Min.   :2019-08-27   Length:3012        Min.   :-19.00
##  Class :character   1st Qu.:2020-02-28   Class :character   1st Qu.:  6.00
##  Mode  :character   Median :2020-08-26   Mode  :character   Median : 15.00
##                     Mean   :2020-08-27                      Mean   : 13.26
##                     3rd Qu.:2021-03-01                      3rd Qu.: 27.00
##                     Max.   :2021-08-27                      Max.   : 62.00
```

Visualizing the Data Series used in Analysis

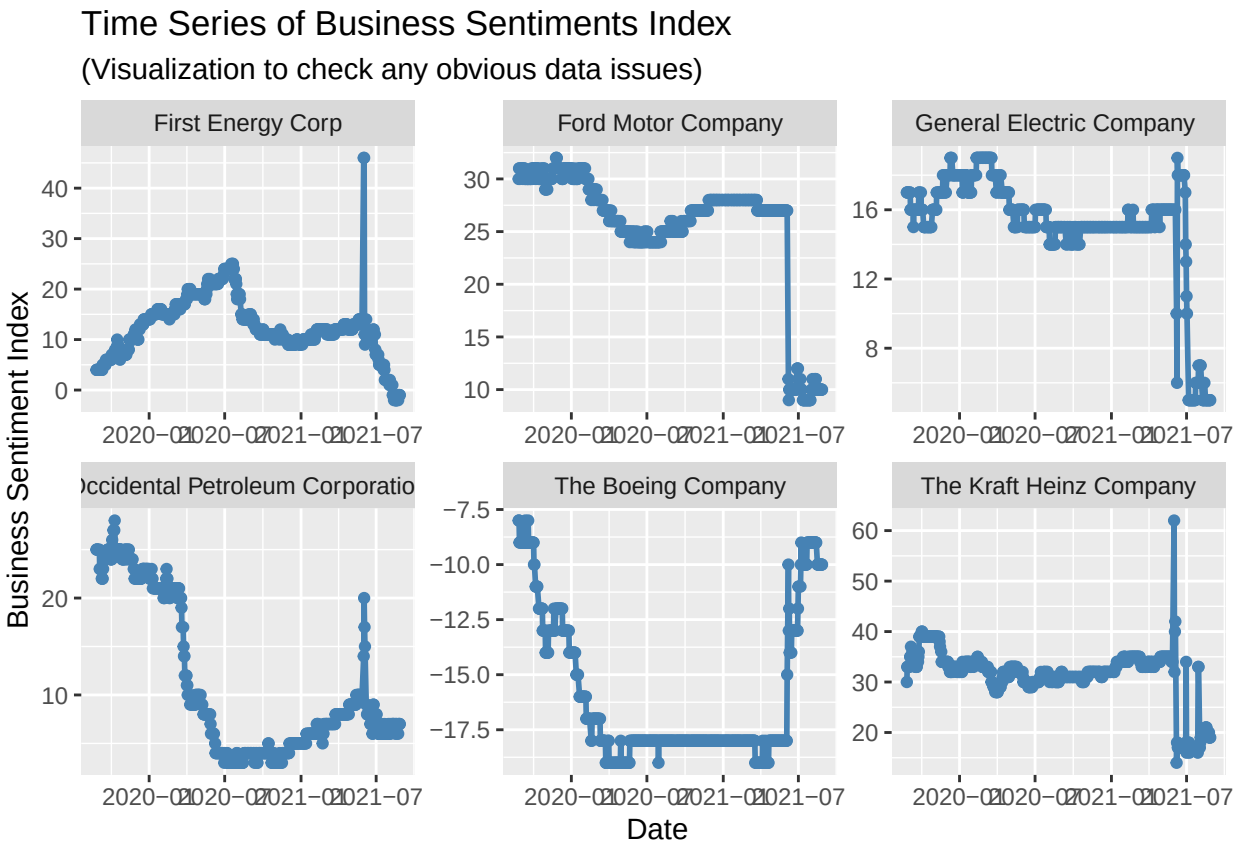
```
ggplot(data = df_c_all, aes(period, Close)) +
  geom_line(color = "steelblue", size = 1) +
  geom_point(color = "steelblue") +
  labs(title = "Time Series of Stock Prices by Company",
       subtitle = "(Visualization to check any obvious data issues)",
       y = "Stock Prices - Closing ", x = " Date") +
  facet_wrap(~name,scales="free",ncol=2)
```



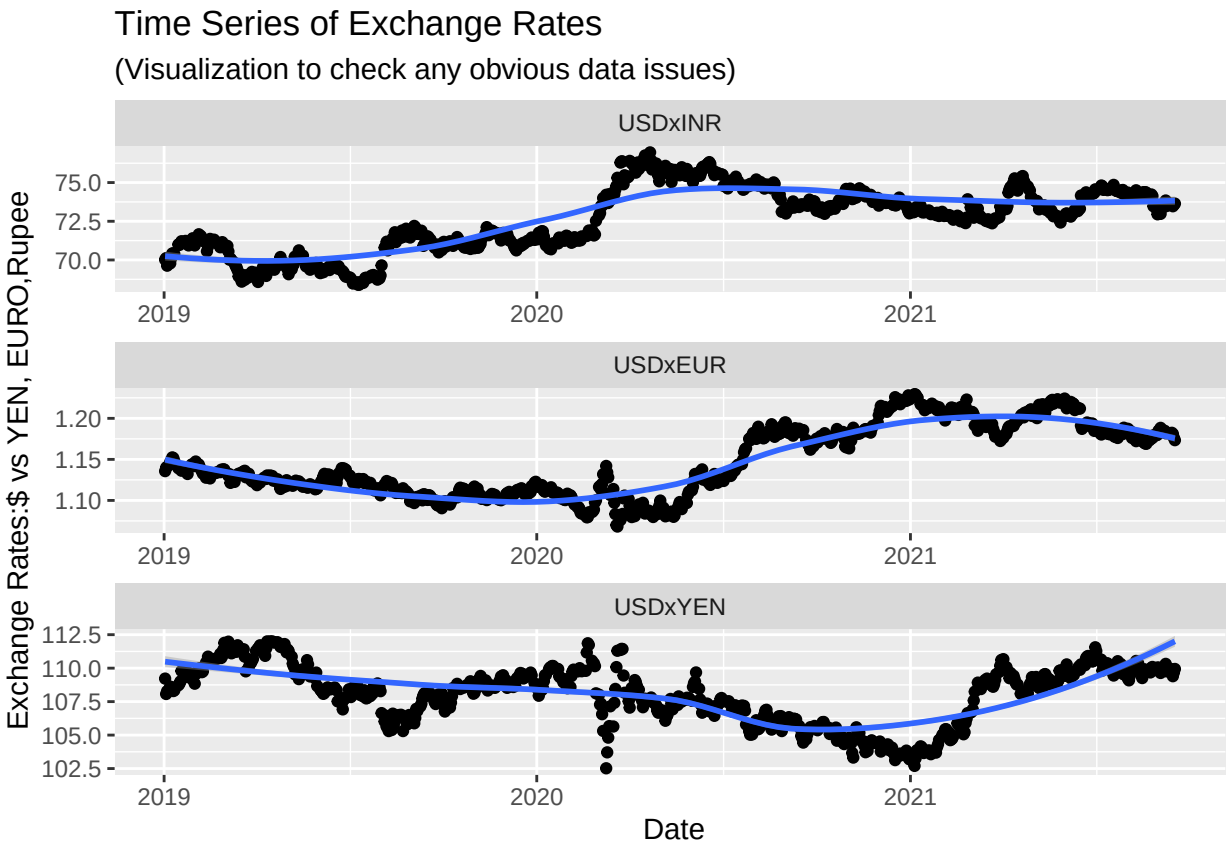

