DATA 624 Spring 2019: Project-1

Ahmed Sajjad, Harpreet Shoker, Jagruti Solao, Chad Smith, Todd Weigel

March 19, 2019

#### Introduction

The innate desire to predict the future stems from the multitude of advantages it delivers. Depending on the quality and processing of the data, the goal can be as easy as predicting a full moon three years from today to as hard as predicting the weather a week from now. The different Machine Learning (ML) techniques that will be described may unveil great insight into how the data can be forecasted or at all. This project shows the process of finding and analyzing what patterns time-series data can hold and the algorithms that discover them.

The purpose of this project is to forecast a dataset 140 days in the future. Using the R programming language and many of its packages at our disposal, we methodically worked through understanding the data structure and the problems it held. The dataset we were using had six different groups of stocks. Each group had seven columns that each represented a different value of a stock price: Open Price, Volume, Low of the day, High of the Day and Close Price. There are 1,622 rows for each group that represent a time period from May of 2011 to October of 2017.   
  
To begin building a model to forecast the future prices, a solid foundation to work from will need to be created. Any data that is missing, unnormal or incorrectly formatted will have to be addressed before it can be used reliably. The process of gathering the data, cleaning it and correcting any errors is described below.

Values to be forecasted for the following series: S01 – Forecast Var01, Var02 S02 – Forecast Var02, Var03 S03 – Forecast Var05, Var07 S04 – Forecast Var01, Var02 S05 – Forecast Var02, Var03 S06 – Forecast Var05, Var07

Loading all the required libraries.

library(utils)  
library('readxl')  
library('xlsx')  
library(tidyr)  
library(dplyr)  
library(ggplot2)  
library(forecast)  
library(fma)  
library(fpp2)  
library(tseries)  
library(gridExtra)  
library(ggcorrplot)  
library(astsa)  
library(janitor)  
library(timeDate)  
library(scales)

#### Exploratory Data Analysis and Data Cleaning

Importing data from Excel, and dates are in a numeric format. We are using as.Date to import these, we simply need to set the origin date and for Excel on Windows, the origin date is December 30, 1899 for dates after 1900.

project\_in\_df <- data.frame(read\_excel("Project1data.xls", sheet = "Set for Class"))  
project\_in\_df = mutate(project\_in\_df, datetime=as.Date(SeriesInd, origin="1899-12-30"))  
project\_in\_df = project\_in\_df[c(1, 8, 2, 3, 4, 5, 6, 7)]  
  
head(project\_in\_df)

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1 40669 2011-05-06 S03 30.64286 123432400 30.34 30.49 30.57286  
## 2 40669 2011-05-06 S02 10.28000 60855800 10.05 10.17 10.28000  
## 3 40669 2011-05-06 S01 26.61000 10369300 25.89 26.20 26.01000  
## 4 40669 2011-05-06 S06 27.48000 39335700 26.82 27.02 27.32000  
## 5 40669 2011-05-06 S05 69.26000 27809100 68.19 68.72 69.15000  
## 6 40669 2011-05-06 S04 17.20000 16587400 16.88 16.94 17.10000

nrow(project\_in\_df)

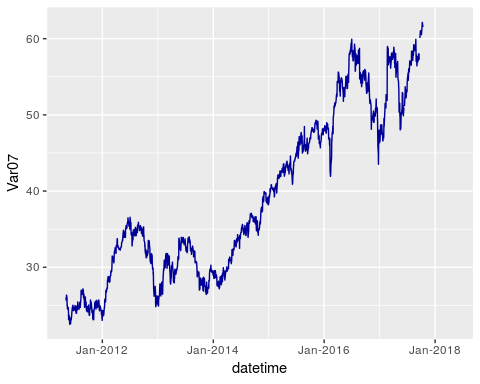
## [1] 10572

# Create separate dataframes for each group  
group\_S01\_df = filter(project\_in\_df, group == 'S01')  
group\_S02\_df = filter(project\_in\_df, group == 'S02')  
group\_S03\_df = filter(project\_in\_df, group == 'S03')  
group\_S04\_df = filter(project\_in\_df, group == 'S04')  
group\_S05\_df = filter(project\_in\_df, group == 'S05')  
group\_S06\_df = filter(project\_in\_df, group == 'S06')

Looking at the data we found that the data is financial stock market data from 6 different companies. The Variables represent the following # Var01 - High of the day # Var02 - Volume # Var03 - Low of the day # Var05 - Open for the day # Var07 - Close for the day

ggplot(group\_S01\_df, aes(datetime, Var07)) + geom\_line(colour="#000099") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



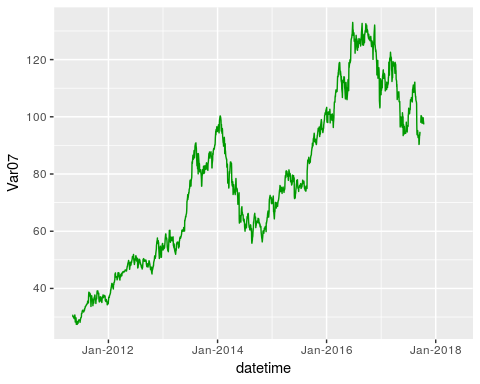
ggplot(group\_S02\_df, aes(datetime, Var07)) + geom\_line(colour="#990000") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



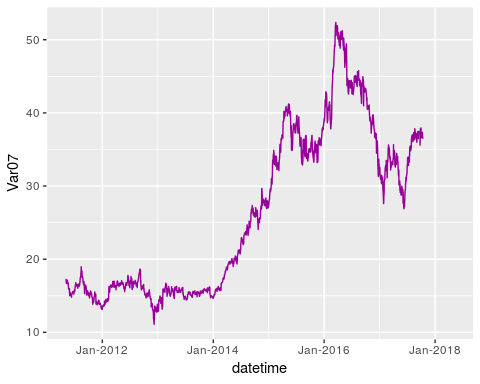
ggplot(group\_S03\_df, aes(datetime, Var07)) + geom\_line(colour="#009900") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



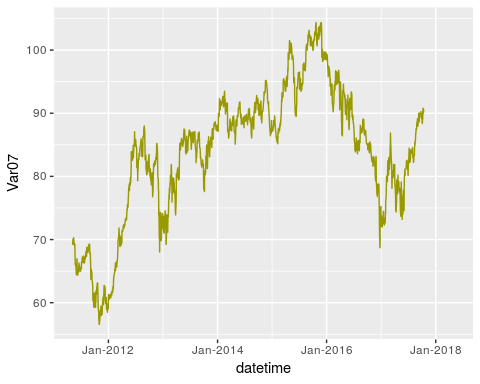
ggplot(group\_S04\_df, aes(datetime, Var07)) + geom\_line(colour="#990099") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



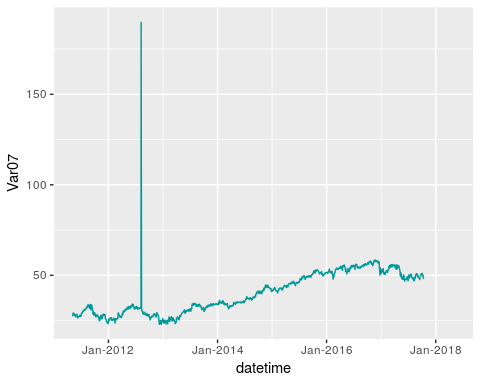
ggplot(group\_S05\_df, aes(datetime, Var07)) + geom\_line(colour="#999900") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



ggplot(group\_S06\_df, aes(datetime, Var07)) + geom\_line(colour="#009999") + scale\_x\_date(labels = date\_format("%b-%Y")) + xlab("datetime") + ylab("Var07")

## Warning: Removed 140 rows containing missing values (geom\_path).



#remove forecast cells for now  
#project\_in\_df <- project\_in\_df[1:9732,]  
group\_S01\_df <- slice(group\_S01\_df, 1:1622)   
group\_S02\_df <- slice(group\_S02\_df, 1:1622)   
group\_S03\_df <- slice(group\_S03\_df, 1:1622)   
group\_S04\_df <- slice(group\_S04\_df, 1:1622)   
group\_S05\_df <- slice(group\_S05\_df, 1:1622)   
group\_S06\_df <- slice(group\_S06\_df, 1:1622)  
  
  
#Change numeric sequence to actual dates for graphing  
#note, we put back to sequence before writing the files back out to excel  
group\_S01\_df[,1] <- excel\_numeric\_to\_date(group\_S01\_df[,1])  
group\_S02\_df[,1] <- excel\_numeric\_to\_date(group\_S02\_df[,1])  
group\_S03\_df[,1] <- excel\_numeric\_to\_date(group\_S03\_df[,1])  
group\_S04\_df[,1] <- excel\_numeric\_to\_date(group\_S04\_df[,1])  
group\_S05\_df[,1] <- excel\_numeric\_to\_date(group\_S05\_df[,1])  
group\_S06\_df[,1] <- excel\_numeric\_to\_date(group\_S06\_df[,1])  
  
#check for rows with NA's  
group\_S01\_df[rowSums(is.na(group\_S01\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1537 2017-06-11 2017-06-11 S01 NA 7329600 NA NA NA  
## 1538 2017-06-12 2017-06-12 S01 NA 6121400 NA NA NA  
## 1607 2017-09-19 2017-09-19 S01 58.83 6337000 NA NA NA  
## 1608 2017-09-22 2017-09-22 S01 59.28 3690900 NA NA NA

group\_S02\_df[rowSums(is.na(group\_S02\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1537 2017-06-11 2017-06-11 S02 NA 38160300 NA NA NA  
## 1538 2017-06-12 2017-06-12 S02 NA 45801300 NA NA NA  
## 1607 2017-09-19 2017-09-19 S02 13.26 19465000 NA NA NA  
## 1608 2017-09-22 2017-09-22 S02 13.20 16234300 NA NA NA

group\_S03\_df[rowSums(is.na(group\_S03\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1537 2017-06-11 2017-06-11 S03 NA 42343600 NA NA NA  
## 1538 2017-06-12 2017-06-12 S03 NA 50074700 NA NA NA  
## 1607 2017-09-19 2017-09-19 S03 95.43 32026000 NA NA NA  
## 1608 2017-09-22 2017-09-22 S03 97.19 38018600 NA NA NA

group\_S04\_df[rowSums(is.na(group\_S04\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1537 2017-06-11 2017-06-11 S04 NA 9098800 NA NA NA  
## 1538 2017-06-12 2017-06-12 S04 NA 11188200 NA NA NA  
## 1607 2017-09-19 2017-09-19 S04 36.72 34330700 NA NA NA  
## 1608 2017-09-22 2017-09-22 S04 36.95 7785800 NA NA NA

group\_S05\_df[rowSums(is.na(group\_S05\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 795 2014-07-01 2014-07-01 S05 NA NA NA NA NA  
## 1537 2017-06-11 2017-06-11 S05 NA 16610900 NA NA NA  
## 1538 2017-06-12 2017-06-12 S05 NA 19331600 NA NA NA  
## 1607 2017-09-19 2017-09-19 S05 90.4 13191900 NA NA NA  
## 1608 2017-09-22 2017-09-22 S05 89.9 11766100 NA NA NA

group\_S06\_df[rowSums(is.na(group\_S06\_df))>0,]

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 20 2011-06-03 2011-06-03 S06 NA NA NA NA NA  
## 1537 2017-06-11 2017-06-11 S06 NA 19885500 NA NA NA  
## 1538 2017-06-12 2017-06-12 S06 NA 32570900 NA NA NA  
## 1607 2017-09-19 2017-09-19 S06 49.21 13222800 NA NA NA  
## 1608 2017-09-22 2017-09-22 S06 48.88 10644000 NA NA NA

From the above results can see missing values in all the variables,we need to impute NA’s with some value. Var01 has 14,Var02 has 2,Var03 has 26,Var05 has 26 and Var07 has 26 missing values.

