Stock_Price_Forecasting_of_Apple.R

sayal

Sat Apr 20 12:11:20 2019

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
library(quantmod)
```

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

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

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

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

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

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

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

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

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

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

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

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

```
library(fpp2)
```

```
##
## 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("2018-01-01")
start_date
```

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

```
end_date
```

```
## [1] "2018-01-01"
```

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

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

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

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

```
getSymbols("AAPL", 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] "AAPL"
```

```
summary(AAPL)
```

```
##      Index              AAPL.Open        AAPL.High         AAPL.Low
##  Min.   :2012-01-03   Min.   : 55.42   Min.   : 57.09   Min.   : 55.01
##  1st Qu.:2013-07-05   1st Qu.: 78.78   1st Qu.: 79.37   1st Qu.: 77.68
##  Median :2015-01-02   Median : 99.25   Median :100.33   Median : 98.40
##  Mean   :2015-01-01   Mean   :102.88   Mean   :103.74   Mean   :101.97
##  3rd Qu.:2016-07-01   3rd Qu.:118.98   3rd Qu.:119.92   3rd Qu.:118.22
##  Max.   :2017-12-29   Max.   :175.11   Max.   :177.20   Max.   :174.86
##    AAPL.Close      AAPL.Volume        AAPL.Adjusted
##  Min.   : 55.79   Min.   : 11475900   Min.   : 39.17
##  1st Qu.: 78.44   1st Qu.: 33257300   1st Qu.: 58.93
##  Median : 99.52   Median : 53583400   Median : 93.16
##  Mean   :102.87   Mean   : 68962505   Mean   : 91.97
##  3rd Qu.:119.25   3rd Qu.: 88569600   3rd Qu.:113.01
##  Max.   :176.42   Max.   :376530000   Max.   :173.07
```

```
head(AAPL)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2012-01-03  58.48571  58.92857 58.42857   58.74714    75555200
## 2012-01-04  58.57143  59.24000 58.46857   59.06286    65005500
## 2012-01-05  59.27857  59.79286 58.95286   59.71857    67817400
## 2012-01-06  59.96714  60.39286 59.88857   60.34286    79573200
## 2012-01-09  60.78571  61.10714 60.19286   60.24714    98506100
## 2012-01-10  60.84428  60.85714 60.21429   60.46286    64549100
##            AAPL.Adjusted
## 2012-01-03      39.17277
## 2012-01-04      39.38329
## 2012-01-05      39.82052
## 2012-01-06      40.23681
## 2012-01-09      40.17297
## 2012-01-10      40.31681
```

```
names(AAPL)
```

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

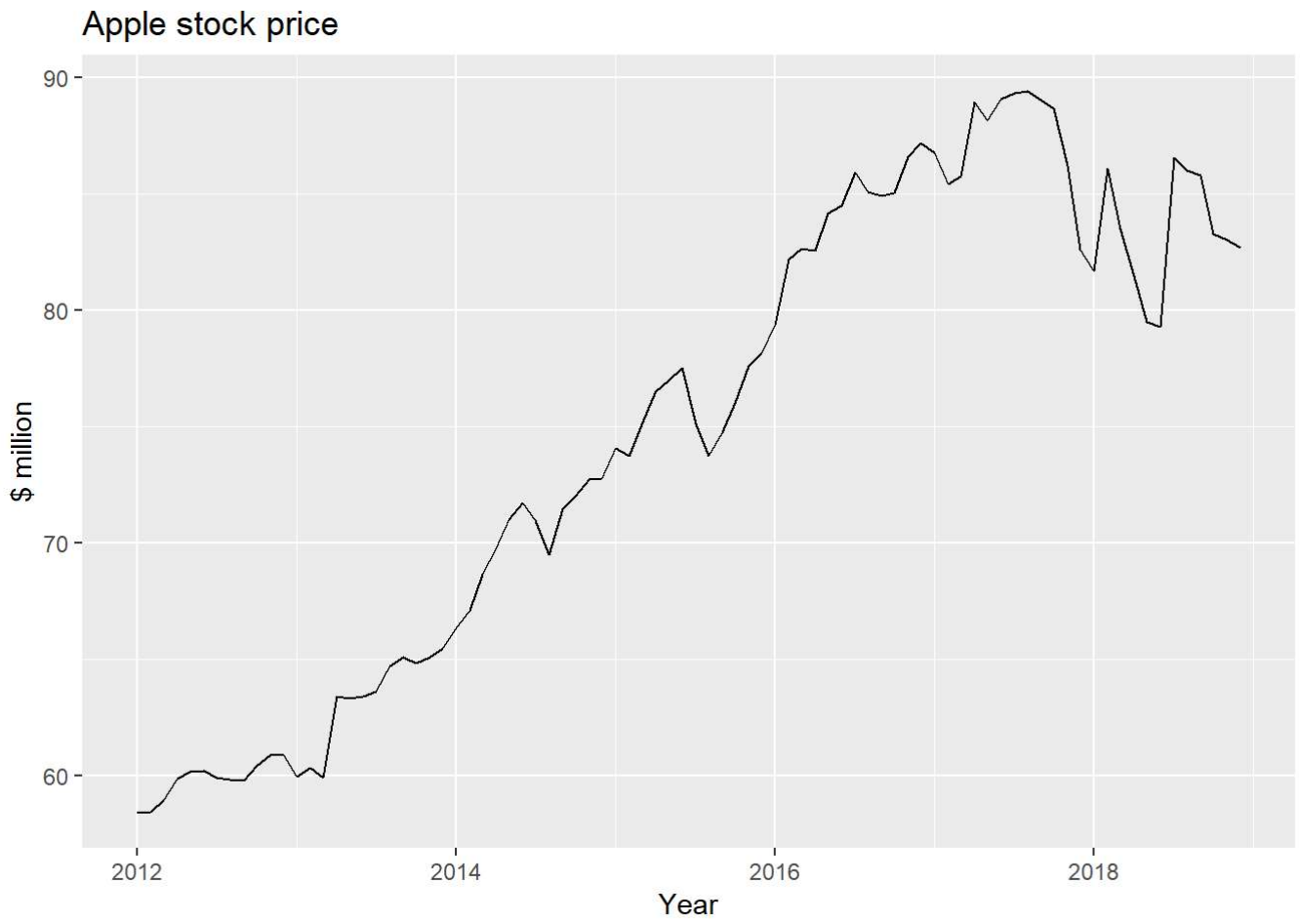
```
data <- ts(AAPL,start=c(2012,1),end=c(2018,12), frequency = 12)
data=data[,3]

