Microsoft_stock_price_forecasting.R

rosha

Fri Sep 20 18:44:01 2019

```
library(ggplot2)
```

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

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 3.5.2
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 3.5.2
```

```
## Loading required package: zoo
```

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

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

```
## Loading required package: TTR
```

```
## Warning: package 'TTR' was built under R version 3.5.2
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(ggplot2)
library(fpp)
```

```
## Warning: package 'fpp' was built under R version 3.5.2
```

```
## Loading required package: forecast
```

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

```
## Loading required package: fma
```

```
## Warning: package 'fma' was built under R version 3.5.2
```

```
## Loading required package: expsmooth
```

```
## Warning: package 'expsmooth' was built under R version 3.5.2
```

```
## Loading required package: lmtest
```

```
## Loading required package: tseries
```

```
## Warning: package 'tseries' was built under R version 3.5.2
```

```
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 3.5.2
```

```
##
## Attaching package: 'fpp2'
```

```
## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec
```

```
start_date <- as.Date("2012-01-01")
end_date <- as.Date("2019-01-01")
start_date
```

```
## [1] "2012-01-01"
```

```
end_date
```

```
## [1] "2019-01-01"
```

```
lapply(start_date, class)
```

```
## [[1]]
## [1] "Date"
```

```
lapply(end_date, class)
```

```
## [[1]]
## [1] "Date"
```

```
#Data scraping from Yahoo finance
getSymbols("MSFT", src = "yahoo", from = start_date, to = end_date)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## [1] "MSFT"
```

```
summary(MSFT)
```

```
##      Index              MSFT.Open        MSFT.High         MSFT.Low
##  Min.   :2012-01-03   Min.   : 26.38   Min.   : 26.63   Min.   : 26.26
##  1st Qu.:2013-10-02   1st Qu.: 34.97   1st Qu.: 35.20   1st Qu.: 34.68
##  Median :2015-07-04   Median : 46.95   Median : 47.45   Median : 46.55
##  Mean   :2015-07-02   Mean   : 54.23   Mean   : 54.69   Mean   : 53.75
##  3rd Qu.:2017-03-31   3rd Qu.: 65.39   3rd Qu.: 65.72   3rd Qu.: 64.95
##  Max.   :2018-12-31   Max.   :115.42   Max.   :116.18   Max.   :114.93
##    MSFT.Close      MSFT.Volume         MSFT.Adjusted
##  Min.   : 26.37   Min.   :  7425600   Min.   : 22.16
##  1st Qu.: 34.98   1st Qu.: 23622625   1st Qu.: 30.24
##  Median : 47.01   Median : 31590050   Median : 42.78
##  Mean   : 54.24   Mean   : 35838061   Mean   : 50.54
##  3rd Qu.: 65.41   3rd Qu.: 42966300   3rd Qu.: 62.54
##  Max.   :115.61   Max.   :248428500   Max.   :113.82
```

```
head(MSFT)
```

```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume
## 2012-01-03     26.55     26.96    26.39      26.77    64731500
## 2012-01-04     26.82     27.47    26.78      27.40    80516100
## 2012-01-05     27.38     27.73    27.29      27.68    56081400
## 2012-01-06     27.53     28.19    27.53      28.11    99455500
## 2012-01-09     28.05     28.10    27.72      27.74    59706800
## 2012-01-10     27.93     28.15    27.75      27.84    60014400
##            MSFT.Adjusted
## 2012-01-03      22.15607
## 2012-01-04      22.67749
## 2012-01-05      22.90923
## 2012-01-06      23.26512
## 2012-01-09      22.95889
## 2012-01-10      23.04165
```

```
View(MSFT)
names(MSFT)
```

```
## [1] "MSFT.Open"     "MSFT.High"     "MSFT.Low"      "MSFT.Close"
## [5] "MSFT.Volume"   "MSFT.Adjusted"
```

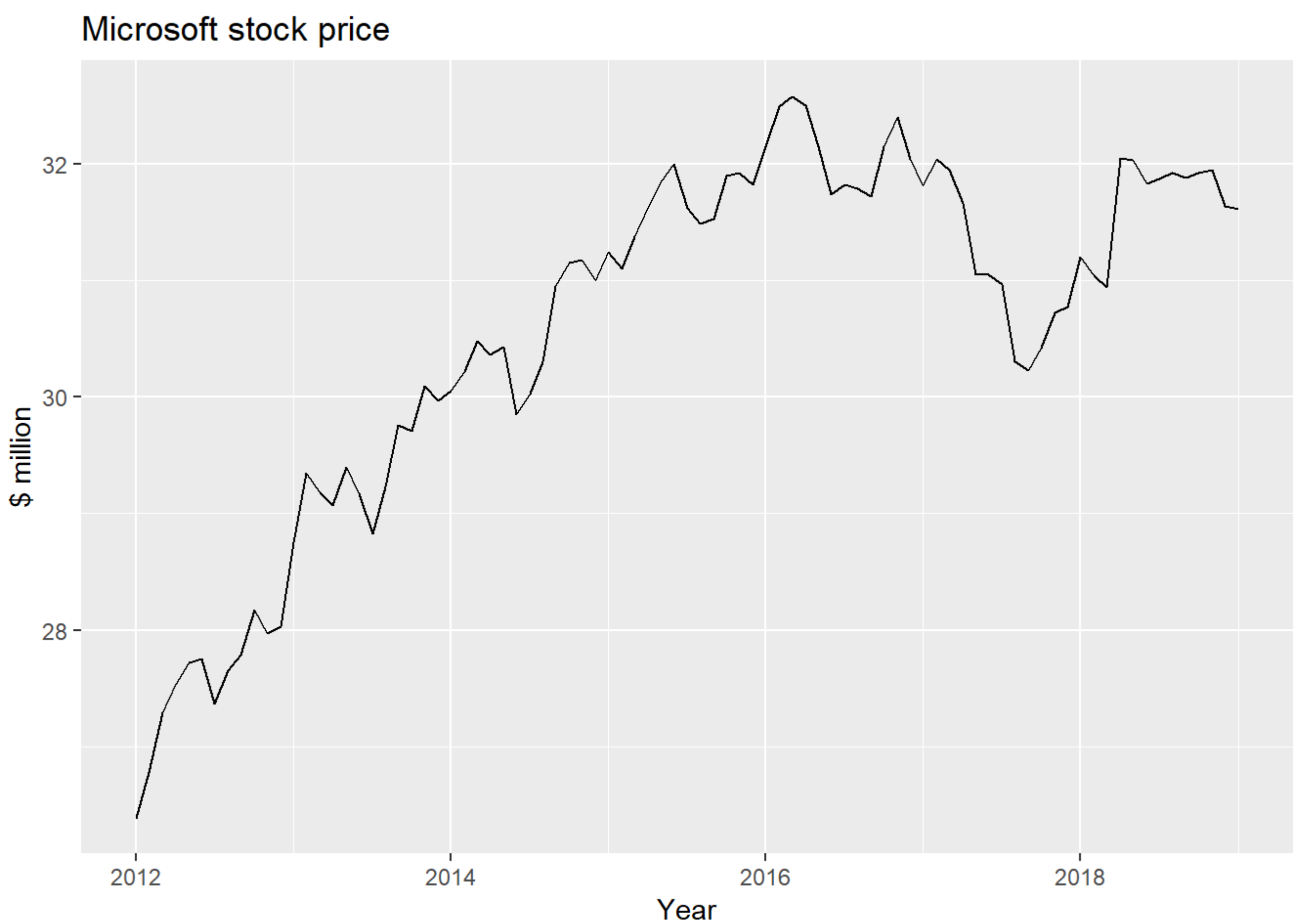
```
data <- ts(MSFT,start=c(2012,1),end=c(2019,01), frequency = 12)
data=data[,3]
View(data)

