Time Series Analysis of AMD Stock Data

Pooria Assarehha

2024-01-06

The Data

The data was downloaded from NASDAQ official website.

We will view the data first to see what must be done with it.

```
df = read.csv("HistoricalData_1704473693714.csv")
str(df)
```

```
## 'data.frame':    2517 obs. of  6 variables:
##  $ Date      : chr  "01/04/2024" "01/03/2024" "01/02/2024" "12/29/2023" ...
##  $ Close.Last: chr  "$136.01" "$135.32" "$138.58" "$147.41" ...
##  $ Volume    : int  58610290 61988580 64902030 62079190 63800680 49033420 47157430 35396580 47179360
##  $ Open      : chr  "$134.30" "$135.71" "$144.28" "$149.50" ...
##  $ High      : chr  "$137.70" "$137.43" "$144.40" "$151.05" ...
##  $ Low       : chr  "$134.00" "$133.7413" "$137.43" "$147.20" ...
```

```
summary(df)
```

```
##      Date            Close.Last            Volume             Open
##  Length:2517        Length:2517        Min.   :  2606577   Length:2517
##  Class :character   Class :character   1st Qu.: 29548270   Class :character
##  Mode  :character   Mode  :character   Median : 48655280   Mode  :character
##                                        Mean   : 54922966
##                                        3rd Qu.: 72964740
##                                        Max.   :323844500
##      High               Low
##  Length:2517        Length:2517
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

we have a Date column which is stored as chr type. we will convert it to a datetime object. We have numerical columns which are stored as chr because of presence of a dollar sign, which we have to remove then convert them to numeric.

Data does not show any missing records, but we will examine Date column for any missing days. Since missing days are probably the days that the market is closed, we can argue that holidays do not count as missing data.

Prepros

– Changing time from chr type to POSIXlt datetime class

```
df$Date <- strptime(df$Date, format = "%m/%d/%Y")
str(df)
```

```
## 'data.frame':    2517 obs. of  6 variables:
##  $ Date      : POSIXlt, format: "2024-01-04" "2024-01-03" ...
##  $ Close.Last: chr  "$136.01" "$135.32" "$138.58" "$147.41" ...
##  $ Volume    : int  58610290 61988580 64902030 62079190 63800680 49033420 47157430 35396580 47179360
##  $ Open      : chr  "$134.30" "$135.71" "$144.28" "$149.50" ...
##  $ High      : chr  "$137.70" "$137.43" "$144.40" "$151.05" ...
##  $ Low       : chr  "$134.00" "$133.7413" "$137.43" "$147.20" ...
```

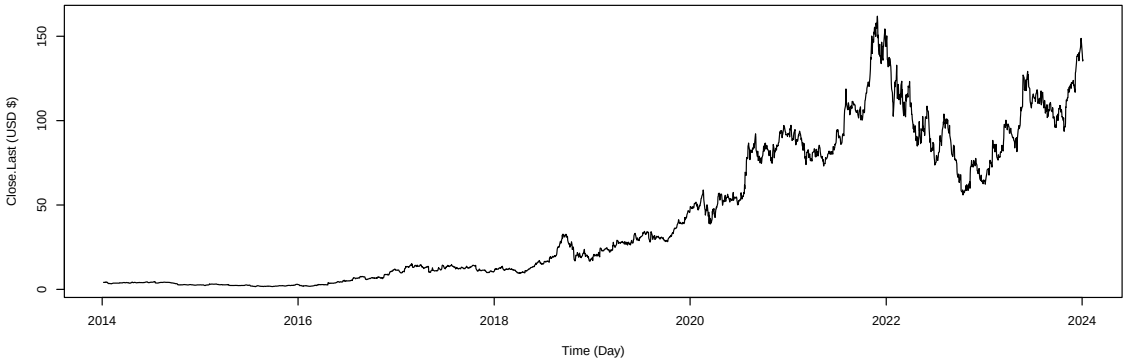
– Changing chr fields to num

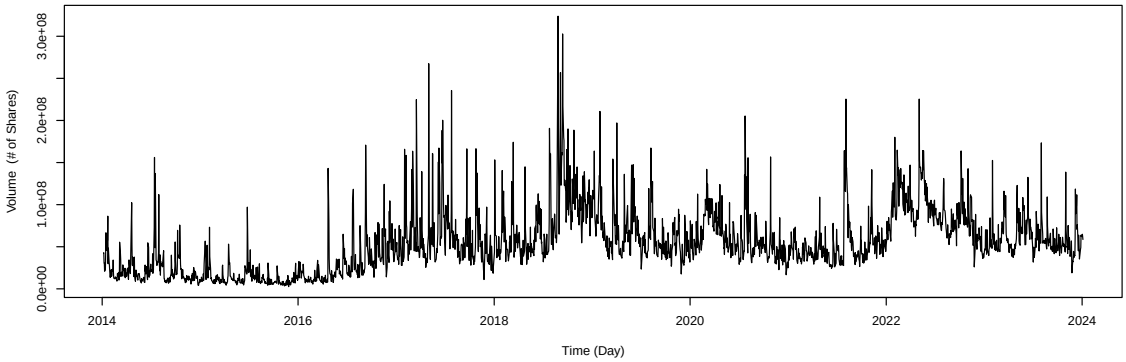
```
rmvdollar <- function(x){return(gsub("\\$", "", x))}

df$Low <- as.numeric(rmvdollar(df$Low))
df$Open <- as.numeric(rmvdollar(df$Open))
df$High <- as.numeric(rmvdollar(df$High))
df$Close.Last <- as.numeric(rmvdollar(df$Close.Last))

str(df)
```