Here we will be imputing these values with the next value in sequence, e.g., if row 1600 is NA for Var01, take value from row 1601 hopefully, since these are stock values, the value from the next day for the few missing values we have, should be close enough

removeNAs <- function(dfTs)  
{  
  
 while(nrow(dfTs[rowSums(is.na(dfTs))>0,]) > 0)  
 {   
   
 dfTs <- transmute(dfTs,   
 SeriesInd = SeriesInd,  
 Var01 = if\_else(is.na(Var01), lead(Var01), Var01),  
 Var02 = if\_else(is.na(Var02), lead(Var02), Var02),  
 Var03 = if\_else(is.na(Var03), lead(Var03), Var03),  
 Var05 = if\_else(is.na(Var05), lead(Var05), Var05),  
 Var07 = if\_else(is.na(Var07), lead(Var07), Var07))  
 }  
 print(dfTs[rowSums(is.na(dfTs))>0,])  
 return(dfTs)  
}  
  
group\_S01\_df <- removeNAs(group\_S01\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

group\_S02\_df <- removeNAs(group\_S02\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

group\_S03\_df <- removeNAs(group\_S03\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

group\_S04\_df <- removeNAs(group\_S04\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

group\_S05\_df <- removeNAs(group\_S05\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

group\_S06\_df <- removeNAs(group\_S06\_df)

## [1] SeriesInd Var01 Var02 Var03 Var05 Var07   
## <0 rows> (or 0-length row.names)

summary(group\_S01\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. :23.01 Min. : 1339900 Min. :22.28 Min. :22.55 Min. :22.50   
## 1st Qu.:2012-12-10 1st Qu.:29.85 1st Qu.: 5347550 1st Qu.:29.34 1st Qu.:29.60 1st Qu.:29.61   
## Median :2014-07-25 Median :35.66 Median : 7895050 Median :35.10 Median :35.36 Median :35.41   
## Mean :2014-07-23 Mean :39.43 Mean : 8907092 Mean :38.69 Mean :39.05 Mean :39.09   
## 3rd Qu.:2016-02-29 3rd Qu.:48.74 3rd Qu.:11321675 3rd Qu.:47.91 3rd Qu.:48.31 3rd Qu.:48.30   
## Max. :2017-10-13 Max. :62.31 Max. :48477500 Max. :61.59 Max. :62.29 Max. :62.14

summary(group\_S02\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. : 9.03 Min. : 7128800 Min. : 8.82 Min. : 8.99 Min. : 8.92   
## 1st Qu.:2012-12-10 1st Qu.:12.18 1st Qu.: 27880300 1st Qu.:11.81 1st Qu.:12.01 1st Qu.:12.00   
## Median :2014-07-25 Median :14.07 Median : 39767500 Median :13.76 Median :13.91 Median :13.91   
## Mean :2014-07-23 Mean :13.99 Mean : 50633098 Mean :13.68 Mean :13.85 Mean :13.84   
## 3rd Qu.:2016-02-29 3rd Qu.:15.84 3rd Qu.: 59050900 3rd Qu.:15.52 3rd Qu.:15.72 3rd Qu.:15.67   
## Max. :2017-10-13 Max. :39.36 Max. :480879500 Max. :38.28 Max. :39.33 Max. :38.40

summary(group\_S03\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. : 28.00 Min. : 13046400 Min. : 27.18 Min. : 27.48 Min. : 27.44   
## 1st Qu.:2012-12-10 1st Qu.: 54.12 1st Qu.: 55186050 1st Qu.: 52.89 1st Qu.: 53.34 1st Qu.: 53.53   
## Median :2014-07-25 Median : 76.24 Median : 85595400 Median : 74.92 Median : 75.66 Median : 75.76   
## Mean :2014-07-23 Mean : 77.65 Mean : 99387244 Mean : 76.15 Mean : 76.95 Mean : 76.91   
## 3rd Qu.:2016-02-29 3rd Qu.: 99.52 3rd Qu.:127036525 3rd Qu.: 97.57 3rd Qu.: 98.53 3rd Qu.: 98.51   
## Max. :2017-10-13 Max. :134.54 Max. :470249500 Max. :131.40 Max. :134.46 Max. :133.00

summary(group\_S04\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. :11.80 Min. : 3468900 Min. :11.09 Min. :11.30 Min. :11.09   
## 1st Qu.:2012-12-10 1st Qu.:15.99 1st Qu.: 12918725 1st Qu.:15.66 1st Qu.:15.83 1st Qu.:15.80   
## Median :2014-07-25 Median :23.45 Median : 17000800 Median :22.77 Median :23.15 Median :23.28   
## Mean :2014-07-23 Mean :26.46 Mean : 20757818 Mean :25.83 Mean :26.15 Mean :26.14   
## 3rd Qu.:2016-02-29 3rd Qu.:36.42 3rd Qu.: 24015700 3rd Qu.:35.59 3rd Qu.:35.98 3rd Qu.:35.93   
## Max. :2017-10-13 Max. :52.62 Max. :233872100 Max. :51.64 Max. :52.28 Max. :52.37

summary(group\_S05\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. : 56.99 Min. : 4156600 Min. : 55.94 Min. : 56.85 Min. : 56.57   
## 1st Qu.:2012-12-10 1st Qu.: 78.86 1st Qu.: 11199775 1st Qu.: 77.25 1st Qu.: 77.95 1st Qu.: 78.11   
## Median :2014-07-25 Median : 85.95 Median : 14575700 Median : 84.88 Median : 85.33 Median : 85.44   
## Mean :2014-07-23 Mean : 84.21 Mean : 16787776 Mean : 82.97 Mean : 83.59 Mean : 83.63   
## 3rd Qu.:2016-02-29 3rd Qu.: 90.99 3rd Qu.: 20014325 3rd Qu.: 89.71 3rd Qu.: 90.37 3rd Qu.: 90.41   
## Max. :2017-10-13 Max. :104.76 Max. :118023500 Max. :103.95 Max. :104.42 Max. :104.38

summary(group\_S06\_df)

## SeriesInd Var01 Var02 Var03 Var05 Var07   
## Min. :2011-05-06 Min. : 23.41 Min. : 4297000 Min. : 22.58 Min. : 22.91 Min. : 22.88   
## 1st Qu.:2012-12-10 1st Qu.: 30.56 1st Qu.: 15427725 1st Qu.: 29.99 1st Qu.: 30.32 1st Qu.: 30.26   
## Median :2014-07-25 Median : 37.14 Median : 21795200 Median : 36.67 Median : 36.94 Median : 36.98   
## Mean :2014-07-23 Mean : 40.21 Mean : 25733544 Mean : 39.51 Mean : 39.86 Mean : 39.87   
## 3rd Qu.:2016-02-29 3rd Qu.: 50.74 3rd Qu.: 31506875 3rd Qu.: 50.06 3rd Qu.: 50.45 3rd Qu.: 50.43   
## Max. :2017-10-13 Max. :195.18 Max. :144985700 Max. :189.36 Max. :195.00 Max. :189.72

From the above data summaries we can see all the null values are removed with apropiate data

# Select relevant columns for each group  
group\_S01\_df = select (group\_S01\_df, matches("SeriesInd|datetime|group|Var01|Var02"))  
group\_S02\_df = select (group\_S02\_df, matches("SeriesInd|datetime|group|Var02|Var03"))  
group\_S03\_df = select (group\_S03\_df, matches("SeriesInd|datetime|group|Var05|Var07"))  
group\_S04\_df = select (group\_S04\_df, matches("SeriesInd|datetime|group|Var01|Var02"))  
group\_S05\_df = select (group\_S05\_df, matches("SeriesInd|datetime|group|Var02|Var03"))  
group\_S06\_df = select (group\_S06\_df, matches("SeriesInd|datetime|group|Var05|Var07"))  
  
# Check number of rows  
print (c(nrow(group\_S01\_df), nrow(group\_S02\_df), nrow(group\_S03\_df), nrow(group\_S04\_df), nrow(group\_S05\_df), nrow(group\_S06\_df)))

## [1] 1622 1622 1622 1622 1622 1622

# Verify dataframes  
head(project\_in\_df, 10)

## SeriesInd datetime group Var01 Var02 Var03 Var05 Var07  
## 1 40669 2011-05-06 S03 30.64286 123432400 30.34000 30.49000 30.57286  
## 2 40669 2011-05-06 S02 10.28000 60855800 10.05000 10.17000 10.28000  
## 3 40669 2011-05-06 S01 26.61000 10369300 25.89000 26.20000 26.01000  
## 4 40669 2011-05-06 S06 27.48000 39335700 26.82000 27.02000 27.32000  
## 5 40669 2011-05-06 S05 69.26000 27809100 68.19000 68.72000 69.15000  
## 6 40669 2011-05-06 S04 17.20000 16587400 16.88000 16.94000 17.10000  
## 7 40670 2011-05-07 S03 30.79857 150476200 30.46428 30.65714 30.62571  
## 8 40670 2011-05-07 S02 11.24000 215620200 10.40000 10.45000 10.96000  
## 9 40670 2011-05-07 S01 26.30000 10943800 25.70000 25.95000 25.86000  
## 10 40670 2011-05-07 S06 28.24000 55416000 27.24000 27.27000 28.07000

head(group\_S01\_df, 10)

## SeriesInd Var01 Var02  
## 1 2011-05-06 26.61 10369300  
## 2 2011-05-07 26.30 10943800  
## 3 2011-05-08 26.03 8933800  
## 4 2011-05-09 25.84 10775400  
## 5 2011-05-10 26.34 12875600  
## 6 2011-05-13 26.49 11677000  
## 7 2011-05-14 26.03 21165300  
## 8 2011-05-15 25.16 18809200  
## 9 2011-05-16 25.00 22908400  
## 10 2011-05-17 24.77 20359100

head(group\_S02\_df, 10)

