Session-23 Assignment-1

setwd("C:/Users/Seshan/Desktop/sv R related/acadgild/assignments/session 23/New folder")  
library(readr)  
AAPLMay10toAug102018 <- read.csv("AAPLMay10toAug102018.csv")  
View(AAPLMay10toAug102018)  
df<-AAPLMay10toAug102018  
head(df)

## Date Open High Low Close Adj.Close Volume  
## 1 2018-05-10 187.74 190.37 187.65 190.04 188.6484 27989300  
## 2 2018-05-11 189.49 190.06 187.45 188.59 187.9309 26212200  
## 3 2018-05-14 189.01 189.53 187.86 188.15 187.4924 20778800  
## 4 2018-05-15 186.78 187.07 185.10 186.44 185.7884 23695200  
## 5 2018-05-16 186.07 188.46 186.00 188.18 187.5223 19183100  
## 6 2018-05-17 188.00 188.91 186.36 186.99 186.3365 17294000

str(df)

## 'data.frame': 65 obs. of 7 variables:  
## $ Date : Factor w/ 65 levels "2018-05-10","2018-05-11",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ Open : num 188 189 189 187 186 ...  
## $ High : num 190 190 190 187 188 ...  
## $ Low : num 188 187 188 185 186 ...  
## $ Close : num 190 189 188 186 188 ...  
## $ Adj.Close: num 189 188 187 186 188 ...  
## $ Volume : int 27989300 26212200 20778800 23695200 19183100 17294000 18297700 18400800 15240700 19467900 ...

new\_date <- as.Date(df$Date)  
new\_date

## [1] "2018-05-10" "2018-05-11" "2018-05-14" "2018-05-15" "2018-05-16"  
## [6] "2018-05-17" "2018-05-18" "2018-05-21" "2018-05-22" "2018-05-23"  
## [11] "2018-05-24" "2018-05-25" "2018-05-29" "2018-05-30" "2018-05-31"  
## [16] "2018-06-01" "2018-06-04" "2018-06-05" "2018-06-06" "2018-06-07"  
## [21] "2018-06-08" "2018-06-11" "2018-06-12" "2018-06-13" "2018-06-14"  
## [26] "2018-06-15" "2018-06-18" "2018-06-19" "2018-06-20" "2018-06-21"  
## [31] "2018-06-22" "2018-06-25" "2018-06-26" "2018-06-27" "2018-06-28"  
## [36] "2018-06-29" "2018-07-02" "2018-07-03" "2018-07-05" "2018-07-06"  
## [41] "2018-07-09" "2018-07-10" "2018-07-11" "2018-07-12" "2018-07-13"  
## [46] "2018-07-16" "2018-07-17" "2018-07-18" "2018-07-19" "2018-07-20"  
## [51] "2018-07-23" "2018-07-24" "2018-07-25" "2018-07-26" "2018-07-27"  
## [56] "2018-07-30" "2018-07-31" "2018-08-01" "2018-08-02" "2018-08-03"  
## [61] "2018-08-06" "2018-08-07" "2018-08-08" "2018-08-09" "2018-08-10"

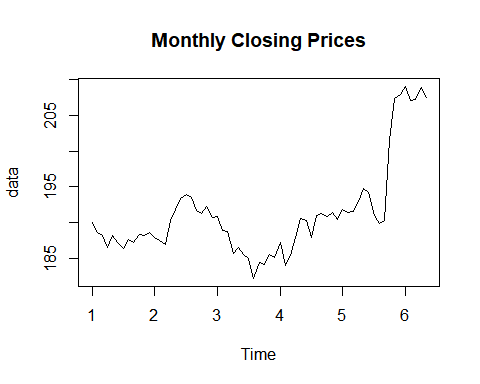
str(df)

## 'data.frame': 65 obs. of 7 variables:  
## $ Date : Factor w/ 65 levels "2018-05-10","2018-05-11",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ Open : num 188 189 189 187 186 ...  
## $ High : num 190 190 190 187 188 ...  
## $ Low : num 188 187 188 185 186 ...  
## $ Close : num 190 189 188 186 188 ...  
## $ Adj.Close: num 189 188 187 186 188 ...  
## $ Volume : int 27989300 26212200 20778800 23695200 19183100 17294000 18297700 18400800 15240700 19467900 ...

format(new\_date,format="%B %d %Y")