```
ggplot(data = df_c_all, aes(period, Volume)) +
  geom_line(color = "steelblue", size = 1) +
  geom_point(color = "steelblue") +
  labs(title = "Time Series of Stock trading volume",
       subtitle = "(Visualization to check any obvious data issues)",
       y = "Daily Trading Volume", x = " Date") +
  facet_wrap(~name,scales="free",ncol=2) +
  scale_y_continuous(labels = label_number(suffix = " M", scale = 1e-6))
```




```
ggplot(data = na.omit(df_bsi_all), aes(period,bsi_score)) +
  geom_line(color = "steelblue", size = 1) +
  geom_point(color = "steelblue") +
  labs(title = "Time Series of Business Sentiments Index",
       subtitle = "(Visualization to check any obvious data issues)",
       y = "Business Sentiment Index", x = " Date") +
  facet_wrap(~name,scales="free",ncol=3)
```




```
ggplot(na.omit(df_forex2), aes(period,value)) +
  geom_point() +
  stat_smooth(formula = y ~ x,method=loess) +
  facet_wrap(~variable,scales="free",ncol=1)+
  labs(title = "Time Series of Exchange Rates",
       subtitle = "(Visualization to check any obvious data issues)",
       y = "Exchange Rates:$ vs YEN, EURO,Rupee", x = " Date")
```



User Defined Function for Company Level Analysis

This is a generic function that takes company name, dependent(x) and independent(y) variables for the regression analysis. It also ensures that necessary data frames to conduct the analysis are existing.

```
fn.regress <- function(company,var.Y,var.X){

  #create local dataframes based on the company names

  if (!exists(paste0("df_",company))) {
     print("Company Datafiles Missing.Process Terminated")
     return(FALSE)
  }else {

  df_b  <- get(paste0("df_",company,"_BSI"))
  df_s  <- get(paste0("df_",company))
  df_cname <- df_companies[df_companies[,1]  == company, ]
  }


  # check all three datasets for data issues and make it global

  d1 <<- dfSummary(df_forex, style = "grid", plain.ascii = TRUE)
  d2 <<- dfSummary(df_s, style = "grid", plain.ascii = TRUE)
  d3 <<- dfSummary(df_b, style = "grid", plain.ascii = TRUE)
```

```
  df_forex <- na.omit(df_forex)
  d4 <<- dfSummary(df_forex, style = "grid", plain.ascii = TRUE)

  # cat(df_cname[1,2])
  # print(d1)
  # print(d2)
  # print(d4)

  # merge all three datasets

  df_all <- merge(df_b, df_forex, by="period",all.x=TRUE )  # merge with Forex file

  df_all <- merge(x=df_all, y=df_s, by="period",all.x=TRUE )  # merge with stock price

  df_all <- subset(na.omit(df_all),select=c(period, ticker.x,   name.x,bsi_score,USDxINR,
                                            USDxEUR,    USDxYEN,    Close,  Volume))

  # dfSummary(df_all, style = "grid", plain.ascii = TRUE)

  df_all <- rename(df_all, c("ticker.x"="ticker", "name.x"="CompanyName",
                             "Close"="Stock.Price", "Volume"="Stock.Volume"))

  d5 <<- dfSummary(df_all, style = "grid", plain.ascii = TRUE) #output saved global

  # Scatterplot the variable relationship to visualize. output saved global

  scatter <<- ggplot(df_all, aes(x=df_all[,var.X], y=df_all[,var.Y])) +
    geom_point()+
    geom_smooth(formula = y ~ x,method=lm)+ ggtitle(df_cname[1,2] ) +
    xlab(var.X) + ylab(var.Y)

  print(scatter)

  # Fit the simple regression model and create a summary

  multi.fit <- lm(Stock.Price~USDxEUR, data=df_all)
  std_results <<- summary(multi.fit) # std model results
  tab_results <<- tab_model(multi.fit) # tabulated results
  residual_plot <<- autoplot(multi.fit) # residuals plot
  return(TRUE)
}
```

Calling the Function to Test a Company with no data

Expectation is that even if the user added a company name in to the list of companies to analyze and there is no data downloaded for that company, this should not result in an error and the user should be alerted about missing data.