## SeriesInd Var02 Var03  
## 1 2011-05-06 60855800 10.05  
## 2 2011-05-07 215620200 10.40  
## 3 2011-05-08 200070600 11.13  
## 4 2011-05-09 130201700 11.32  
## 5 2011-05-10 130463000 11.46  
## 6 2011-05-13 170626200 11.78  
## 7 2011-05-14 162995900 11.72  
## 8 2011-05-15 154527100 11.47  
## 9 2011-05-16 116531200 11.51  
## 10 2011-05-17 96149800 11.55

head(group\_S03\_df, 10)

## SeriesInd Var05 Var07  
## 1 2011-05-06 30.49000 30.57286  
## 2 2011-05-07 30.65714 30.62571  
## 3 2011-05-08 30.62571 30.13857  
## 4 2011-05-09 30.25000 30.08286  
## 5 2011-05-10 30.04286 30.28286  
## 6 2011-05-13 30.40000 30.01571  
## 7 2011-05-14 29.88428 29.67429  
## 8 2011-05-15 29.69571 30.09286  
## 9 2011-05-16 30.01571 29.91857  
## 10 2011-05-17 30.13286 29.41857

head(group\_S04\_df, 10)

## SeriesInd Var01 Var02  
## 1 2011-05-06 17.20 16587400  
## 2 2011-05-07 17.23 11718100  
## 3 2011-05-08 17.30 16422000  
## 4 2011-05-09 16.90 31816300  
## 5 2011-05-10 16.76 15470000  
## 6 2011-05-13 16.83 16181900  
## 7 2011-05-14 16.86 15672400  
## 8 2011-05-15 16.98 16955600  
## 9 2011-05-16 17.23 16715600  
## 10 2011-05-17 17.25 18415000

head(group\_S05\_df, 10)

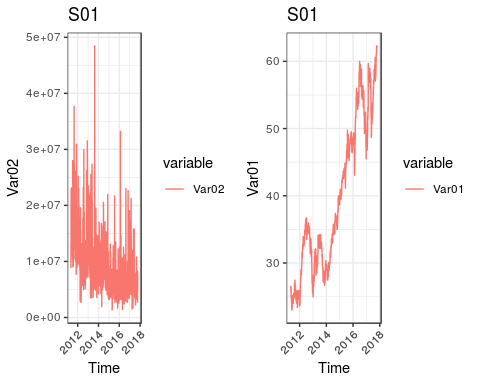
## SeriesInd Var02 Var03  
## 1 2011-05-06 27809100 68.19  
## 2 2011-05-07 30174700 68.80  
## 3 2011-05-08 35044700 69.34  
## 4 2011-05-09 27192100 69.42  
## 5 2011-05-10 24891800 69.22  
## 6 2011-05-13 30685000 69.65  
## 7 2011-05-14 31496700 69.52  
## 8 2011-05-15 24884400 69.26  
## 9 2011-05-16 18630800 69.35  
## 10 2011-05-17 29411900 68.65

head(group\_S06\_df, 10)

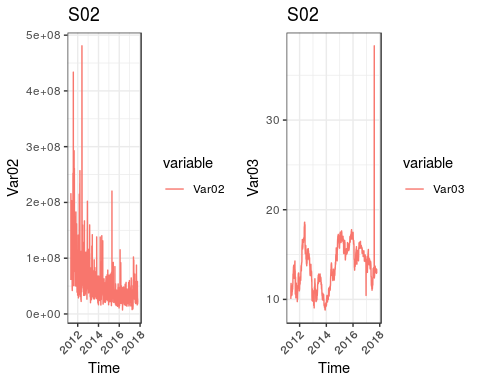
## SeriesInd Var05 Var07  
## 1 2011-05-06 27.02 27.32  
## 2 2011-05-07 27.27 28.07  
## 3 2011-05-08 28.03 28.11  
## 4 2011-05-09 28.12 29.13  
## 5 2011-05-10 28.90 28.86  
## 6 2011-05-13 29.09 28.80  
## 7 2011-05-14 28.47 28.08  
## 8 2011-05-15 27.99 28.58  
## 9 2011-05-16 28.50 28.99  
## 10 2011-05-17 28.82 28.08

Lets start looking at the data. First we are making line plots for all the groups and variables we will be forecasting.

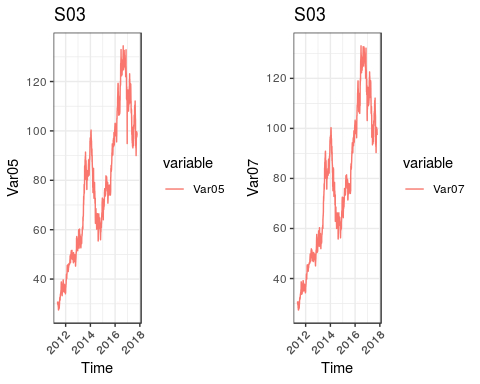
SeriesInd1 <- as.Date(group\_S01\_df$SeriesInd,origin = "1899-12-30")  
SeriesInd2 <- as.Date(group\_S02\_df$SeriesInd,origin = "1899-12-30")  
SeriesInd3 <- as.Date(group\_S03\_df$SeriesInd,origin = "1899-12-30")  
SeriesInd4 <- as.Date(group\_S04\_df$SeriesInd,origin = "1899-12-30")  
SeriesInd5 <- as.Date(group\_S05\_df$SeriesInd,origin = "1899-12-30")  
SeriesInd6 <- as.Date(group\_S06\_df$SeriesInd,origin = "1899-12-30")  
p1 <- ggplot(group\_S01\_df, aes(SeriesInd1, y = Var02, color = variable)) + ggtitle("S01") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p2 <- ggplot(group\_S01\_df, aes(SeriesInd1, y = Var01, color = variable)) + ggtitle("S01") + geom\_line(aes(y = Var01, col = "Var01")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p3 <- ggplot(group\_S02\_df, aes(SeriesInd2, y = Var02,color = variable))+ ggtitle("S02") + geom\_line(aes(y = Var02, col = "Var02")) +xlab("Time")+ theme\_bw() + theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
  
p4 <- ggplot(group\_S02\_df, aes(SeriesInd2, y = Var03,color = variable))+ ggtitle("S02") + geom\_line(aes(y = Var03, col = "Var03")) +xlab("Time")+ theme\_bw() + theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p5 <- ggplot(group\_S03\_df, aes(SeriesInd3, y = Var05, color = variable)) + geom\_line(aes(y = Var05, col = "Var05")) + ggtitle("S03")+ xlab("Time") + theme\_bw() + theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p6 <- ggplot(group\_S03\_df, aes(SeriesInd3, y = Var07, color = variable)) + geom\_line(aes(y = Var07, col = "Var07")) + ggtitle("S03") + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
   
p7 <- ggplot(group\_S04\_df, aes(SeriesInd4, y = Var01, color = variable)) + geom\_line(aes(y = Var01, col = "Var01")) + ggtitle("S04") +xlab("Time")+ theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))   
p8 <- ggplot(group\_S04\_df, aes(SeriesInd4, y = Var02,color = variable))+ ggtitle("S04") + geom\_line(aes(y = Var02, col = "Var02")) +xlab("Time")+ theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
  
p9 <- ggplot(group\_S05\_df, aes(SeriesInd5, y = Var03, color = variable)) + ggtitle("S05") +geom\_line(aes(y = Var03, col = "Var03")) + xlab("Time")+theme\_bw()+  
theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p10 <- ggplot(group\_S05\_df, aes(SeriesInd5, y = Var02,color = variable))+ ggtitle("S05") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p11 <- ggplot(group\_S06\_df, aes(SeriesInd6, y = Var05, color = variable)) + geom\_line(aes(y = Var05, col = "Var05")) +  
ggtitle("S06") +xlab("Time")+ theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p12 <- ggplot(group\_S06\_df, aes(SeriesInd6, y = Var07, color = variable)) + ggtitle("S06") + geom\_line(aes(y = Var07, col = "Var07")) + xlab("Time")+theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
grid.arrange(p1, p2, nrow = 1)



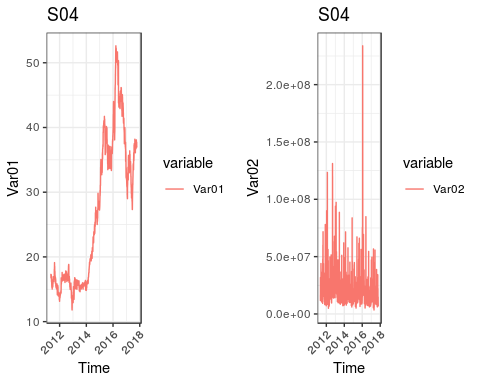
grid.arrange(p3, p4, nrow = 1)



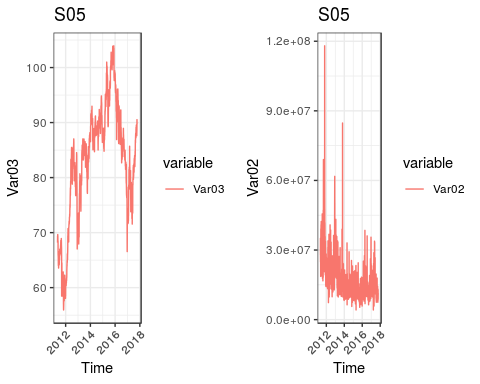
grid.arrange(p5, p6, nrow = 1)



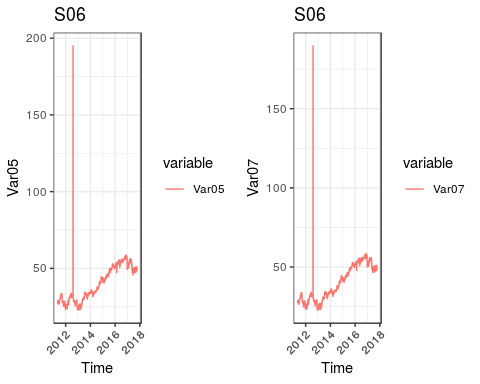
grid.arrange(p7,p8, nrow = 1)



grid.arrange(p9,p10, nrow = 1)



grid.arrange(p11,p12, nrow = 1)



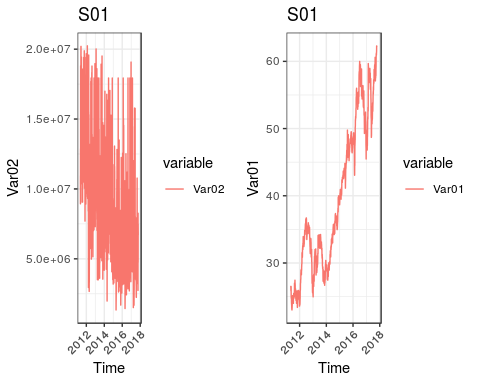
Comparing to other variable Var02 seems to be noisy than any other variable and having outliers. These outliers needs to be fixed before producing forecasts S03 and S06 variables Var05 and Var07 seems to be quite similar Also we see in S02 - Var03 and S06-Var05 and Var07 plot some outlier values that also needs to be fixed before forecasting. We can also observe some seasonality and trend pattern in Var01,Var03,Var05 and var07

Removing outliers We are using here IQR to fix the outliers in our data For missing values that lie outside the 1.5\*IQR limits, we are capping it by replacing those observations outside the lower limit with the value of 5th %ile and those that lie above the upper limit, with the value of 95th %ile.