## [1] "May 10 2018" "May 11 2018" "May 14 2018" "May 15 2018"   
## [5] "May 16 2018" "May 17 2018" "May 18 2018" "May 21 2018"   
## [9] "May 22 2018" "May 23 2018" "May 24 2018" "May 25 2018"   
## [13] "May 29 2018" "May 30 2018" "May 31 2018" "June 01 2018"   
## [17] "June 04 2018" "June 05 2018" "June 06 2018" "June 07 2018"   
## [21] "June 08 2018" "June 11 2018" "June 12 2018" "June 13 2018"   
## [25] "June 14 2018" "June 15 2018" "June 18 2018" "June 19 2018"   
## [29] "June 20 2018" "June 21 2018" "June 22 2018" "June 25 2018"   
## [33] "June 26 2018" "June 27 2018" "June 28 2018" "June 29 2018"   
## [37] "July 02 2018" "July 03 2018" "July 05 2018" "July 06 2018"   
## [41] "July 09 2018" "July 10 2018" "July 11 2018" "July 12 2018"   
## [45] "July 13 2018" "July 16 2018" "July 17 2018" "July 18 2018"   
## [49] "July 19 2018" "July 20 2018" "July 23 2018" "July 24 2018"   
## [53] "July 25 2018" "July 26 2018" "July 27 2018" "July 30 2018"   
## [57] "July 31 2018" "August 01 2018" "August 02 2018" "August 03 2018"  
## [61] "August 06 2018" "August 07 2018" "August 08 2018" "August 09 2018"  
## [65] "August 10 2018"

# %d - day as number 1-31  
# %a - weekday such as Mon  
# %A- complete day name ex.Monday  
# %m - month as a number  
# %b - short form of month Jan, Feb  
# %B - full form of month, January  
# %y - two digit year  
# %Y- four digit year  
  
data = ts(df$Close,frequency =12)  
  
plot(data,main="Monthly Closing Prices")



# Additive Time Series  
# Trend + Seasonality+ Cyclicity+ error   
# Multiplicative Time Series  
## Trend \* Seasonality \* Cyclicity \* error  
  
# additive model is easy to explain, easy to forecast and interpret  
# multiplicate models can be converted to additive models using log of the time series  
log(data)

## Jan Feb Mar Apr May Jun Jul Aug  
## 1 5.247235 5.239575 5.237239 5.228109 5.237399 5.231055 5.227412 5.234472  
## 2 5.235910 5.233779 5.230413 5.248286 5.256610 5.264295 5.267755 5.265071  
## 3 5.251226 5.240900 5.240370 5.224079 5.228431 5.222839 5.219923 5.204940  
## 4 5.232071 5.214501 5.222516 5.236282 5.250072 5.248865 5.235803 5.252430  
## 5 5.256870 5.254574 5.255462 5.262690 5.272076 5.268940 5.252169 5.246550  
## 6 5.342669 5.333250 5.333926 5.341760 5.335276   
## Sep Oct Nov Dec  
## 1 5.231964 5.238355 5.237239 5.239522  
## 2 5.255932 5.253477 5.258953 5.250701  
## 3 5.217270 5.215805 5.223055 5.220950  
## 4 5.254000 5.251802 5.254627 5.249127  
## 5 5.248549 5.305789 5.334601 5.337490  
## 6

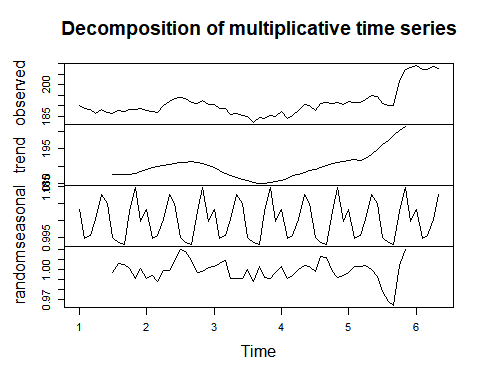
# assumption for time series forecst:  
#1- the time series should be stationary  
  
# Identify the stationarity of a time series  
#1- mean value of the time series is constant over time, the trend should not be present in the series  
#2- the variance does not increase over time  
#3- the seasonality impact is minimal, deseasonalization of the time series data  
  