###############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
## 2012 58.42857 58.46857 58.95286 59.88857 60.19286 60.21429 59.90143
## 2013 59.96429 60.32857 59.93572 63.39000 63.30571 63.39571 63.62714
## 2014 66.36857 67.10000 68.65143 69.79285 71.01286 71.71429 70.98428
## 2015 74.09143 73.75429 75.12143 76.52857 76.96714 77.50285 75.14286
## 2016 79.39286 82.20000 82.65000 82.57143 84.15000 84.49715 85.91572
##           Aug      Sep      Oct      Nov      Dec
## 2012 59.82143 59.80857 60.42285 60.90000 60.93000
## 2013 64.72429 65.07858 64.85429 65.08000 65.45715
## 2014 69.51857 71.47143 72.01714 72.72429 72.78571
## 2015 73.74571 74.75714 76.01714 77.58714 78.14286
## 2016 85.07571 84.91428 85.03714 86.58000 87.18714
```

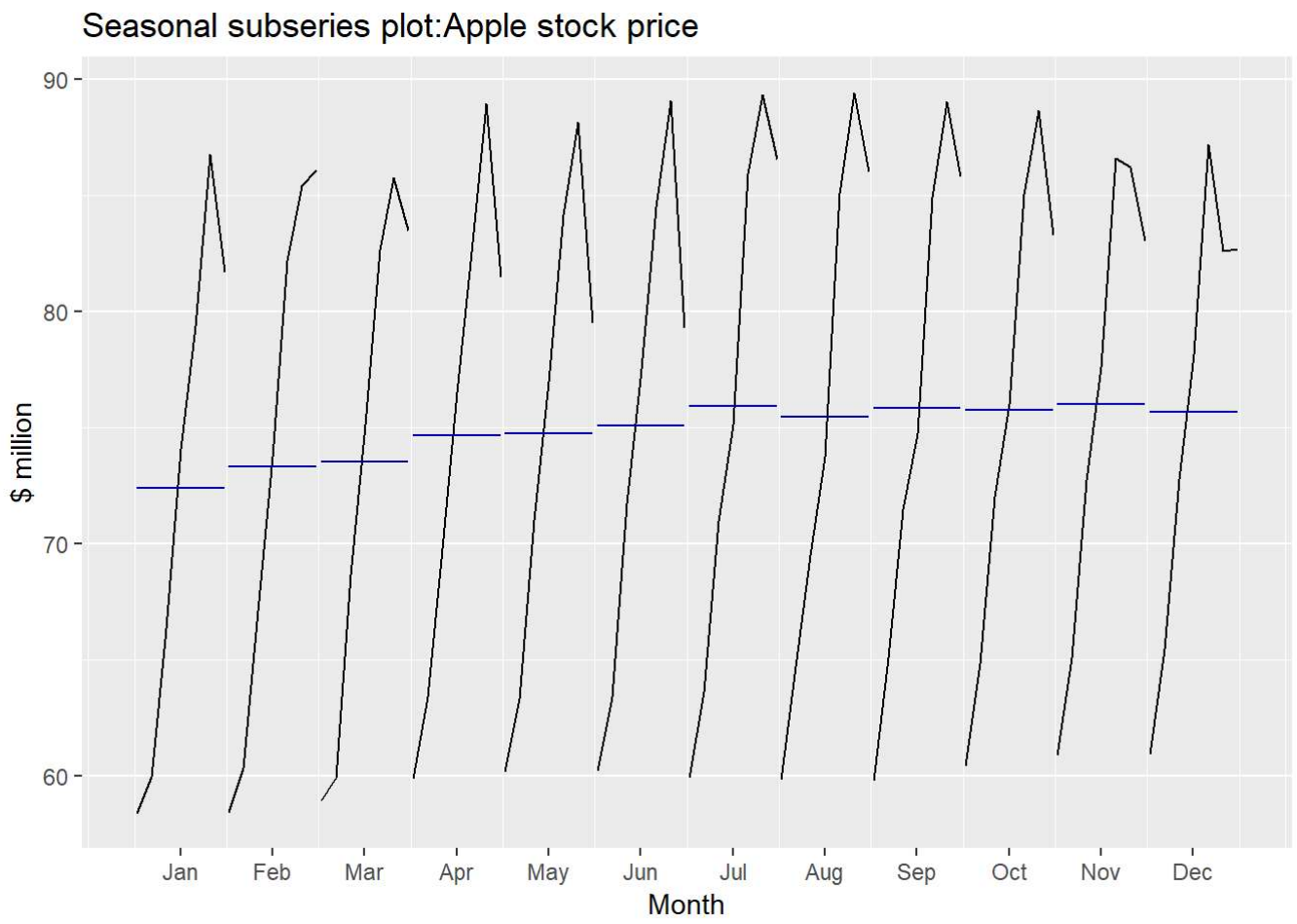
```
test_data
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2017 86.74715 85.42000 85.76857 88.93000 88.14286 89.05714 89.32858
## 2018 81.70143 86.10143 83.50285 81.48857 79.51714 79.28571 86.57143
##           Aug      Sep      Oct      Nov      Dec
## 2017 89.42857 89.04857 88.64286 86.21571 82.60714
## 2018 86.01857 85.78571 83.28571 83.03286 82.69428
```

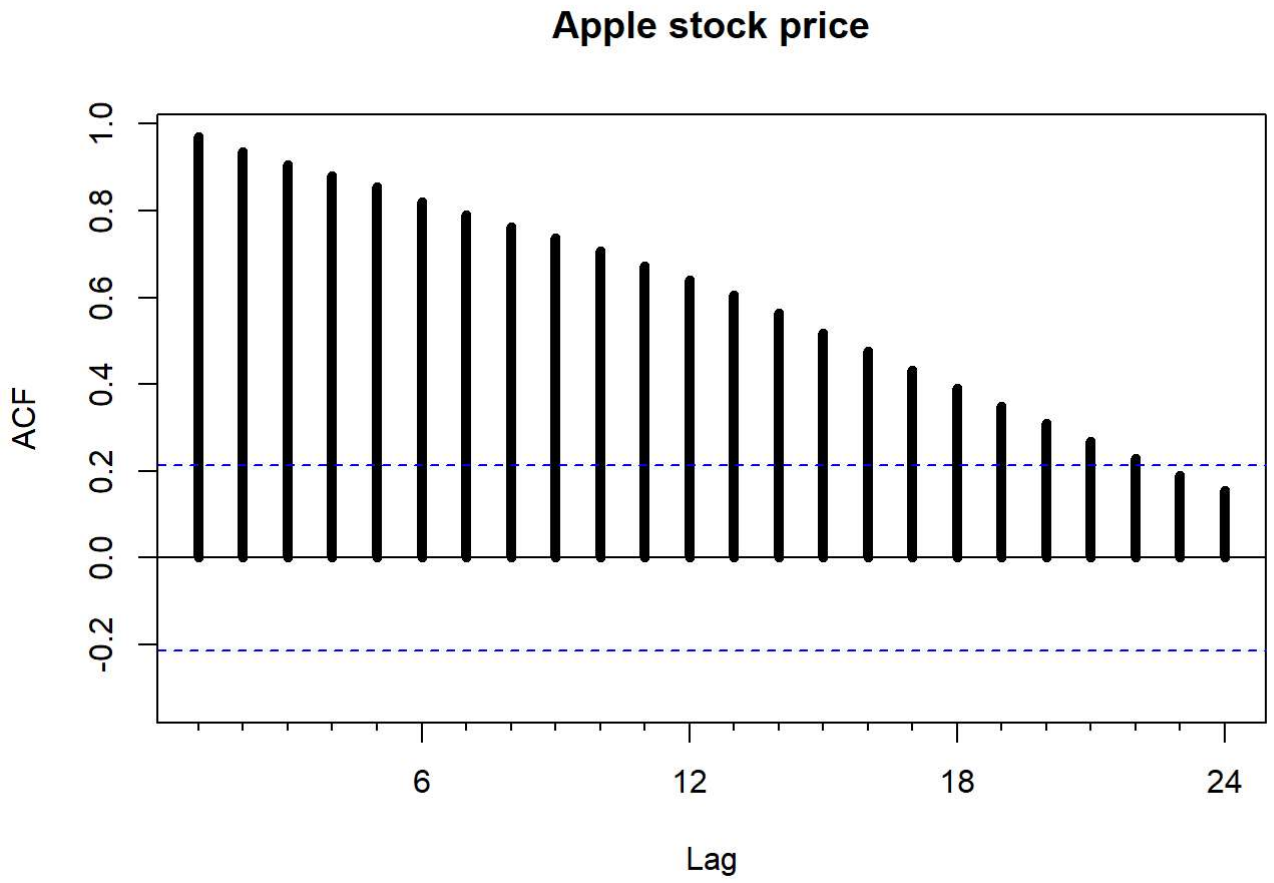
```
autoplot(data) + ggtitle("Apple stock price") + ylab("$ million") + xlab("Year")
```



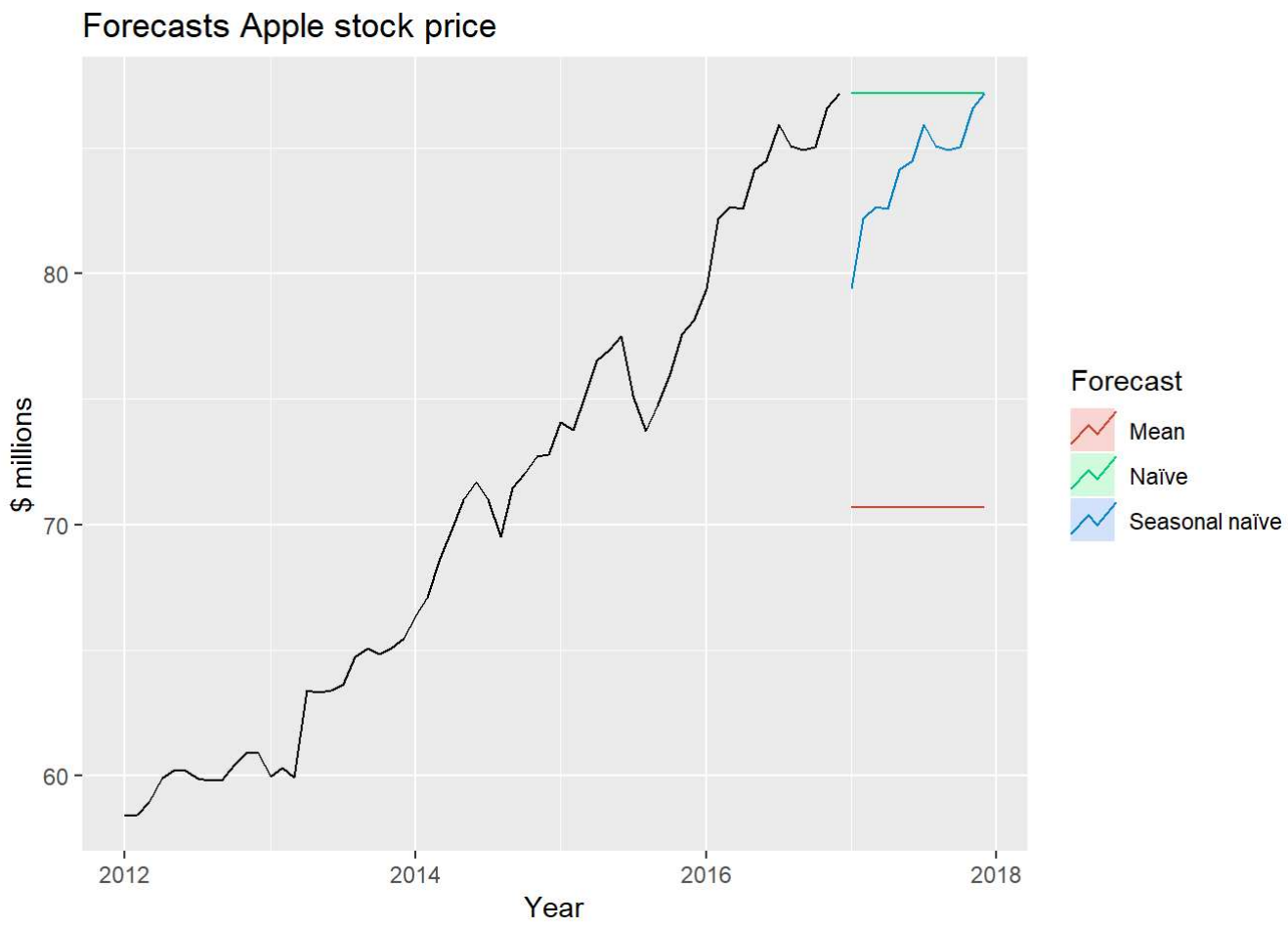

```
ggsubseriesplot(data) + ylab("$ million") + ggtitle("Seasonal subseries plot:Apple stock price")
```