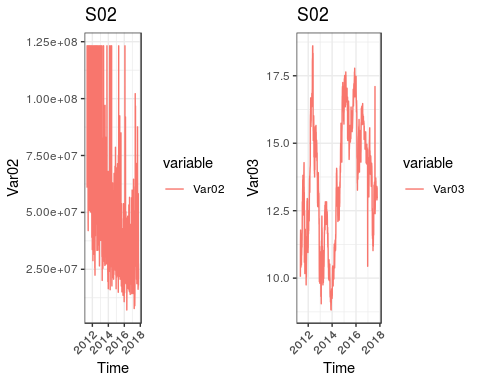
Remove\_Outlier <- function(x){  
 repeat{  
 qnt <- quantile(x, probs=c(.25, .75), na.rm = T)  
 caps <- quantile(x, probs=c(.05, .95), na.rm = T)  
 H <- 1.5 \* IQR(x, na.rm = T)  
 x[x < (qnt[1] - H)] <- caps[1]  
 x[x > (qnt[2] + H)] <- caps[2]  
   
 return(x)  
 if(x < (qnt[1] - H)){break}  
 if(x > (qnt[1] + H)){break}  
 }  
   
}  
group\_S01\_df$Var01=Remove\_Outlier(group\_S01\_df$Var01)  
group\_S01\_df$Var02=Remove\_Outlier(group\_S01\_df$Var02)  
group\_S02\_df$Var03=Remove\_Outlier(group\_S02\_df$Var03)  
group\_S02\_df$Var02=Remove\_Outlier(group\_S02\_df$Var02)  
group\_S04\_df$Var01=Remove\_Outlier(group\_S04\_df$Var01)  
group\_S04\_df$Var02=Remove\_Outlier(group\_S04\_df$Var02)  
group\_S05\_df$Var03=Remove\_Outlier(group\_S05\_df$Var03)  
group\_S05\_df$Var02=Remove\_Outlier(group\_S05\_df$Var02)  
group\_S06\_df$Var05=Remove\_Outlier(group\_S06\_df$Var05)  
group\_S06\_df$Var07=Remove\_Outlier(group\_S06\_df$Var07)

Lets plot again plots for the variables to check if the outliers are fixed

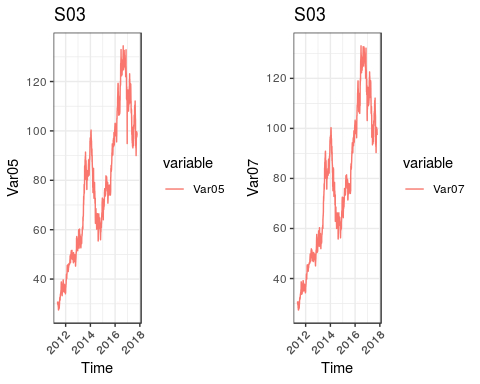
p1 <- ggplot(group\_S01\_df, aes(SeriesInd1, y = Var02, color = variable)) + ggtitle("S01") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw() + theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p2 <- ggplot(group\_S01\_df, aes(SeriesInd1, y = Var01, color = variable)) + ggtitle("S01") + geom\_line(aes(y = Var01, col = "Var01")) +xlab("Time")+ theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p3 <- ggplot(group\_S02\_df, aes(SeriesInd2, y = Var02,color = variable))+ ggtitle("S02") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw()+  
theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
  
p4 <- ggplot(group\_S02\_df, aes(SeriesInd2, y = Var03,color = variable))+ ggtitle("S02") + geom\_line(aes(y = Var03, col = "Var03")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p5 <- ggplot(group\_S03\_df, aes(SeriesInd3, y = Var05, color = variable)) + geom\_line(aes(y = Var05, col = "Var05")) +  
ggtitle("S03") + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p6 <- ggplot(group\_S03\_df, aes(SeriesInd3, y = Var07, color = variable)) + geom\_line(aes(y = Var07, col = "Var07")) +  
ggtitle("S03") + xlab("Time")+theme\_bw()+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
   
p7 <- ggplot(group\_S04\_df, aes(SeriesInd4, y = Var01, color = variable)) + geom\_line(aes(y = Var01, col = "Var01")) +   
 ggtitle("S04") + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
p8 <- ggplot(group\_S04\_df, aes(SeriesInd4, y = Var02,color = variable))+ ggtitle("S04") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
  
p9 <- ggplot(group\_S05\_df, aes(SeriesInd5, y = Var03, color = variable)) + ggtitle("S05") + geom\_line(aes(y = Var03, col = "Var03")) + xlab("Time")+theme\_bw()+  
theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p10 <- ggplot(group\_S05\_df, aes(SeriesInd5, y = Var02,color = variable))+ ggtitle("S05") + geom\_line(aes(y = Var02, col = "Var02")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p11 <- ggplot(group\_S06\_df, aes(SeriesInd6, y = Var05, color = variable)) + geom\_line(aes(y = Var05, col = "Var05")) +  
ggtitle("S06") + xlab("Time")+theme\_bw()+  
theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
p12 <- ggplot(group\_S06\_df, aes(SeriesInd6, y = Var07, color = variable)) + ggtitle("S06") + geom\_line(aes(y = Var07, col = "Var07")) + xlab("Time")+theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
grid.arrange(p1, p2, nrow = 1)



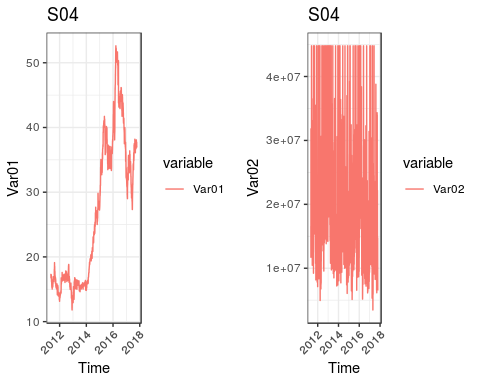
grid.arrange(p3, p4, nrow = 1)



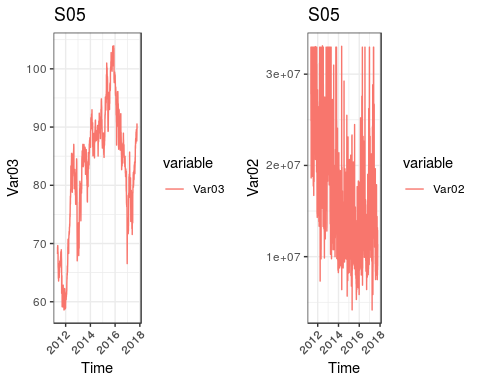
grid.arrange(p5, p6, nrow = 1)



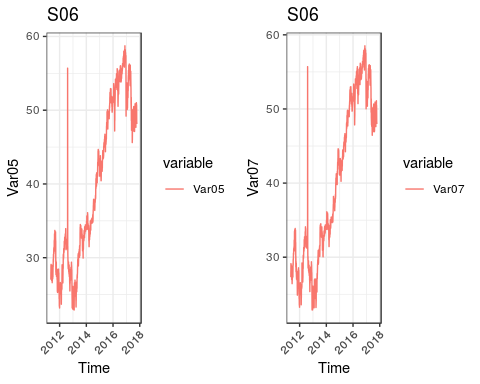
grid.arrange(p7,p8, nrow = 1)



grid.arrange(p9,p10, nrow = 1)



grid.arrange(p11,p12, nrow = 1)



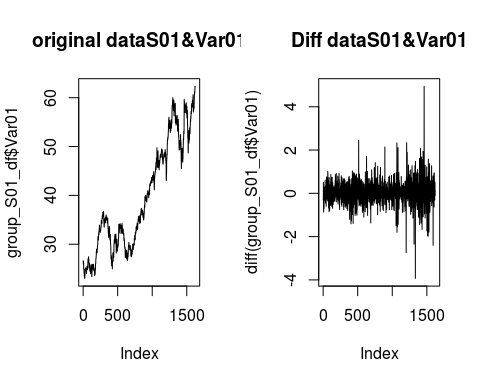
The plots looks it has removed the extreme outliers.

The plots for Var01,Var03,Var05,Var07 shows increase in general during the period of time. But there is no obvious pattern in the fluctuation. In other words, there might be seasonality, but an obvious upward trend. Also, the variance is not stable seeing from the plots and it seems to increase. Thus, we may use difference and logarithm or square root transformation on original data to stabilize the variance.

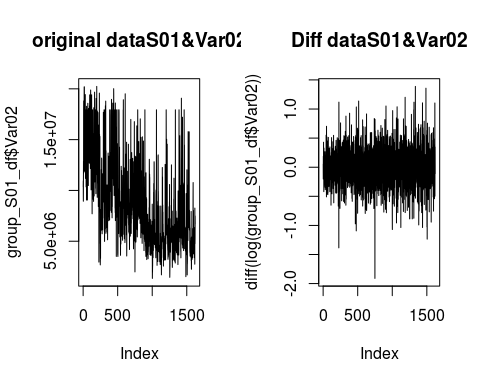
We are using here difference of each value over previous value and difference logarithm transformation for Var02

Lets create plots to have compare how the orignal data and transformed data

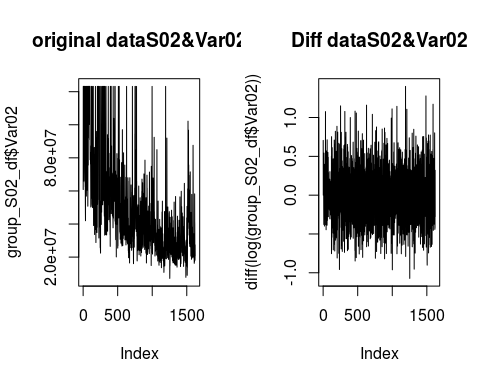
par(mfrow=c(1,2))  
p1 <- plot(group\_S01\_df$Var01, type = "l", main = "original dataS01&Var01")  
p2 <- plot(diff(group\_S01\_df$Var01), type = "l", main = "Diff dataS01&Var01")



