Time series forecasting

Introduction: Building a predictive forecasting model for close price values of Amazon (AMZN) and Microsoft(MSFT)

Read the data

stock\_price<-read.csv("C:/Users/glane/Downloads/prices.csv")

stock\_price$date<- as.Date(stock\_price$date,"%d-%m-%Y")

Since we know that this is daily data and it begins in 2010 let’s update the frequency and start arguments. Selecting only two stocks for prediction as shown

library(dplyr)

##   
## Attaching package: 'dplyr'

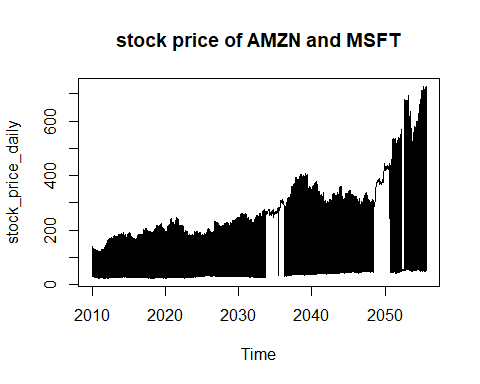
## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

stock\_price <- filter(stock\_price , symbol == c("AMZN" , "MSFT"))  
View(stock\_price)  
stock\_price\_daily <- ts(stock\_price$close, frequency = 52 ,start = 2010)

Plot the data with the autoplot() function which is convenient for working with time series.

plot(stock\_price\_daily, main = "stock price of AMZN and MSFT")

 From this we can see a few things

1.There is a general updward trend 2.The trend is not constant, it moves down during the recession 3.There is differences in sales based on month

Time Series Decomposition You should decompose your data to get or present a basic understanding of your data. Outputs of a decomposition

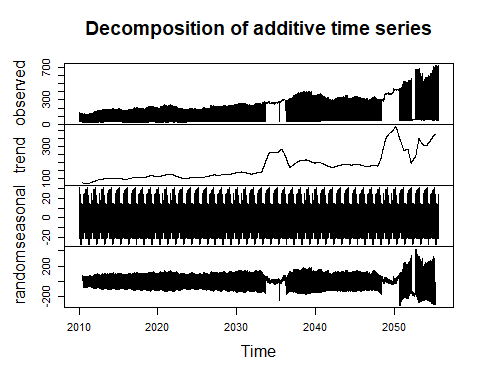
1.The underlying trend of your data 2.A seasonal factor 3.A remainder which explains what the trend and seasonal factor do not

Types of seasonal decompositions

Additive Seasonal Decomposition: Each season gets moved by a constant number that is added or subtracted from the trend. Multiplicative Seasonal Decomposition: Each season has a number we multiply to the trend.

Decomposing the Stock of AMZN and MSFT

decomposed\_stock\_additive <- decompose(stock\_price\_daily, type = "additive")  
plot(decomposed\_stock\_additive)

 Because our decomposition was additive, we can add the series in panels 2, 3, and 4 and get the top panel.

data = trend + seasonal + remainder

Splitting the data into train and test

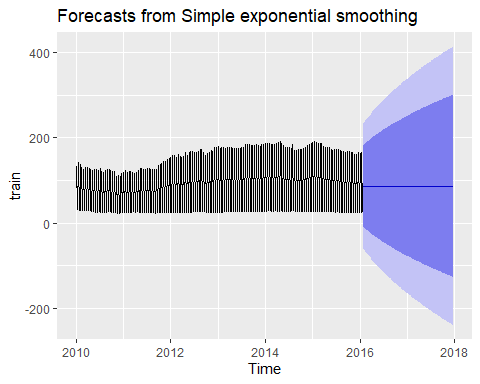
train <- window(stock\_price\_daily, end = c(2016,4))  
test <- window(stock\_price\_daily, start = c(2013, 1))

Exponential Smoothing for forecasting

library(forecast)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

sesmodel <- ses(train, alpha = .2, h = 100)  
p1<-autoplot(forecast(sesmodel))  
p1