```
source("./src/fn_regression.R") # call the function to global environment to be sure

# Call Function with parameters company ticker symbol(name), Y variable, X Vairable

fn.regress("UNK", "Stock.Price","bsi_score")
```

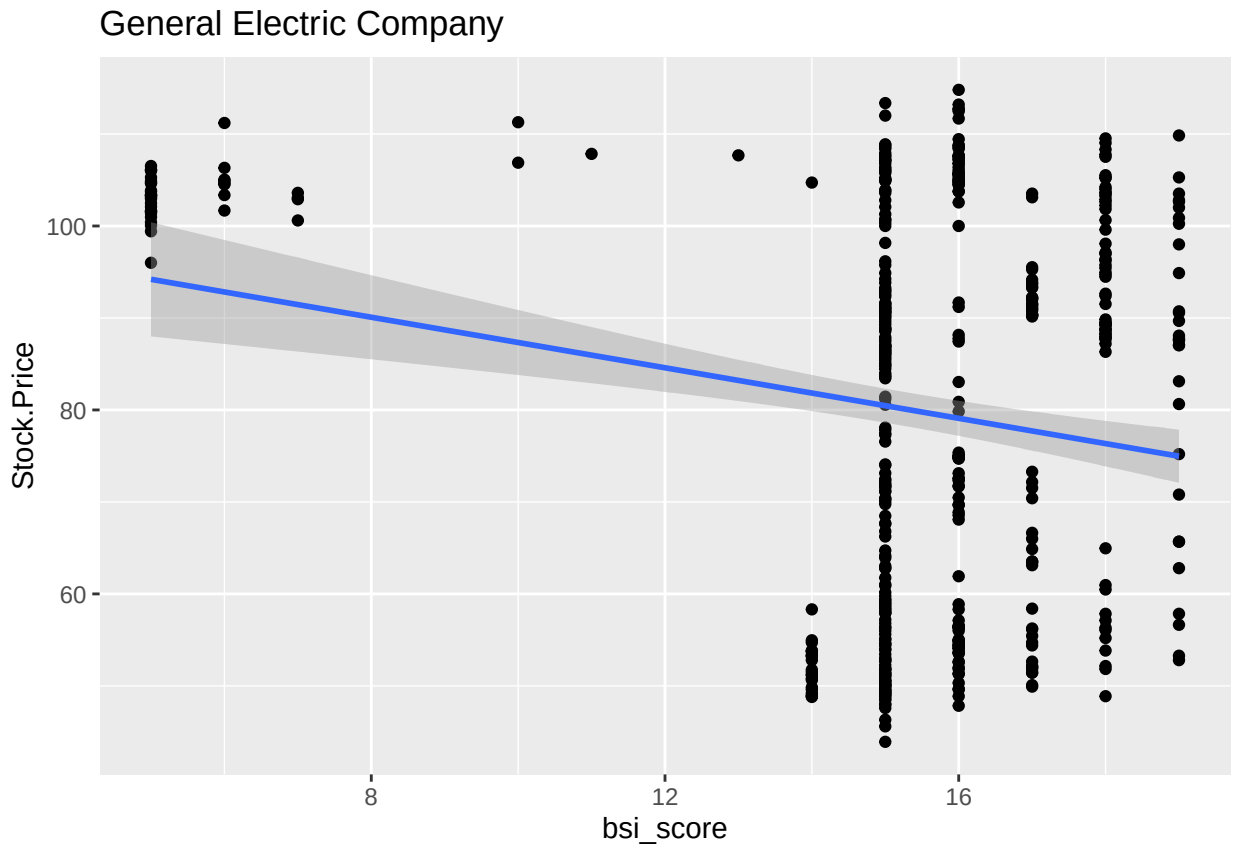
```
## [1] "Company Datafiles Missing.Process Terminated"
```

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

Calling the Function to show analysis

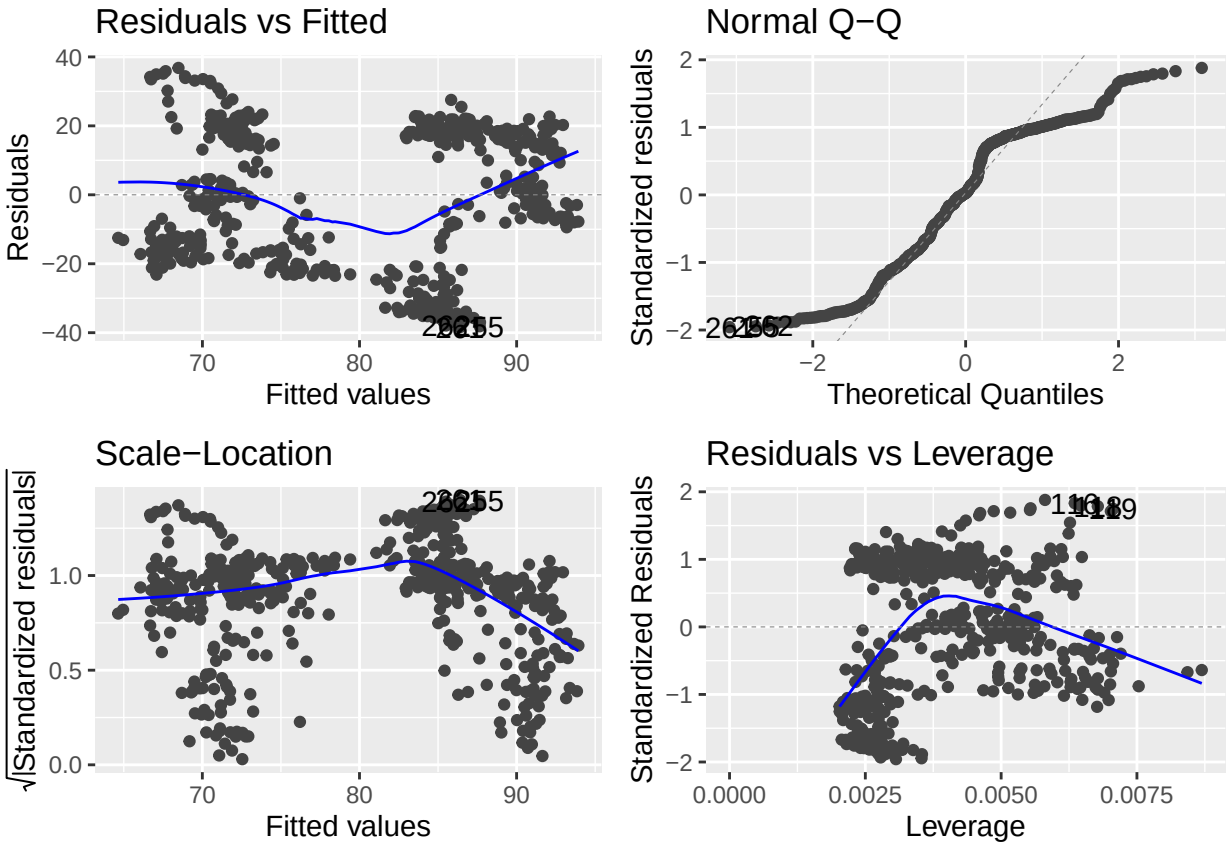
This outputs a scatterplot that shows the relationship between the variables, Standard Summary results of a regression function, Tabulated results of a regression function with coefficients and plots residuals for better diagnostics. Simple regression can be performed for any company and any set of variables. The coefficients and p values quantify the sensitivity of the variable and degree of confidence. Multiple regression and tests of time series properties are for the future development.