```
Acf(data, lwd=5, main="Apple stock price")
```



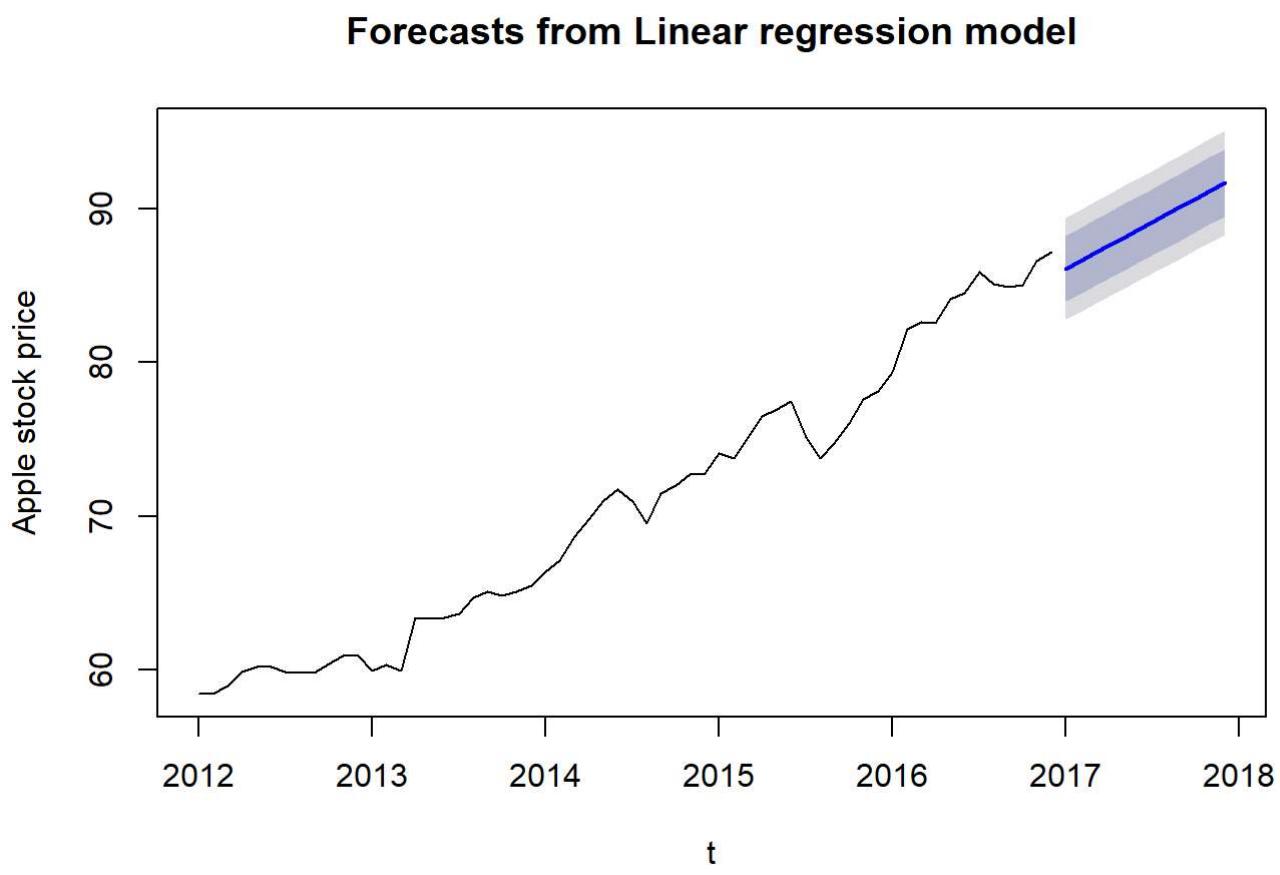

```
###############Mean, Naive, Seasonal Naive###############
fit.mean=meanf(train_data,h=12)
fit.naive=naive(train_data,h=12)
fit.snaive=snaive(train_data,h=12)

autoplot(train_data) +
  autolayer(meanf(train_data, h=12),
            series="Mean", PI=FALSE) +
  autolayer(naive(train_data, h=12),
            series="Naïve", PI=FALSE) +
  autolayer(snaive(train_data, h=12),
            series="Seasonal naïve", PI=FALSE) +
  ggtitle("Forecasts Apple 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
##      55.246        0.506
##
##
## Error measures:
##                         ME     RMSE      MAE          MPE     MAPE      MASE
## Training set 1.184961e-16 1.572859 1.299547 -0.03258308 1.868646 0.2134477
##                   ACF1
## Training set 0.7859229
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       86.11242 83.96893 88.25592 82.80253 89.42231
## Feb 2017       86.61843 84.47148 88.76538 83.30321 89.93365
## Mar 2017       87.12445 84.97394 89.27495 83.80373 90.44516
## Apr 2017       87.63046 85.47629 89.78463 84.30408 90.95683
## May 2017       88.13647 85.97853 90.29441 84.80427 91.46866
## Jun 2017       88.64248 86.48067 90.80430 85.30431 91.98066
## Jul 2017       89.14849 86.98270 91.31428 85.80418 92.49281
## Aug 2017       89.65451 87.48464 91.82437 86.30389 93.00512
## Sep 2017       90.16052 87.98647 92.33457 86.80345 93.51759
## Oct 2017       90.66653 88.48820 92.84486 87.30285 94.03021
## Nov 2017       91.17254 88.98983 93.35526 87.80209 94.54300
## Dec 2017       91.67856 89.49135 93.86576 88.30118 95.05593
```

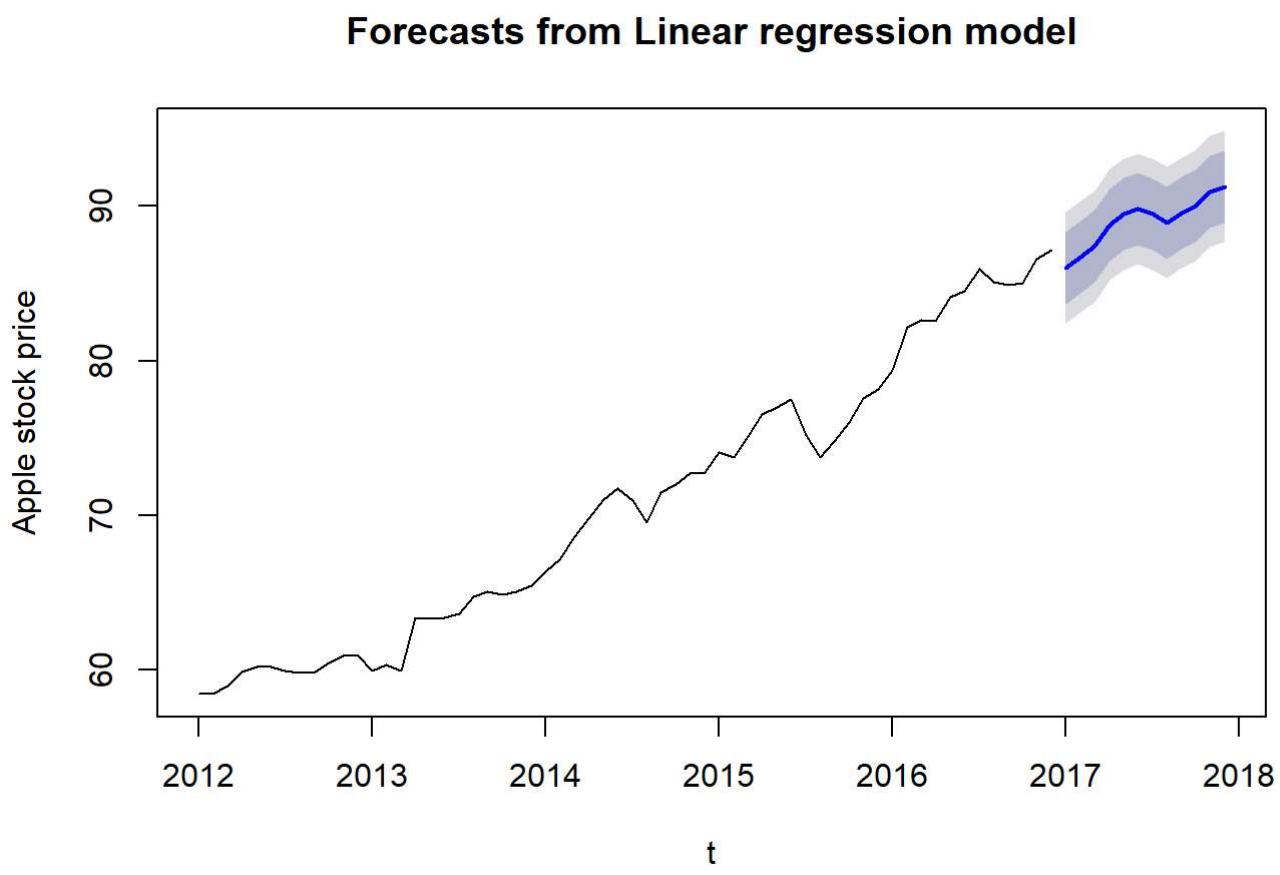
```
plot(fit.tslm1, ylab="Apple stock price",
     xlab="t")
```



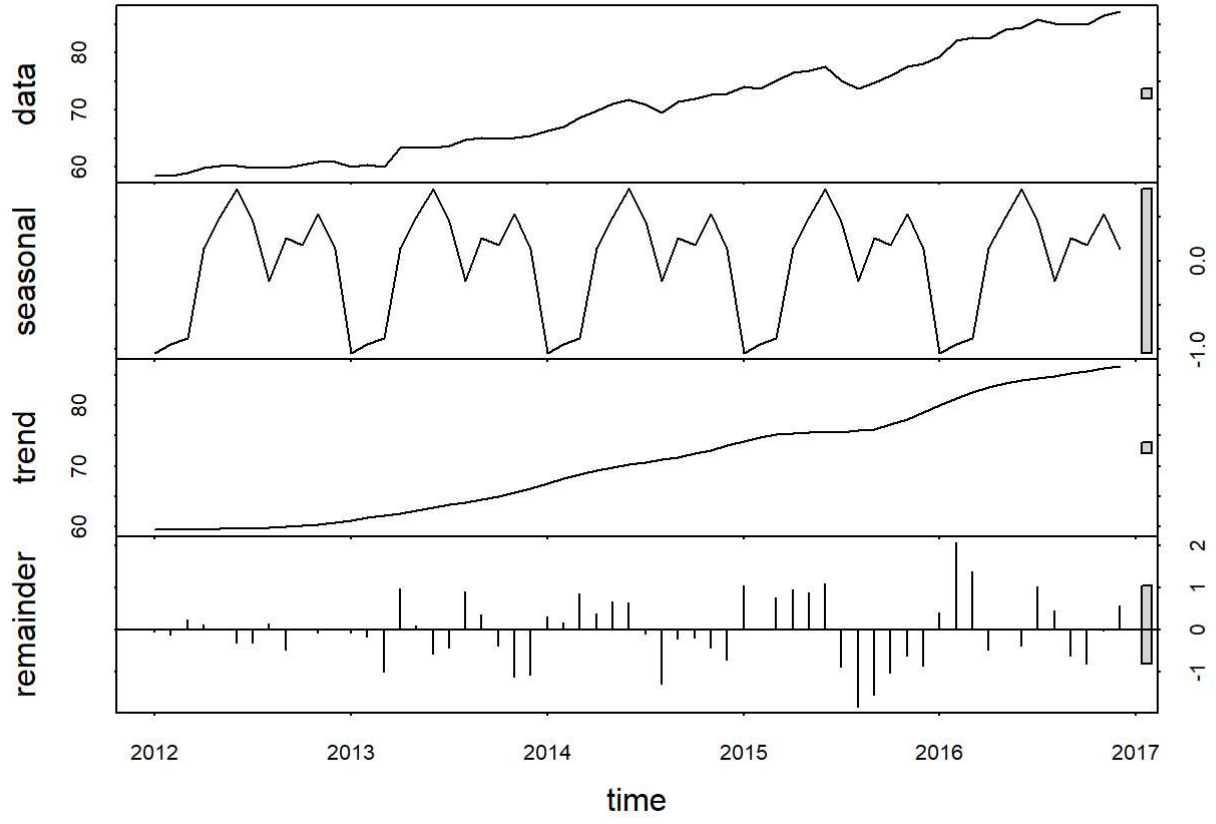