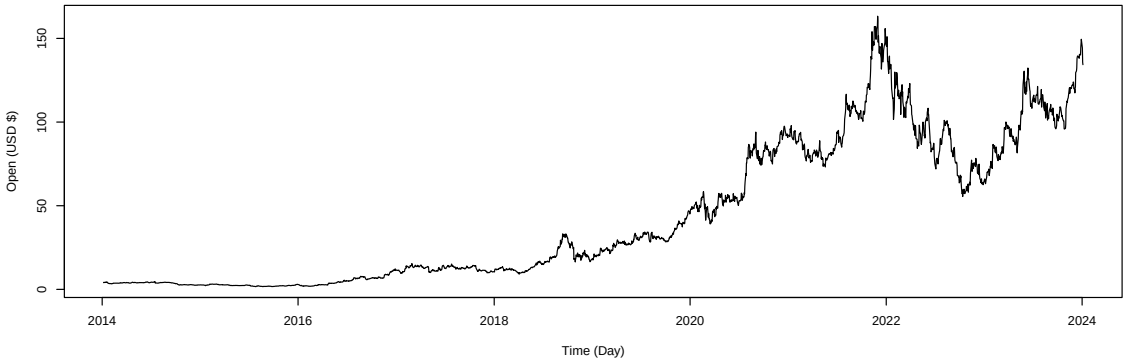
```
## 'data.frame':    2517 obs. of  6 variables:
##  $ Date      : POSIXlt, format: "2024-01-04" "2024-01-03" ...
##  $ Close.Last: num  136 135 139 147 149 ...
##  $ Volume    : int  58610290 61988580 64902030 62079190 63800680 49033420 47157430 35396580 47179360
##  $ Open      : num  134 136 144 150 147 ...
##  $ High      : num  138 137 144 151 150 ...
##  $ Low       : num  134 134 137 147 146 ...
```

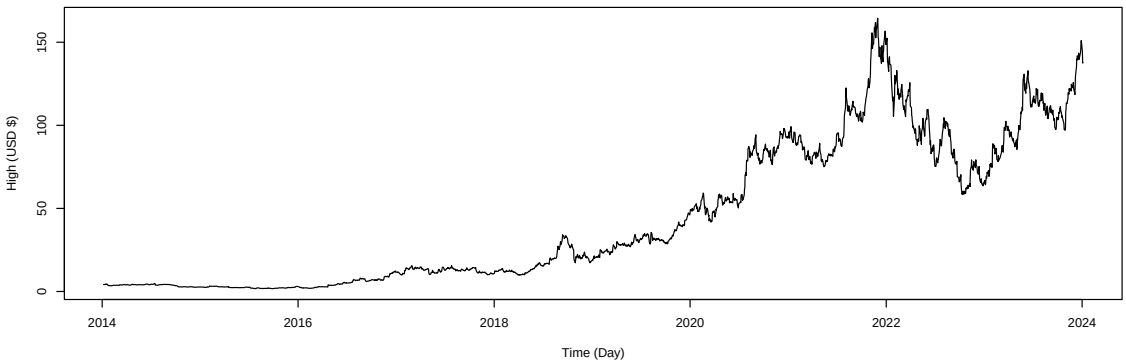
Finding a Good Transfromation

```
library(TSA)
par(mfrow=c(4,1))
for(i in 1:5){
  if(is.numeric(df[,i])){
    plot(
      df$Date,