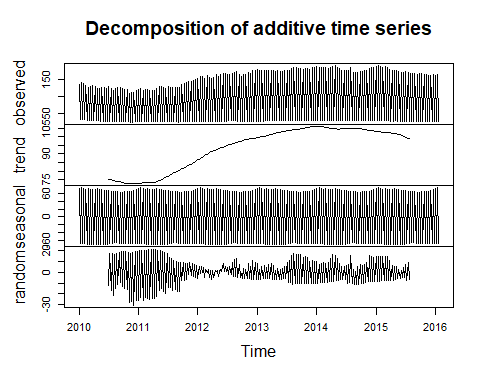
etc\_acc<-accuracy(sesmodel, test)  
etc\_acc

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.02213727 74.62859 73.43059 -115.55703 161.5713 8.360977  
## Test set 20.43161548 84.82059 82.05180 -94.63457 148.6115 9.342607  
## ACF1 Theil's U  
## Training set -0.9956641 NA  
## Test set -0.9856809 0.6233917

Arima model When to use Arima model: 1. Data should be stationary – by stationary it means that the properties of the series doesn’t depend on the time when it is captured. A white noise series and series with cyclic behavior can also be considered as stationary series.

1. Data should be univariate – ARIMA works on a single variable. Auto-regression is all about regression with the past values.

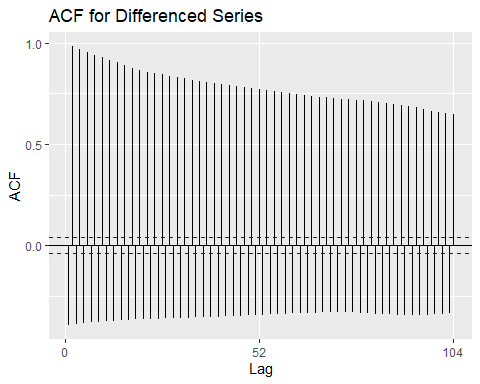
plot(decompose(train))



Autocorrelation

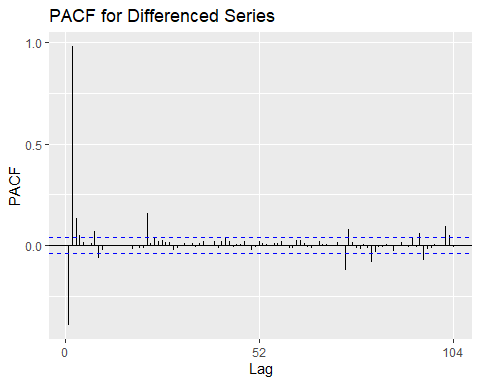
ggAcf(stock\_price\_daily, main='ACF for Differenced Series')

## Warning: Ignoring unknown parameters: main



ggPacf(stock\_price\_daily, main='PACF for Differenced Series')

## Warning: Ignoring unknown parameters: main

 On the x-axis we have previous time periods. On the y-axis we see the correlation between elec and its quantity from the time period on the x-axis.

Here we see a strong correlation with present values and the previous value, as represented by the vertical bar. This makes sense as the most recent value gives an indicator of trend. We also see a pretty good correlation with its value a year ago. This makes sense because the value peaks the same time each year.

Since these values are closely related (ie high correlation). We can create an effective autoregressive model with the ar() function. This will create a linear regression, finding coefficients between the past and current values.

Unit Root Test

library(tseries)  
adf.test( stock\_price\_daily, alternative = "stationary")

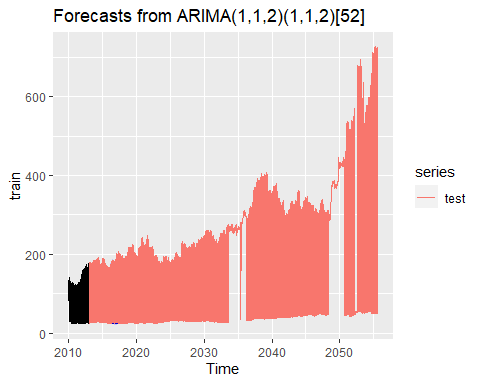
## Warning in adf.test(stock\_price\_daily, alternative = "stationary"): p-value  
## smaller than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: stock\_price\_daily  
## Dickey-Fuller = -4.8698, Lag order = 13, p-value = 0.01  
## alternative hypothesis: stationary