```
source("./src/fn_regression.R") # call the function to global environment to be sure

# Call Function with parameters company ticker symbol(name), Y variable, X Vairable

fn.regress("GE", "Stock.Price","bsi_score")
```



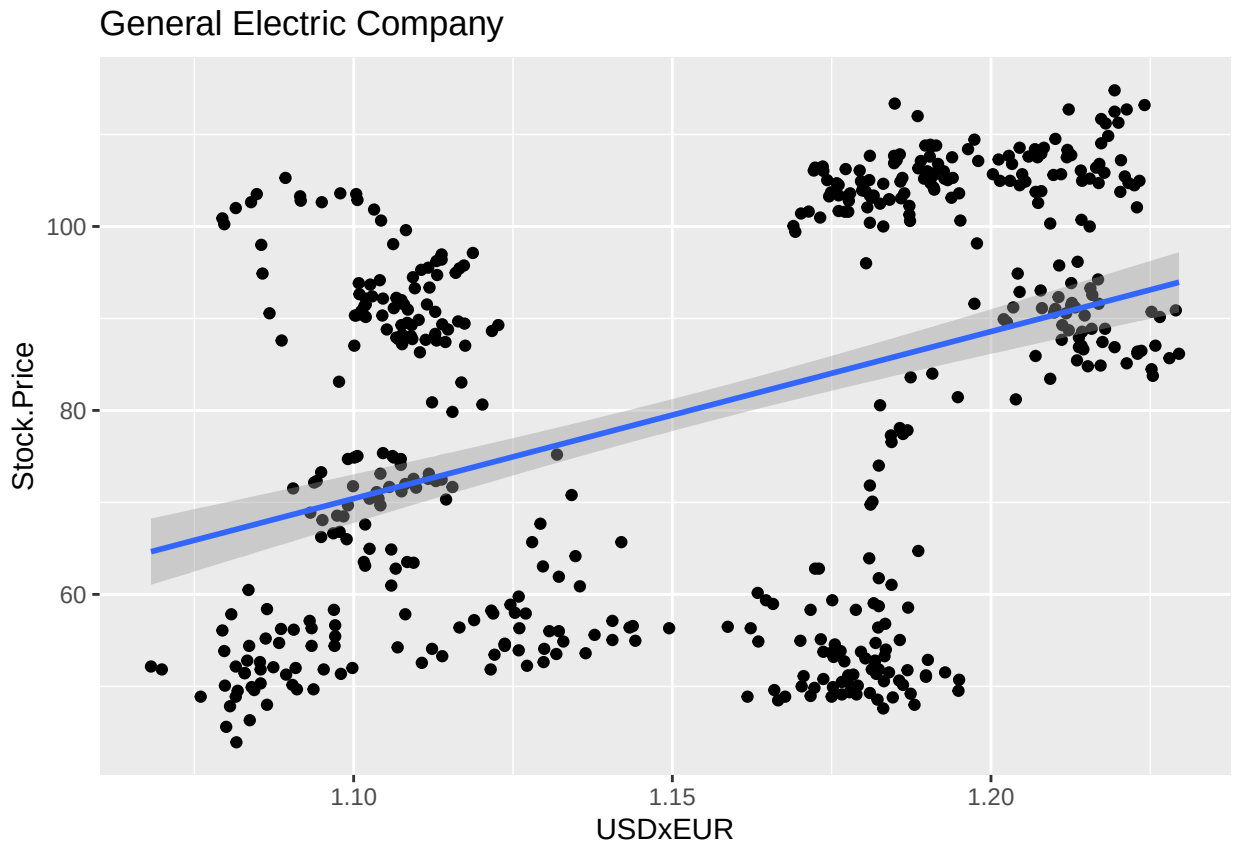