#Calculating Train and Test data
train_data = window(data,start=c(2012,1), end=c(2016,12))
test_data = window(data,start=c(2017,1), end=c(2018,12))
train_data
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 2012 26.39 26.78 27.29 27.53 27.72 27.75 27.37 27.65 27.79 28.17 27.97
## 2013 28.75 29.35 29.18 29.07 29.40 29.17 28.83 29.23 29.76 29.71 30.09
## 2014 30.05 30.22 30.48 30.36 30.43 29.85 30.03 30.30 30.95 31.15 31.18
## 2015 31.24 31.10 31.38 31.61 31.85 32.00 31.62 31.49 31.53 31.90 31.92
## 2016 32.15 32.49 32.58 32.50 32.15 31.74 31.82 31.79 31.72 32.15 32.40
##        Dec
## 2012 28.03
## 2013 29.97
## 2014 31.00
## 2015 31.82
## 2016 32.04
```

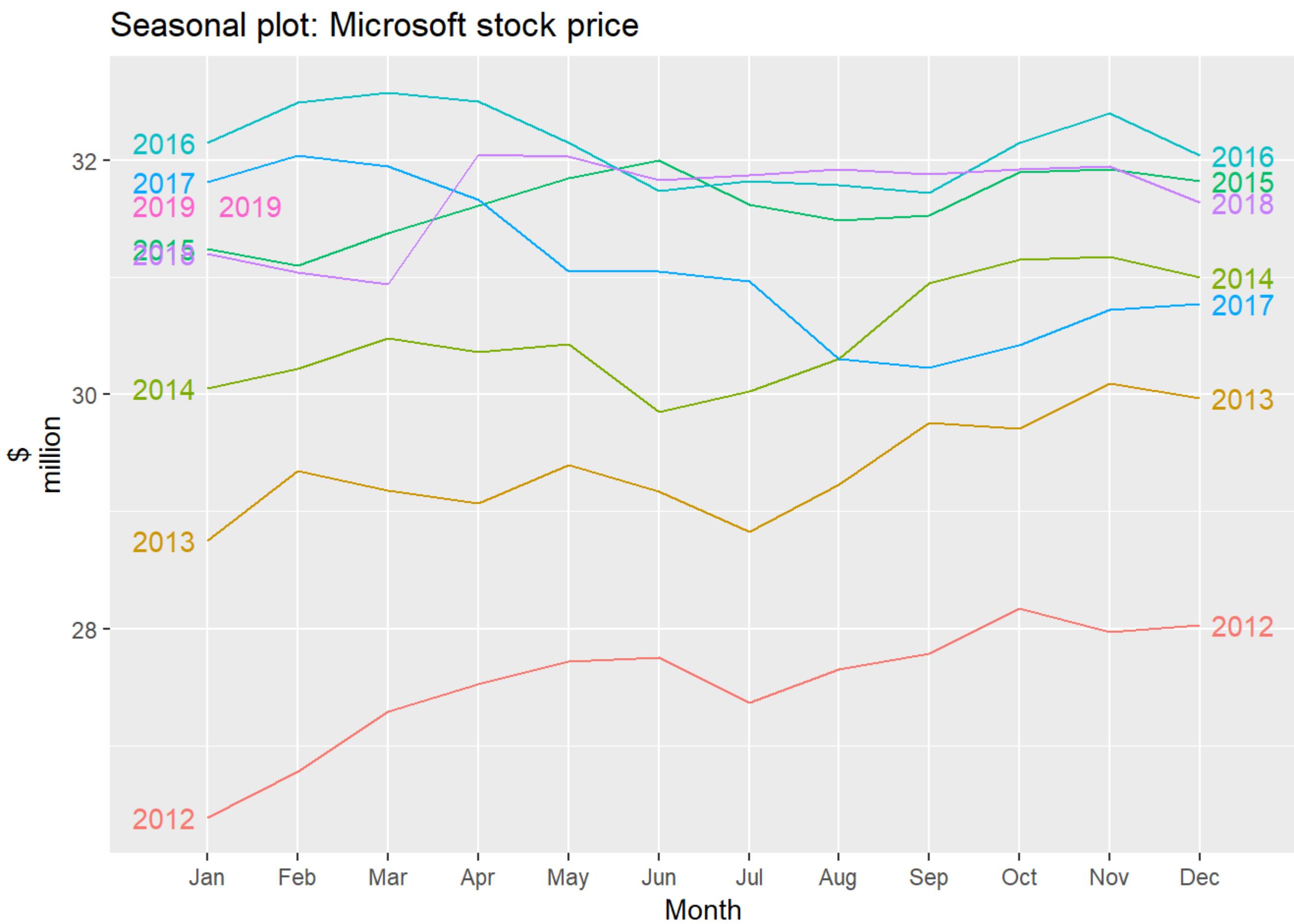
```
test_data
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 2017 31.81 32.04 31.95 31.66 31.05 31.05 30.97 30.30 30.23 30.42 30.72
## 2018 31.20 31.04 30.94 32.05 32.03 31.83 31.87 31.92 31.88 31.92 31.95
##        Dec
## 2017 30.77
## 2018 31.64
```

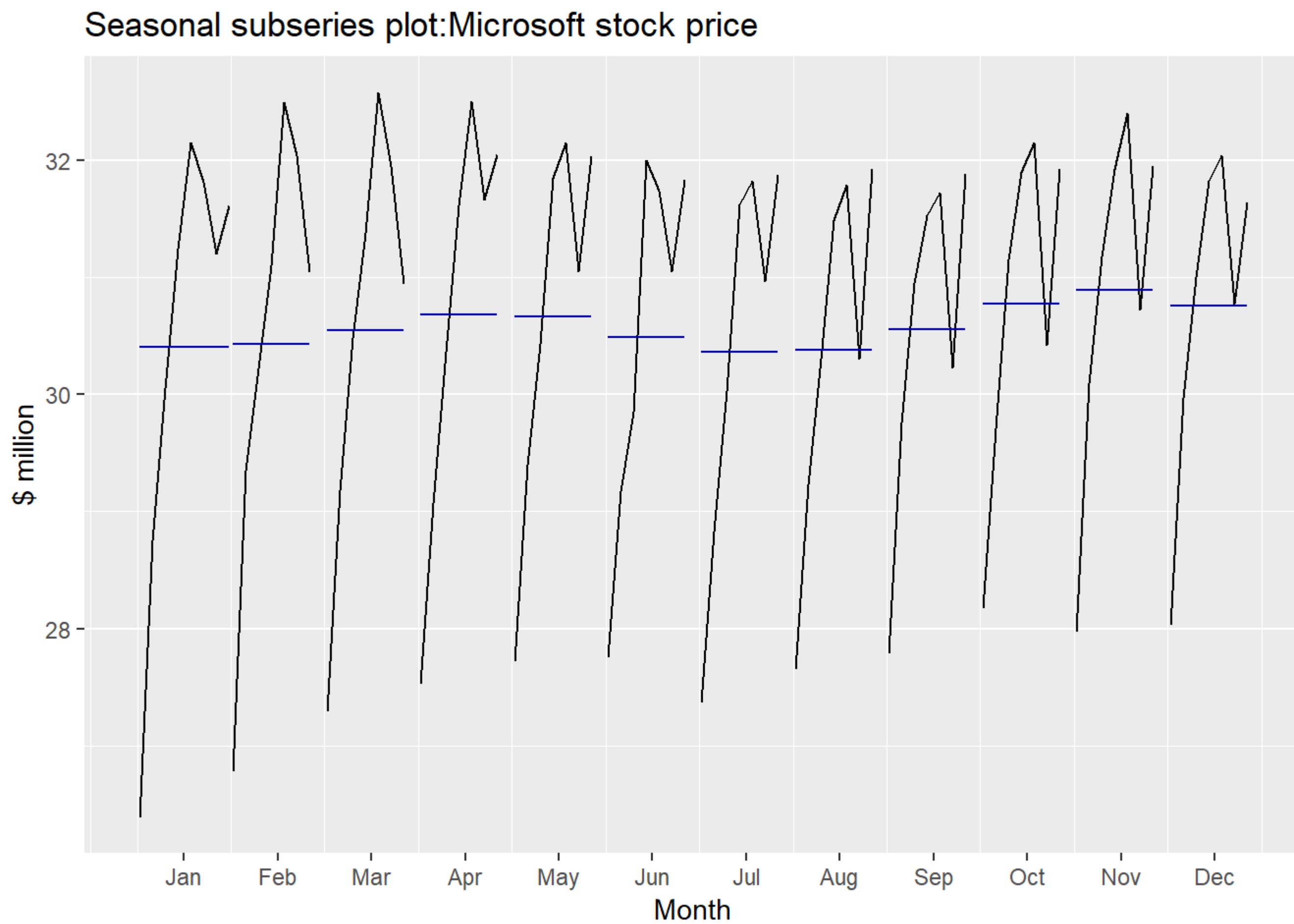
```
#DataPlot
autoplot(data) + ggtitle("Microsoft stock price") + ylab("$ million") + xlab("Year")
```




```
#Seasonal DataPlot
ggseasonplot(data, year.labels=TRUE, year.labels.left=TRUE) + ylab("$
million") + ggtitle("Seasonal plot: Microsoft stock price")
```




```
#Seasonal subseries plot
ggsubseriesplot(data) + ylab("$ million") + ggtitle("Seasonal subseries plot:Microsoft stock pri
ce")
```



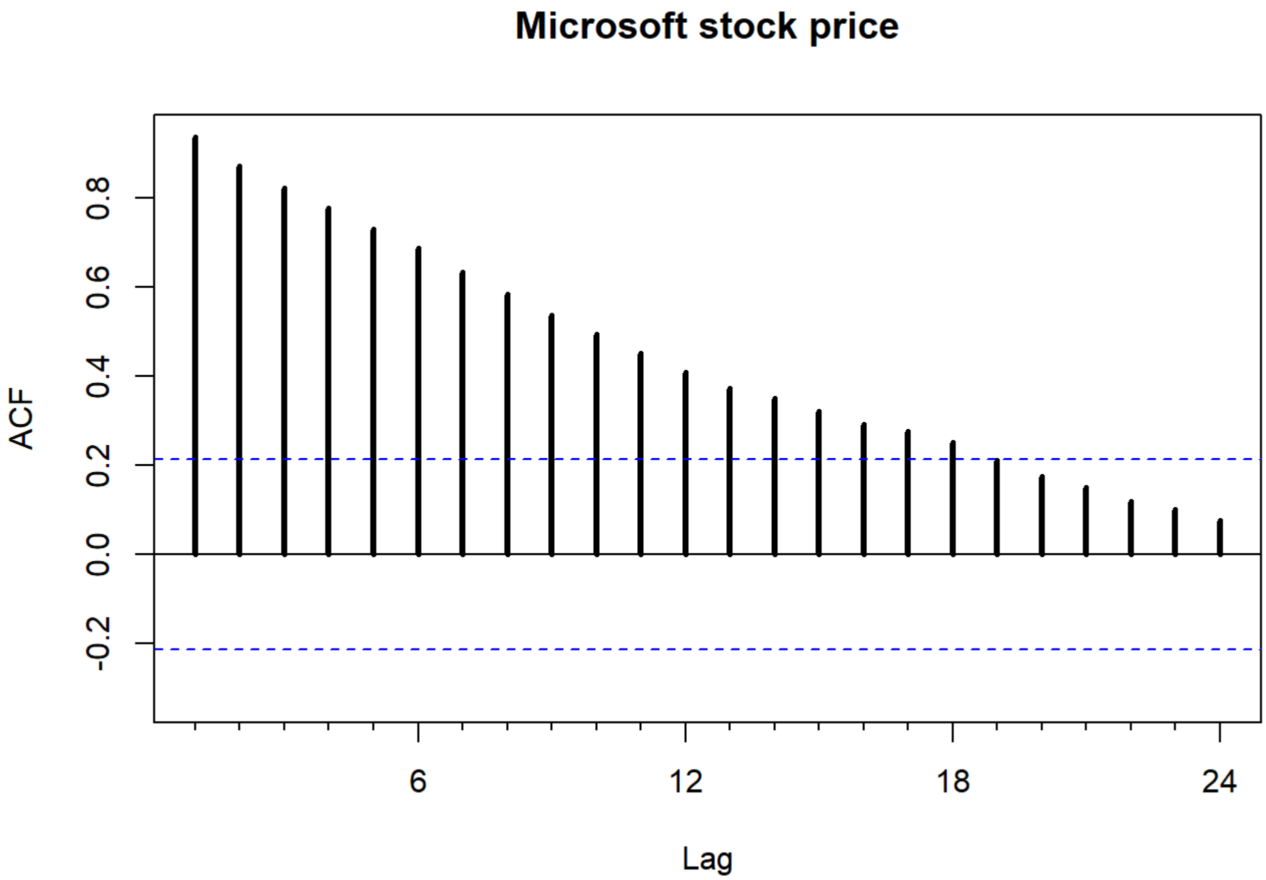