As per ADF test the series is stationary since p value is less than 0.05

Arima(p,d,q) The values of p and q are then chosen by minimising the AICc after differencing the data d times. p = number of autoregressive terms d = the number of nonseasonal differences needed for stationarity, and q = number of moving average terms

stocktrain <- Arima(train, order=c(1,1,2),  
 seasonal=c(1,1,2), lambda=0)  
stocktrain %>%  
 forecast(h=60) %>%  
 autoplot() + autolayer(test)

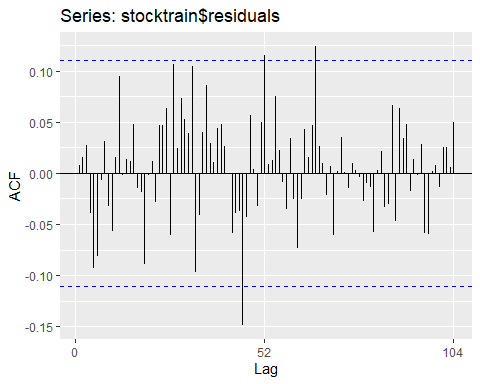


summary(stocktrain)

## Series: train   
## ARIMA(1,1,2)(1,1,2)[52]   
## Box Cox transformation: lambda= 0   
##   
## Coefficients:  
## ar1 ma1 ma2 sar1 sma1 sma2  
## -0.9973 0.3086 0.0063 0.1585 -0.8765 0.1593  
## s.e. 0.0037 0.0621 0.0613 1.1142 1.0968 0.7891  
##   
## sigma^2 estimated as 0.0005773: log likelihood=593.49  
## AIC=-1172.97 AICc=-1172.53 BIC=-1147.97  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.01106237 2.932236 1.56854 0.04273195 1.467081 0.1785975  
## ACF1  
## Training set -0.002792191

ACF and residual part

ggAcf(stocktrain$residuals)



library(FitAR)

## Warning: package 'FitAR' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: leaps

## Warning: package 'leaps' was built under R version 4.0.3

## Loading required package: ltsa

## Warning: package 'ltsa' was built under R version 4.0.3

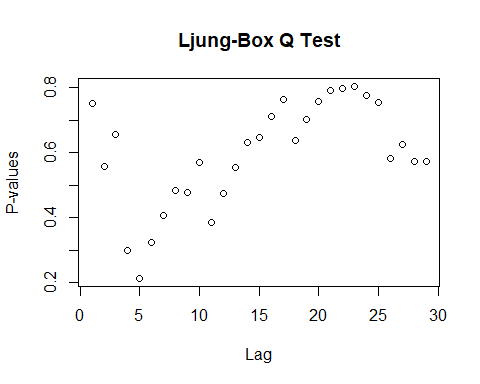
## Loading required package: bestglm

## Warning: package 'bestglm' was built under R version 4.0.3

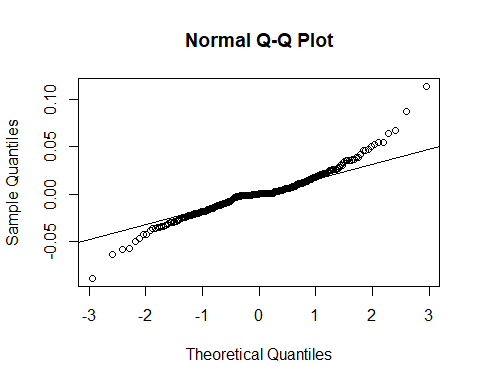
##   
## Attaching package: 'FitAR'

## The following object is masked from 'package:forecast':  
##   
## BoxCox

boxresult<-LjungBoxTest (stocktrain$residuals,k=2,StartLag=2)  
plot(boxresult[,3],main= "Ljung-Box Q Test", ylab= "P-values", xlab= "Lag")



qqnorm(stocktrain$residuals)  
qqline(stocktrain$residuals)

 The p-values for the Ljung-Box Q test all are well above 0.05, indicating “non-significance.”

As all the graphs are in support of the assumption that there is no pattern in the residuals, we can go ahead and calculate the forecast.