```
###############Season + Trend###############
linear_season <- tslm(train_data ~ trend + season)
fit.tslm2=forecast(linear_season, h=12)
summary(fit.tslm2)
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = train_data ~ trend + season)
##
## Coefficients:
## (Intercept)        trend      season2      season3      season4
##     54.8939       0.5102       0.2109       0.3927       1.2545
##     season5      season6      season7      season8      season9
##      1.4357       1.2647       0.4039      -0.6435      -0.5248
##    season10     season11     season12
##     -0.5713      -0.1769      -0.3609
##
##
## Error measures:
##                         ME     RMSE      MAE         MPE    MAPE      MASE
## Training set -1.184455e-16 1.401696 1.133838 -0.02238192 1.65084 0.1862304
##                    ACF1
## Training set 0.7761691
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       86.01664 83.69219 88.34110 82.41909 89.61420
## Feb 2017       86.73778 84.41333 89.06224 83.14023 90.33534
## Mar 2017       87.42979 85.10533 89.75424 83.83223 91.02734
## Apr 2017       88.80178 86.47733 91.12624 85.20423 92.39934
## May 2017       89.49321 87.16876 91.81767 85.89566 93.09077
## Jun 2017       89.83236 87.50791 92.15681 86.23480 93.42991
## Jul 2017       89.48179 87.15733 91.80624 85.88423 93.07934
## Aug 2017       88.94464 86.62019 91.26910 85.34709 92.54220
## Sep 2017       89.57350 87.24905 91.89795 85.97594 93.17106
## Oct 2017       90.03721 87.71276 92.36167 86.43966 93.63477
## Nov 2017       90.94179 88.61734 93.26624 87.34423 94.53934
## Dec 2017       91.26807 88.94362 93.59252 87.67051 94.86563
```

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



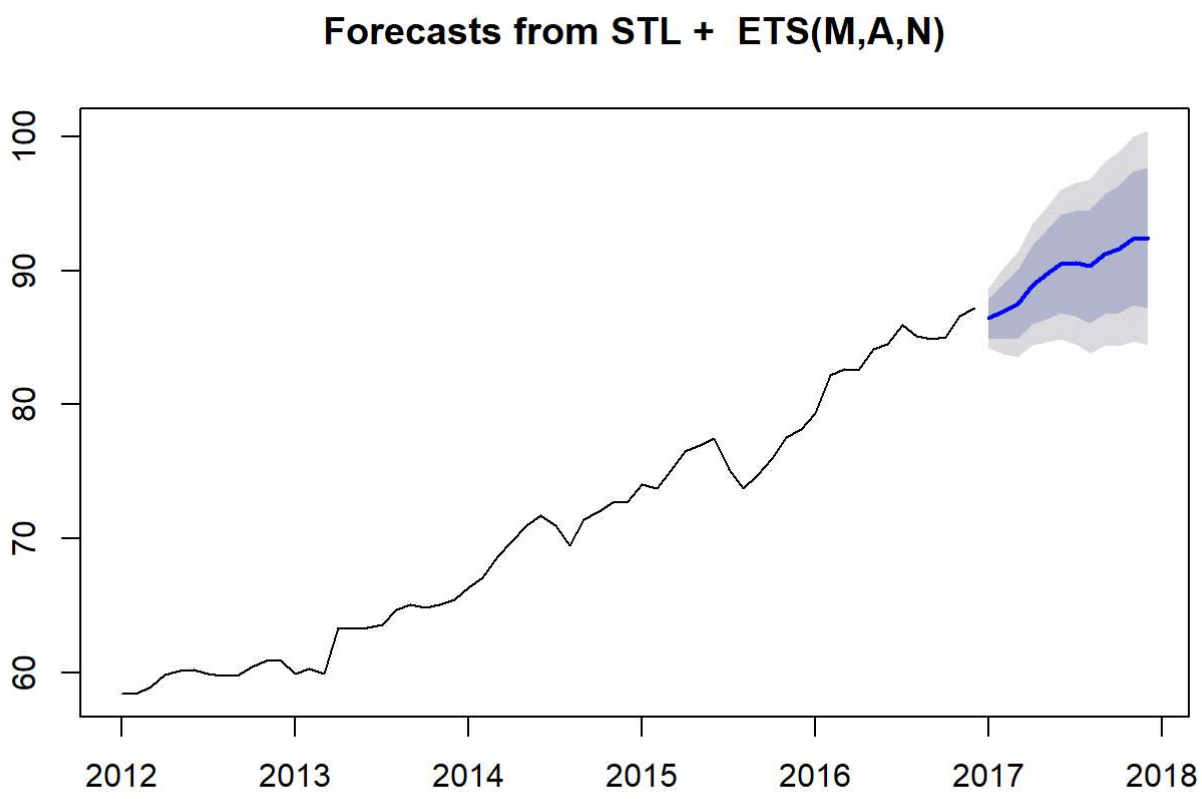

```
###############STL decomposition###############
stl_decomp <- stl(train_data, t.window=12, s.window="periodic")
plot(stl_decomp)
```




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

