Forecasting Stock Price Using Time Series Analysis

Jiali Han, Junhua Zhu, Wei Li, Zhuofan Dong

4/23/2020

A picture containing clock

Description automatically generated

## Abstract

In this project, we first used the autoregressive integrated moving average model (ARIMA) in time series analysis to predict the stock price of Google. A time series dataset is a dataset that lists data in order of time. The data in a time series are equally spaced. The daily closing value of the stock price is a typical application of time series.

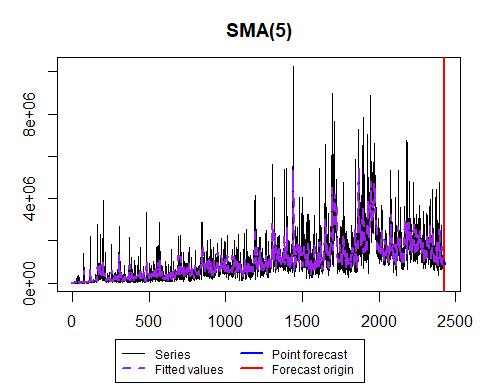
In the first example, we used the method of moving average. Moving average is an indicator of the overall trend of a dataset, which is the average of any subset of the data. Besides, It is also a useful tool for forecasting long-term trends or circles. The second method we used is the exponential smoothing. It is a powerful technique used for forecasting univariate time series data that can be extended to support the analysis of data with systematic trends. In the last example, we used the ARIMA model. ARIMA model is a generalized version of the autoregressive moving average model. The ARIMA model is widely used to fit time series dataset and forecast the future points in the series.

## Analysis

### Simple Moving Average

This method is used to predict the stock price M at time t by averaging the time series values over adjacent time periods. It depends on choosing n point moving, where n should be larger or equal to 1. We first average the data value over an adjacent time period depending on the type of n (n is odd or even). Then we divide it by n, and we can get predicted value M at time t. Here is an example:

library(smooth)  
stock <- read.csv("acwi.us.txt",header = TRUE)  
MovingAverage <- sma(stock$Volume,order=5,silent=FALSE)



# visualize the forecast plot with order 5

Here we get the output plot by using a simple moving average and we choose 5 data points to average the value at different times. In this case, we use sma() function to plot the forecast graph and order=5 to use 5 data points doing moving average method. The graph shows a fitted value line and an actual value line. The difference between these two lines displays the accuracy of the forecast.

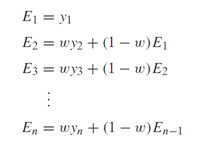
# suppose we want to forecast value at quarter 10 using 5 data points,  
M10 = (stock[8,6]+stock[9,6]+stock[10,6]+stock[11,6]+stock[12,6])/5  
M10 # then we can get predicted value using simple smoothing average

## [1] 3730.2

In the second case, we want to forecast the stock price at specific time period 10, and then we predict that value by averaging the data point at time period 8,9, 10, 11, and 12. The result is 3730.2. However, the result is much smaller than the actual value. The analysis is that the stock price increases dramatically that year, and the forecast model cannot follow the trend of increasing price.

### Exponential Smoothing

To compute exponential smoothing forecasts，first we choose a weight w between 0 and 1. Smaller w gives less weight to the current value of time series and yields a smoother series, whereas larger w has the opposite effect. Using weight w, we can forecast value based on current value as a list of values:

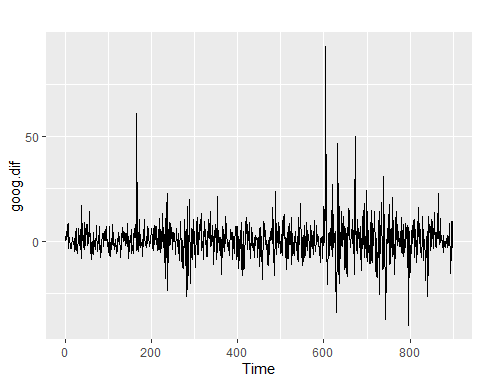


For future value, it could be:

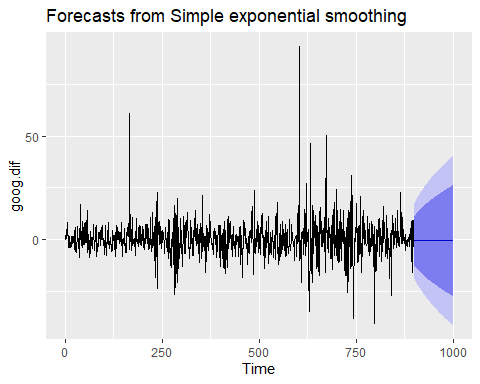


Here is an example code of exponential smoothing method:

library(tidyverse)  
library(fpp2)  
goog.train <- window(goog, end = 900) #traning the dataset of stock data  
goog.test <- window(goog, start = 901)  
goog.dif <- diff(goog.train)  
autoplot(goog.dif)



ses.goog.dif <- ses(goog.dif, alpha = .2, h = 100) #using parameter 0.2 to forecast  
autoplot(ses.goog.dif) # visualize the fexponential forecast



# to view the accuracy of forecast  
goog.dif.test <- diff(goog.test)  
accuracy(ses.goog.dif, goog.dif.test)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.01368221 9.317223 6.398819 99.97907 253.7069 0.7572009  
## Test set 0.97219517 8.141450 6.117483 109.93320 177.9684 0.7239091  
## ACF1 Theil's U  
## Training set -0.05440377 NA  
## Test set 0.12278141 0.9900678

Here we get the forecast plot of stock price by exponential smoothing method. Depending on the weight we choose, we can get different plots of exponential smoothing. After we compute the exponential smoothing forecast, we can use the accuracy()function to get the value of ME and RMSE so that we can estimate the accuracy of the forecast. Moreover, from the graph we can see that the predicted values resemble beyond the available time period in the dataset. Therefore, one drawback of exponential smoothing is that it cannot predict value at future time periods, which means it is an application for short-term period data points.

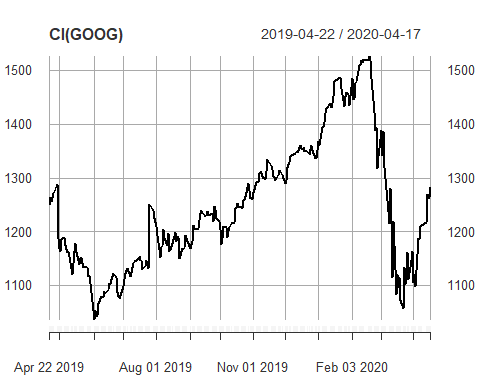
### ARIMA Model

Before applying the ARIMA model, there’s two assumptions that we should keep in mind: one is the data should be stationary, which means the means, variance and autocorrelation don’t change over time; the other is the data should be univariate.

library(quantmod)  
library(forecast)  
library(xlsx)  
library(tseries)  
library(timeSeries)  
library(dplyr)  
library(prophet)  
library(fGarch)  
getSymbols("GOOG", src = "yahoo", from = "2019-04-20",to = "2020-4-19")

## [1] "GOOG"

plot(Cl(GOOG))



close\_price<-c(Cl(GOOG))

After getting our data, we can use the auto.arima() function to get the best fitted model

#conduct adf test for the dataset  
print(adf.test(close\_price))

##   
## Augmented Dickey-Fuller Test  
##   
## data: close\_price  
## Dickey-Fuller = -1.7685, Lag order = 6, p-value = 0.6733  
## alternative hypothesis: stationary

#apply auto arima to the dataset  
modelfit <- auto.arima(close\_price, lambda = "auto")

We are going to conduct a Box-Ljung test here to the residuals of the fitted ARIMA model. This is a test about independence. It tests the overall randomness instead of testing the individual randomness at each distinct lag. The result of the Box-Ljung test tells us the residuals have no autocorrelation.

#we can check the residuals of the model with ARIMA parameters selected.  
summary(modelfit)

## Series: close\_price   
## ARIMA(1,1,0)   
## Box Cox transformation: lambda= 0.6865685   
##   
## Coefficients:  
## ar1  
## -0.2925  
## s.e. 0.0604  
##   
## sigma^2 estimated as 7.999: log likelihood=-614.19  
## AIC=1232.37 AICc=1232.42 BIC=1239.42  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.1784899 26.13307 17.0544 -0.01433272 1.390662 0.9883102  
## ACF1  
## Training set 0.01646473

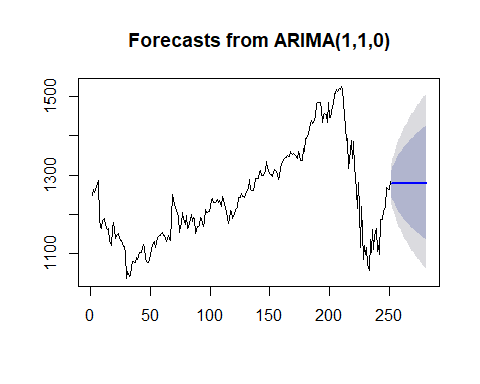
#Box test for lag=2  
Box.test(modelfit$residuals, lag= 2, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: modelfit$residuals  
## X-squared = 1.0044, df = 2, p-value = 0.6052