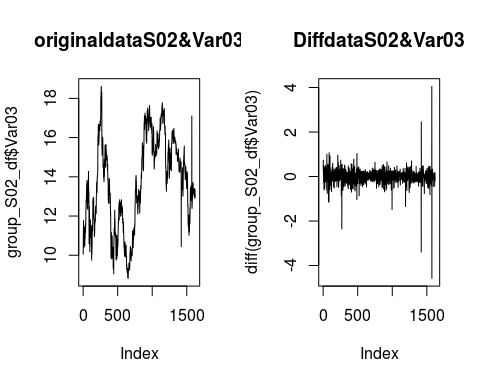
p3 <- plot(group\_S01\_df$Var02, type = "l", main = "original dataS01&Var02")  
p4 <- plot(diff(log(group\_S01\_df$Var02)), type = "l", main = "Diff dataS01&Var02")



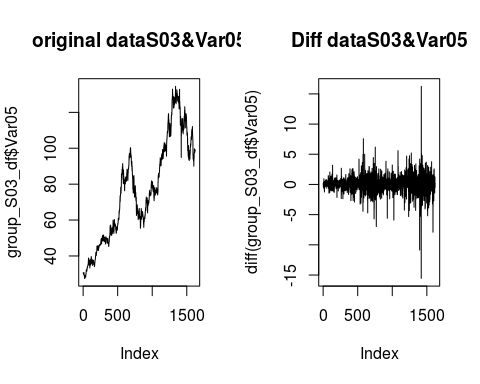
p5 <- plot(group\_S02\_df$Var02, type = "l", main = "original dataS02&Var02")  
p6 <- plot(diff(log(group\_S02\_df$Var02)), type = "l", main = "Diff dataS02&Var02")



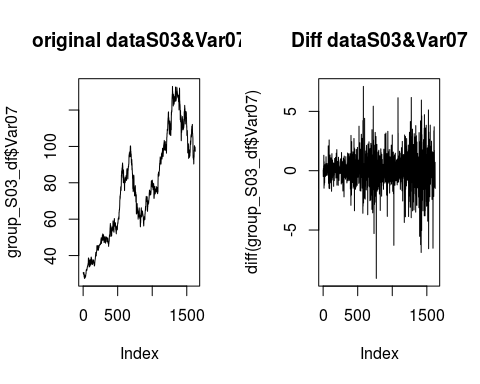
p7 <- plot(group\_S02\_df$Var03, type = "l", main = "originaldataS02&Var03")  
p8 <- plot(diff(group\_S02\_df$Var03), type = "l", main = "DiffdataS02&Var03")



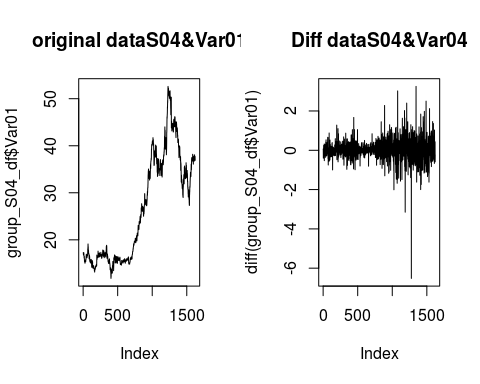
p9 <- plot(group\_S03\_df$Var05, type = "l", main = "original dataS03&Var05")  
p10 <- plot(diff(group\_S03\_df$Var05), type = "l", main = "Diff dataS03&Var05")



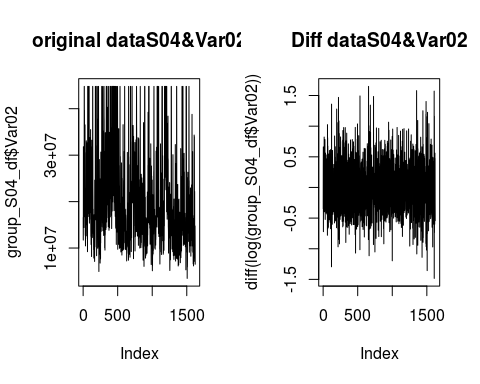
p11 <- plot(group\_S03\_df$Var07, type = "l", main = "original dataS03&Var07")  
p12 <- plot(diff(group\_S03\_df$Var07), type = "l", main = "Diff dataS03&Var07")



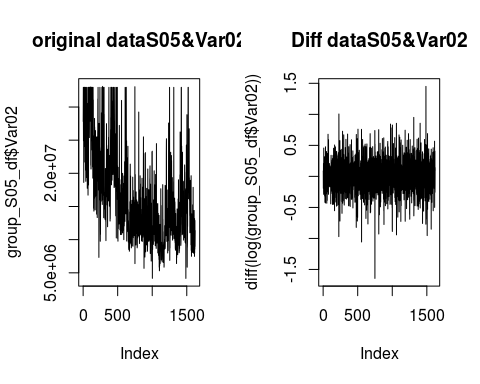
p13 <- plot(group\_S04\_df$Var01, type = "l", main = "original dataS04&Var01")  
p14 <- plot(diff(group\_S04\_df$Var01), type = "l", main = "Diff dataS04&Var04")



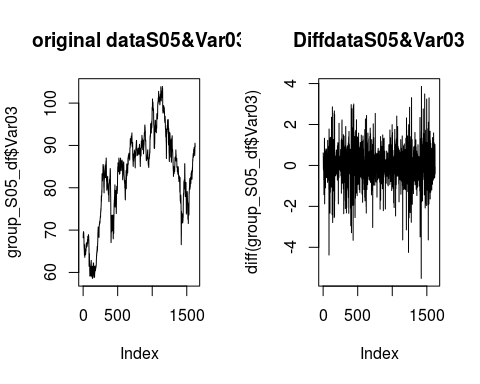
p15 <- plot(group\_S04\_df$Var02, type = "l", main = "original dataS04&Var02")  
p16 <- plot(diff(log(group\_S04\_df$Var02)), type = "l", main = "Diff dataS04&Var02")



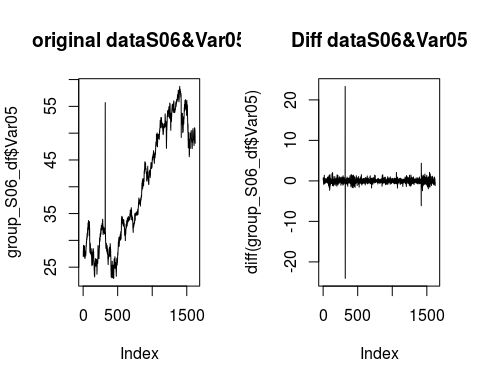
p17 <- plot(group\_S05\_df$Var02, type = "l", main = "original dataS05&Var02")  
p18 <- plot(diff(log(group\_S05\_df$Var02)), type = "l", main = "Diff dataS05&Var02")



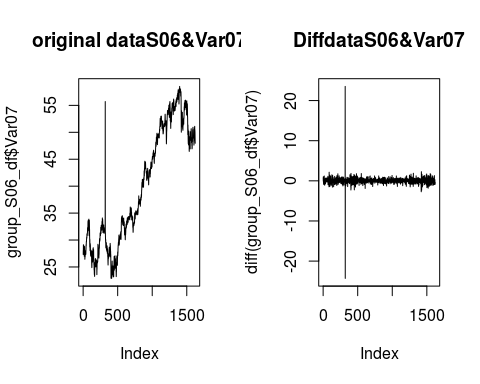
p19 <- plot(group\_S05\_df$Var03, type = "l", main = "original dataS05&Var03")  
p20 <- plot(diff(group\_S05\_df$Var03), type = "l", main = "DiffdataS05&Var03")



p21 <- plot(group\_S06\_df$Var05, type = "l", main = "original dataS06&Var05")  
p22 <- plot(diff(group\_S06\_df$Var05), type = "l", main = "Diff dataS06&Var05")



p23 <- plot(group\_S06\_df$Var07, type = "l", main = "original dataS06&Var07")  
p24 <- plot(diff(group\_S06\_df$Var07), type = "l", main = "DiffdataS06&Var07")



Data looks like it eliminated noise compared to the orginal. Data looks more stationary after applying differencing to Var01,Var02,Var03,Var05,Var07.

### ARIMA

ARIMA (autoregressive integrated moving average) is a commonly used technique utilized to fit time series data and forecasting. It is a generalized version of ARMA (autoregressive moving average) process, where the ARMA process is applied for a differenced version of the data rather than original.

Three numbers p, d and q specify ARIMA model and the ARIMA model is said to be of order (p,d,q). Here p, d and q are the orders of AR part, Difference and the MA part respectively.

AR and MA- both are different techniques to fit stationary time series data. ARMA (and ARIMA) is a combination of these two methods for better fit of the model.

#### Group S05 Forecast

##### (Note we start analysis with S05, and not S01, S01 and the other stocks will be examined later.)

First let’s get the dataframe data into a timeseries for variable 02. The data is stored as a series of dates, 5 days with a break after, so essentially 52 weeks times 5 days will divide it up nicely into 7 years of data. We will then fit the data using auto.arima to find the appropriate values for P,D,Q (p = the number of lag observations, d = degree of differencing, and q = size of the moving average window, for lagged forecast errors ).

We see that a 1,1,2 model appears to be most appropriate.

h <- 140  
TsVar02 <- ts(group\_S05\_df$Var02, frequency = 5\*52)  
#TsVar02 <- ts(group\_S01\_df[,1:2])  
TsVar02 <- ts(group\_S05\_df$Var02, start = c(2011, 85), end = c(2017, 166), frequency = 260)  
  
arimaFitVar02 <- auto.arima(TsVar02, seasonal = FALSE)  
arimaFitVar02

## Series: TsVar02   
## ARIMA(1,1,2)   
##   
## Coefficients:  
## ar1 ma1 ma2  
## 0.7600 -1.3275 0.3646  
## s.e. 0.0619 0.0725 0.0587  
##   
## sigma^2 estimated as 1.545e+13: log likelihood=-27244.99  
## AIC=54497.99 AICc=54498.01 BIC=54519.6

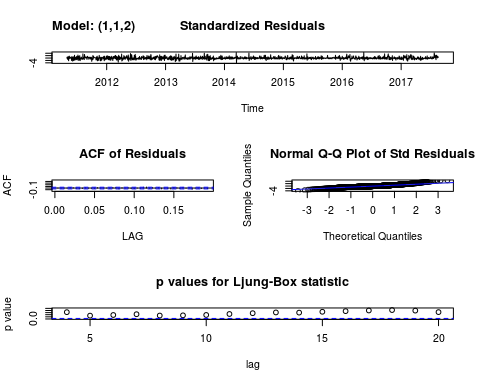
We will use the package Applied Statistical Time Series Analysis (astsa) and it’s wrapper module, sarima around the arima set of tools to do our forecasting.

Looking at the residuals from the package output, we see that they look like white noise, which is what we hope, as we don’t wish to see a pattern. The ACF of the residuals look acceptable too. There are a couple of small spikes just above the threshold of significance, but otherwise fine.

Note the box test results show that there is no correlation as all value are outside the threshold.