      df[,i],
      type = "l",
      xlab="Time (Day)",
      ylab = paste(colnames(df)[i], if(i == 3) " (# of Shares)"  else "(USD $)" )
    )
  }
}
```


Close.Last (USD $)
0
50
100
150
2014
2016
2018
2020
2022
2024
Time (Day)

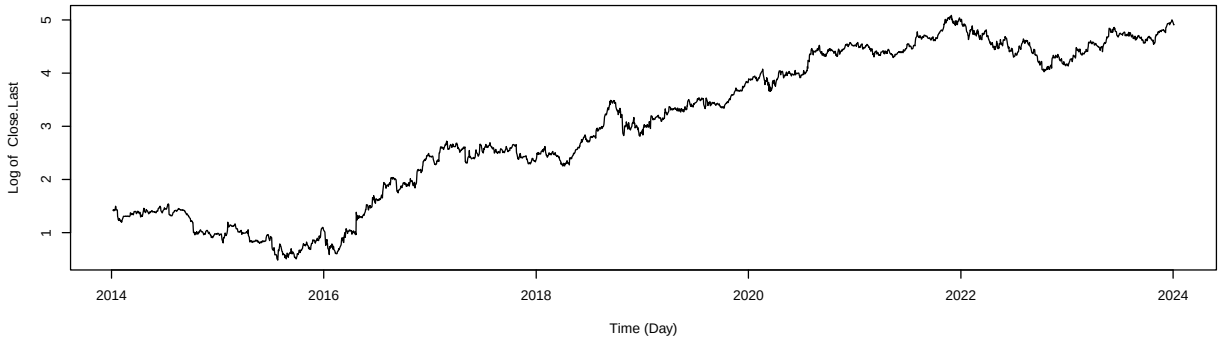

Volume (# of Shares)
0.0e+00
1.0e+08
2.0e+08
3.0e+08
2014
2016
2018
2020
2022
2024
Time (Day)

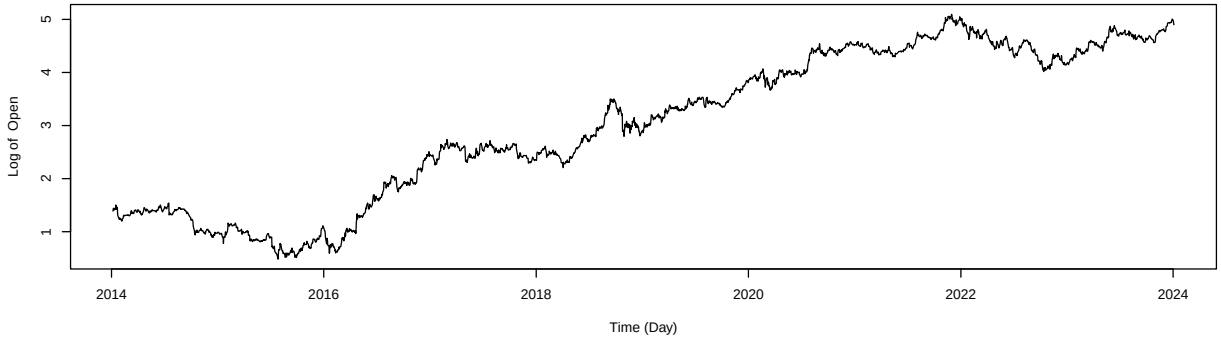

Open (USD $)
0
50
100
150
2014
2016
2018
2020
2022
2024
Time (Day)

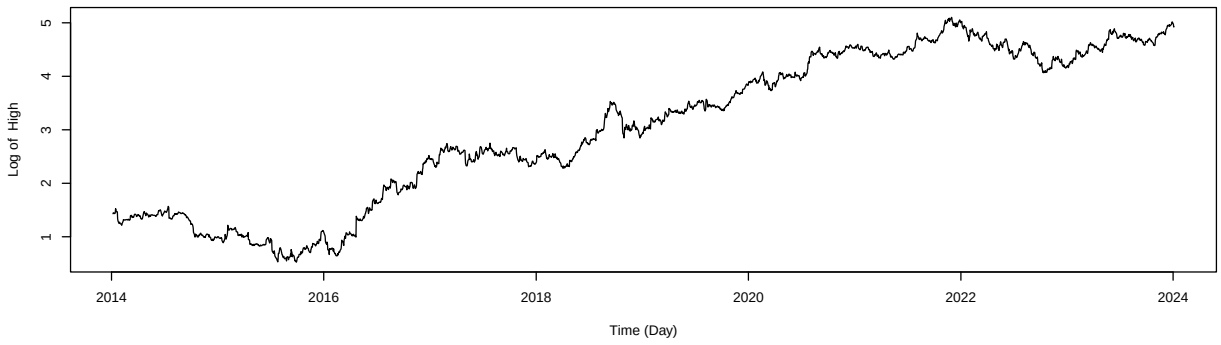

High (USD $)
0
50
100
150
2014
2016
2018
2020
2022
2024
Time (Day)

We can argue an exponential trend for USD variables.

