Simple Forecasts

Dayanara M.

January 2, 2021

```r
library(rmarkdown)
```

```
## Warning: package 'rmarkdown' was built under R version 3.5.1
```

```r
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Exploring the dataset

```r
setwd("C:/Users/ddaya/OneDrive/Quantitative Finance")


# import data
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 3.5.1
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
StockData <- read.zoo("GOOG.csv",header = TRUE, sep = ",",format="%Y-%m-%d")
PriceData<-ts(StockData$Adj.Close, frequency = 5) # Frequency=5 because it is set on only business days
summary(StockData)
```

```
##      Index                 Open             High             Low        
##  Min.   :2014-12-30   Min.   : 493.3   Min.   : 494.6   Min.   : 486.2  
##  1st Qu.:2016-06-29   1st Qu.: 752.4   1st Qu.: 758.0   1st Qu.: 745.6  
##  Median :2017-12-28   Median :1027.2   Median :1040.4   Median :1016.3  
##  Mean   :2017-12-29   Mean   :1006.5   Mean   :1016.3   Mean   : 997.1  
##  3rd Qu.:2019-07-01   3rd Qu.:1195.3   3rd Qu.:1204.2   3rd Qu.:1185.8  
##  Max.   :2020-12-29   Max.   :1824.5   Max.   :1847.2   Max.   :1822.7  
##      Close          Adj.Close          Volume        
##  Min.   : 491.2   Min.   : 491.2   Min.   :  346800  
##  1st Qu.: 751.1   1st Qu.: 751.1   1st Qu.: 1242050  
##  Median :1027.8   Median :1027.8   Median : 1525200  
##  Mean   :1007.1   Mean   :1007.1   Mean   : 1737910  
##  3rd Qu.:1195.6   3rd Qu.:1195.6   3rd Qu.: 1973300  
##  Max.   :1828.0   Max.   :1828.0   Max.   :11164900
```

```r
head(StockData, nrow=10)
```

```
##                Open     High      Low    Close Adj.Close  Volume
## 2014-12-30 526.6441 529.6957 525.6867 528.9677  528.9677  876200
## 2014-12-31 529.7955 531.1417 524.3604 524.9587  524.9587 1368200
## 2015-01-02 527.5616 529.8154 522.6650 523.3731  523.3731 1447500
## 2015-01-05 521.8273 522.8944 511.6552 512.4630  512.4630 2059800
## 2015-01-06 513.5900 514.7617 499.6781 500.5856  500.5856 2899900
## 2015-01-07 505.6118 505.8552 498.2820 499.7280  499.7280 2065000
```

Forecasting

```r
#### Forecasting models
       ####################################
  ##########################################################################################################################
########
  ### Point Forecasts
  # Note: A point forecast is the mean of all possible future sample paths. So the point forecasts are usually much less var
iable than the data.
     # we will forecast for the next 10 days

library(forecast)
```

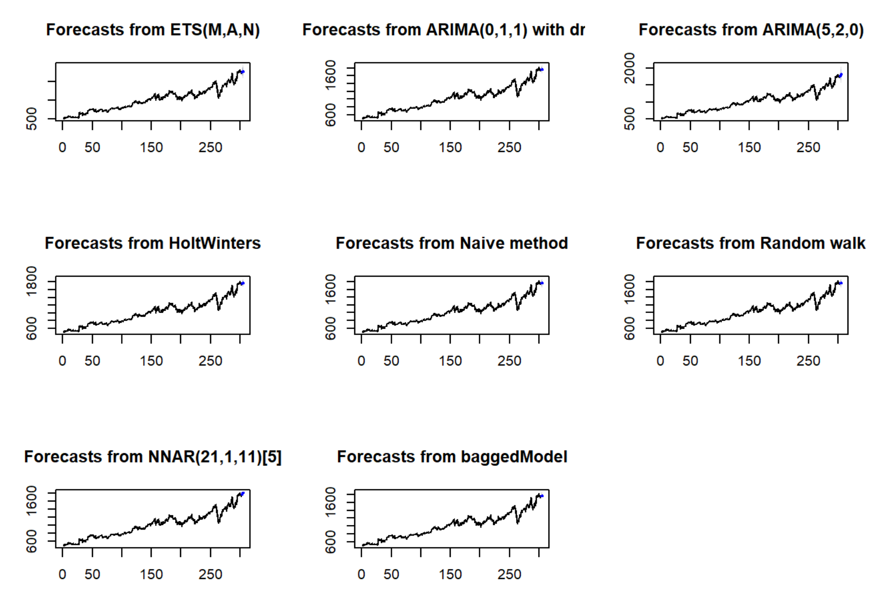
```
## Warning: package 'forecast' was built under R version 3.5.1
```