decompose(data) # default method is additive

## $x  
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 1 190.04 188.59 188.15 186.44 188.18 186.99 186.31 187.63 187.16 188.36  
## 2 187.90 187.50 186.87 190.24 191.83 193.31 193.98 193.46 191.70 191.23  
## 3 190.80 188.84 188.74 185.69 186.50 185.46 184.92 182.17 184.43 184.16  
## 4 187.18 183.92 185.40 187.97 190.58 190.35 187.88 191.03 191.33 190.91  
## 5 191.88 191.44 191.61 193.00 194.82 194.21 190.98 189.91 190.29 201.50  
## 6 209.07 207.11 207.25 208.88 207.53   
## Nov Dec  
## 1 188.15 188.58  
## 2 192.28 190.70  
## 3 185.50 185.11  
## 4 191.45 190.40  
## 5 207.39 207.99  
## 6   
##   
## $seasonal  
## Jan Feb Mar Apr May Jun  
## 1 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961 0.94269711  
## 2 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961 0.94269711  
## 3 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961 0.94269711  
## 4 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961 0.94269711  
## 5 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961 0.94269711  
## 6 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961   
## Jul Aug Sep Oct Nov Dec  
## 1 -0.94316286 -1.23007568 -1.40158058 0.50225588 1.87600261 -0.03032988  
## 2 -0.94316286 -1.23007568 -1.40158058 0.50225588 1.87600261 -0.03032988  
## 3 -0.94316286 -1.23007568 -1.40158058 0.50225588 1.87600261 -0.03032988  
## 4 -0.94316286 -1.23007568 -1.40158058 0.50225588 1.87600261 -0.03032988  
## 5 -0.94316286 -1.23007568 -1.40158058 0.50225588 1.87600261 -0.03032988  
## 6   
##   
## $trend  
## Jan Feb Mar Apr May Jun Jul Aug  
## 1 NA NA NA NA NA NA 187.7925 187.6579  
## 2 188.9729 189.5354 189.9675 190.2762 190.5679 190.8283 191.0375 191.2142  
## 3 189.5708 188.7229 187.9496 187.3521 186.7750 186.2596 185.8758 185.5200  
## 4 186.0975 186.5900 187.2467 187.8154 188.3446 188.8129 189.2292 189.7383  
## 5 191.7925 191.8750 191.7850 192.1829 193.2883 194.6854 196.1346 197.5038  
## 6 NA NA NA NA NA   
## Sep Oct Nov Dec  
## 1 187.5592 187.6642 187.9746 188.3900  
## 2 191.3479 191.2362 190.8246 190.2754  
## 3 185.1758 185.1317 185.3967 185.7704  
## 4 190.3104 190.7788 191.1650 191.5025  
## 5 198.8083 200.1217 201.3129 NA  
## 6   
##   
## $random  
## Jan Feb Mar Apr May Jun  
## 1 NA NA NA NA NA NA  
## 2 -1.66123862 -1.03633707 -2.27207090 -0.11133461 -0.18321332 1.53896851  
## 3 0.64085296 1.11615847 1.61585564 -1.73717174 -1.72029982 -1.74227386  
## 4 0.49417750 -1.67092228 -1.02123907 0.07949335 0.79011876 0.59439235  
## 5 -0.50081275 0.56407997 0.65043343 0.74199210 0.08637347 -1.41810790  
## 6 NA NA NA NA NA   
## Jul Aug Sep Oct Nov Dec  
## 1 -0.53933810 1.20216485 1.00241853 0.19357891 -1.70059198 0.22033175  
## 2 3.88565965 3.47591660 1.75366124 -0.50850984 -0.42058669 0.45491017  
## 3 -0.01267264 -2.11992615 0.65574078 -1.47391751 -1.77266827 -0.63008483  
## 4 -0.40599889 2.52174060 2.42116470 -0.37100334 -1.59100723 -1.07217800  
## 5 -4.21142614 -6.36367202 -7.11676138 0.87607566 4.20107806 NA  
## 6   
##   
## $figure  
## [1] 0.58831620 -0.99907935 -0.82543364 0.07509057 1.44529961  
## [6] 0.94269711 -0.94316286 -1.23007568 -1.40158058 0.50225588  
## [11] 1.87600261 -0.03032988  
##   
## $type  
## [1] "additive"  
##   
## attr(,"class")  
## [1] "decomposed.ts"

decompose(data, type='multi')