#not reject null hypothesis  
Box.test(modelfit$residuals, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: modelfit$residuals  
## X-squared = 0.023405, df = 1, p-value = 0.8784

#not reject our null hypothesis, we can keep moving on our study.  
price\_forecast <- forecast(modelfit, h=30)  
#forecast the next 30 days stock prices  
plot(price\_forecast)



#plot for the forecasting data with 80% and 95% confidence intervals for lower and upper price scenarios.

The blue line represents the mean of the prediction, and the darker and lighter area represents 80% and 95% CI in lower and upper scenarios.

Next, we train our dataset to make our model more robust and prevent overfitting.

head(price\_forecast$mean)

## Time Series:  
## Start = 252   
## End = 257   
## Frequency = 1   
## [1] 1277.454 1279.149 1278.653 1278.798 1278.756 1278.768

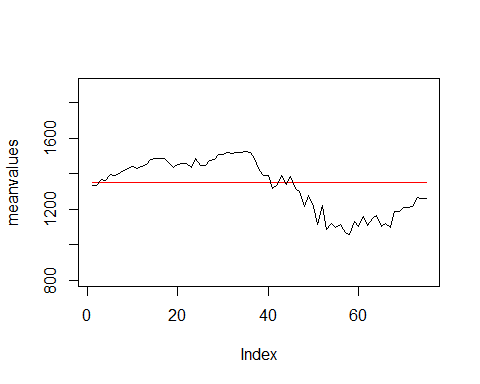
head(price\_forecast$lower)

## Time Series:  
## Start = 252   
## End = 257   
## Frequency = 1   
## 80% 95%  
## 252 1243.488 1225.624  
## 253 1237.563 1215.724  
## 254 1229.169 1203.222  
## 255 1222.921 1193.661  
## 256 1217.057 1184.785  
## 257 1211.790 1176.794

head(price\_forecast$upper)

## Time Series:  
## Start = 252   
## End = 257   
## Frequency = 1   
## 80% 95%  
## 252 1311.706 1329.952  
## 253 1321.162 1343.575  
## 254 1328.745 1355.506  
## 255 1335.451 1365.751  
## 256 1341.402 1374.943  
## 257 1346.865 1383.358

#we can train the 70% of our dataset and set the test set as the rest(30%).  
N = length(close\_price)  
n = 0.7\*N  
train = close\_price[1:n, ]  
test = close\_price[(n+1):N, ]  
trainarimafit <- auto.arima(train, lambda = "auto")  
predlen=length(test)  
trainarimafit <- forecast(trainarimafit, h=predlen)  
meanvalues <- as.vector(trainarimafit$mean)  
precios <- as.vector(test$GOOG.Close)  
plot(meanvalues, type= "l", col= "red")  
lines(precios, type = "l")



#Finally, we have a plot for trained dataset of mean tendency.

The main result in ARIMA(p, d, q), where p denotes the number of autoregressive terms, d denotes the number of times that the set should be differentiated for making it stationary, and q denotes the number of invertible moving average terms.

From the result, we can see that the ARIMA model selects the best model parameters so that it will give a good fit to the data. Consequently, the ARIMA model generally guarantees a very good estimation. Therefore, with past predictions, we can use ARIMA to forecast daily close price values in the future.

## Discussion

There are tons of methods used to predicting stock price and we only practiced a few of them here. We compared these three methods through this experiment and concluded the pros and cons for each of them in the following chart:

|  |  |  |
| --- | --- | --- |
| Model | Advantages | Disadvantages |
| Simple Moving Average | Useful for forecasting long-term trends | The forecast will be less accurate for short-term dramatic growth of value |
| Exponential Smoothing | Can control weight w to get more accurate result for current time period | Cannot predict value at future time period, only apply to short-term time period time series model |
| ARIMA Model | Fits data well thus accuracy can be guaranteed | Only works for one variable. Requires stationary data |

Though these methods are nice and reliable, in reality we need to consider other elements that can affect the stock price from many other aspects to make our prediction.

## Appendix

<https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average> <https://en.wikipedia.org/wiki/Time_series> <http://uc-r.github.io/ts_exp_smoothing> <https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs/version/3> <https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/>