```
#Test for Stationary
Box.test(data, lag = 20, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 584.91, df = 20, p-value < 2.2e-16
```

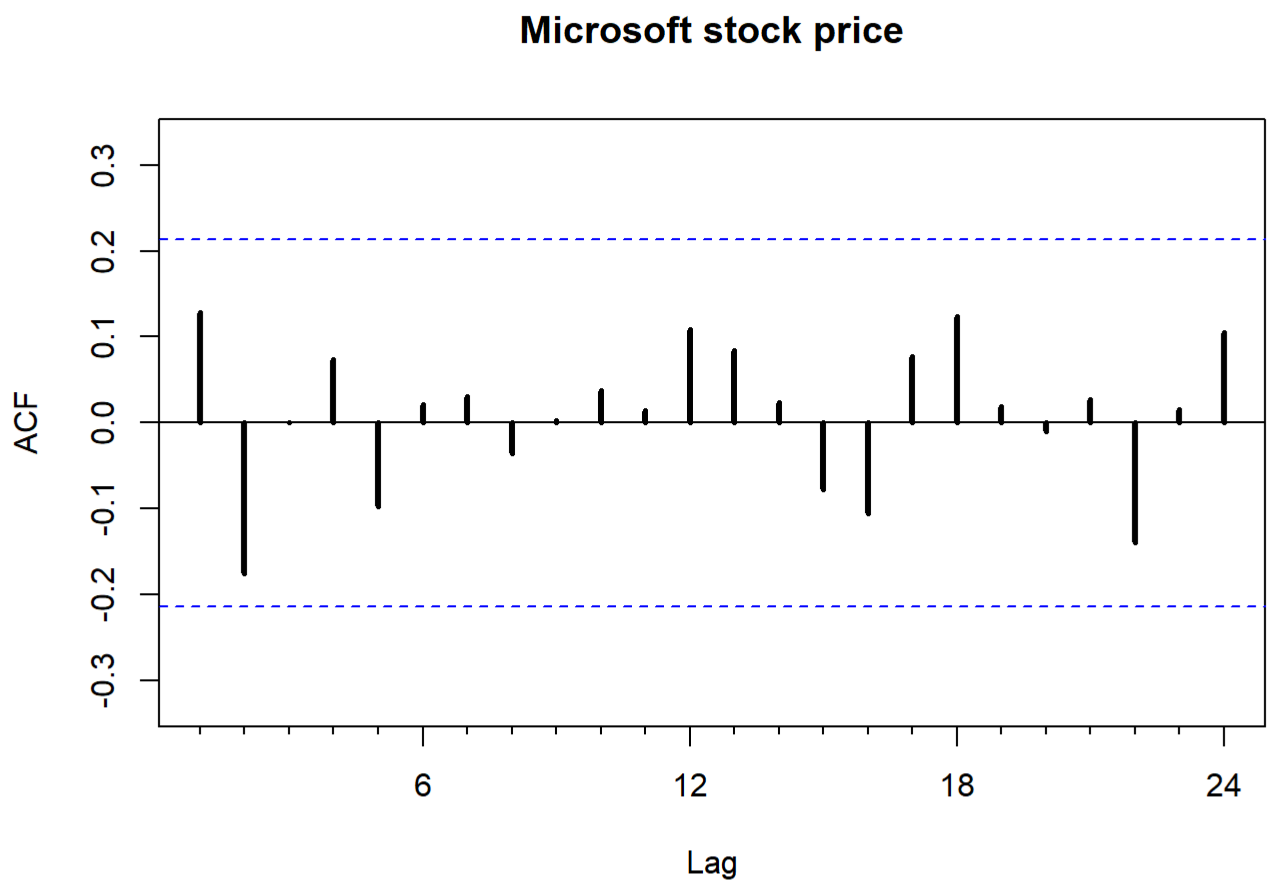
```
#Adf test
adf.test(data)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -1.5314, Lag order = 4, p-value = 0.7682
## alternative hypothesis: stationary
```

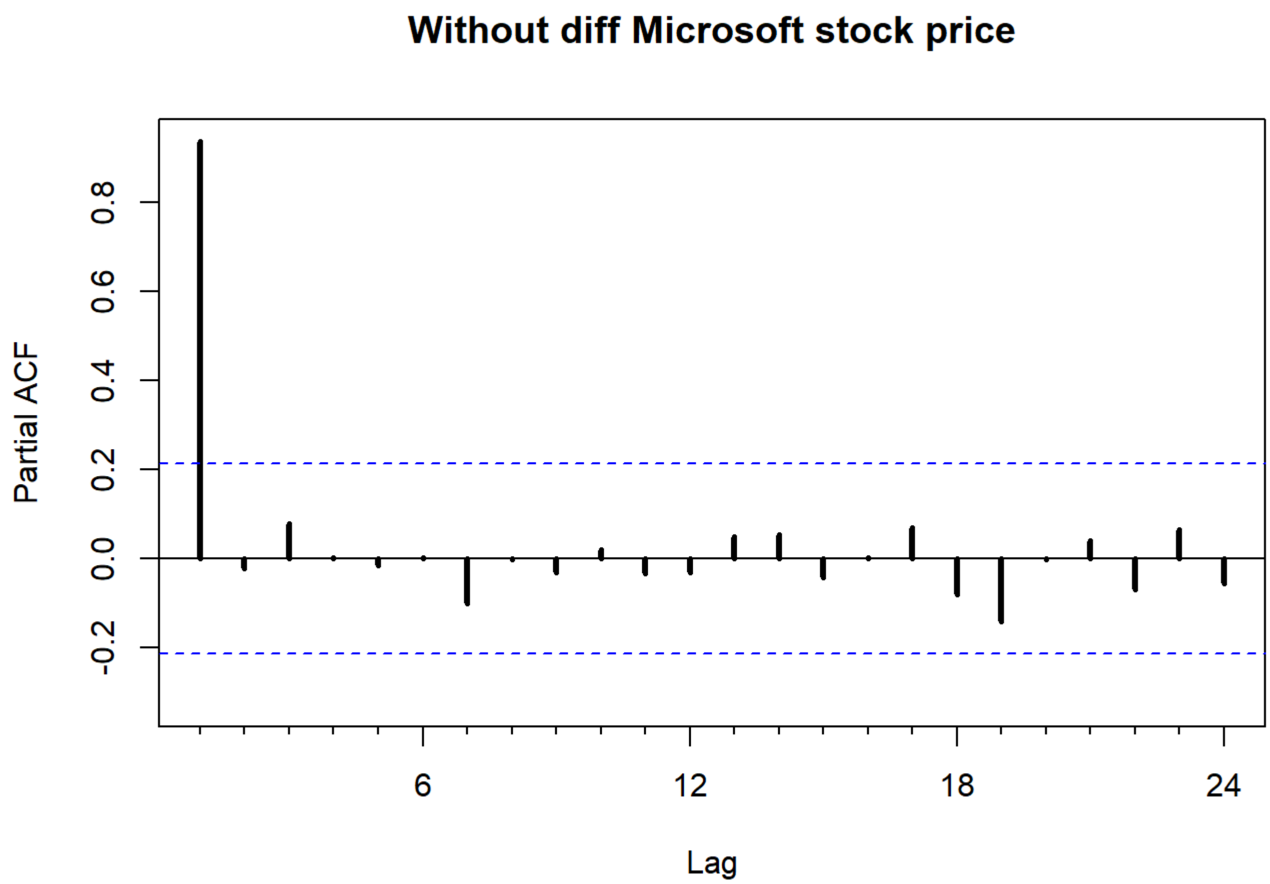
```
#Autocorelation Function
Acf(data, lwd=3,main="Microsoft stock price") #By seeing the plot, we can make out, it is not th
e stationary hence, we are using differencing
```



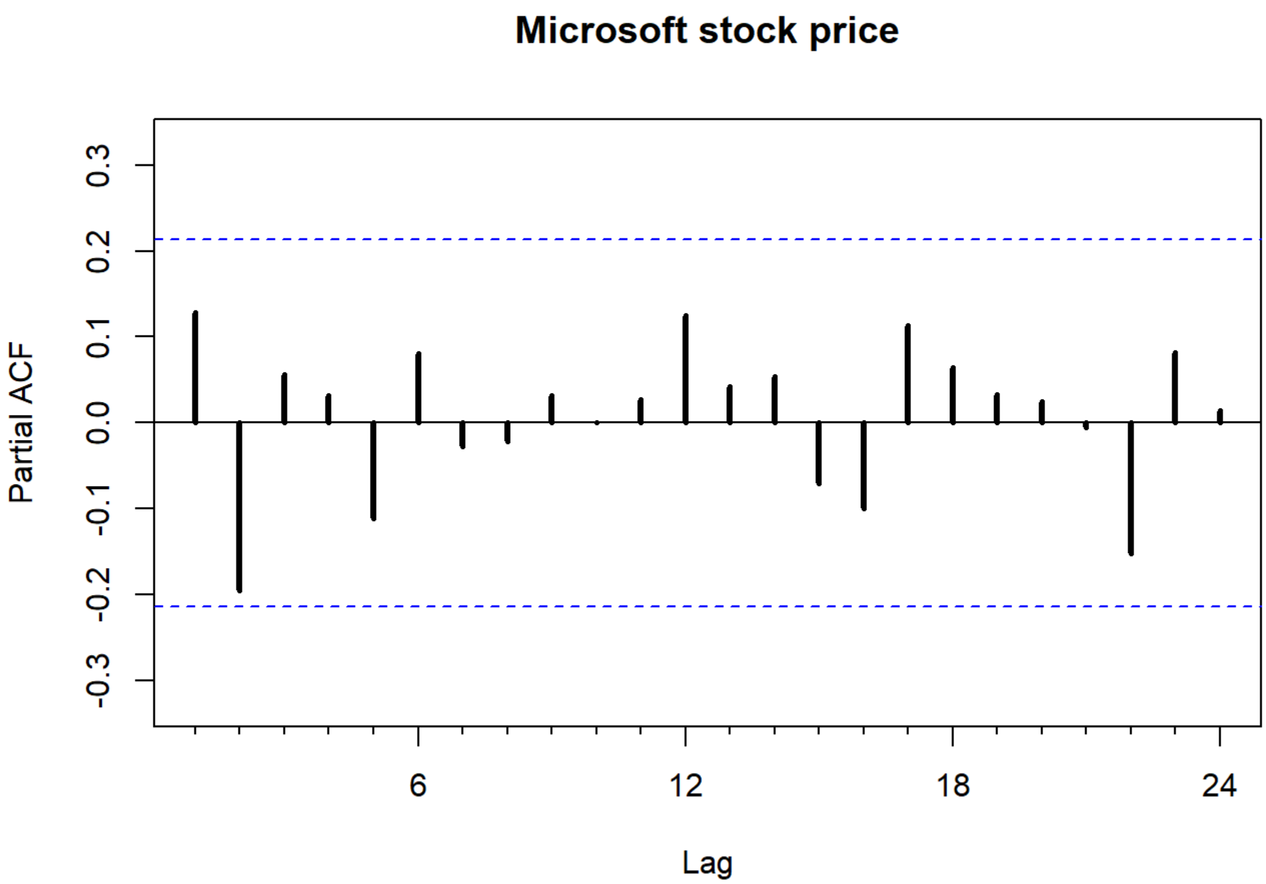