## $x  
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 1 190.04 188.59 188.15 186.44 188.18 186.99 186.31 187.63 187.16 188.36  
## 2 187.90 187.50 186.87 190.24 191.83 193.31 193.98 193.46 191.70 191.23  
## 3 190.80 188.84 188.74 185.69 186.50 185.46 184.92 182.17 184.43 184.16  
## 4 187.18 183.92 185.40 187.97 190.58 190.35 187.88 191.03 191.33 190.91  
## 5 191.88 191.44 191.61 193.00 194.82 194.21 190.98 189.91 190.29 201.50  
## 6 209.07 207.11 207.25 208.88 207.53   
## Nov Dec  
## 1 188.15 188.58  
## 2 192.28 190.70  
## 3 185.50 185.11  
## 4 191.45 190.40  
## 5 207.39 207.99  
## 6   
##   
## $seasonal  
## Jan Feb Mar Apr May Jun Jul  
## 1 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## 2 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## 3 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## 4 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## 5 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## 6 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902   
## Aug Sep Oct Nov Dec  
## 1 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
## 2 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
## 3 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
## 4 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
## 5 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
## 6   
##   
## $trend  
## Jan Feb Mar Apr May Jun Jul Aug  
## 1 NA NA NA NA NA NA 187.7925 187.6579  
## 2 188.9729 189.5354 189.9675 190.2762 190.5679 190.8283 191.0375 191.2142  
## 3 189.5708 188.7229 187.9496 187.3521 186.7750 186.2596 185.8758 185.5200  
## 4 186.0975 186.5900 187.2467 187.8154 188.3446 188.8129 189.2292 189.7383  
## 5 191.7925 191.8750 191.7850 192.1829 193.2883 194.6854 196.1346 197.5038  
## 6 NA NA NA NA NA   
## Sep Oct Nov Dec  
## 1 187.5592 187.6642 187.9746 188.3900  
## 2 191.3479 191.2362 190.8246 190.2754  
## 3 185.1758 185.1317 185.3967 185.7704  
## 4 190.3104 190.7788 191.1650 191.5025  
## 5 198.8083 200.1217 201.3129 NA  
## 6   
##   
## $random  
## Jan Feb Mar Apr May Jun Jul  
## 1 NA NA NA NA NA NA 0.9969622  
## 2 0.9912315 0.9945689 0.9880225 0.9994619 0.9990398 1.0080115 1.0203733  
## 3 1.0033553 1.0059892 1.0086237 0.9907840 0.9910057 0.9907993 0.9997277  
## 4 1.0026902 0.9909792 0.9944940 1.0004751 1.0042463 1.0031716 0.9977305  
## 5 0.9973463 1.0030862 1.0034832 1.0039024 1.0003316 0.9926410 0.9784856  
## 6 NA NA NA NA NA   
## Aug Sep Oct Nov Dec  
## 1 1.0061748 1.0049329 1.0011569 0.9914980 1.0011829  
## 2 1.0181439 1.0089291 0.9974259 0.9981287 1.0024060  
## 3 0.9881529 1.0030199 0.9922233 0.9911258 0.9966185  
## 4 1.0131751 1.0124715 0.9981447 0.9920504 0.9944160  
## 5 0.9676326 0.9639259 1.0043284 1.0204763 NA  
## 6   
##   
## $figure  
## [1] 1.0031182 0.9946632 0.9956196 1.0003478 1.0075902 1.0049535 0.9951287  
## [8] 0.9937153 0.9929736 1.0025480 1.0095161 0.9998259  
##   
## $type  
## [1] "multiplicative"  
##   
## attr(,"class")  
## [1] "decomposed.ts"

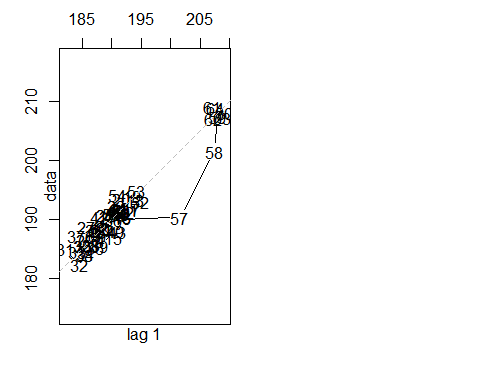
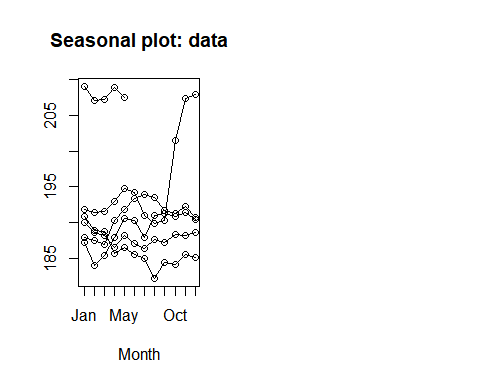
par(mfrow=c(1,2))  
plot(decompose(data, type='multi'))  
library(forecast)



seasonplot(data)  
  
lag(data,10)

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 0 190.04 188.59 188.15 186.44 188.18 186.99 186.31 187.63  
## 1 188.15 188.58 187.90 187.50 186.87 190.24 191.83 193.31 193.98 193.46  
## 2 192.28 190.70 190.80 188.84 188.74 185.69 186.50 185.46 184.92 182.17  
## 3 185.50 185.11 187.18 183.92 185.40 187.97 190.58 190.35 187.88 191.03  
## 4 191.45 190.40 191.88 191.44 191.61 193.00 194.82 194.21 190.98 189.91  
## 5 207.39 207.99 209.07 207.11 207.25 208.88 207.53   
## Nov Dec  
## 0 187.16 188.36  
## 1 191.70 191.23  
## 2 184.43 184.16  
## 3 191.33 190.91  
## 4 190.29 201.50  
## 5

lag.plot(data)