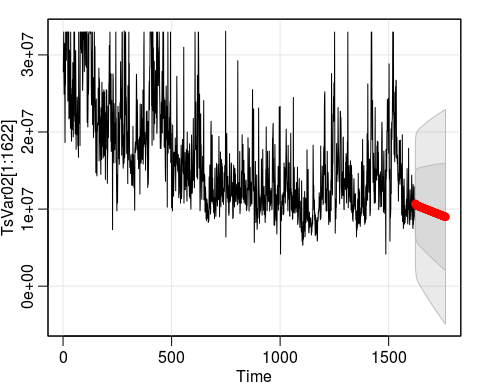
sarima(TsVar02, 1, 1, 2)

## initial value 15.326257   
## iter 2 value 15.244408  
## iter 3 value 15.199934  
## iter 4 value 15.197369  
## iter 5 value 15.197323  
## iter 6 value 15.197246  
## iter 7 value 15.196955  
## iter 8 value 15.196021  
## iter 9 value 15.194363  
## iter 10 value 15.192923  
## iter 11 value 15.192371  
## iter 12 value 15.191646  
## iter 13 value 15.191477  
## iter 14 value 15.191121  
## iter 15 value 15.190463  
## iter 16 value 15.189146  
## iter 17 value 15.187523  
## iter 18 value 15.186481  
## iter 19 value 15.186372  
## iter 20 value 15.186201  
## iter 21 value 15.186107  
## iter 22 value 15.185764  
## iter 23 value 15.185437  
## iter 24 value 15.185324  
## iter 25 value 15.185287  
## iter 26 value 15.185280  
## iter 27 value 15.185279  
## iter 27 value 15.185279  
## final value 15.185279   
## converged  
## initial value 15.184058   
## iter 2 value 15.184037  
## iter 3 value 15.184028  
## iter 4 value 15.184001  
## iter 5 value 15.183934  
## iter 6 value 15.183843  
## iter 7 value 15.183756  
## iter 8 value 15.183735  
## iter 9 value 15.183735  
## iter 9 value 15.183735  
## iter 9 value 15.183735  
## final value 15.183735   
## converged



## $fit  
##   
## Call:  
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,   
## Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,   
## reltol = tol))  
##   
## Coefficients:  
## ar1 ma1 ma2 constant  
## 0.7586 -1.3264 0.3638 -1470.274  
## s.e. 0.0619 0.0724 0.0587 15191.714  
##   
## sigma^2 estimated as 1.542e+13: log likelihood = -27244.99, aic = 54499.97  
##   
## $degrees\_of\_freedom  
## [1] 1637  
##   
## $ttable  
## Estimate SE t.value p.value  
## ar1 0.7586 0.0619 12.2483 0.0000  
## ma1 -1.3264 0.0724 -18.3114 0.0000  
## ma2 0.3638 0.0587 6.2020 0.0000  
## constant -1470.2739 15191.7140 -0.0968 0.9229  
##   
## $AIC  
## [1] 31.37181  
##   
## $AICc  
## [1] 31.37305  
##   
## $BIC  
## [1] 30.38498

fSarima <- sarima.for(TsVar02[1:1622], 140, 1, 1, 2, plot.all = TRUE)



fSarima$pred[1:20]

## [1] 10708749 10652147 10606869 10570099 10539723 10514149 10492185 10472931 10455715 10440030 10425494 10411823 10398801 10386266 10374098 10362206 10350520 10338990 10327576 10316251

#save prediction into df  
group\_S05\_df[1623:1762,2] <- fSarima$pred

### ARIMA Fit for Var03

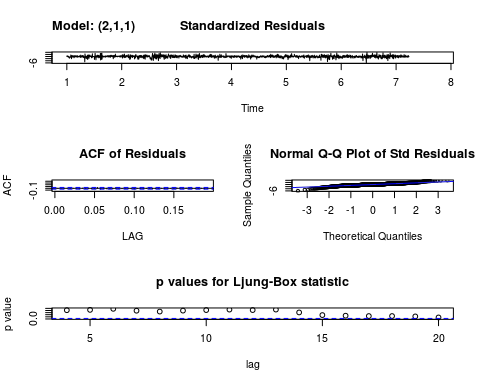
Doing the same process for Var03, we see that a 2,1,1 model is best. Again as before the box test results look good as do the ACF of the residuals.

library(astsa)  
h <- 140  
TsVar03 <- ts(group\_S05\_df$Var03, frequency = 5\*52)  
  
arimaFitVar03 <- auto.arima(TsVar03, seasonal = FALSE)  
arimaFitVar03

## Series: TsVar03   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## 0.7785 -0.1368 -0.6654  
## s.e. 0.1458 0.0251 0.1461  
##   
## sigma^2 estimated as 0.8007: log likelihood=-2118.51  
## AIC=4245.01 AICc=4245.04 BIC=4266.58

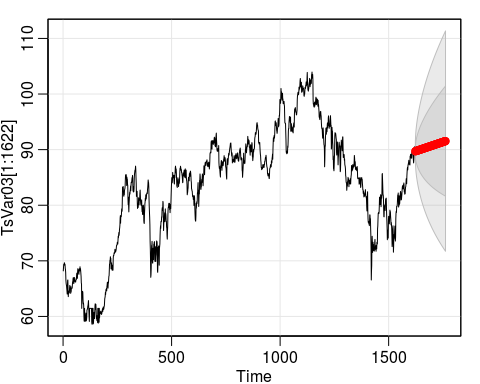
sarima(TsVar03, 2, 1, 1)

## initial value -0.102328   
## iter 2 value -0.104048  
## iter 3 value -0.110049  
## iter 4 value -0.110131  
## iter 5 value -0.110136  
## iter 6 value -0.110149  
## iter 7 value -0.110186  
## iter 8 value -0.110279  
## iter 9 value -0.110542  
## iter 10 value -0.110723  
## iter 11 value -0.110793  
## iter 12 value -0.111166  
## iter 13 value -0.111173  
## iter 14 value -0.111195  
## iter 15 value -0.111257  
## iter 16 value -0.111381  
## iter 17 value -0.111520  
## iter 18 value -0.111771  
## iter 19 value -0.112038  
## iter 20 value -0.112038  
## iter 21 value -0.112051  
## iter 22 value -0.112053  
## iter 23 value -0.112099  
## iter 24 value -0.112133  
## iter 25 value -0.112135  
## iter 26 value -0.112146  
## iter 27 value -0.112149  
## iter 28 value -0.112150  
## iter 29 value -0.112150  
## iter 29 value -0.112150  
## iter 29 value -0.112150  
## final value -0.112150   
## converged



## $fit  
##   
## Call:  
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,   
## Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,   
## reltol = tol))  
##   
## Coefficients:  
## ar1 ar2 ma1 constant  
## 0.7799 -0.1370 -0.6671 0.0131  
## s.e. 0.1451 0.0251 0.1454 0.0207  
##   
## sigma^2 estimated as 0.7991: log likelihood = -2118.3, aic = 4246.61  
##   
## $degrees\_of\_freedom  
## [1] 1617  
##   
## $ttable  
## Estimate SE t.value p.value  
## ar1 0.7799 0.1451 5.3731 0.0000  
## ar2 -0.1370 0.0251 -5.4585 0.0000  
## ma1 -0.6671 0.1454 -4.5872 0.0000  
## constant 0.0131 0.0207 0.6321 0.5274  
##   
## $AIC  
## [1] 0.7802196  
##   
## $AICc  
## [1] 0.7813741  
##   
## $BIC  
## [1] -0.2073532

fitArimaV03 <- sarima(TsVar03, 2, 1, 1, details = FALSE)  
  
fSarima <- sarima.for(TsVar03[1:1622], n.ahead = 140, 2, 1, 1, plot.all = TRUE)



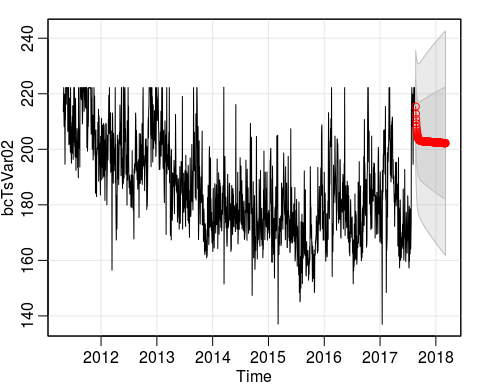
fSarima$pred[1:20]

## [1] 89.69658 89.70958 89.72489 89.73975 89.75394 89.76767 89.78114 89.79445 89.80770 89.82090 89.83408 89.84726 89.86042 89.87359 89.88675 89.89991 89.91308 89.92624 89.93940 89.95256

#save prediction into df  
group\_S05\_df[1623:1762,3] <- fSarima$pred

### Attempted a box cox transform to see how that worked out, really other than scale is pretty similar.

bcTsVar02 <- BoxCox(TsVar02, lambda = "auto")  
fSarima <- sarima.for(bcTsVar02, 140, 1, 1, 3, plot.all = TRUE)



#### Sanity checks for differencing. Run some difference tests, we see that one difference is absolutely needed, and perhaps two differences could be warranted for var03 as one order of difference gives .07 value for the test statistic (i.e.it is greater than .05), but it is questionable, as extra differencing in itself can lead to errors.

#Reset values  
TsVar02 <- ts(group\_S05\_df$Var02, frequency = 5\*52)  
TsVar03 <- ts(group\_S05\_df$Var03, frequency = 5\*52)  
  
library(urca)  
summary(ur.kpss(TsVar02))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 8 lags.   
##   
## Value of test-statistic is: 10.105   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(TsVar02)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 8 lags.   
##   
## Value of test-statistic is: 0.0054   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(TsVar03))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 8 lags.   
##   
## Value of test-statistic is: 8.3755   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(TsVar03)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 8 lags.   
##   
## Value of test-statistic is: 0.0658   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

print("Do we actually need a second order of differencing?")

## [1] "Do we actually need a second order of differencing?"

summary(ur.kpss(diff(diff(TsVar03))))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 8 lags.   
##   
## Value of test-statistic is: 0.003   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

print("Checking for seasonality:")

## [1] "Checking for seasonality:"

nsdiffs(TsVar02, test = "seas")

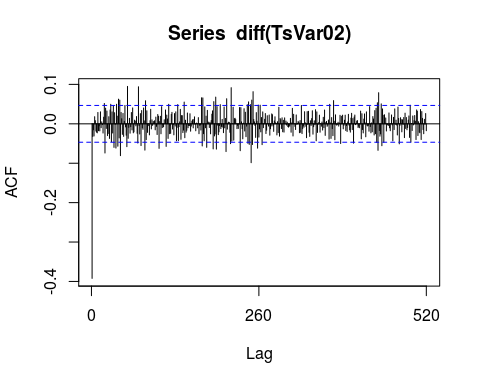
## [1] 0

nsdiffs(TsVar03, test = "seas")

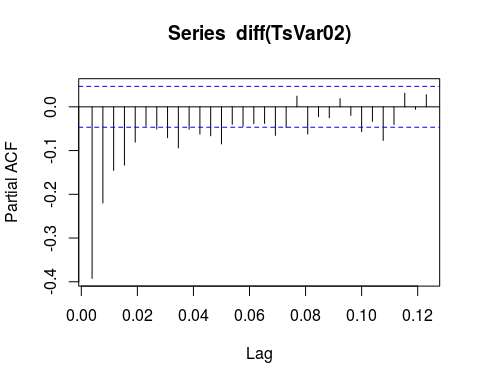
## [1] 0

Checking ACF graphs for the p and q values.