```
par(mfrow=c(3,1))
for(i in 1:5){
  if(all(class(df[,i]) == "numeric")){
    plot(
      df$Date,
      log(df[,i]),
      type = "l",
      xlab="Time (Day)",
      ylab = paste("Log of " , colnames(df)[i])
    )
  }
}
```

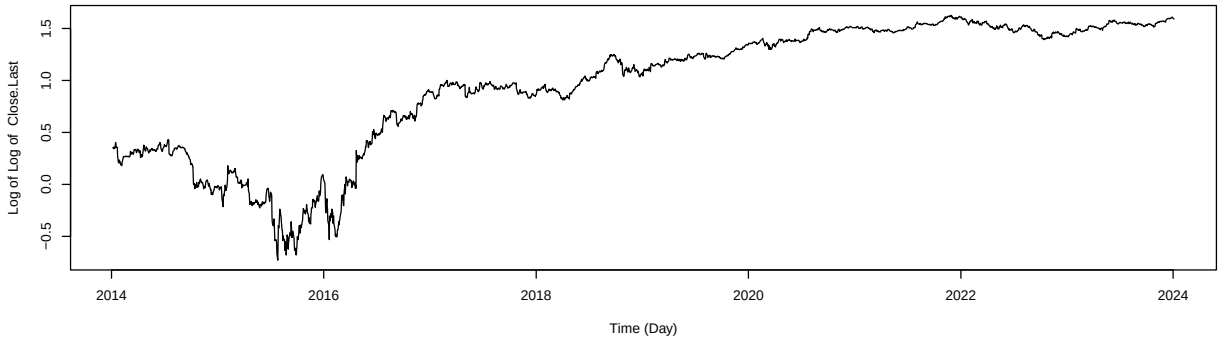


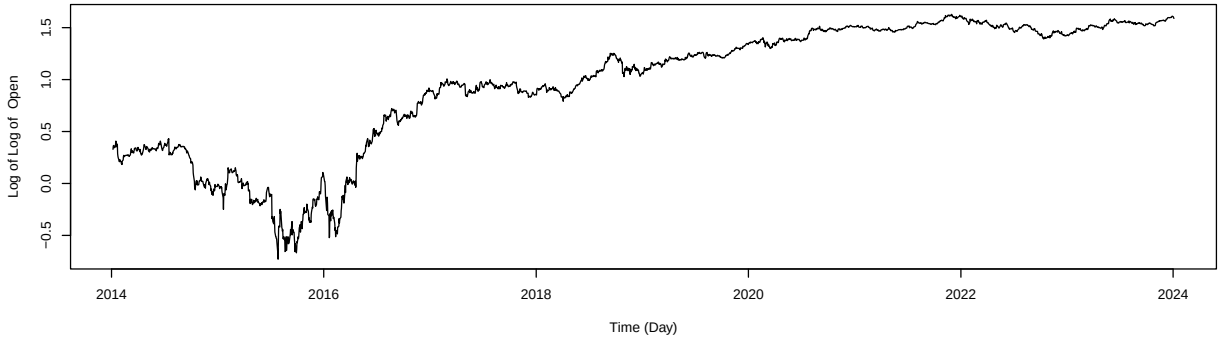


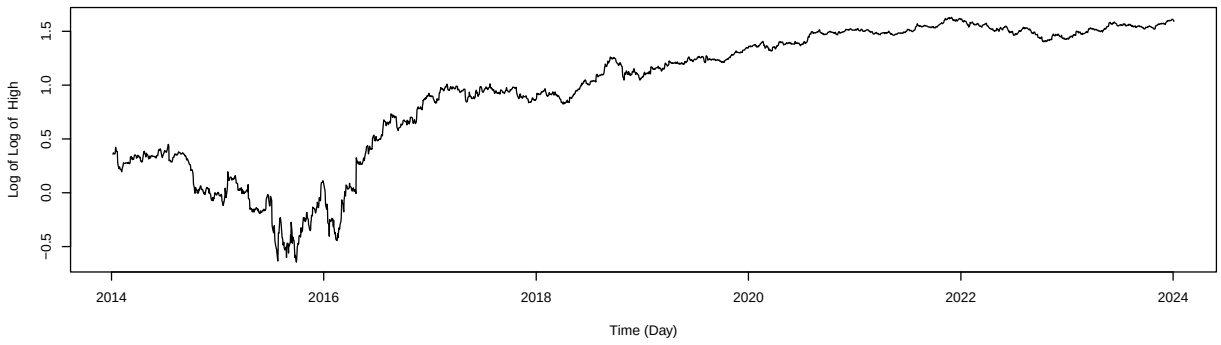


it still has a trend. what should we do?

```
par(mfrow=c(3,1))
for(i in 1:5){
  if(all(class(df[,i]) == "numeric")){ ## Using all() to avoid a warning
    plot(
      df$Date,
      log(log(df[,i])),
      type = "l",
      xlab="Time (Day)",