```
##
## Forecast method: STL +  ETS(M,A,N)
##
## Model Information:
## ETS(M,A,N)
##
## Call:
##  ets(y = x, model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.9924
##     beta  = 1e-04
##
##   Initial states:
##     l = 59.244
##     b = 0.4363
##
##   sigma:  0.0131
##
##      AIC     AICc      BIC
## 241.7742 242.8853 252.2459
##
## Error measures:
##                       ME      RMSE      MAE         MPE      MAPE      MASE
## Training set 0.02738604 0.9168425 0.697892 0.007787811 0.9767164 0.1146272
##                     ACF1
## Training set 0.08221281
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## Jan 2017       86.43741 84.96352 87.91131 84.18328  88.69154
## Feb 2017       86.97168 84.88977 89.05360 83.78766  90.15570
## Mar 2017       87.47681 84.92364 90.02999 83.57207  91.38156
## Apr 2017       88.94046 85.98657 91.89436 84.42288  93.45805
## May 2017       89.72354 86.41375 93.03334 84.66164  94.78544
## Jun 2017       90.48050 86.84639 94.11460 84.92261  96.03838
## Jul 2017       90.54774 86.61309 94.48238 84.53021  96.56526
## Aug 2017       90.30809 86.09157 94.52462 83.85948  96.75671
## Sep 2017       91.23445 86.75118 95.71772 84.37789  98.09102
## Oct 2017       91.58504 86.84759 96.32249 84.33974  98.83034
## Nov 2017       92.37649 87.39549 97.35749 84.75871  99.99428
## Dec 2017       92.41983 87.20440 97.63526 84.44352 100.39614
```

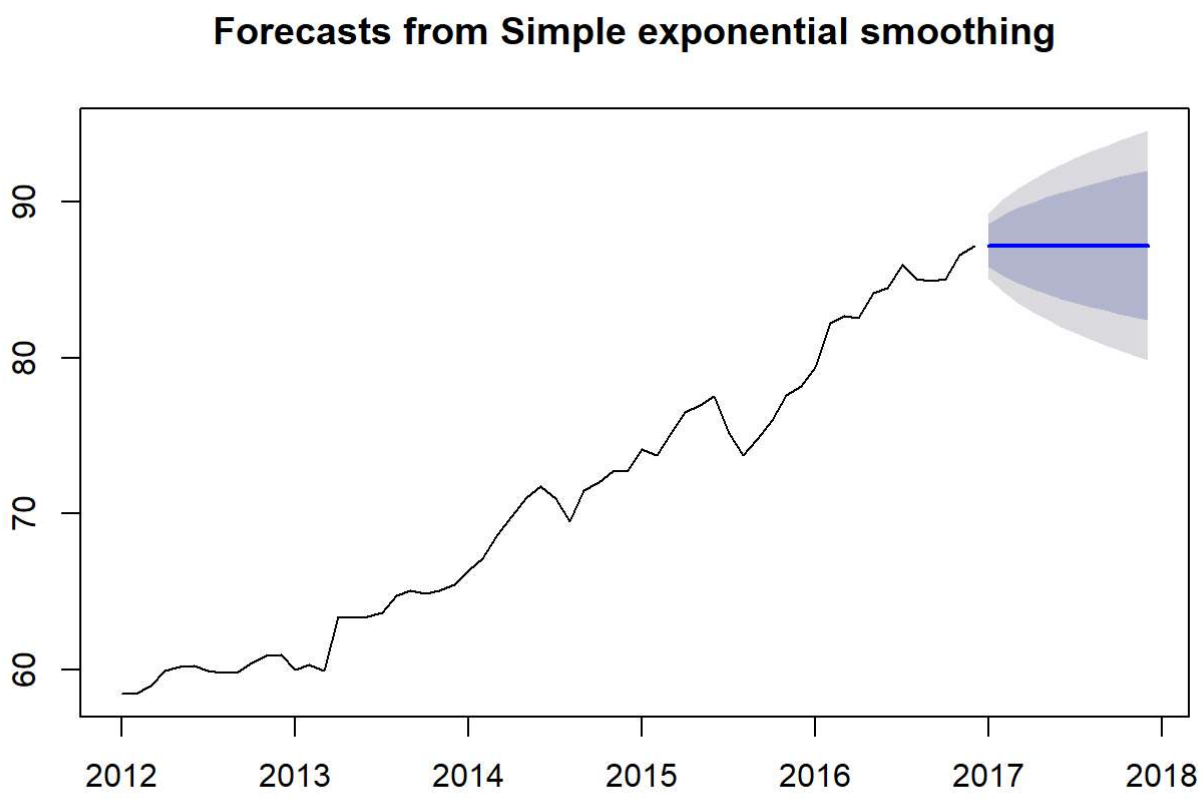
```
plot(fit.stl)
```



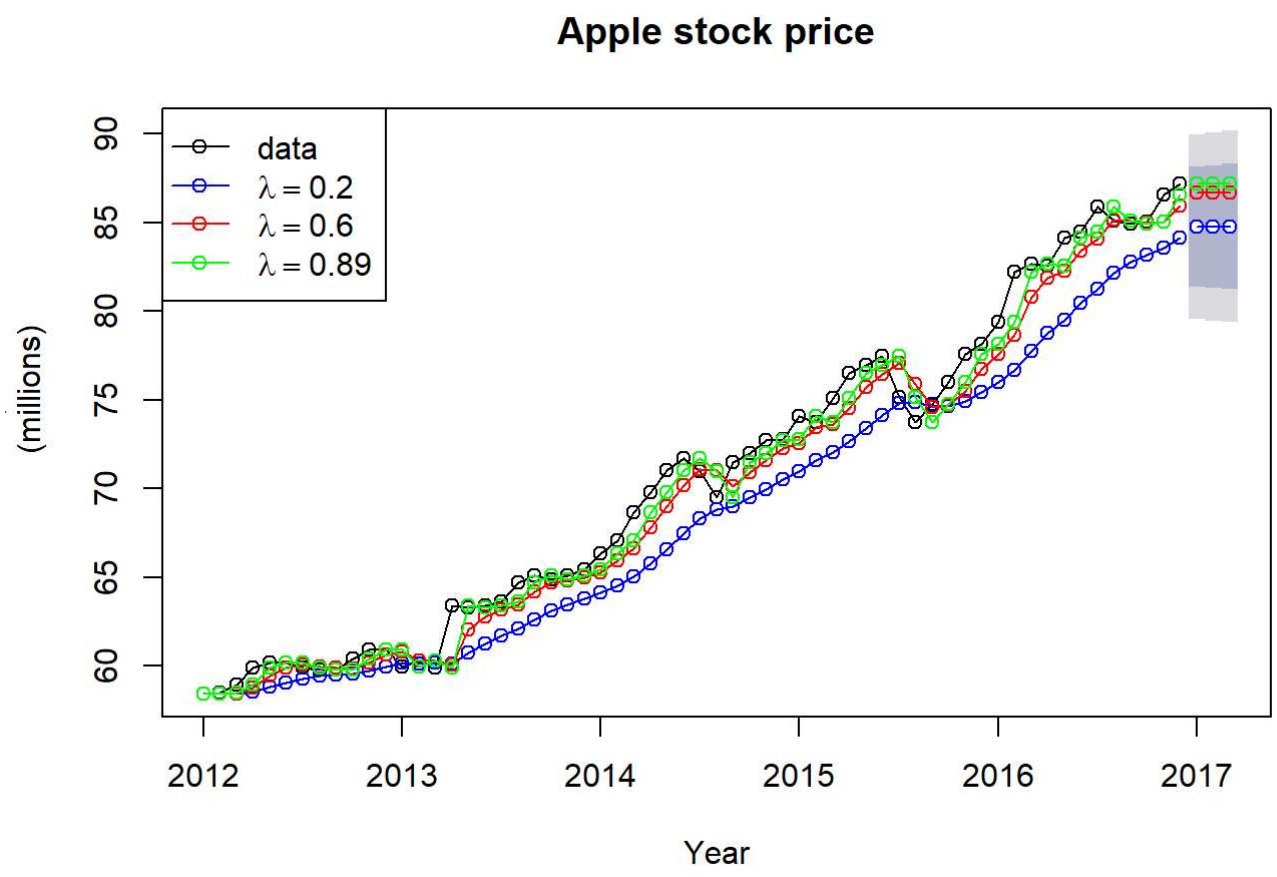

```
###############ETS###############
library(forecast)
ets_forecast<-ets(train_data)
fit.ets_forecast <- forecast(ets_forecast)

###############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 = 58.4301
##
##   sigma:  1.0819
##
##      AIC     AICc      BIC
## 259.0724 259.5010 265.3555
##
## Error measures:
##                     ME     RMSE     MAE       MPE    MAPE      MASE
## Training set 0.4793308 1.063712 0.79413 0.6560745 1.10028 0.1304341
##                    ACF1
## Training set 0.07868901
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       87.18708 85.80057 88.57359 85.06660 89.30756
## Feb 2017       87.18708 85.22636 89.14780 84.18842 90.18574
## Mar 2017       87.18708 84.78574 89.58842 83.51455 90.85961
## Apr 2017       87.18708 84.41428 89.95988 82.94644 91.42772
## May 2017       87.18708 84.08701 90.28715 82.44593 91.92823
## Jun 2017       87.18708 83.79113 90.58303 81.99342 92.38074
## Jul 2017       87.18708 83.51904 90.85512 81.57730 92.79686
## Aug 2017       87.18708 83.26579 91.10837 81.18999 93.18417
## Sep 2017       87.18708 83.02793 91.34623 80.82621 93.54795
## Oct 2017       87.18708 82.80296 91.57120 80.48214 93.89202
## Nov 2017       87.18708 82.58898 91.78518 80.15489 94.21927
## Dec 2017       87.18708 82.38452 91.98964 79.84220 94.53196
```

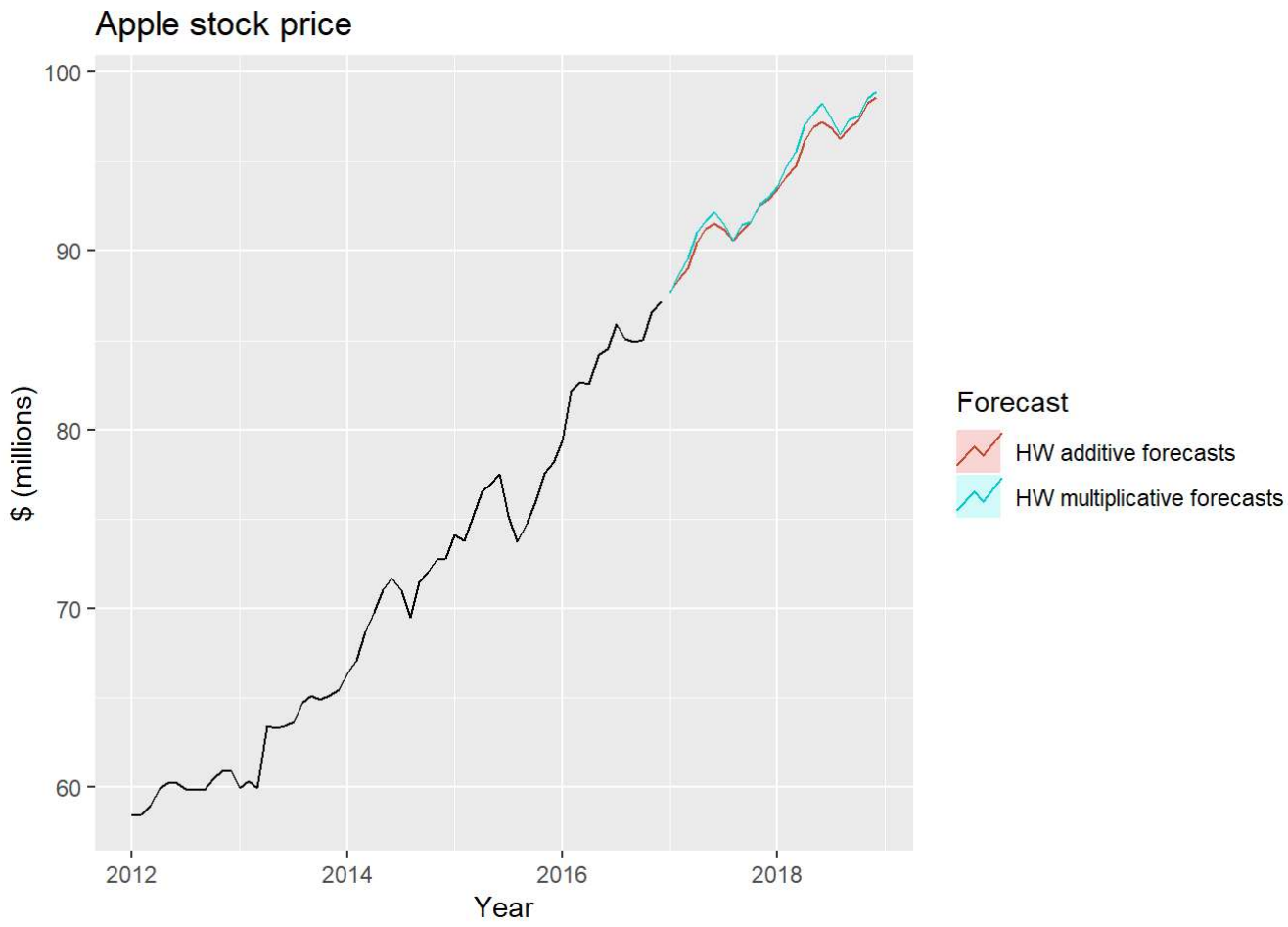
```
plot(fit.ses)
```