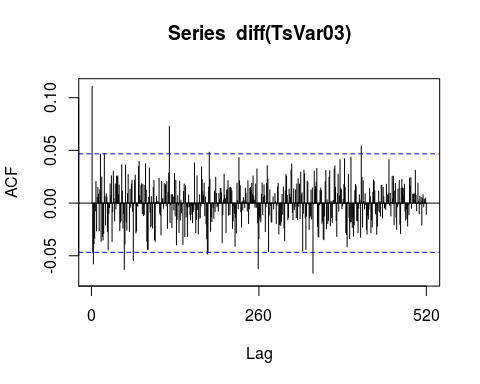
#113  
Acf(diff(TsVar02))



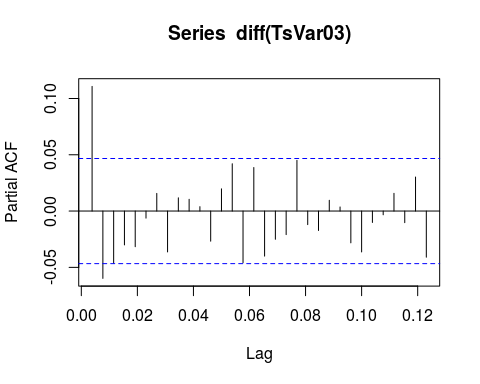
pacf(diff(TsVar02))



Acf(diff(TsVar03))

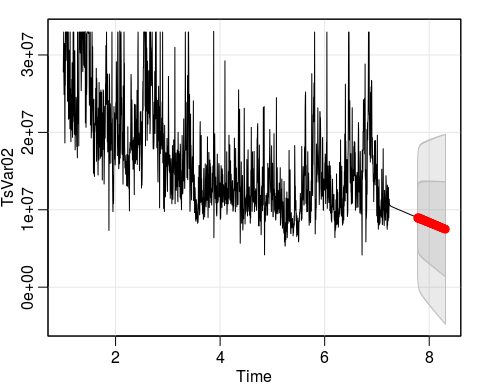


pacf(diff(TsVar03))

 Looking at the plots is seems like a (1,1,4) model might be better for Var02 considering the number of spikes in the acf graph. Rerunning using that model, the box test results look better, althought the AIC numbers are virtually unchanged.

The 2,1,1 ARIMA model based on the acf graphs looks reasonable for Var03.

fSarima <- sarima.for(TsVar02, 140, 4, 1,4,plot.all = TRUE)



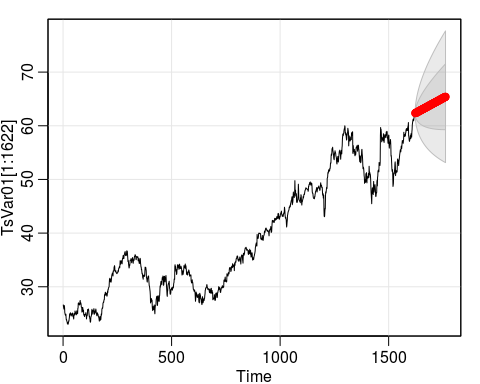
#### Group S01 Forecast

##### Note for this and future groups, detailed textual analysis is not given, as the concepts for each are similar to the above discusion given for S05. The graphs which correspond with the predictions given on the excel file, are given for the other groups so that the reader can make their own examination of that data.

##### Note as well, the code is hidden for brevity of output, as well as some of the output from the SARIMA command.

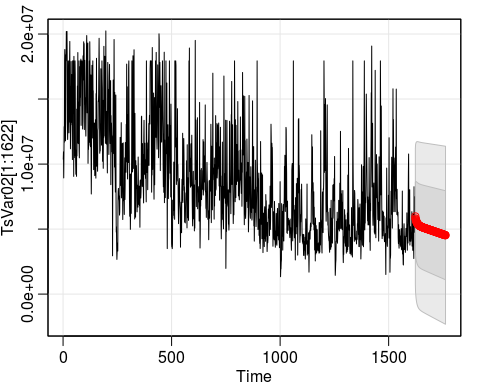
## Series: TsVar01   
## ARIMA(0,1,2) with drift   
##   
## Coefficients:  
## ma1 ma2 drift  
## 0.0875 -0.0727 0.0220  
## s.e. 0.0248 0.0250 0.0129  
##   
## sigma^2 estimated as 0.2612: log likelihood=-1210.51  
## AIC=2429.02 AICc=2429.04 BIC=2450.58

## Series: TsVar02   
## ARIMA(1,1,3) with drift   
##   
## Coefficients:  
## ar1 ma1 ma2 ma3 drift  
## 0.872 -1.4349 0.3060 0.1340 -5738.623  
## s.e. 0.042 0.0511 0.0447 0.0346 2703.689  
##   
## sigma^2 estimated as 6.799e+12: log likelihood=-26247.04  
## AIC=52506.07 AICc=52506.12 BIC=52538.42



## [1] "Subset of Predicted values: "

## [1] 62.34861 62.36244 62.38447 62.40650 62.42852 62.45055 62.47257 62.49460 62.51663 62.53865 62.56068 62.58270 62.60473 62.62676 62.64878 62.67081 62.69283 62.71486 62.73689 62.75891



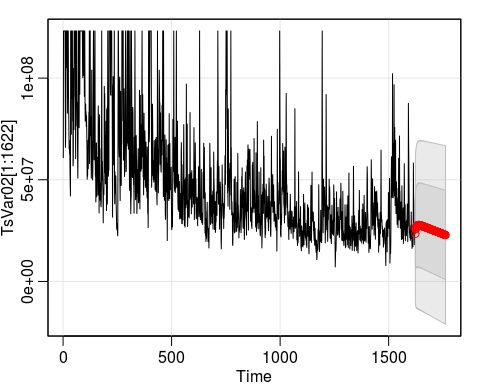
## [1] "Subset of Predicted values: "

## [1] 5976242 5815753 5746395 5685180 5631065 5583142 5540618 5502802 5469091 5438961 5411953 5387667 5365755 5345913 5327876 5311413 5296323 5282429 5269580 5257640

#### Group S02 Forecast

## Series: TsVar02   
## ARIMA(3,1,4) with drift   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 ma4 drift  
## 0.0818 0.8584 -0.1525 -0.5703 -1.0397 0.5069 0.1112 -41486.54  
## s.e. 0.2240 0.0611 0.1669 0.2245 0.0889 0.2168 0.0786 16530.25  
##   
## sigma^2 estimated as 2.745e+14: log likelihood=-29243.06  
## AIC=58504.12 AICc=58504.23 BIC=58552.63

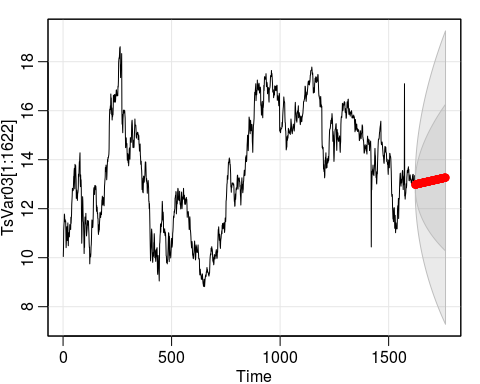
## Series: TsVar03   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## 0.7197 0.0683 -0.8283  
## s.e. 0.2518 0.0450 0.2499  
##   
## sigma^2 estimated as 0.09362: log likelihood=-378.95  
## AIC=765.9 AICc=765.92 BIC=787.46



## [1] "Subset of Predicted values: "

## [1] 23391972 25880151 25881126 26758001 26442267 27160232 26805375 27432028 27060353 27613212 27234991 27726531 27348929 27788880 27416942 27812975 27450175 27808391 27457041 27782336

## Warning in stats::arima(xdata, order = c(p, d, q), seasonal = list(order = c(P, : possible convergence problem: optim gave code = 1



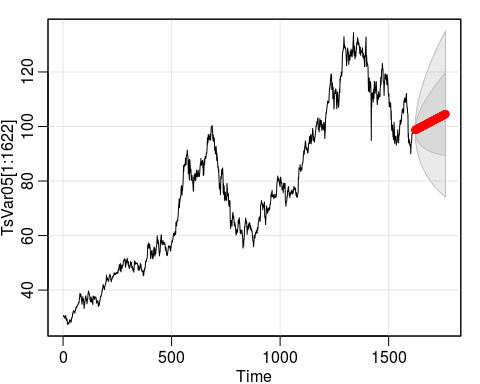
## [1] "Subset of Predicted values: "

## [1] 12.97521 12.98033 12.98516 12.98920 12.99270 12.99580 12.99861 13.00121 13.00366 13.00599 13.00824 13.01043 13.01257 13.01468 13.01676 13.01883 13.02089 13.02293 13.02497 13.02700

#### Group S03 Forecast

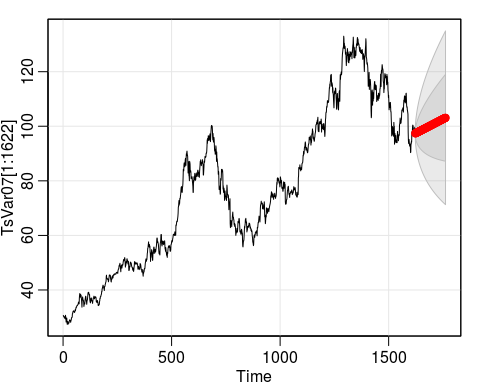
## Series: TsVar05   
## ARIMA(1,1,0)   
##   
## Coefficients:  
## ar1  
## -0.1634  
## s.e. 0.0245  
##   
## sigma^2 estimated as 2.248: log likelihood=-2956.16  
## AIC=5916.32 AICc=5916.33 BIC=5927.1

## Series: TsVar07   
## ARIMA(0,1,0)   
##   
## sigma^2 estimated as 1.811: log likelihood=-2781.33  
## AIC=5564.65 AICc=5564.66 BIC=5570.04



## [1] "Subset of Predicted values: "

## [1] 98.71268 98.75792 98.79945 98.84159 98.88363 98.92569 98.96774 99.00980 99.05185 99.09391 99.13596 99.17802 99.22008 99.26213 99.30419 99.34624 99.38830 99.43035 99.47241 99.51446