```
Acf(diff(data), lwd=3,main="Microsoft stock price")
```




```
#Partial ACF
Pacf(data, lwd=3,main="Without diff Microsoft stock price")
```




```
Pacf(diff(data), lwd=3,main="Microsoft stock price")
```

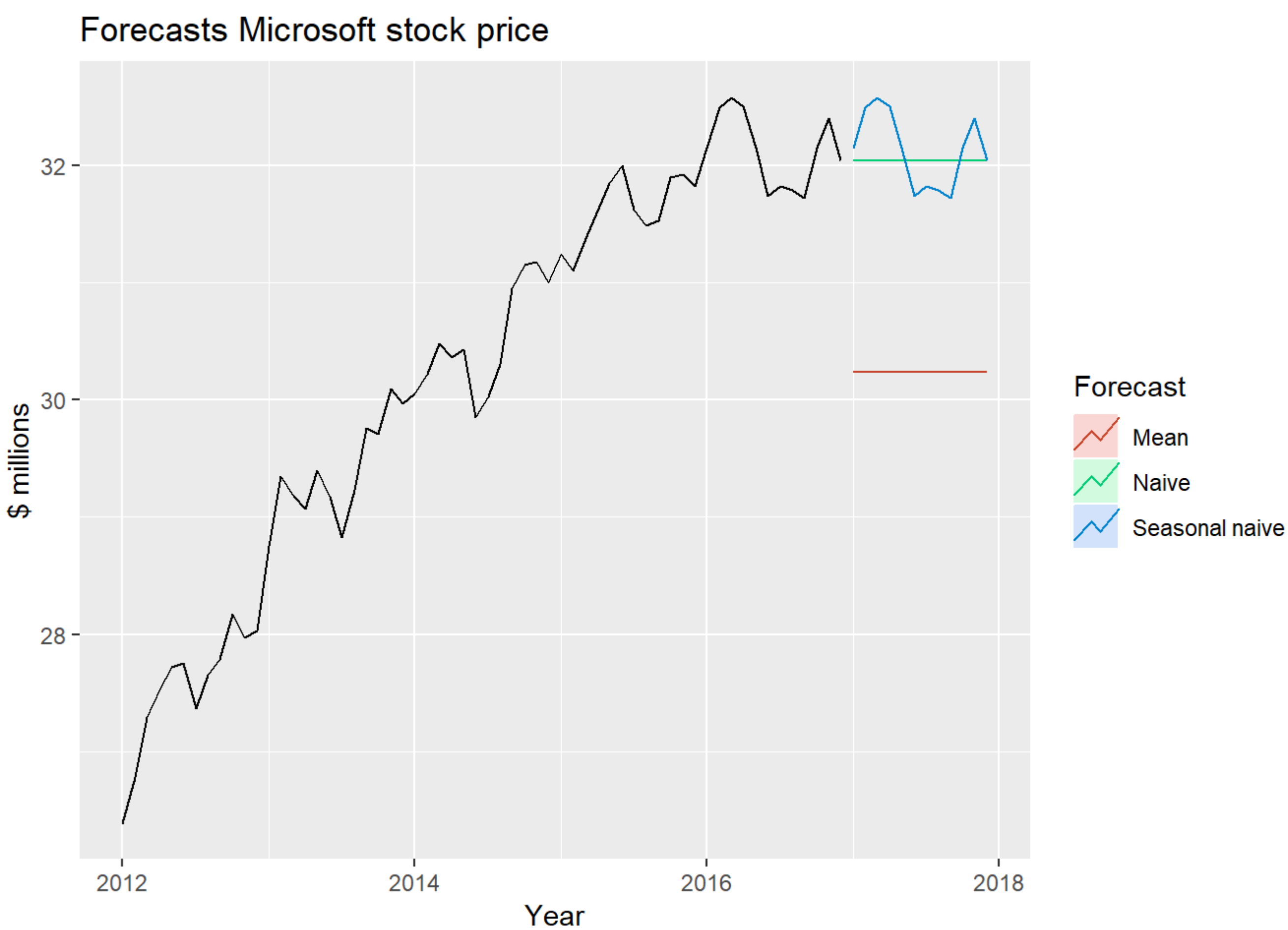



```
adf.test(diff(data)) #here pvalue is lesss than 0.05, hence series is stationary
```

```
## Warning in adf.test(diff(data)): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(data)
## Dickey-Fuller = -4.7477, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

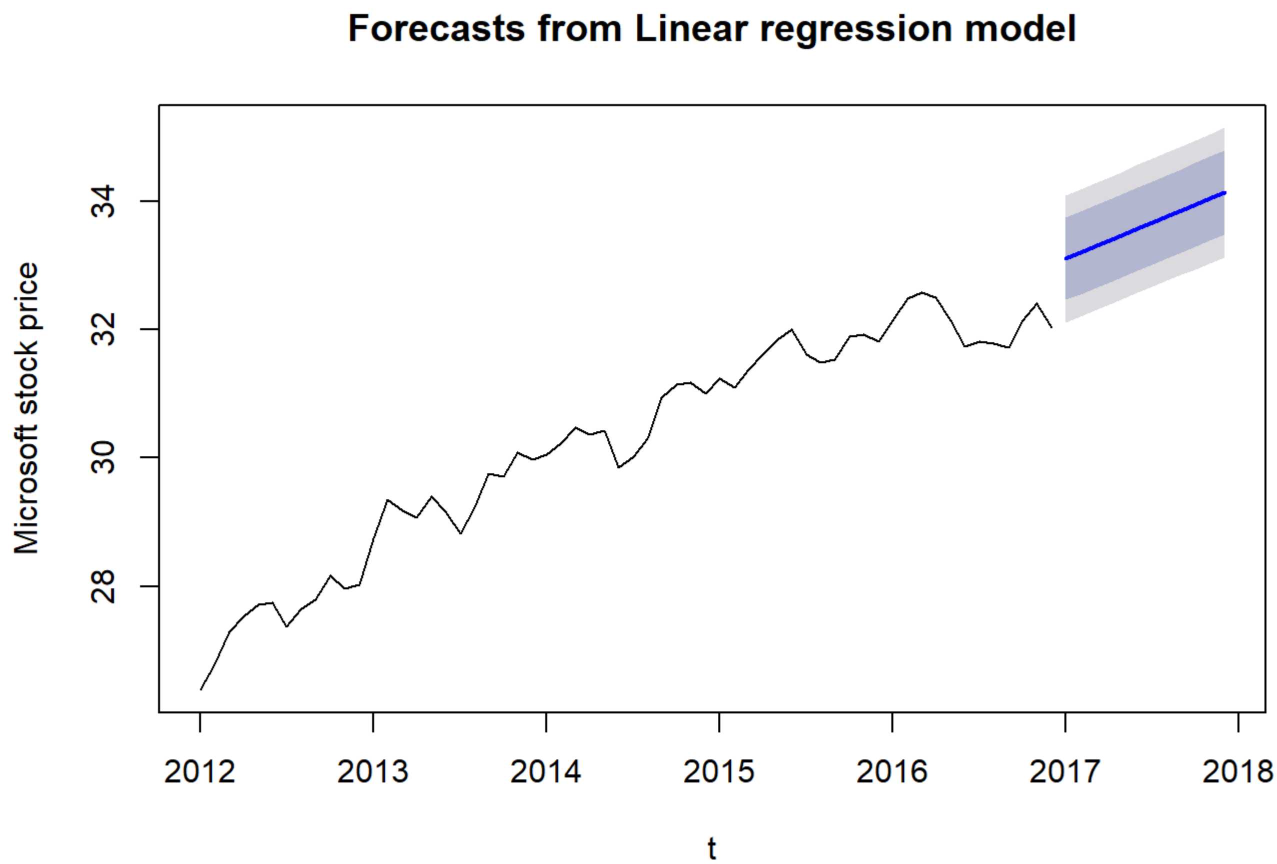
```
#Mean, Naive, Seasonal Naive
fit.mean=meanf(train_data,h=12)
fit.naive=naive(train_data,h=12)
fit.snaive=snaive(train_data,h=12)
autoplot(train_data) +
  autolayer(meanf(train_data, h=12),
            series="Mean", PI=FALSE) +
  autolayer(naive(train_data, h=12),
            series="Naive", PI=FALSE) +
  autolayer(snaive(train_data, h=12),
            series="Seasonal naive", PI=FALSE) +
  ggtitle("Forecasts Microsoft stock price") +
  xlab("Year") + ylab("$ millions") +
  guides(colour=guide_legend(title="Forecast"))
```




```
#Linear Trend
linear_reg <- tslm(train_data ~ trend)
fit.tslm1=forecast(linear_reg, h=12)
summary(fit.tslm1)
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = train_data ~ trend)
##
## Coefficients:
## (Intercept)        trend
##    27.36347      0.09406
##
##
## Error measures:
##                        ME      RMSE       MAE         MPE     MAPE
## Training set 1.775977e-16 0.4690483 0.4000656 -0.02727281 1.327682
##                   MASE     ACF1
## Training set 0.3453182 0.740629
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       33.10120 32.46198 33.74042 32.11414 34.08825
## Feb 2017       33.19526 32.55501 33.83551 32.20661 34.18390
## Mar 2017       33.28932 32.64801 33.93063 32.29904 34.27960
## Apr 2017       33.38338 32.74098 34.02579 32.39141 34.37535
## May 2017       33.47744 32.83392 34.12097 32.48374 34.47115
## Jun 2017       33.57150 32.92682 34.21619 32.57601 34.56699
## Jul 2017       33.66557 33.01970 34.31143 32.66824 34.66289
## Aug 2017       33.75963 33.11254 34.40671 32.76043 34.75883
## Sep 2017       33.85369 33.20536 34.50202 32.85256 34.85481
## Oct 2017       33.94775 33.29814 34.59736 32.94465 34.95085
## Nov 2017       34.04181 33.39089 34.69273 33.03669 35.04693
## Dec 2017       34.13587 33.48362 34.78812 33.12869 35.14305
```