```
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="Apple 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)
```



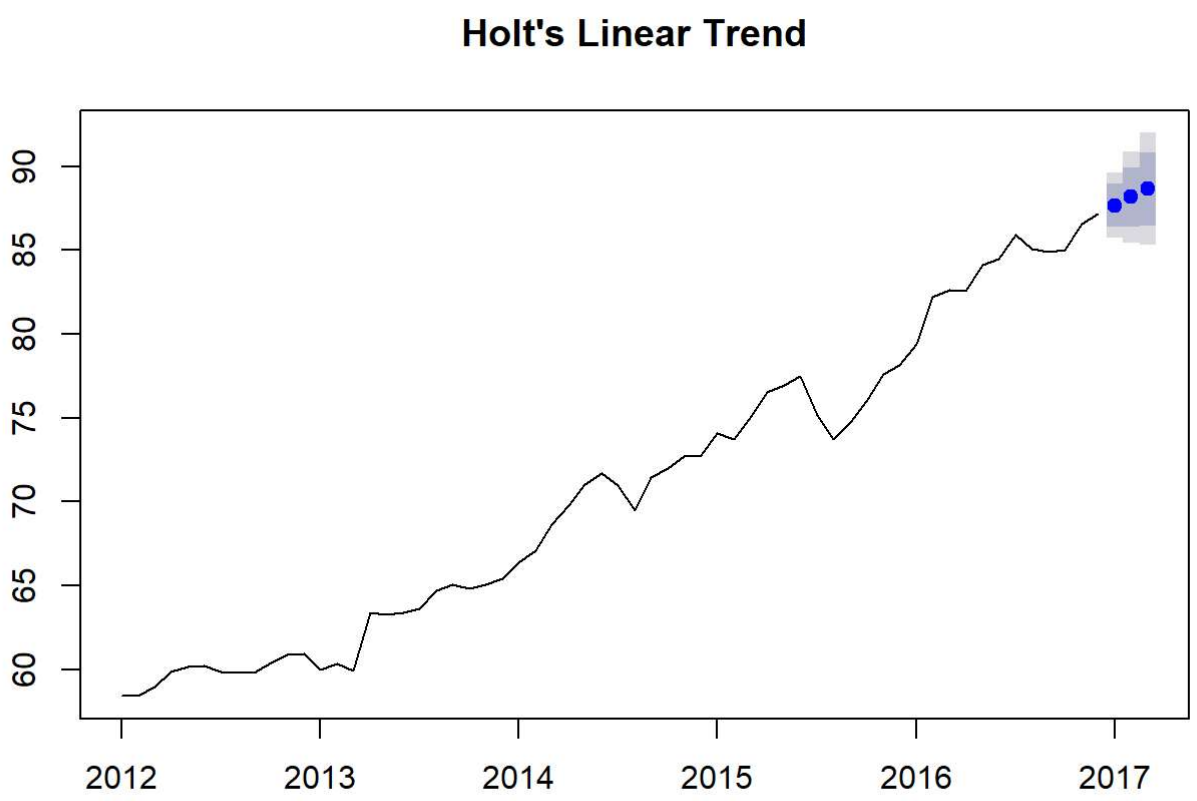

```
###############HW Additive and Multiplicative###############
fit1_add <- hw(train_data,seasonal="additive")
fit1_add <- forecast(fit1_add)
fit2_multi <- hw(train_data,seasonal="multiplicative")
fit2_multi <- forecast(fit2_multi)
autoplot(train_data) +
  autolayer(fit1_add, series="HW additive forecasts", PI=FALSE) +
  autolayer(fit2_multi, series="HW multiplicative forecasts", PI=FALSE) +
  xlab("Year") +
  ylab("$ (millions)") +
  ggtitle("Apple stock price") +
  guides(colour=guide_legend(title="Forecast"))
```




```
###############Holt's Linear trend###############
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 = 0.9999
##     beta  = 1e-04
##
##   Initial states:
##     l = 57.9212
##     b = 0.4885
##
##   sigma:  0.9808
##
##      AIC     AICc      BIC
## 249.1990 250.3101 259.6707
##
## Error measures:
##                         ME     RMSE       MAE         MPE      MAPE
## Training set -0.0004784057 0.947577 0.6877916 -0.03113668 0.9677082
##                    MASE       ACF1
## Training set 0.1129682 0.07482566
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       87.67558 86.41859 88.93258 85.75318 89.59799
## Feb 2017       88.16404 86.38639 89.94169 85.44535 90.88272
## Mar 2017       88.65249 86.47525 90.82974 85.32269 91.98230
```

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




```
###############Check stationarity###############
ndiffs(train_data)
```

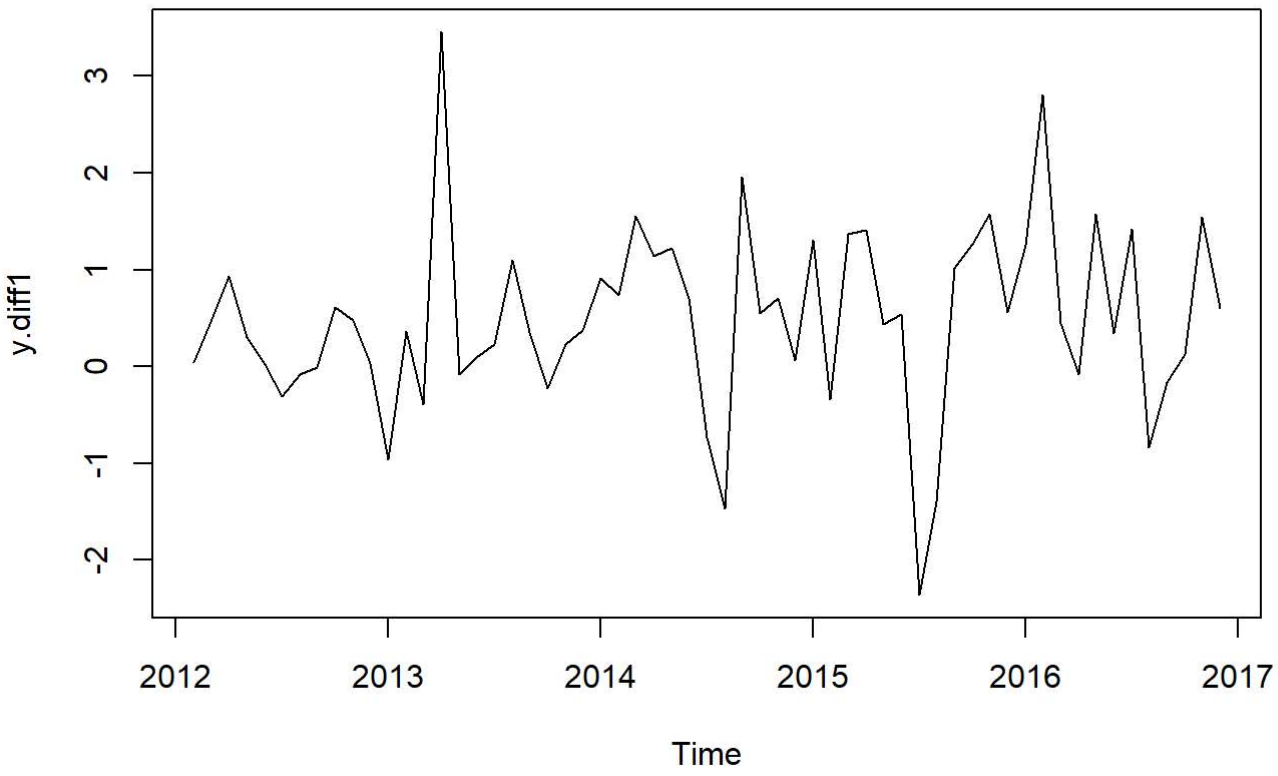
```
## [1] 1
```