## [1] "Subset of Predicted values: "

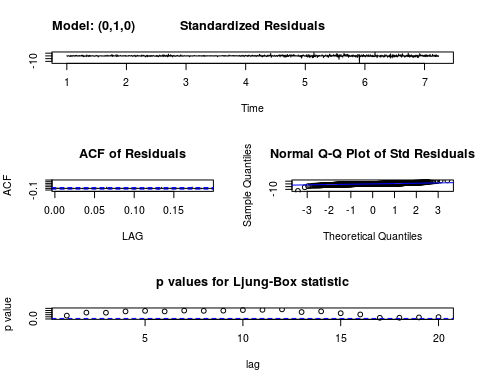
## [1] 97.38118 97.42237 97.46356 97.50475 97.54594 97.58713 97.62832 97.66951 97.71070 97.75188 97.79307 97.83426 97.87545 97.91664 97.95783 97.99902 98.04021 98.08140 98.12258 98.16377

#### Group S04 Forecast

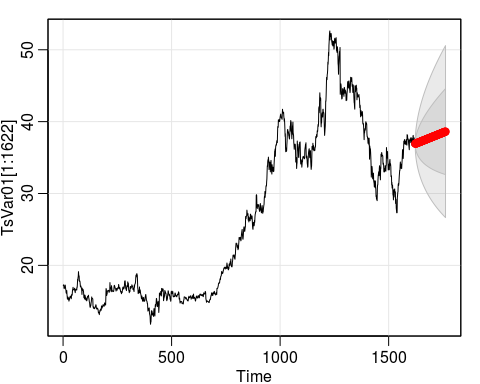
## Series: TsVar01   
## ARIMA(0,1,0)   
##   
## sigma^2 estimated as 0.2557: log likelihood=-1194.84  
## AIC=2391.68 AICc=2391.68 BIC=2397.07

## Series: TsVar02   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## 0.4778 0.0138 -0.9523  
## s.e. 0.0282 0.0274 0.0134  
##   
## sigma^2 estimated as 5.723e+13: log likelihood=-27974.37  
## AIC=55956.73 AICc=55956.76 BIC=55978.3

## initial value -0.682128   
## iter 1 value -0.682128  
## final value -0.682128   
## converged  
## initial value -0.682128   
## iter 1 value -0.682128  
## final value -0.682128   
## converged

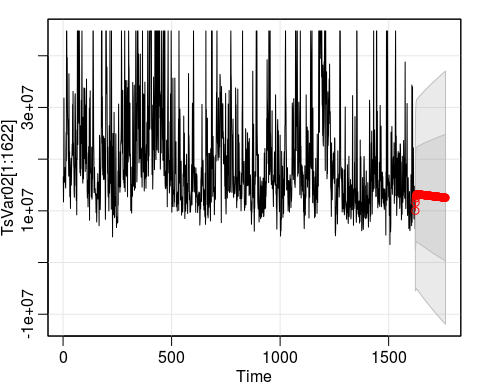


## $fit  
##   
## Call:  
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,   
## Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,   
## reltol = tol))  
##   
## Coefficients:  
## constant  
## 0.0122  
## s.e. 0.0126  
##   
## sigma^2 estimated as 0.2556: log likelihood = -1194.37, aic = 2392.74  
##   
## $degrees\_of\_freedom  
## [1] 1620  
##   
## $ttable  
## Estimate SE t.value p.value  
## constant 0.0122 0.0126 0.9684 0.333  
##   
## $AIC  
## [1] -0.3630232  
##   
## $AICc  
## [1] -0.3617856  
##   
## $BIC  
## [1] -1.359699



## [1] "Subset of Predicted values: "

## [1] 36.92216 36.93432 36.94648 36.95864 36.97080 36.98295 36.99511 37.00727 37.01943 37.03159 37.04375 37.05591 37.06807 37.08023 37.09239 37.10455 37.11671 37.12886 37.14102 37.15318



## [1] "Subset of Predicted values: "

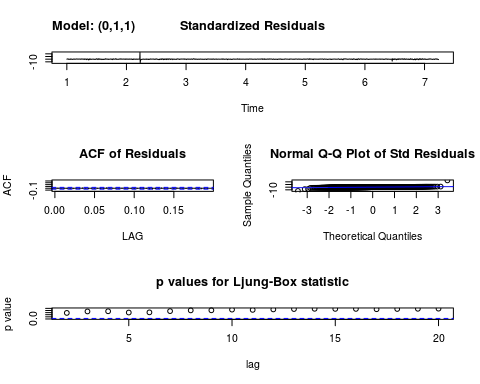
## [1] 10000727 11596210 12404977 12811308 13014265 13114379 13162487 13184297 13192810 13194598 13192986 13189654 13185453 13180812 13175949 13170973 13165941 13160879 13155804 13150720

#### Group S06 Forecast

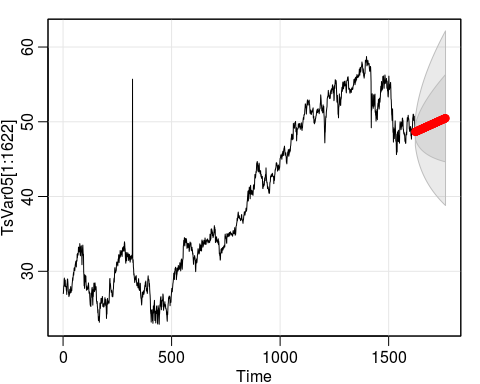
## Series: TsVar05   
## ARIMA(0,1,1)   
##   
## Coefficients:  
## ma1  
## -0.4661  
## s.e. 0.0230  
##   
## sigma^2 estimated as 0.844: log likelihood=-2162.24  
## AIC=4328.48 AICc=4328.48 BIC=4339.26

## Series: TsVar07   
## ARIMA(0,1,1)   
##   
## Coefficients:  
## ma1  
## -0.4553  
## s.e. 0.0232  
##   
## sigma^2 estimated as 0.8512: log likelihood=-2169.12  
## AIC=4342.25 AICc=4342.25 BIC=4353.03

## initial value 0.008912   
## iter 2 value -0.080612  
## iter 3 value -0.084606  
## iter 4 value -0.085420  
## iter 5 value -0.085437  
## iter 6 value -0.085437  
## iter 7 value -0.085437  
## iter 8 value -0.085437  
## iter 8 value -0.085437  
## iter 8 value -0.085437  
## final value -0.085437   
## converged  
## initial value -0.085407   
## iter 2 value -0.085407  
## iter 3 value -0.085407  
## iter 3 value -0.085407  
## iter 3 value -0.085407  
## final value -0.085407   
## converged

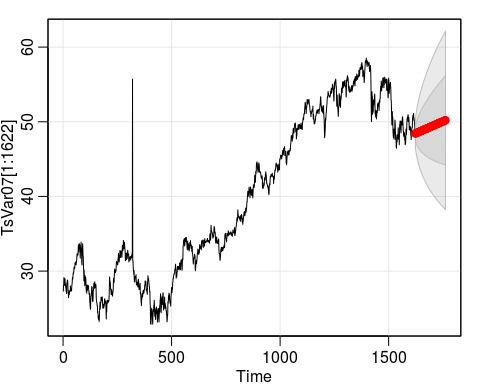


## $fit  
##   
## Call:  
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,   
## Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,   
## reltol = tol))  
##   
## Coefficients:  
## ma1 constant  
## -0.4673 0.0131  
## s.e. 0.0230 0.0122  
##   
## sigma^2 estimated as 0.8429: log likelihood = -2161.65, aic = 4329.31  
##   
## $degrees\_of\_freedom  
## [1] 1619  
##   
## $ttable  
## Estimate SE t.value p.value  
## ma1 -0.4673 0.0230 -20.3260 0.0000  
## constant 0.0131 0.0122 1.0813 0.2797  
##   
## $AIC  
## [1] 0.8315002  
##   
## $AICc  
## [1] 0.8327424  
##   
## $BIC  
## [1] -0.1618519



## [1] "Subset of Predicted values: "

## [1] 48.65134 48.66448 48.67763 48.69077 48.70391 48.71705 48.73020 48.74334 48.75648 48.76962 48.78277 48.79591 48.80905 48.82219 48.83534 48.84848 48.86162 48.87476 48.88791 48.90105



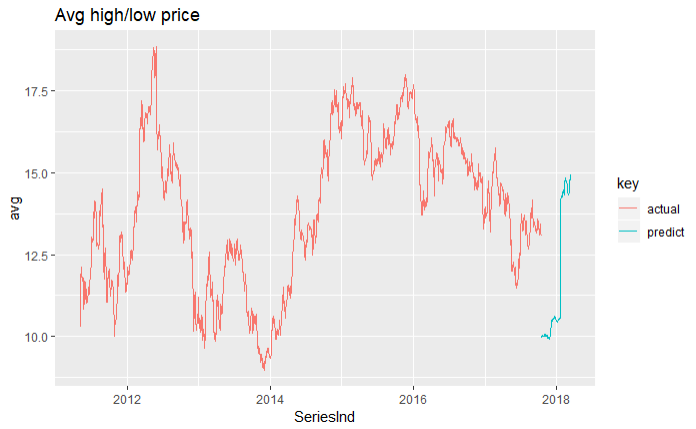
## [1] "Subset of Predicted values: "

## [1] 48.42725 48.44001 48.45276 48.46552 48.47827 48.49103 48.50378 48.51653 48.52929 48.54204 48.55480 48.56755 48.58031 48.59306 48.60582 48.61857 48.63133 48.64408 48.65683 48.66959

### Deep Learning

A promising and more complicated technique involves a process called deep learning. Deep learning can provide very accurate results at the cost of computer processing power. To build and train a model can take hours or days compared to other ML techniques. A deep learning model popular for time series is called Long Short-Term Memory (LSTM). Like ARIMA, this process requires data transformation to give the model the best chance at accurate forecast predictions and involves the constant tinkering of model parameters such as batch sizes, number of iterations, a loss function, optimizing function and other options. Each change of the model will need to be retrained.

Our attempt didn’t provide desirable results. The errors we calculated such as the RSME were high so model couldn’t be trusted to provide a best fit to the data. To simplify the model, the daily high and low price were averaged so the model only had to forecast one variable.



Caption for the picture.

### Conclusion

Each variable from each series in the dataset provides its own challenges. There isn’t a one-solution-fits-all for every time-series dataset though ARIMA may come close. It can eliminate residual autocorrelation or white noise, and it can take into account long-term trends and seasonality. It can auto-adjust and vary its p, d, and q parameters to build the best model.

Of course, many forecasting models will never be perfect. But the progressive to improve them may lead to the discovery of other characteristics from the data that you would have not seen otherwise. The goal to turn the random into predictable is exhausting but we have found the ARIMA model to be a proven starting point for any forecasting model.