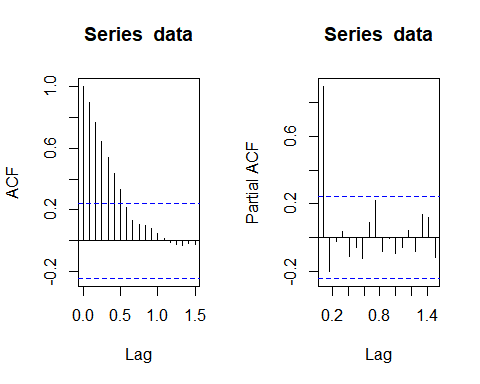
# Calculation of Autocorrelation and Partial Autocorrelation  
data

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 1 190.04 188.59 188.15 186.44 188.18 186.99 186.31 187.63 187.16 188.36  
## 2 187.90 187.50 186.87 190.24 191.83 193.31 193.98 193.46 191.70 191.23  
## 3 190.80 188.84 188.74 185.69 186.50 185.46 184.92 182.17 184.43 184.16  
## 4 187.18 183.92 185.40 187.97 190.58 190.35 187.88 191.03 191.33 190.91  
## 5 191.88 191.44 191.61 193.00 194.82 194.21 190.98 189.91 190.29 201.50  
## 6 209.07 207.11 207.25 208.88 207.53   
## Nov Dec  
## 1 188.15 188.58  
## 2 192.28 190.70  
## 3 185.50 185.11  
## 4 191.45 190.40  
## 5 207.39 207.99  
## 6

ac<-acf(data)  
  
ac$acf

## , , 1  
##   
## [,1]  
## [1,] 1.000000000  
## [2,] 0.897834549  
## [3,] 0.766959609  
## [4,] 0.642380728  
## [5,] 0.540362058  
## [6,] 0.435258811  
## [7,] 0.329717557  
## [8,] 0.213959913  
## [9,] 0.130089131  
## [10,] 0.108939188  
## [11,] 0.096442343  
## [12,] 0.081448406  
## [13,] 0.048226570  
## [14,] 0.012083704  
## [15,] -0.008036572  
## [16,] -0.024501683  
## [17,] -0.027889568  
## [18,] -0.018651269  
## [19,] -0.020409708

# data time series may not have stationarity  
  
pac<-pacf(data)



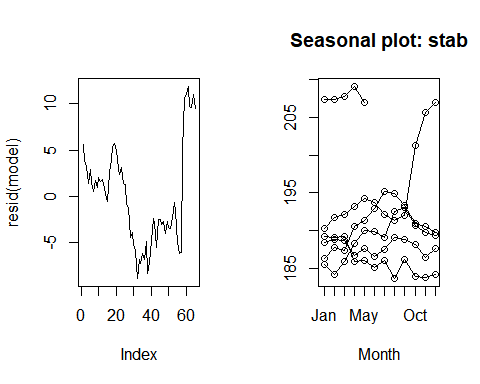
pac$acf

## , , 1  
##   
## [,1]  
## [1,] 0.897834549  
## [2,] -0.201901271  
## [3,] -0.021388111  
## [4,] 0.033830089  
## [5,] -0.113123217  
## [6,] -0.061245804  
## [7,] -0.127001529  
## [8,] 0.089593010  
## [9,] 0.222548229  
## [10,] -0.084860931  
## [11,] -0.006016842  
## [12,] -0.096866419  
## [13,] -0.060046996  
## [14,] 0.039563483  
## [15,] -0.084282971  
## [16,] 0.133672152  
## [17,] 0.117993466  
## [18,] -0.118439370

# looking at the ACF and PACF graph we can conclude that the time series is not stationary  
  
model <- lm(data~c(1:length(data)))  
  
summary(model)

##   
## Call:  
## lm(formula = data ~ c(1:length(data)))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.8666 -4.0286 -0.5626 2.9954 11.8853   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 184.25256 1.33272 138.253 < 2e-16 \*\*\*  
## c(1:length(data)) 0.21200 0.03511 6.039 9.13e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.31 on 63 degrees of freedom  
## Multiple R-squared: 0.3666, Adjusted R-squared: 0.3566   
## F-statistic: 36.46 on 1 and 63 DF, p-value: 9.126e-08

plot(resid(model),type='l')  
  
# the series is not stationary  
  
# deseasonalize the time series  
  
  
tbl <- stl(data,'periodic')  
  
stab<-seasadj(tbl)  
  
seasonplot(stab,12)



# statistically we need to test out if the series is stationary or not  
# Augmented Dickey Fuller Test  
  
library(tseries)  
  
adf.test(data)

##   
## Augmented Dickey-Fuller Test  
##   
## data: data  
## Dickey-Fuller = -0.86015, Lag order = 3, p-value = 0.9516  
## alternative hypothesis: stationary

# if the p-value is less than 0.05, then the time series is stationary, else not  
  
# Time Series Forecasting Models  
  
# Simple Exponential Smoothing  
# Double Expo. Smoothing  
# Tripple Expo. Smoothing   
# AR-I-MA model  
  
#PACF- p   
#diff - d  
#ACF- q  
  
  
model2<-auto.arima(data)  
accuracy(model2)