```
y.diff1 = diff(train_data, differences = 1)
adf.test(y.diff1, alternative = "stationary")
```

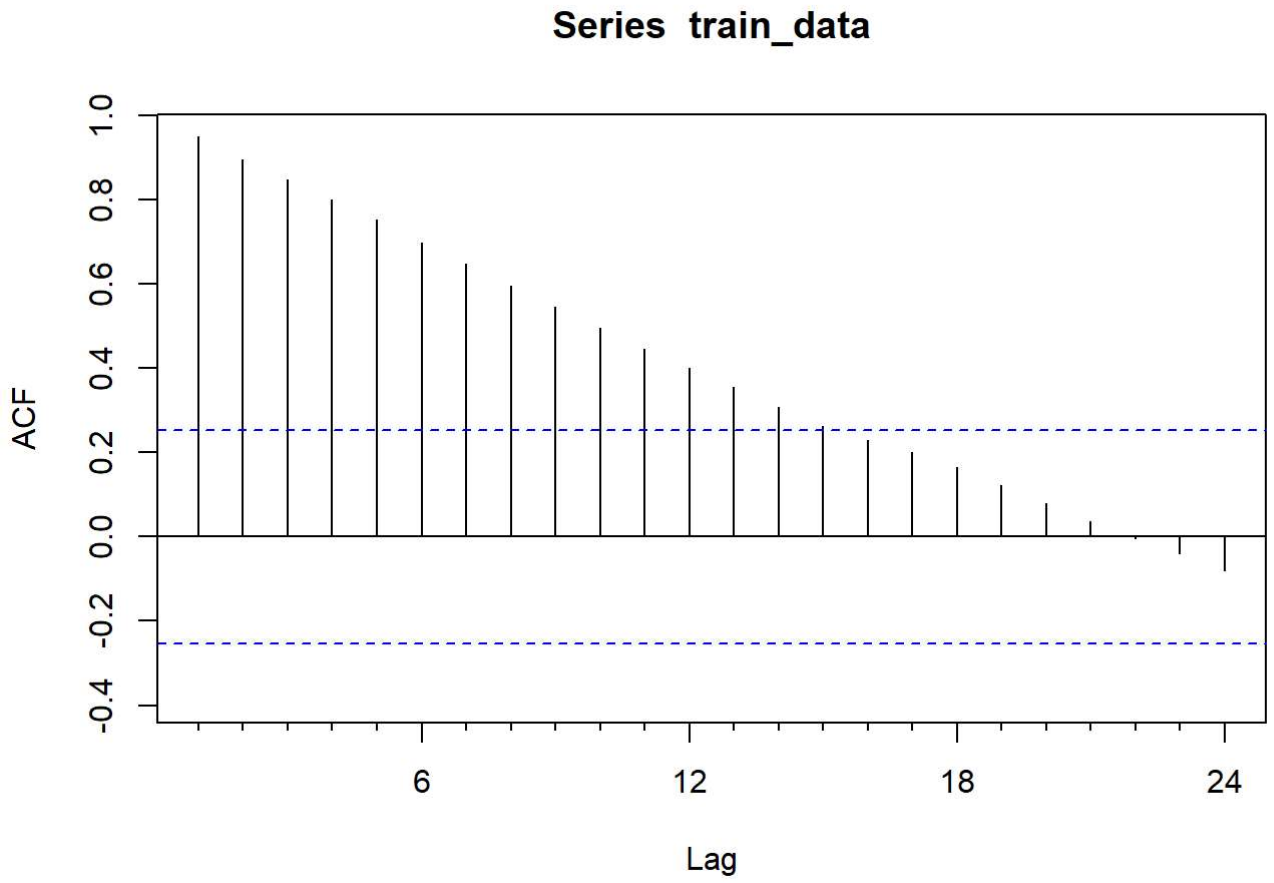
```
## Warning in adf.test(y.diff1, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y.diff1
## Dickey-Fuller = -4.3407, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

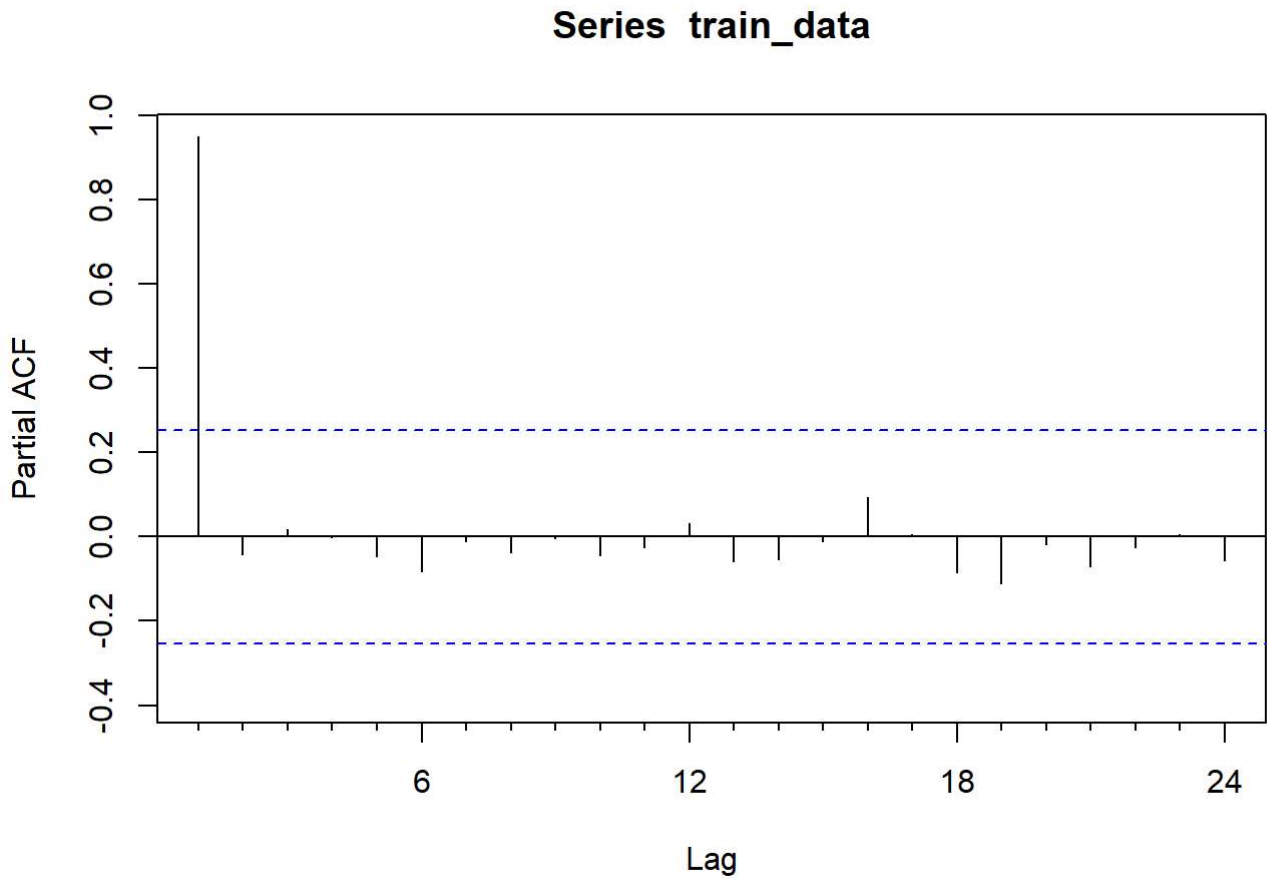
```
plot(y.diff1)
```




```
Acf(train_data)
```



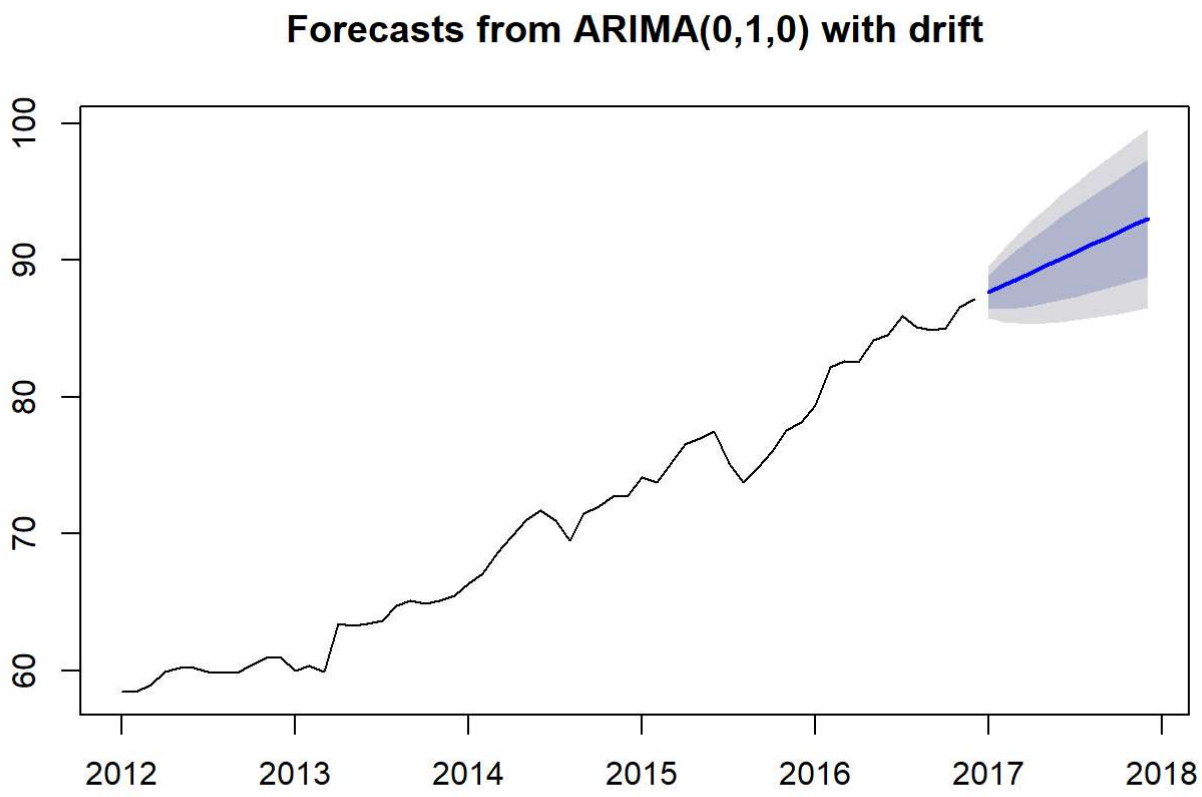

```
Pacf(train_data)
```




```
################Auto ARIMA###############
y.arima <- auto.arima(train_data)
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
##       0.4874
## s.e.  0.1244
##
## sigma^2 estimated as 0.9288:  log likelihood=-81.03
## AIC=166.07   AICc=166.28   BIC=170.22
##
## Error measures:
##                        ME      RMSE       MAE         MPE      MAPE
## Training set 0.0009656851 0.9475482 0.6883235 -0.02889026 0.9686066
##                    MASE       ACF1
## Training set 0.1130556 0.07441283
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2017       87.67457 86.43948 88.90967 85.78567 89.56348
## Feb 2017       88.16201 86.41533 89.90869 85.49069 90.83333
## Mar 2017       88.64944 86.51020 90.78868 85.37775 91.92113
## Apr 2017       89.13687 86.66669 91.60706 85.35906 92.91469
## May 2017       89.62431 86.86256 92.38606 85.40058 93.84804
## Jun 2017       90.11174 87.08640 93.13708 85.48488 94.73860
## Jul 2017       90.59917 87.33143 93.86692 85.60159 95.59676
## Aug 2017       91.08661 87.59324 94.57997 85.74397 96.42925
## Sep 2017       91.57404 87.86877 95.27932 85.90731 97.24077
## Oct 2017       92.06148 88.15577 95.96718 86.08822 98.03473
## Nov 2017       92.54891 88.45257 96.64524 86.28411 98.81371
## Dec 2017       93.03634 88.75786 97.31482 86.49297 99.57971
```