      ylab = paste("Log of Log of " , colnames(df)[i])
    )
  }
}
```

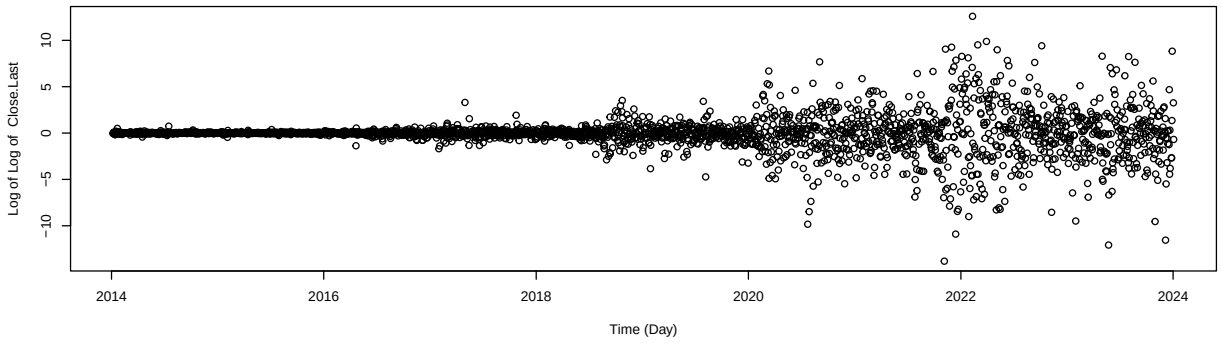


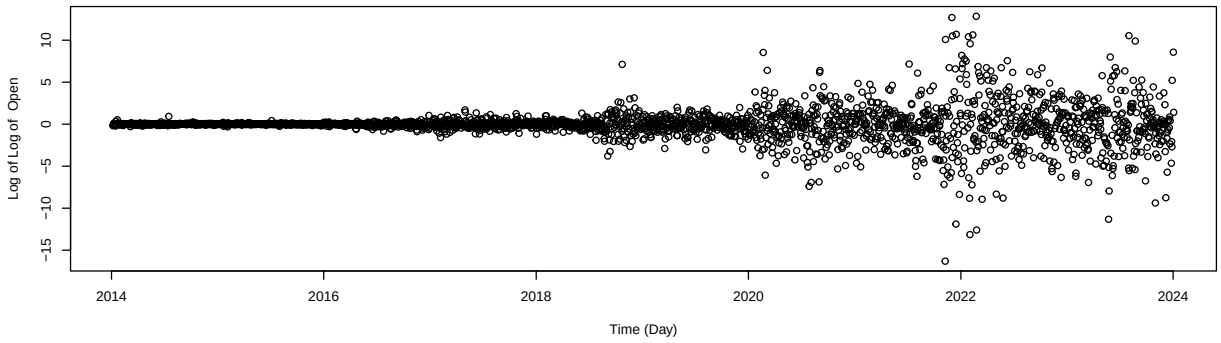


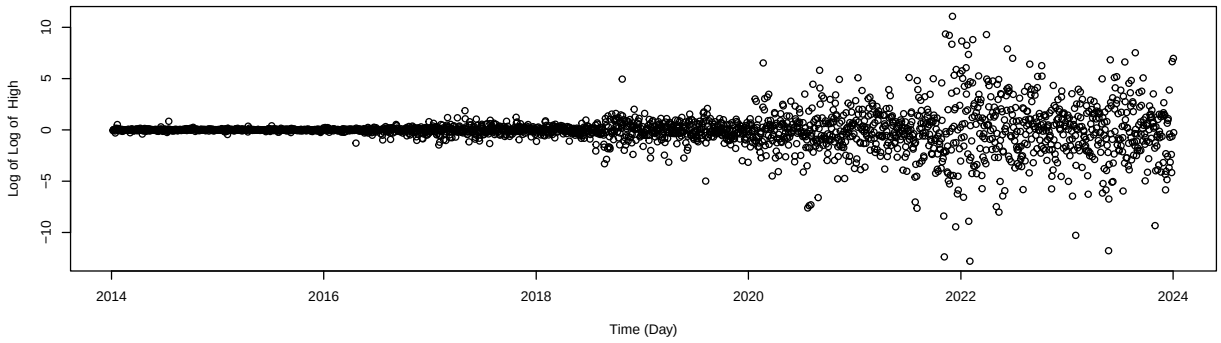


Let's plot a first difference to see whats what.

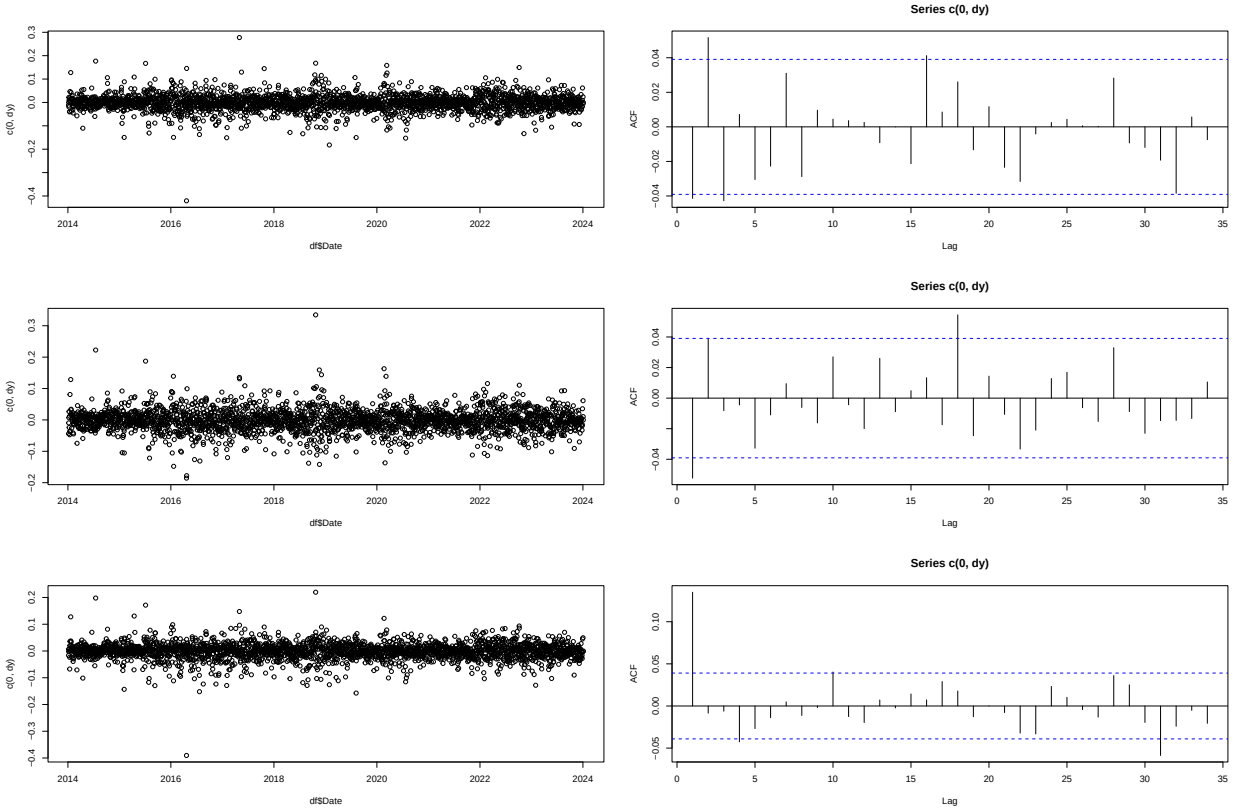
```
par(mfrow=c(3,1))
for(i in 1:5){
  if(all(class(df[,i]) == "numeric")){ ## Using all() to avoid a warning
    y = df[,i]
    dy = y - zlag(y)
    plot(
      df$Date,
      dy,
      # c(0,diff(df[,i])), equiv
      type = "p",
      xlab="Time (Day)",
      ylab = paste("Log of Log of " , colnames(df)[i])
    )
  }
}