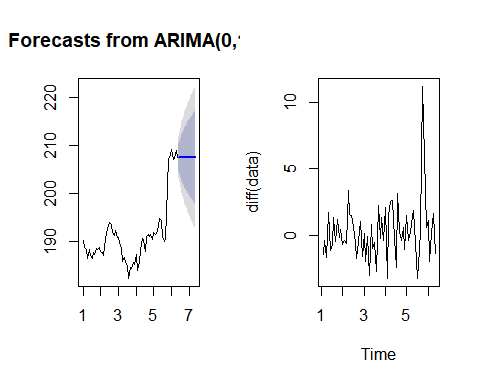
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.2720007 2.171997 1.452924 0.1307485 0.7549219 0.2304089  
## ACF1  
## Training set 0.1700406

plot(forecast(model2,h=12))  
  
adf.test(diff(data))

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

##   
## Augmented Dickey-Fuller Test  
##   
## data: diff(data)  
## Dickey-Fuller = -4.5932, Lag order = 3, p-value = 0.01  
## alternative hypothesis: stationary

plot(diff(data))



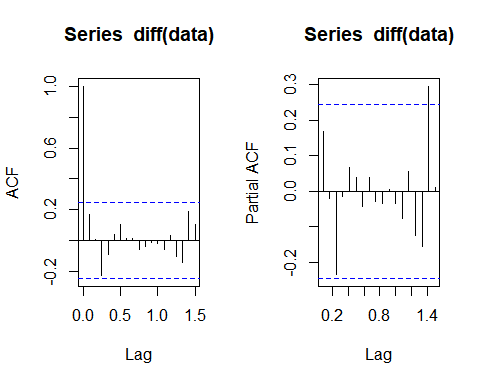
diff(data,differences = 3)

## Jan Feb Mar Apr May Jun  
## 1 -2.279985 4.719973 -6.379962  
## 2 -1.750031 1.390030 -0.510025 4.230026 -5.780028 1.670012  
## 3 4.310013 -3.740021 3.920029 -4.810028 6.810013 -5.709992  
## 4 4.189986 -7.789978 10.069978 -3.649980 -1.050017 -2.879991  
## 5 4.120010 -4.450028 2.530016 0.609998 -0.789992 -2.860015  
## 6 5.769989 -3.520004 5.140013 -0.609999 -4.470017   
## Jul Aug Sep Oct Nov Dec  
## 1 3.439960 1.490033 -3.790022 3.460006 -3.080002 2.050019  
## 2 -0.699997 -0.379989 -0.050034 2.530030 0.229995 -4.150009  
## 3 2.349975 -2.709975 7.219986 -7.539979 4.139969 -3.339980  
## 4 0.599992 7.860000 -8.469986 2.129990 1.679992 -2.549987  
## 5 -0.190004 4.780030 -0.710038 9.380037 -16.150026 0.030015  
## 6

#running a model on diff data  
model3<-auto.arima(diff(data))  
  
accuracy(model3)

## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 0.2732813 2.188771 1.472657 100 100 0.7590256 0.1695623

acf(diff(data))  
  
pacf(diff(data))



#taking random order  
model4 <- Arima(diff(data),order=c(4,0,5))  
model4

## Series: diff(data)   
## ARIMA(4,0,5) with non-zero mean   
##   
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs produced

## ar1 ar2 ar3 ar4 ma1 ma2 ma3 ma4  
## -0.2803 -1.4150 -0.0641 -0.4746 0.4821 1.7941 0.1249 0.6766  
## s.e. NaN 0.1265 NaN 0.1483 NaN NaN NaN NaN  
## ma5 mean  
## -0.2638 0.2614  
## s.e. 0.1724 0.2747  
##   
## sigma^2 estimated as 4.095: log likelihood=-133.78  
## AIC=289.56 AICc=294.64 BIC=313.31

accuracy(model4)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0006347165 1.858786 1.406267 92.29461 131.9952 0.7248077  
## ACF1  
## Training set 0.01349565

model5 <- Arima(diff(data),order=c(4,0,4))  
model5

## Series: diff(data)   
## ARIMA(4,0,4) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ma1 ma2 ma3 ma4  
## 0.4456 0.0444 -0.1777 0.5375 -0.3143 -0.0540 -0.0711 -0.5606  
## s.e. 0.6682 0.5270 0.4503 0.4737 0.6114 0.5523 0.3712 0.6958  
## mean  
## 0.2479  
## s.e. 0.1483  
##   
## sigma^2 estimated as 4.847: log likelihood=-137.17  
## AIC=294.34 AICc=298.49 BIC=315.93