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




```
###############Comparison##############
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.ets, a.ses, a.stl, a.hol
t, a.add, a.multi, a.arima)
a.table
```

```
##                        ME       RMSE        MAE           MPE       MAPE
## Training set  4.503053e-15  8.9032051  7.7063175  -1.574440099 11.0144268
## Test set      1.676571e+01 16.8872841 16.7657148  19.128589860 19.1285899
## Training set  4.874334e-01  1.0726601  0.8075307   0.667174022  1.1188413
## Test set      2.576216e-01  2.0390045  1.7871442   0.239925930  2.0567580
## Training set  6.088363e+00  6.5115185  6.0883632   8.140173840  8.1401738
## Test set      3.263810e+00  4.4078055  4.0878580   3.676284667  4.6707592
## Training set  1.184961e-16  1.5728594  1.2995469  -0.032583082  1.8686463
## Test set     -1.450725e+00  3.1333447  1.8732924  -1.719282838  2.1971936
## Training set -1.184455e-16  1.4016964  1.1338384  -0.022381915  1.6508397
## Test set     -1.601785e+00  2.9902011  1.8255601  -1.887899865  2.1424698
## Training set  3.771897e-03  0.9523994  0.6980538  -0.024680434  0.9840451
## Test set     -7.960162e+00  9.8077564  7.9601624  -9.551741436  9.5517414
## Training set  4.793308e-01  1.0637119  0.7941300   0.656074513  1.1002801
## Test set      2.576823e-01  2.0390122  1.7871544   0.239995419  2.0567682
## Training set  2.738604e-02  0.9168425  0.6978920   0.007787811  0.9767164
## Test set     -2.430410e+00  3.6501703  2.4820320  -2.833431113  2.8929399
## Training set -4.784057e-04  0.9475770  0.6877916  -0.031136676  0.9677082
## Test set     -2.185467e+00  2.3599983  2.1854675  -2.548379708  2.5483797
## Training set  2.287631e-02  1.0435227  0.8090299   0.031483026  1.1445557
## Test set     -8.603157e+00 10.3822584  8.6031574 -10.315422546 10.3154225
## Training set  1.305267e-02  0.8090733  0.6158147  -0.014935041  0.8737645
## Test set     -8.205216e+00  9.9776680  8.2052156  -9.843121038  9.8431210
## Training set  9.656851e-04  0.9475482  0.6883235  -0.028890263  0.9686066
## Test set     -2.910696e+00  3.9930551  2.9106956  -3.389624529  3.3896245
##                    MASE        ACF1 Theil's U
## Training set 1.2657454  0.94843626        NA
## Test set     2.7537311  0.42429116  9.689557
## Training set 0.1326351  0.07489653        NA
## Test set     0.2935344  0.42429116  1.224775
## Training set 1.0000000  0.79681952        NA
## Test set     0.6714215  0.29812931  2.318251
## Training set 0.2134477  0.78592295        NA
## Test set     0.3076841  0.44862792  1.892951
## Training set 0.1862304  0.77616906        NA
## Test set     0.2998442  0.38271319  1.805318
## Training set 0.1146538  0.11073137        NA
## Test set     1.3074388  0.83806826  4.204542
## Training set 0.1304341  0.07868901        NA
## Test set     0.2935361  0.42429116  1.224779
## Training set 0.1146272  0.08221281        NA
## Test set     0.4076682  0.45506900  2.200972
## Training set 0.1129682  0.07482566        NA
## Test set     0.3589581 -0.13109590  2.921910
## Training set 0.1328814  0.49839658        NA
## Test set     1.4130493  0.82611868  4.460046
## Training set 0.1011462  0.01198980        NA
## Test set     1.3476883  0.82765257  4.285148
## Training set 0.1130556  0.07441283        NA
## Test set     0.4780752  0.44378624  2.404093
```

```
row.names(a.table)<-c('Mean training','Mean test', 'Naive training', 'Naive test', 'S. Naive tra
ining', 'S. Naive test' ,'Linear training', 'Linear test','season-trend training', 'season-trend
test', 'ets training', 'ets test',"ses training", "ses test",'STL training', 'STL test',"Holt's
 Linear training", "Holt's Linear test", 'Add training', 'Add test','Multi training', 'Multi tes
t','ARIMA training', 'ARIMA test')

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

```
##                                    ME       RMSE        MAE           MPE
## Mean training           4.503053e-15  8.9032051  7.7063175  -1.574440099
## Mean test               1.676571e+01 16.8872841 16.7657148  19.128589860
## Naive training          4.874334e-01  1.0726601  0.8075307   0.667174022
## Naive test              2.576216e-01  2.0390045  1.7871442   0.239925930
## S. Naive training       6.088363e+00  6.5115185  6.0883632   8.140173840
## S. Naive test           3.263810e+00  4.4078055  4.0878580   3.676284667
## Linear training         1.184961e-16  1.5728594  1.2995469  -0.032583082
## Linear test            -1.450725e+00  3.1333447  1.8732924  -1.719282838
## season-trend training  -1.184455e-16  1.4016964  1.1338384  -0.022381915
## season-trend test      -1.601785e+00  2.9902011  1.8255601  -1.887899865
## ets training            3.771897e-03  0.9523994  0.6980538  -0.024680434
## ets test               -7.960162e+00  9.8077564  7.9601624  -9.551741436
## ses training            4.793308e-01  1.0637119  0.7941300   0.656074513
## ses test                2.576823e-01  2.0390122  1.7871544   0.239995419
## STL training            2.738604e-02  0.9168425  0.6978920   0.007787811
## STL test               -2.430410e+00  3.6501703  2.4820320  -2.833431113
## Holt's Linear training -4.784057e-04  0.9475770  0.6877916  -0.031136676
## Holt's Linear test     -2.185467e+00  2.3599983  2.1854675  -2.548379708
## Add training            2.287631e-02  1.0435227  0.8090299   0.031483026
## Add test               -8.603157e+00 10.3822584  8.6031574 -10.315422546
## Multi training          1.305267e-02  0.8090733  0.6158147  -0.014935041
## Multi test             -8.205216e+00  9.9776680  8.2052156  -9.843121038
## ARIMA training          9.656851e-04  0.9475482  0.6883235  -0.028890263
## ARIMA test             -2.910696e+00  3.9930551  2.9106956  -3.389624529
##                             MAPE      MASE        ACF1 Theil's U
## Mean training          11.0144268 1.2657454  0.94843626        NA
## Mean test              19.1285899 2.7537311  0.42429116  9.689557
## Naive training          1.1188413 0.1326351  0.07489653        NA
## Naive test              2.0567580 0.2935344  0.42429116  1.224775
## S. Naive training       8.1401738 1.0000000  0.79681952        NA
## S. Naive test           4.6707592 0.6714215  0.29812931  2.318251
## Linear training         1.8686463 0.2134477  0.78592295        NA
## Linear test             2.1971936 0.3076841  0.44862792  1.892951
## season-trend training   1.6508397 0.1862304  0.77616906        NA
## season-trend test       2.1424698 0.2998442  0.38271319  1.805318
## ets training            0.9840451 0.1146538  0.11073137        NA
## ets test                9.5517414 1.3074388  0.83806826  4.204542
## ses training            1.1002801 0.1304341  0.07868901        NA
## ses test                2.0567682 0.2935361  0.42429116  1.224779
## STL training            0.9767164 0.1146272  0.08221281        NA
## STL test                2.8929399 0.4076682  0.45506900  2.200972
## Holt's Linear training  0.9677082 0.1129682  0.07482566        NA
## Holt's Linear test      2.5483797 0.3589581 -0.13109590  2.921910
## Add training            1.1445557 0.1328814  0.49839658        NA
## Add test               10.3154225 1.4130493  0.82611868  4.460046
## Multi training          0.8737645 0.1011462  0.01198980        NA
## Multi test              9.8431210 1.3476883  0.82765257  4.285148
## ARIMA training          0.9686066 0.1130556  0.07441283        NA
## ARIMA test              3.3896245 0.4780752  0.44378624  2.404093
```