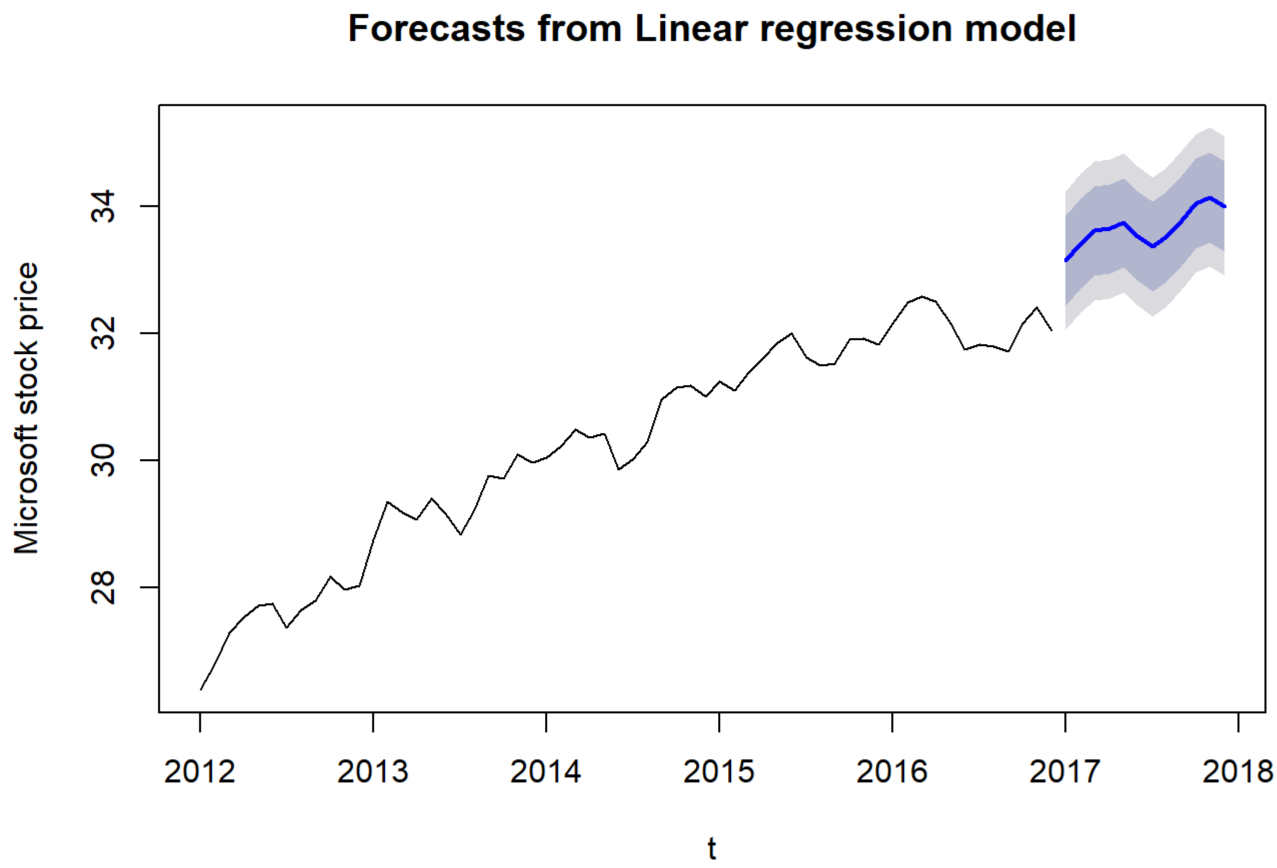
```
plot(fit.tslm1, ylab="Microsoft stock price",
     xlab="t")
```



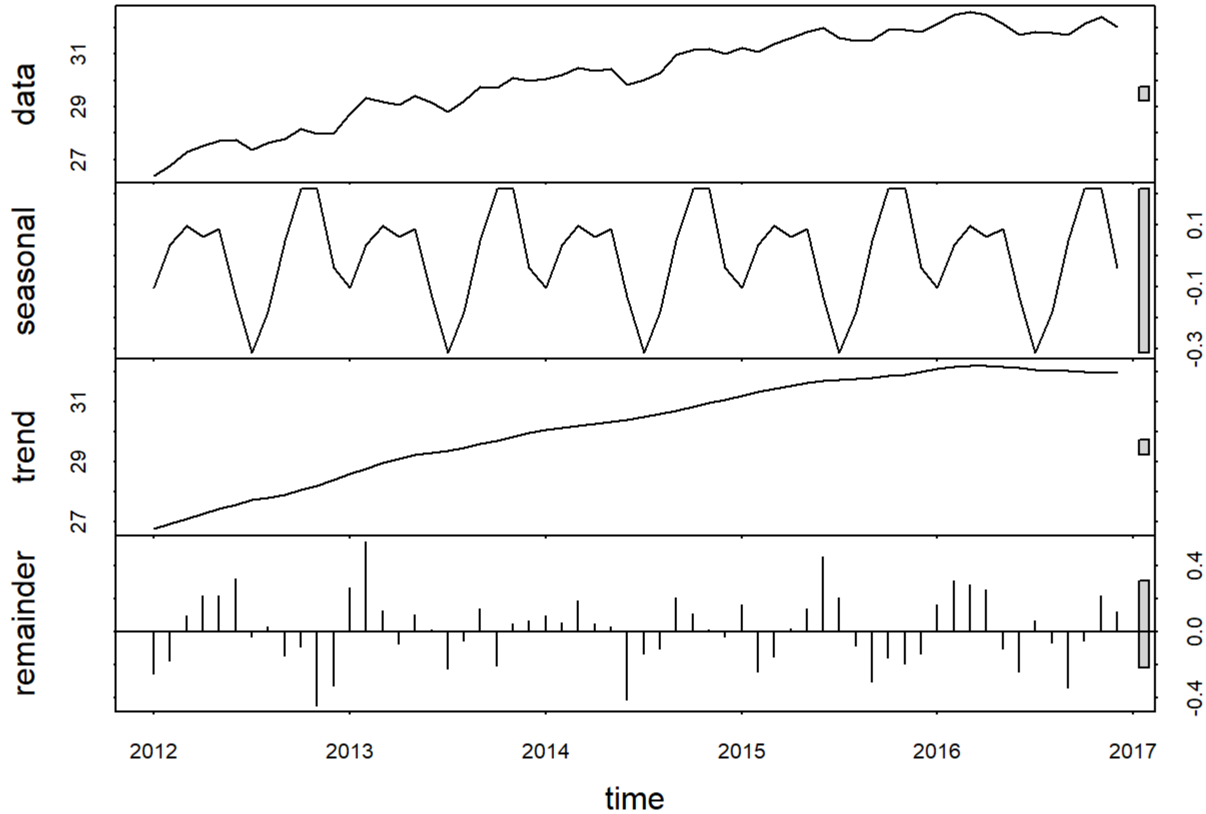

```r
#Season + Trend
linear_season <- tslm(train_data ~ trend + season)
fit.tslm2=forecast(linear_season, h=12)
summary(fit.tslm2)
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = train_data ~ trend + season)
##
## Coefficients:
## (Intercept)        trend      season2      season3      season4
##    27.33527      0.09523      0.17677      0.27554      0.21231
##     season5      season6      season7      season8      season9
##     0.21308     -0.09015     -0.35337     -0.29060     -0.12783
##    season10     season11     season12
##     0.04294      0.04371     -0.19152
##
##
## Error measures:
##                         ME      RMSE       MAE         MPE     MAPE
## Training set -1.184437e-16 0.4256142 0.3455998 -0.02311721 1.149542
##                   MASE      ACF1
## Training set 0.2983058 0.7844564
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       33.14425 32.43845 33.85005 32.05188 34.23662
## Feb 2017       33.41625 32.71045 34.12205 32.32388 34.50862
## Mar 2017       33.61025 32.90445 34.31605 32.51788 34.70262
## Apr 2017       33.64225 32.93645 34.34805 32.54988 34.73462
## May 2017       33.73825 33.03245 34.44405 32.64588 34.83062
## Jun 2017       33.53025 32.82445 34.23605 32.43788 34.62262
## Jul 2017       33.36225 32.65645 34.06805 32.26988 34.45462
## Aug 2017       33.52025 32.81445 34.22605 32.42788 34.61262
## Sep 2017       33.77825 33.07245 34.48405 32.68588 34.87062
## Oct 2017       34.04425 33.33845 34.75005 32.95188 35.13662
## Nov 2017       34.14025 33.43445 34.84605 33.04788 35.23262
## Dec 2017       34.00025 33.29445 34.70605 32.90788 35.09262
```

```
plot(fit.tslm2, ylab="Microsoft stock price",
     xlab="t")
```