```
##
## Call:
## lm(formula = Stock.Price ~ USDxEUR, data = df_all)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.395 -16.665   0.575  17.901  36.805
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  -129.30      21.63  -5.978 4.33e-09 ***
## USDxEUR       181.56      18.73   9.694  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.64 on 496 degrees of freedom
## Multiple R-squared:  0.1593, Adjusted R-squared:  0.1576
## F-statistic: 93.97 on 1 and 496 DF,  p-value: < 2.2e-16
```

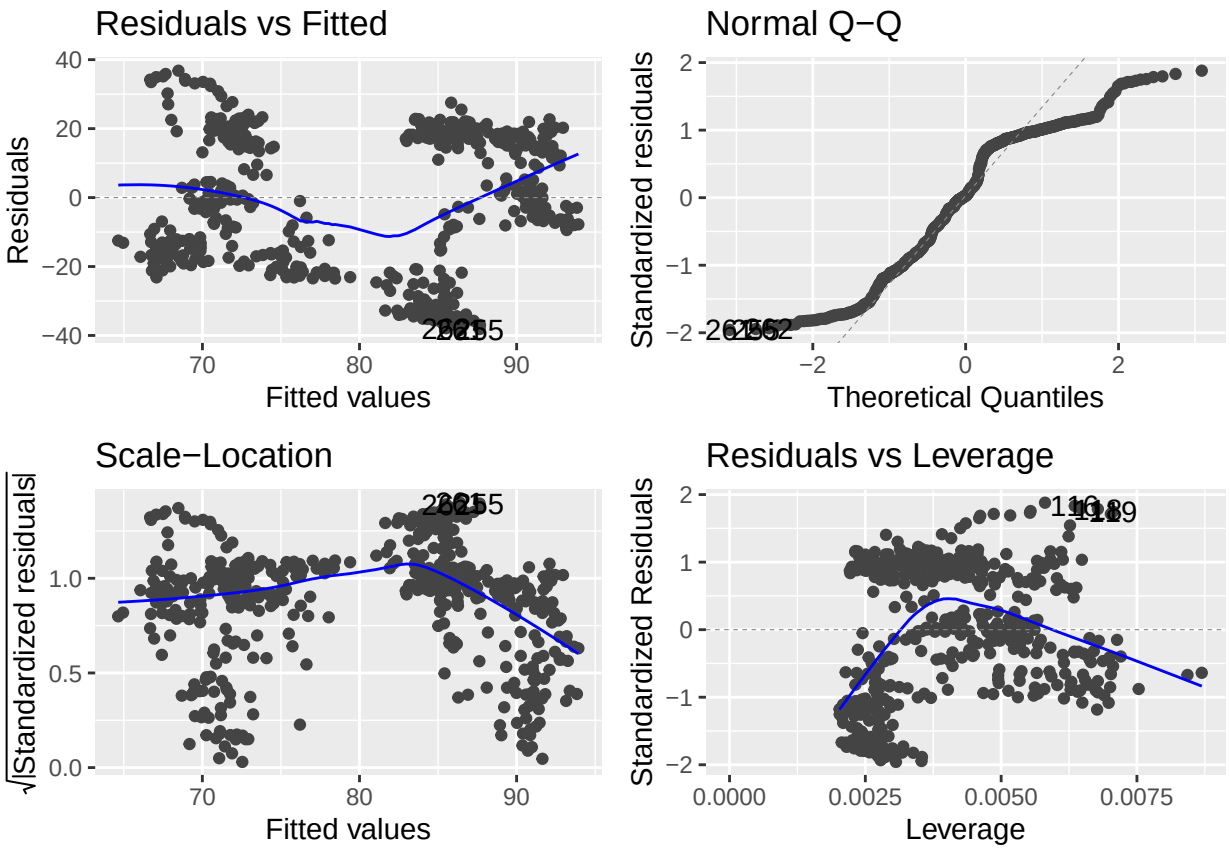



```
## [1] TRUE
```

```
fn.regress("GE", "Stock.Price","USDxEUR")
```




```
##
## Call:
## lm(formula = Stock.Price ~ USDxEUR, data = df_all)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.395 -16.665   0.575  17.901  36.805
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -129.30      21.63  -5.978 4.33e-09 ***
## USDxEUR       181.56      18.73   9.694  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.64 on 496 degrees of freedom
## Multiple R-squared:  0.1593, Adjusted R-squared:  0.1576
## F-statistic: 93.97 on 1 and 496 DF,  p-value: < 2.2e-16
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q−Q
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized Residuals
Leverage

```
## [1] TRUE
```