accuracy(model5)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.1080514 2.040884 1.479593 108.6686 139.0467 0.7626008  
## ACF1  
## Training set 0.01210928

model6<-Arima(data,order=c(3,0,5))  
model6

## Series: data   
## ARIMA(3,0,5) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 ma4 ma5  
## 0.7731 -0.7050 0.8166 0.3971 1.1226 0.1251 0.0658 -0.1386  
## s.e. 0.2638 0.1838 0.1065 0.2880 0.2760 0.2585 0.2129 0.1412  
## mean  
## 193.6317  
## s.e. 4.8020  
##   
## sigma^2 estimated as 4.67: log likelihood=-140.1  
## AIC=300.2 AICc=304.27 BIC=321.94

accuracy(model6)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0750624 2.005746 1.468935 0.02667925 0.763838 0.2329479  
## ACF1  
## Training set 0.02072704

model7<-Arima(diff(data),order=c(4,0,4))  
model7

## Series: diff(data)   
## ARIMA(4,0,4) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ma1 ma2 ma3 ma4  
## 0.4456 0.0444 -0.1777 0.5375 -0.3143 -0.0540 -0.0711 -0.5606  
## s.e. 0.6682 0.5270 0.4503 0.4737 0.6114 0.5523 0.3712 0.6958  
## mean  
## 0.2479  
## s.e. 0.1483  
##   
## sigma^2 estimated as 4.847: log likelihood=-137.17  
## AIC=294.34 AICc=298.49 BIC=315.93

accuracy(model7)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.1080514 2.040884 1.479593 108.6686 139.0467 0.7626008  
## ACF1  
## Training set 0.01210928

model8<-Arima(diff(data),order=c(0,0,1))  
model8

## Series: diff(data)   
## ARIMA(0,0,1) with non-zero mean   
##   
## Coefficients:  
## ma1 mean  
## 0.1565 0.2648  
## s.e. 0.1127 0.3090  
##   
## sigma^2 estimated as 4.734: log likelihood=-139.56  
## AIC=285.12 AICc=285.52 BIC=291.59

accuracy(model8)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.00374748 2.141417 1.506563 111.3245 117.7739 0.7765016  
## ACF1  
## Training set 0.01275007

model9<-Arima(diff(data),order=c(1,0,0))  
model9

## Series: diff(data)   
## ARIMA(1,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 mean  
## 0.1702 0.2626  
## s.e. 0.1233 0.3214  
##   
## sigma^2 estimated as 4.726: log likelihood=-139.51  
## AIC=285.01 AICc=285.41 BIC=291.49

accuracy(model9)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.004944389 2.139586 1.507873 110.3545 118.8004 0.7771767  
## ACF1  
## Training set 0.004464458

model10<-Arima(diff(data),order=c(1,0,1))  
model10

## Series: diff(data)   
## ARIMA(1,0,1) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 mean  
## 0.1386 0.0329 0.2629  
## s.e. 0.3916 0.3809 0.3199  
##   
## sigma^2 estimated as 4.802: log likelihood=-139.5  
## AIC=287 AICc=287.68 BIC=295.64

accuracy(model10)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.004743348 2.139462 1.508693 110.7547 119.1916 0.7775994  
## ACF1  
## Training set 0.001901816

model11<-Arima(diff(data),order=c(1,0,2))  
model11

## Series: diff(data)   
## ARIMA(1,0,2) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 ma2 mean  
## -0.4792 0.6771 0.2237 0.2612  
## s.e. 0.6182 0.5892 0.1329 0.3388  
##   
## sigma^2 estimated as 4.782: log likelihood=-138.87  
## AIC=287.74 AICc=288.77 BIC=298.53

accuracy(model11)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.003565339 2.117405 1.510722 105.7682 123.4363 0.7786448  
## ACF1  
## Training set -0.009002027

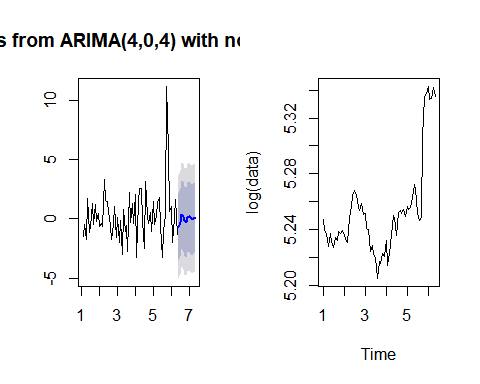
model12<-Arima(diff(data),order=c(1,1,3))  
model12