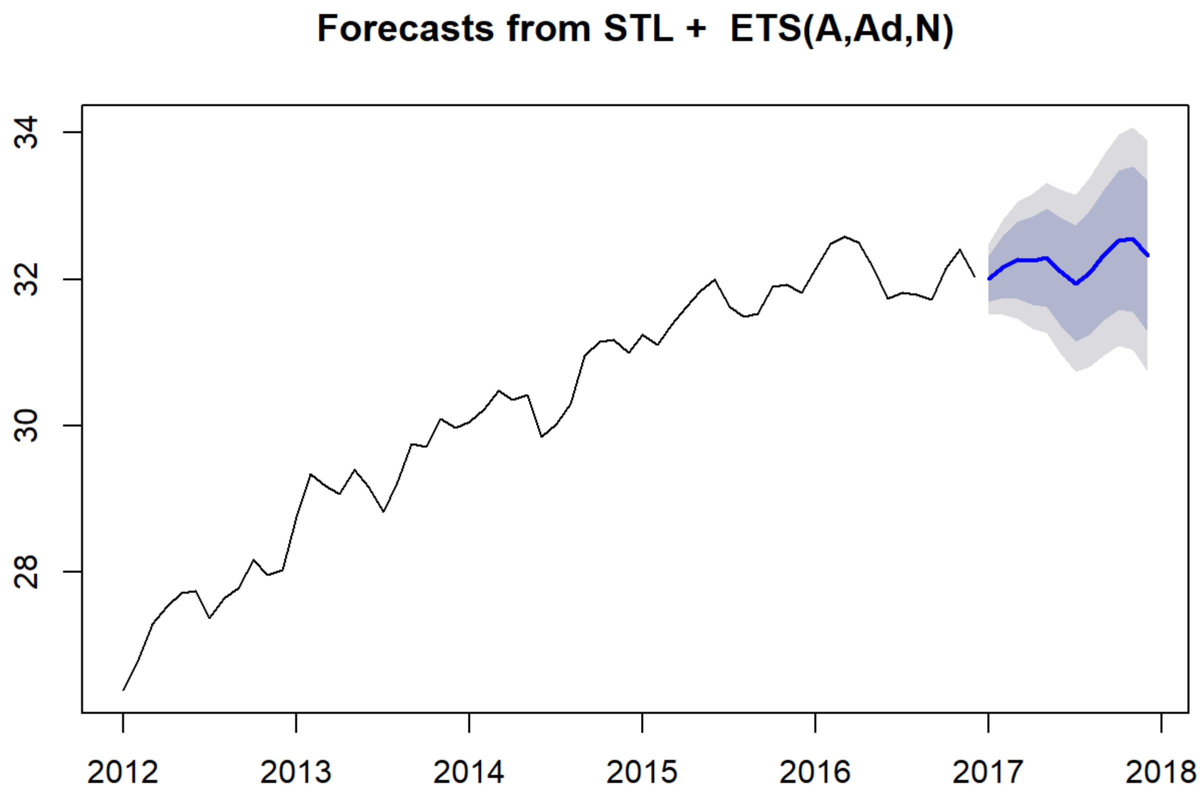

```
#STL decomposition
stl_decomp <- stl(train_data, t.window=12, s.window="periodic")
plot(stl_decomp)
```



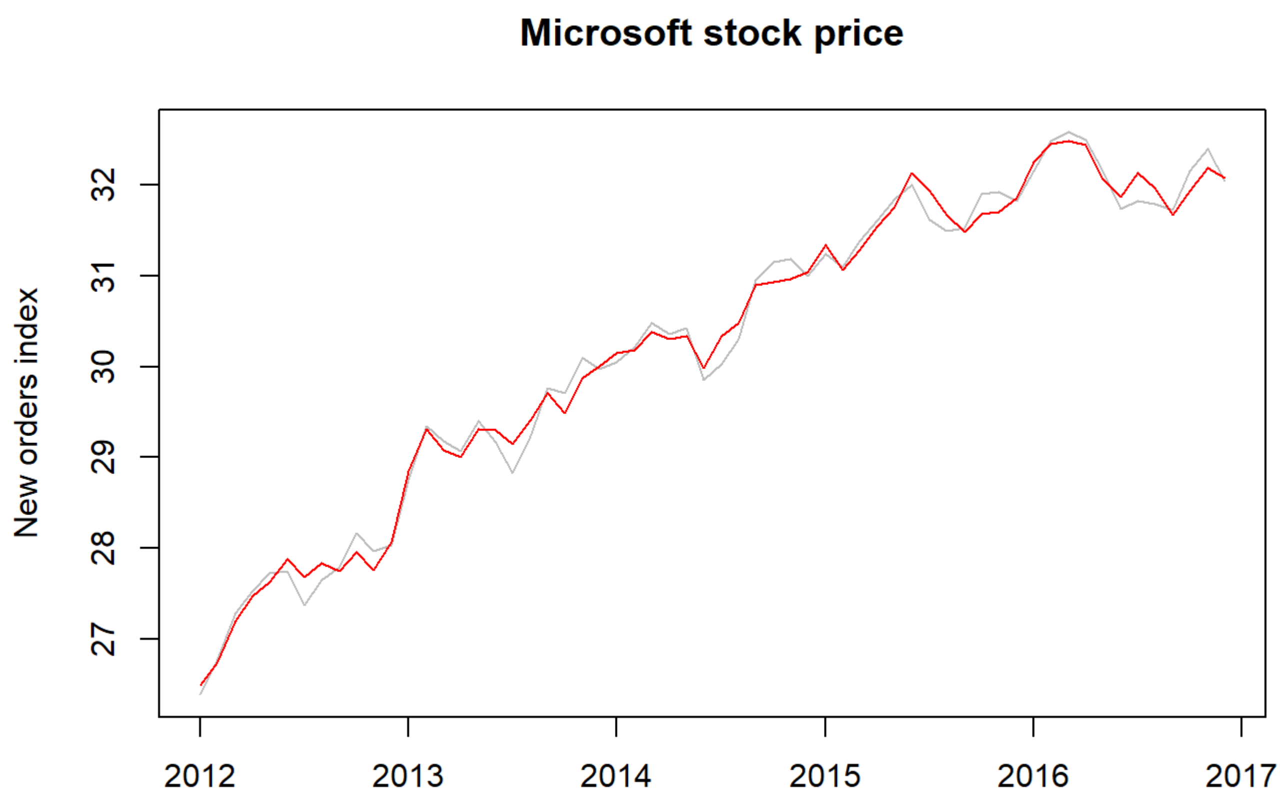

```
fit.stl <- forecast(stl_decomp,h=12)
summary(fit.stl)
```

```
##
## Forecast method: STL +  ETS(A,Ad,N)
##
## Model Information:
## ETS(A,Ad,N)
##
## Call:
##  ets(y = x, model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.9483
##     beta  = 1e-04
##     phi   = 0.9647
##
##   Initial states:
##     l = 26.3104
##     b = 0.244
##
##   sigma:  0.2443
##
##      AIC     AICc      BIC
## 83.30105 84.88596 95.86712
##
## Error measures:
##                        ME      RMSE       MAE        MPE      MAPE
## Training set -0.002177407 0.2338684 0.1920148 -0.0112109 0.6355899
##                   MASE       ACF1
## Training set 0.1657383 0.05932208
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       32.00689 31.69385 32.31993 31.52814 32.48565
## Feb 2017       32.17339 31.74197 32.60482 31.51358 32.83320
## Mar 2017       32.26097 31.73725 32.78468 31.46001 33.06192
## Apr 2017       32.24838 31.64634 32.85041 31.32765 33.16911
## May 2017       32.29893 31.62764 32.97022 31.27228 33.32558
## Jun 2017       32.10160 31.36755 32.83565 30.97896 33.22423
## Jul 2017       31.94346 31.15160 32.73533 30.73242 33.15451
## Aug 2017       32.09445 31.24871 32.94019 30.80101 33.38789
## Sep 2017       32.34469 31.44830 33.24108 30.97378 33.71560
## Oct 2017       32.53347 31.58913 33.47780 31.08923 33.97771
## Nov 2017       32.55155 31.56158 33.54152 31.03753 34.06558
## Dec 2017       32.31671 31.28311 33.35030 30.73596 33.89746
```

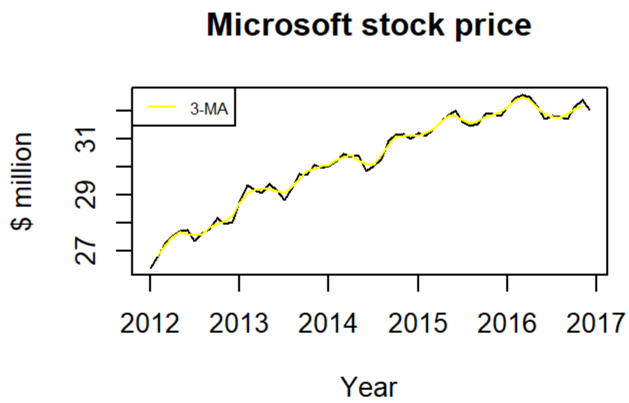
```
plot(fit.stl)
```

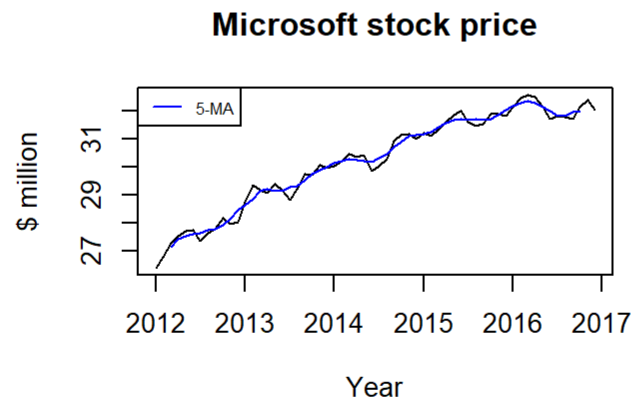


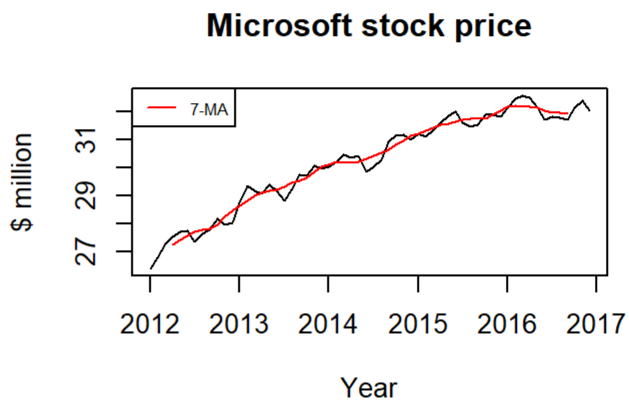

```
#Seasonally adjusted data
plot(train_data, col="grey",
     main="Microsoft stock price",
     xlab="", ylab="New orders index")
lines(seasadj(stl_decomp),col="red",ylab="Seasonally adjusted")
```