```
## Warning: As of rlang 0.4.0, dplyr must be at least version 0.8.0.
## * dplyr 0.7.5 is too old for rlang 0.4.5.
## * Please update dplyr to the latest version.
## * Updating packages on Windows requires precautions:
##   <https://github.com/jennybc/what-they-forgot/issues/62>
```

```r
     m_ets = ets(PriceData) #exponential smoothing
      f_ets = forecast(m_ets, h=10) # forecast de exponential smoothing
         # plot( f_ets)

     m_aa = auto.arima(PriceData)  # Auto ARIMA
      f_aa = forecast(m_aa, h=10)   # forecast de ARIMA
       # plot(f_aa)
    m_ar<- arima(PriceData, order = c(5,2,0)) # ARIMA
      f_ar<-forecast(m_ar, h=10)
       # plot(f_ar)
   # m_tbats = tbats(PriceData) # Model for series exhibiting multiple complex seasonalities
    #  f_tbats = forecast(m_tbats, h=10) # Trigonometric regressors to model multiple-seasonalities
       # plot(f_tbats)

        # TBATS is an acronym for the following:
          # T for trigonometric regressors to model multiple-seasonalities
          # B for Box-Cox transformations
          # A for ARMA errors
          # T for trend
          # S for seasonality
      m_holt <- HoltWinters(PriceData, gamma=FALSE)
        f_holt=forecast(m_holt, h = 10)   # forecast Holt's Exponential Smoothing
         # plot(f_holt)
     m_nai<- naive(PriceData) #naive bayes
      f_nai<-forecast(m_nai, h=10) # forecast naive bayes
       # plot(f_nai)
    m_rwf<-rwf(PriceData) # random walk with drift model
      f_rwf<-forecast(m_rwf, h=10) # drift forecast
         #   plot(f_rwf)
    m_nn <- nnetar(PriceData)  # Neural Network
      f_nn <- forecast(m_nn, h=10)       # forecast Neural Network
       # plot(f_nn)
  #  m_stlf<-stlm(PriceData)# Loess Forecasting model
    #  f_stlf<-forecast(m_stlf, h=10) # Forecast Loess
      #  plot(f_stlf)
  #  library(fracdiff)
     #   m_arf = arfima(PriceData)  # Auto ARIMA
       #   f_arf = forecast(m_arf, h=10)   # forecast de ARIMA
          #  plot(f_arf) # ARFIMA(p,d,q) model is selected and estimated automatically using
                        # the Hyndman-Khandakar (2008) algorithm  to select
                        # p and q and the Haslett and Raftery (1989) algorithm to estimate the parameters including d.
     m_a <- ma(PriceData,order=5) # Moving Average
      f_ma<-forecast(m_a, h=10) # forecast MA
```

```
## Warning in ets(object, lambda = lambda, biasadj = biasadj,
## allow.multiplicative.trend = allow.multiplicative.trend, : Missing values
## encountered. Using longest contiguous portion of time series
```

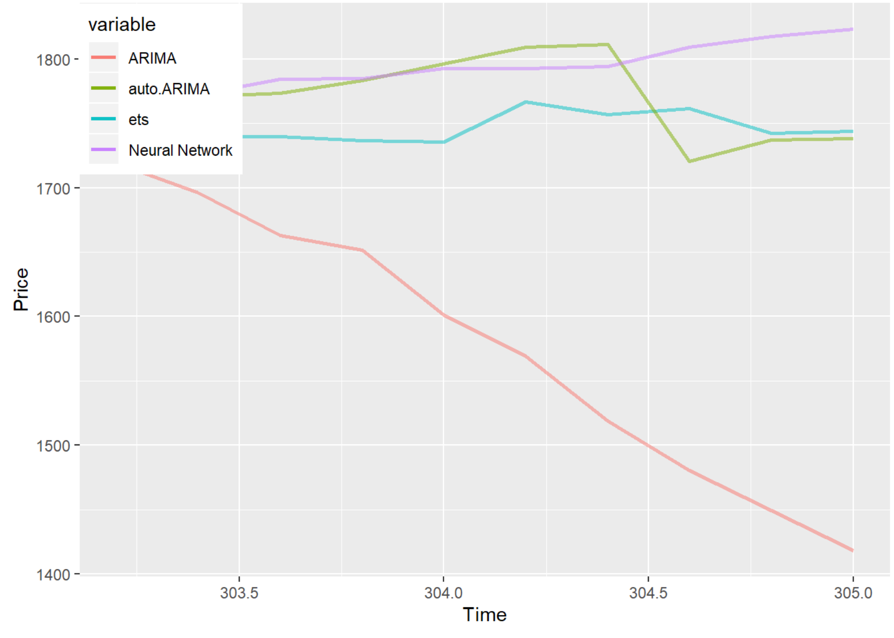
```r
       # plot(f_ma)

     m_ba<-baggedModel(PriceData, fn="auto.arima") # bagged ARIMA
      f_ba<-forecast(m_ba, h=10)
       # plot(f_ba)