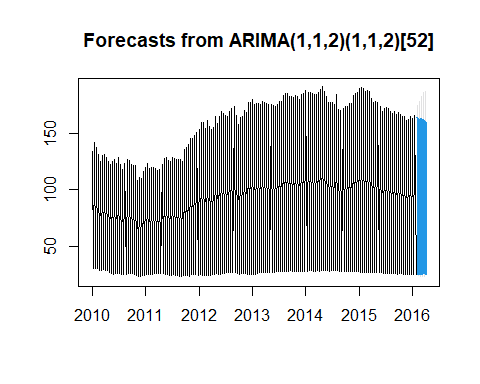
The ACF of residuals shows The mean of the residuals is close to zero and there is no significant correlation in the residuals series

Forecast

predict(stocktrain,n.ahead = 5)

## $pred  
## Time Series:  
## Start = c(2016, 5)   
## End = c(2016, 9)   
## Frequency = 52   
## [1] 5.100529 3.238284 5.089299 3.235553 5.092055  
##   
## $se  
## Time Series:  
## Start = c(2016, 5)   
## End = c(2016, 9)   
## Frequency = 52   
## [1] 0.02406030 0.02519930 0.03491443 0.03571842 0.04310029

futurVal <- forecast(stocktrain,h=10, level=c(99.5))  
plot(futurVal)

 A forecast error is the difference between the actual or real and the predicted or forecast value of a time series

COmpare ETS vs ARIMA

etc\_acc<-accuracy(sesmodel, test)  
etc\_acc

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.02213727 74.62859 73.43059 -115.55703 161.5713 8.360977  
## Test set 20.43161548 84.82059 82.05180 -94.63457 148.6115 9.342607  
## ACF1 Theil's U  
## Training set -0.9956641 NA  
## Test set -0.9856809 0.6233917

acc\_arima<-accuracy(futurVal, test)  
acc\_arima

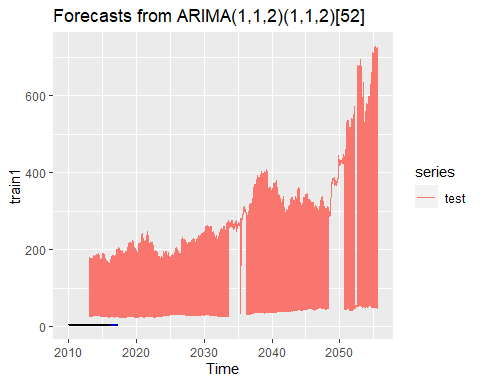
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.01106237 2.932236 1.568540 0.04273195 1.467081 0.1785975  
## Test set 5.37199360 8.232069 5.454249 3.14793210 3.471098 0.6210334  
## ACF1 Theil's U  
## Training set -0.002792191 NA  
## Test set -0.718789750 0.08373

Arima model is better since error rate (MAPE) is 1.46% compare to 161.57% of ETS

Natural logarithms of the series

stocktrain\_log<- log(stock\_price\_daily)  
train1 <- window(stocktrain\_log, end = c(2016, 4))  
test1 <- window(stocktrain\_log, start = c(2013, 1))

stocktrain1 <- Arima(train1, order=c(1,1,2),  
 seasonal=c(1,1,2), lambda=0)  
stocktrain1 %>%  
 forecast(h=60) %>%  
 autoplot() + autolayer(test)



summary(stocktrain1)

## Series: train1   
## ARIMA(1,1,2)(1,1,2)[52]   
## Box Cox transformation: lambda= 0   
##   
## Coefficients:  
## ar1 ma1 ma2 sar1 sma1 sma2  
## -0.9968 0.3208 0.0627 0.1482 -0.8535 0.2031  
## s.e. 0.0044 0.0622 0.0596 0.6756 0.6654 0.4665  
##   
## sigma^2 estimated as 3.33e-05: log likelihood=971.49  
## AIC=-1928.98 AICc=-1928.54 BIC=-1903.97  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0005312203 0.02257443 0.01523751 0.01516045 0.3638153 0.1815565  
## ACF1  
## Training set 0.0138673

You can see than the error rate (MAPE) has fallen down to 0.36%