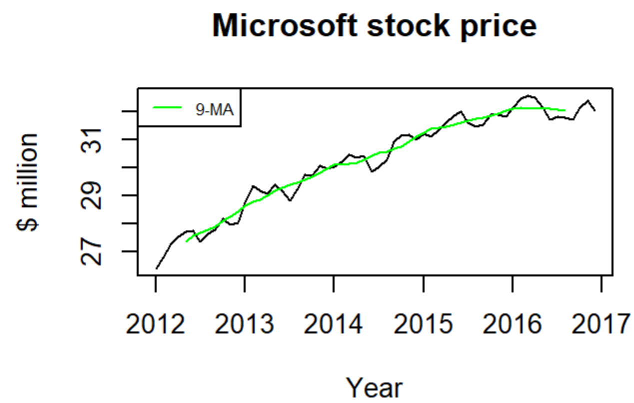



```
#Moving Average
par(mfrow=c(2,2))

plot(train_data, main="Microsoft stock price",
     ylab="$ million", xlab="Year")
lines(ma(train_data,3),col="yellow")
legend("topleft",lty=1,col="yellow",cex=0.6,
       legend=c("3-MA"))
plot(train_data, main="Microsoft stock price",
     ylab="$ million", xlab="Year")
lines(ma(train_data,5),col="blue")
legend("topleft",lty=1,col="blue",cex=0.6,
       legend=c("5-MA"))
plot(train_data, main="Microsoft stock price",
     ylab="$ million", xlab="Year")
lines(ma(train_data,7),col="red")
legend("topleft",lty=1,col="red",cex=0.6,
       legend=c("7-MA"))
plot(train_data, main="Microsoft stock price",
     ylab="$ million", xlab="Year")
lines(ma(train_data,9),col="green")
legend("topleft",lty=1,col="green",cex=0.6,
       legend=c("9-MA"))
```










```
#SES
fit.ses <- ses(train_data, h = 12)
fit.ses <- forecast(fit.ses)
summary(fit.ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = train_data, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 26.3931
##
##   sigma:  0.3027
##
##      AIC     AICc      BIC
## 106.2177 106.6463 112.5007
##
## Error measures:
##                      ME      RMSE      MAE       MPE      MAPE      MASE
## Training set 0.09412525 0.2975941 0.245221 0.3181611 0.8173299 0.2116635
##                   ACF1
## Training set 0.1088122
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       32.04004 31.65214 32.42794 31.44679 32.63328
## Feb 2017       32.04004 31.49149 32.58859 31.20110 32.87897
## Mar 2017       32.04004 31.36822 32.71186 31.01258 33.06750
## Apr 2017       32.04004 31.26429 32.81578 30.85364 33.22644
## May 2017       32.04004 31.17273 32.90734 30.71361 33.36647
## Jun 2017       32.04004 31.08995 32.99012 30.58701 33.49306
## Jul 2017       32.04004 31.01383 33.06624 30.47059 33.60948
## Aug 2017       32.04004 30.94298 33.13709 30.36223 33.71784
## Sep 2017       32.04004 30.87643 33.20364 30.26046 33.81961
## Oct 2017       32.04004 30.81349 33.26658 30.16420 33.91587
## Nov 2017       32.04004 30.75363 33.32645 30.07265 34.00743
## Dec 2017       32.04004 30.69643 33.38365 29.98516 34.09491
```

```
plot(fit.ses)

fit1 <-ses(train_data, alpha=0.2, initial="simple", h=3)
fit2 <-ses(train_data, alpha=0.6, initial="simple", h=3)
fit3 <-ses(train_data, h=3)
plot(fit1,main="Microsoft stock price", ylab="$
     (millions)", xlab="Year", fcol="white", type="o")
lines(fitted(fit1), col="blue", type="o")
lines(fitted(fit2), col="red", type="o")
lines(fitted(fit3), col="green", type="o")
lines(fit1$mean, col="blue", type="o")

lines(fit2$mean, col="red", type="o")


lines(fit3$mean, col="green", type="o")
legend("topleft",lty=1, col=c(1,"blue","red","green"),
       c("data", expression(lambda == 0.2), expression(lambda == 0.6),
         expression(lambda == 0.89)),pch=1)

#Holt's Linear trend
#The SES model usually doesnâ□□t work well when the data shows a long term trend. This method u
ses two smoothing techniques instead of just the alpha one
fit.hlinear <- holt(train_data, h=3)
fit.hlinear <- forecast(fit.hlinear)
summary(fit.hlinear)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = train_data, h = 3)
##
##   Smoothing parameters:
##     alpha = 0.9871
##     beta  = 1e-04
##
##   Initial states:
##     l = 26.4354
##     b = 0.0866
##
##   sigma:  0.293
##
##      AIC     AICc      BIC
## 104.2296 105.3407 114.7013
##
## Error measures:
##                       ME      RMSE       MAE        MPE      MAPE
## Training set 0.006896459 0.2831083 0.2326707 0.02840631 0.7752235
##                   MASE      ACF1
## Training set 0.2008306 0.1152219
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       32.13237 31.75682 32.50792 31.55801 32.70673
## Feb 2017       32.21899 31.69128 32.74671 31.41192 33.02607
## Mar 2017       32.30562 31.66067 32.95057 31.31925 33.29198
```