```





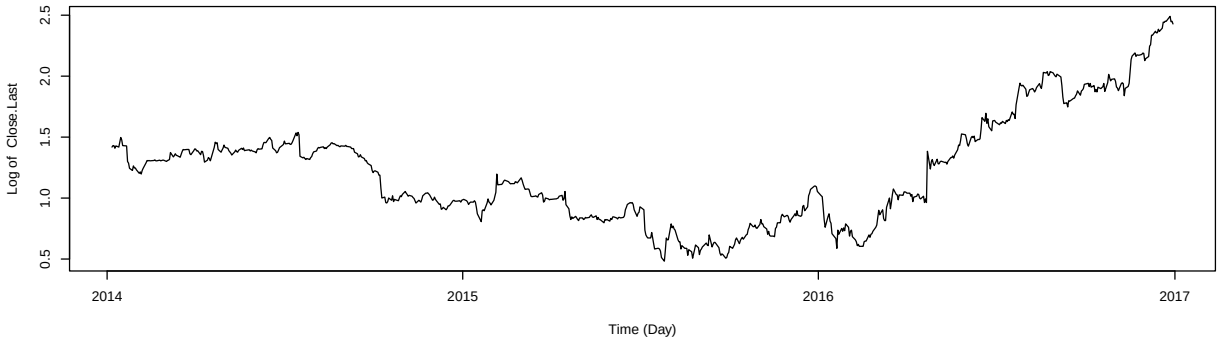


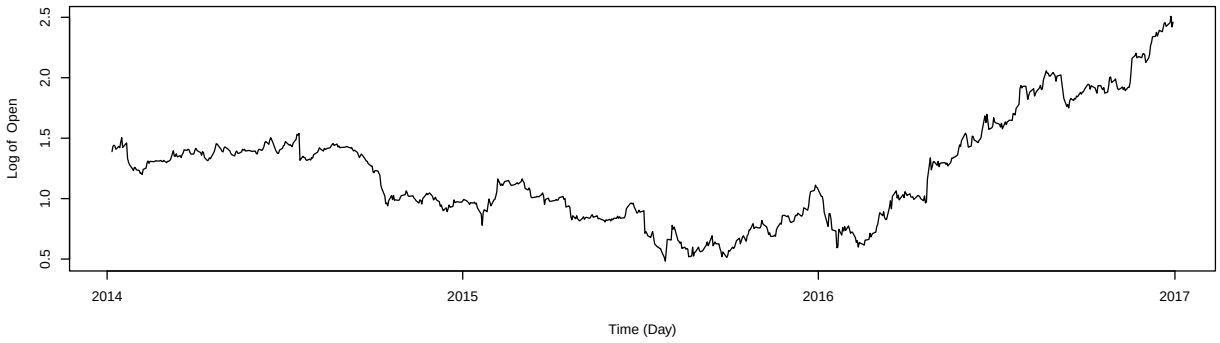

```r
par(mfrow=c(3,2))
for(i in 1:5){
  if(all(class(df[,i]) == "numeric")){ ## Using all() to avoid a warning
    y = log(df[,i])
    dy = diff(y)
    #plot(df$Date, y)
    plot(df$Date, c(0,dy))
    acf(c(0,dy))
  }
}
```

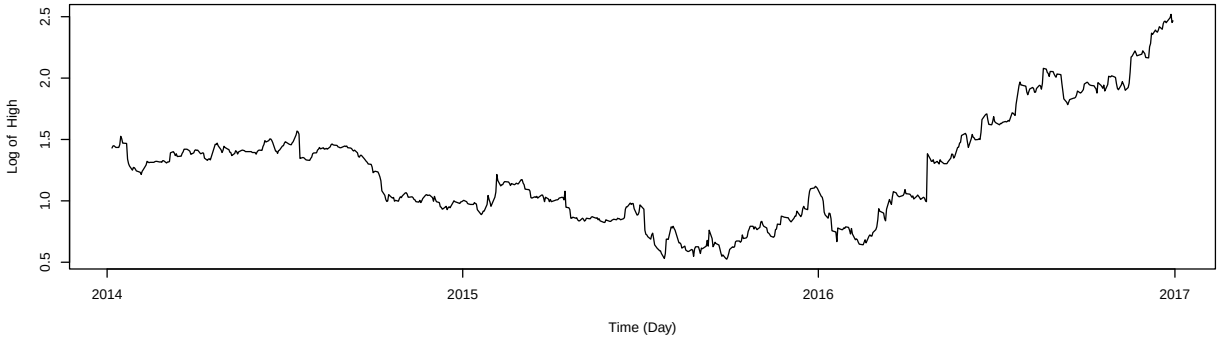


We may be better off segmenting data:

```
df2014_16 = df[1764:nrow(df),]
par(mfrow=c(3,1))
for(i in c(2,4,5)){
  plot(
    df2014_16$Date,
    log(df2014_16[,i]),
    type = "l",
    xlab="Time (Day)",
    ylab = paste("Log of " , colnames(df)[i])
  )
}
```







The industry-standard transformation is to calculate percent returns, via a slight alteration of first-order differencing:

$$R_t = \frac{P_t}{P_{t-1}} - 1$$

Where P is price, and t is time. We call this simply "returns."

Many academics especially prefer to also take the logarithm:

$$r_t = \ln(1 + R_t) = \ln(\frac{P_t}{P_{t-1}}) = \ln(P_t) - \ln(P_{t-1})$$

Predicting price isn't all that useful. Predicting returns is.

Some notes. This can be viewed as exact first-order difference on the log price. It's nice for a few reasons.

– It's symmetrical, meaning that if you reverse the time series it just flips the signs.

– You can sum sequential r[t] to get the cumulative log returns over a period.

– Log returns are very similar numerically to returns, and are monotonic with them

– Any transformation you do to predict log returns maps back to a valid (positive) price, because the reverse transform involves the exponent. That's not true with log-returns.

To a first order approximation, for a really small time scale (like daily returns or hourly returns), the expected value of both returns and log returns is 0, which is also nice.

If this is your first rodeo, I'd do log-returns because they're easier to manipulate.