 ### plot all forecasting models
        # png(file='gtemps1.png', width=600, height=320)
        par(mfrow=c(3,3)) # 3 columns and 3 rows of graphs
        plot(f_ets)
        plot(f_aa)
        plot(f_ar)
       # plot(f_tbats)
        plot(f_holt)
        plot(f_nai)
        plot(f_rwf)
        plot(f_nn)
       # plot(f_stlf)
       # plot(f_arf)
       # plot(f_ma)
        plot(f_ba)
        #  dev.off()
```




```r
### Simulations of the most accurate forecasts

      s_ets<-simulate(m_ets, nsim=10, future=TRUE, bootstrap=TRUE)
      s_aa<-simulate(m_aa, nsim=10, future=TRUE, bootstrap=TRUE)
      s_ar<-simulate(m_ar, nsim=10, future=TRUE, bootstrap=TRUE)
      s_nn<-simulate(m_nn, nsim=10, future=TRUE, bootstrap=TRUE)
          ### Fit
            si_ets<-simulate(m_ets, nsim=length(PriceData), bootstrap=TRUE)
            si_aa<-simulate(m_aa, nsim=length(PriceData), bootstrap=TRUE)
            si_ar<-simulate(m_ar, nsim=length(PriceData), bootstrap=TRUE)
            si_nn<-simulate(m_nn, nsim=length(PriceData), bootstrap=TRUE)

   ### we generate graphs of the predicted values
            library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```r
gtemp.df    = data.frame(Time=c(time(s_ets)), gtemp=c(s_aa), gtempk=c(s_ets), gtempl=c(s_ar), gtempm=c(s_nn))
ggplot(data = gtemp.df, aes(x=Time, y=value, color=variable )  )             +
  ylab('Price')                                +
  geom_line(aes(y=gtemp , col='ets'), size=1, alpha=.5)   +
  geom_line(aes(y=gtempk, col='auto.ARIMA'),  size=1, alpha=.5)   +
  geom_line(aes(y=gtempl, col='ARIMA'),  size=1, alpha=.5)   +
  geom_line(aes(y=gtempm, col='Neural Network'),  size=1, alpha=.5)   +
  theme(legend.position=c(.1,.85))
```



Comparison between models

```r
                         ################################  Comparison between models  ####################################
#####################

                         f1<-accuracy(m_ets)
                         f2<-accuracy(m_aa)
                         f3<-accuracy(m_ar)
                         f4<-accuracy(m_nn)


                         Train<-rbind(f1,f2,f3,f4)
                         rownames(Train) <- c("m_ets","m_aa","m_ar","m_nn")

                         Train<-data.frame(Train)
                         Train[order(-Train$MAPE),]   # <---- select the model with smalles MAPE
```

	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	MAPE <dbl>	MASE <dbl>	ACF1 <dbl>
m_ar	0.024634515	20.00959	13.062853	0.0002912513	1.2665151	1.1197463	-0.026917685
m_aa	0.001191957	18.31200	11.637269	-0.0265650792	1.1278264	0.4399607	-0.003127487
m_ets	-0.012635962	18.32183	11.617459	-0.0277767405	1.1256803	0.4392118	-0.033098758
m_nn	0.044759366	14.52322	9.852118	-0.0166998760	0.9760794	0.3724710	0.010015418

4 rows

Test

We will test t=our two best model choices and select a final one

```r
                         dm.test(residuals(m_nn),residuals(m_ets), alternative = "less",
                                 h=10) # <--- Diebold-Mariano test compares the forecast accuracy of two forecast methods
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(m_nn)residuals(m_ets)
## DM = 9.559, Forecast horizon = 10, Loss function power = 2,
## p-value = 1
## alternative hypothesis: less
```

```r
                         # For alternative="less", the alternative hypothesis is that method 2 is less accurate than method
1.
```

According to the Diebold-Mariano Test, we conclude that ETS model is less accurate than the Neural Network model.

Predict stock prices

```r
    Forecasted_GOOG<-as.data.frame(f_nn)
     names(Forecasted_GOOG)[1] <- "Stock"
        dates<-seq(as.Date("2020/12/29"), by = 5, length.out = 10)
          data.frame(dates,Forecasted_GOOG$Stock)
```

dates <date>	Forecasted_GOOG.Stock <dbl>
2020-12-29	1778.919
2021-01-03	1790.785
2021-01-08	1791.596
2021-01-13	1796.858
2021-01-18	1803.261
2021-01-23	1806.153
2021-01-28	1807.050
2021-02-02	1810.802
2021-02-07	1811.095
2021-02-12	1818.785

1-10 of 10 rows