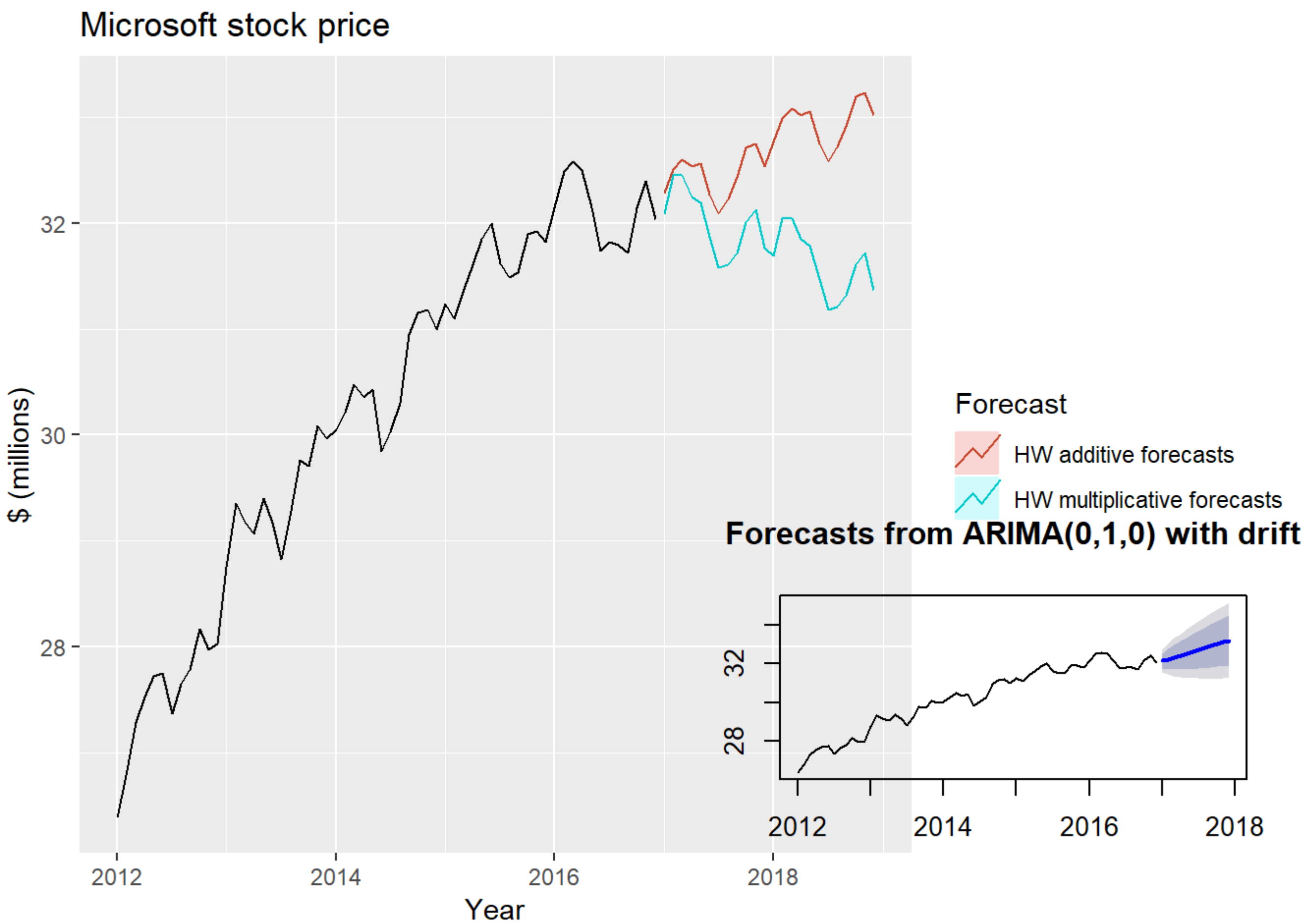
```
plot(fit.hlinear, main = "Holt's Linear Trend")
lines(train_data)


#Holt's Winter Additive and Multiplicative
fit1_add <- hw(train_data,seasonal="additive")
fit1_add <- forecast(fit1_add)
fit2_multi <- hw(train_data,seasonal="multiplicative")
fit2_multi <- forecast(fit2_multi)


autoplot(train_data) +
  autolayer(fit1_add, series="HW additive forecasts", PI=FALSE) +
  autolayer(fit2_multi, series="HW multiplicative forecasts", PI=FALSE) +
  xlab("Year") +
  ylab("$ (millions)") +
  ggtitle("Microsoft stock price") +
  guides(colour=guide_legend(title="Forecast"))



#Auto ARIMA
#One of the most used Time Series analysis models. We will use the auto.arima() function to find
the values automatically
y.arima <- auto.arima(train_data)
summary(y.arima)
```

```
## Series: train_data
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##       0.0958
## s.e.  0.0370
##
## sigma^2 estimated as 0.0823:  log likelihood=-9.53
## AIC=23.07   AICc=23.28   BIC=27.22
##
## Training set error measures:
##                        ME      RMSE       MAE         MPE      MAPE
## Training set 0.0004382371 0.2820501 0.2305344 0.008244221 0.7671619
##                   MASE      ACF1
## Training set 0.1989867 0.1164604
```

```
fit.arima <- forecast(y.arima, h=12)
summary(fit.arima)
```

```
##
## Forecast method: ARIMA(0,1,0) with drift
##
## Model Information:
## Series: train_data
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##       0.0958
## s.e.  0.0370
##
## sigma^2 estimated as 0.0823:  log likelihood=-9.53
## AIC=23.07   AICc=23.28   BIC=27.22
##
## Error measures:
##                        ME      RMSE       MAE         MPE      MAPE
## Training set 0.0004382371 0.2820501 0.2305344 0.008244221 0.7671619
##                   MASE      ACF1
## Training set 0.1989867 0.1164604
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       32.13576 31.76812 32.50340 31.57351 32.69802
## Feb 2017       32.23153 31.71160 32.75145 31.43637 33.02668
## Mar 2017       32.32729 31.69052 32.96406 31.35343 33.30115
## Apr 2017       32.42305 31.68777 33.15833 31.29853 33.54757
## May 2017       32.51881 31.69674 33.34089 31.26157 33.77606
## Jun 2017       32.61458 31.71404 33.51511 31.23733 33.99182
## Jul 2017       32.71034 31.73765 33.68303 31.22274 34.19794
## Aug 2017       32.80610 31.76626 33.84595 31.21580 34.39641
## Sep 2017       32.90187 31.79894 34.00479 31.21509 34.58864
## Oct 2017       32.99763 31.83505 34.16021 31.21961 34.77565
## Nov 2017       33.09339 31.87406 34.31272 31.22859 34.95819
## Dec 2017       33.18915 31.91561 34.46270 31.24143 35.13687
```

```
plot(fit.arima)
```




```
#Comparison between models
#To see the whole picture of our analysis, now we will display all the accuracies of each model
 we have created so far
a.mean=accuracy(fit.mean,test_data)
a.naive=accuracy(fit.naive,test_data)
a.snaive=accuracy(fit.snaive,test_data)
a.linear=accuracy(fit.tslm1,test_data)
a.linear_season=accuracy(fit.tslm2,test_data)
#a.ets=accuracy(fit.ets_forecast,test_data)
a.ses=accuracy(fit.ses, test_data)
a.stl=accuracy(fit.stl, test_data)
a.holt=accuracy(fit.hlinear, test_data)
a.multi=accuracy(fit1_add, test_data)
a.add=accuracy(fit2_multi, test_data)
a.arima=accuracy(fit.arima, test_data)


a.table<-rbind(a.mean, a.naive, a.snaive, a.linear, a.linear_season, a.ses, a.stl, a.holt, a.ad
d, a.multi, a.arima)
a.table
```

```
##                         ME      RMSE       MAE          MPE      MAPE
## Training set -5.323234e-16 1.6951456 1.4468447 -0.325461790 4.8658328
## Test set      8.484994e-01 1.0485431 0.8488883  2.691918233 2.6932048
## Training set  9.576275e-02 0.3000990 0.2493222  0.323717954 0.8309759
## Test set     -9.591680e-01 1.1399538 0.9591680 -3.126377155 3.1263772
## Training set  1.147709e+00 1.2963548 1.1585419  3.762540281 3.7966718
## Test set     -1.046668e+00 1.1437379 1.0466681 -3.394789086 3.3947891
## Training set  1.775977e-16 0.4690483 0.4000656 -0.027272809 1.3276824
## Test set     -2.537702e+00 2.6955342 2.5377018 -8.224693412 8.2246934
## Training set -1.184437e-16 0.4256142 0.3455998 -0.023117205 1.1495424
## Test set     -2.579751e+00 2.7032265 2.5797512 -8.352383995 8.3523840
## Training set  9.412525e-02 0.2975941 0.2452210  0.318161119 0.8173299
## Test set     -9.592040e-01 1.1399841 0.9592040 -3.126493048 3.1264930
## Training set -2.177407e-03 0.2338684 0.1920148 -0.011210904 0.6355899
## Test set     -1.158708e+00 1.3545716 1.1587077 -3.772781515 3.7727815
## Training set  6.896459e-03 0.2831083 0.2326707  0.028406309 0.7752235
## Test set     -2.856593e-01 0.2957607 0.2856593 -0.895037878 0.8950379
## Training set -6.817596e-02 0.3668989 0.3040211 -0.221314038 1.0157251
## Test set     -4.229613e-01 0.8380907 0.7225811 -1.386762876 2.3263047
## Training set -3.504017e-02 0.2277567 0.1896310 -0.117710555 0.6292572
## Test set     -1.317480e+00 1.4271857 1.3174799 -4.228000402 4.2280004
## Training set  4.382371e-04 0.2820501 0.2305344  0.008244221 0.7671619
## Test set     -1.581626e+00 1.8268917 1.5816258 -5.147711997 5.1477120
##                   MASE        ACF1 Theil's U
## Training set 1.2488497  0.93229932        NA
## Test set     0.7327213  0.80958040  3.063001
## Training set 0.2152035  0.11489776        NA
## Test set     0.8279097  0.80958040  3.818160
## Training set 1.0000000  0.76185069        NA
## Test set     0.9034357  0.69880868  3.815306
## Training set 0.3453182  0.74062901        NA
## Test set     2.1904273  0.81644887  8.927445
## Training set 0.2983058  0.78445643        NA
## Test set     2.2267224  0.77090467  8.938057
## Training set 0.2116635  0.10881218        NA
## Test set     0.8279407  0.80958040  3.818259
## Training set 0.1657383  0.05932208        NA
## Test set     1.0001431  0.77935917  4.545454
## Training set 0.2008306  0.11522187        NA
## Test set     0.2465679 -0.64575605  1.604234
## Training set 0.2624170  0.61576754        NA
## Test set     0.6236988  0.83382675  2.508968
## Training set 0.1636808 -0.00490785        NA
## Test set     1.1371880  0.72420513  4.256209
## Training set 0.1989867  0.11646041        NA
## Test set     1.3651866  0.81635093  6.121313
```

```
row.names(a.table)<-c('Mean training','Mean test', 'Naive training', 'Naive test', 'Seasonal. Na
ive training', 'Seasonal. Naive test' ,'Linear training', 'Linear test','season-trend training',
'season-trend test',"ses training", "ses test",'STL training', 'STL test',"Holt's Linear trainin
g", "Holt's Linear test", 'Add training', 'Add test','Multi training', 'Multi test','ARIMA train
ing', 'ARIMA test')


#Final Tabular format
a.table<-as.data.frame(a.table)
a.table
```

```
##                                   ME      RMSE       MAE          MPE
## Mean training          -5.323234e-16 1.6951456 1.4468447 -0.325461790
## Mean test               8.484994e-01 1.0485431 0.8488883  2.691918233
## Naive training          9.576275e-02 0.3000990 0.2493222  0.323717954
## Naive test             -9.591680e-01 1.1399538 0.9591680 -3.126377155
## Seasonal. Naive training  1.147709e+00 1.2963548 1.1585419  3.762540281
## Seasonal. Naive test   -1.046668e+00 1.1437379 1.0466681 -3.394789086
## Linear training         1.775977e-16 0.4690483 0.4000656 -0.027272809
## Linear test            -2.537702e+00 2.6955342 2.5377018 -8.224693412
## season-trend training  -1.184437e-16 0.4256142 0.3455998 -0.023117205
## season-trend test      -2.579751e+00 2.7032265 2.5797512 -8.352383995
## ses training            9.412525e-02 0.2975941 0.2452210  0.318161119
## ses test               -9.592040e-01 1.1399841 0.9592040 -3.126493048
## STL training           -2.177407e-03 0.2338684 0.1920148 -0.011210904
## STL test               -1.158708e+00 1.3545716 1.1587077 -3.772781515
## Holt's Linear training  6.896459e-03 0.2831083 0.2326707  0.028406309
## Holt's Linear test     -2.856593e-01 0.2957607 0.2856593 -0.895037878
## Add training           -6.817596e-02 0.3668989 0.3040211 -0.221314038
## Add test               -4.229613e-01 0.8380907 0.7225811 -1.386762876
## Multi training         -3.504017e-02 0.2277567 0.1896310 -0.117710555
## Multi test             -1.317480e+00 1.4271857 1.3174799 -4.228000402
## ARIMA training          4.382371e-04 0.2820501 0.2305344  0.008244221
## ARIMA test             -1.581626e+00 1.8268917 1.5816258 -5.147711997
##                             MAPE      MASE        ACF1 Theil's U
## Mean training          4.8658328 1.2488497  0.93229932        NA
## Mean test              2.6932048 0.7327213  0.80958040  3.063001
## Naive training         0.8309759 0.2152035  0.11489776        NA
## Naive test             3.1263772 0.8279097  0.80958040  3.818160
## Seasonal. Naive training 3.7966718 1.0000000  0.76185069        NA
## Seasonal. Naive test   3.3947891 0.9034357  0.69880868  3.815306
## Linear training        1.3276824 0.3453182  0.74062901        NA
## Linear test            8.2246934 2.1904273  0.81644887  8.927445
## season-trend training  1.1495424 0.2983058  0.78445643        NA
## season-trend test      8.3523840 2.2267224  0.77090467  8.938057
## ses training           0.8173299 0.2116635  0.10881218        NA
## ses test               3.1264930 0.8279407  0.80958040  3.818259
## STL training           0.6355899 0.1657383  0.05932208        NA
## STL test               3.7727815 1.0001431  0.77935917  4.545454
## Holt's Linear training 0.7752235 0.2008306  0.11522187        NA
## Holt's Linear test     0.8950379 0.2465679 -0.64575605  1.604234
## Add training           1.0157251 0.2624170  0.61576754        NA
## Add test               2.3263047 0.6236988  0.83382675  2.508968
## Multi training         0.6292572 0.1636808 -0.00490785        NA
## Multi test             4.2280004 1.1371880  0.72420513  4.256209
## ARIMA training         0.7671619 0.1989867  0.11646041        NA
## ARIMA test             5.1477120 1.3651866  0.81635093  6.121313
```