## Series: diff(data)   
## ARIMA(1,1,3)   
##   
## Coefficients:  
## ar1 ma1 ma2 ma3  
## -0.4163 -0.3667 -0.3947 -0.2341  
## s.e. 0.5693 0.7264 0.5525 0.1834  
##   
## sigma^2 estimated as 4.876: log likelihood=-139.02  
## AIC=288.04 AICc=289.09 BIC=298.75

accuracy(model12)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.2818303 2.120099 1.452976 88.80578 112.3296 0.748882  
## ACF1  
## Training set -0.03927055

# MAPE = mean absolute percentage error (should be < 10%) for a good model  
par(mfrow=c(1,2))  
plot(forecast(model5,h=12))  
  
plot(log(data))



# Holt Winters Exponential Smoothing Model  
  
# if series is stationary then use simple exponential smoothing model  
model4<-HoltWinters(data,beta = F, gamma = F)  
summary(model4)

## Length Class Mode   
## fitted 128 mts numeric   
## x 65 ts numeric   
## alpha 1 -none- numeric   
## beta 1 -none- logical   
## gamma 1 -none- logical   
## coefficients 1 -none- numeric   
## seasonal 1 -none- character  
## SSE 1 -none- numeric   
## call 4 -none- call

model4

## Holt-Winters exponential smoothing without trend and without seasonal component.  
##   
## Call:  
## HoltWinters(x = data, beta = F, gamma = F)  
##   
## Smoothing parameters:  
## alpha: 0.9999498  
## beta : FALSE  
## gamma: FALSE  
##   
## Coefficients:  
## [,1]  
## a 207.5301

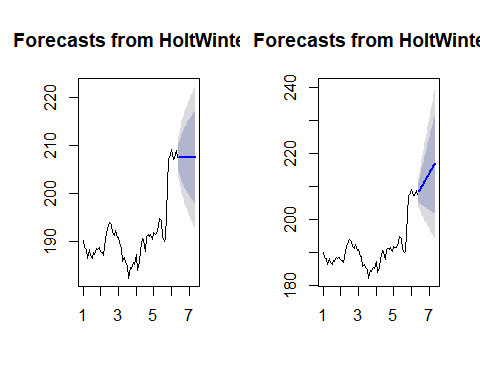
library(forecast)  
plot(forecast(model4,12))  
  
# Holt Winters Exponential Smoothing Model  
  
# if series is not stationary and only trend component is present, then use double exponential smoothing model  
model5<-HoltWinters(data,gamma = F)  
summary(model5)

## Length Class Mode   
## fitted 189 mts numeric   
## x 65 ts numeric   
## alpha 1 -none- numeric   
## beta 1 -none- numeric   
## gamma 1 -none- logical   
## coefficients 2 -none- numeric   
## seasonal 1 -none- character  
## SSE 1 -none- numeric   
## call 3 -none- call

model5

## Holt-Winters exponential smoothing with trend and without seasonal component.  
##   
## Call:  
## HoltWinters(x = data, gamma = F)  
##   
## Smoothing parameters:  
## alpha: 1  
## beta : 0.08842156  
## gamma: FALSE  
##   
## Coefficients:  
## [,1]  
## a 207.5299990  
## b 0.7924614

plot(forecast(model5,12))



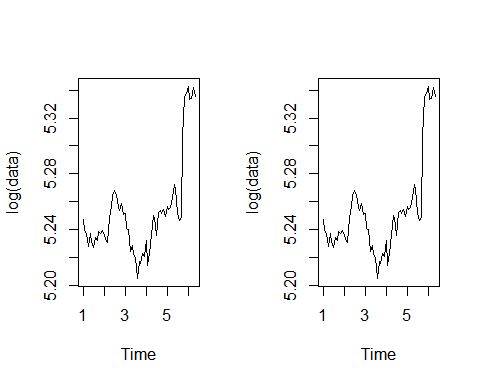
plot(log(data))  
  
# Holt Winters Exponential Smoothing Model  
# if series is not stationary and trend, seasonality component is present, then use tripple exponential smoothing model  
model6<-HoltWinters(data)  
summary(model6)

## Length Class Mode   
## fitted 212 mts numeric   
## x 65 ts numeric   
## alpha 1 -none- numeric   
## beta 1 -none- numeric   
## gamma 1 -none- numeric   
## coefficients 14 -none- numeric   
## seasonal 1 -none- character  
## SSE 1 -none- numeric   
## call 2 -none- call

model6

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = data)  
##   
## Smoothing parameters:  
## alpha: 0.9039743  
## beta : 0  
## gamma: 1  
##   
## Coefficients:  
## [,1]  
## a 206.47238517  
## b 0.33351689  
## s1 2.15703648  
## s2 -0.63318618  
## s3 -0.23391222  
## s4 -0.11991297  
## s5 1.47358988  
## s6 1.27297528  
## s7 0.29666706  
## s8 -0.05308394  
## s9 -1.96662951  
## s10 -2.49227872  
## s11 -0.54306702  
## s12 1.05761383

plot(log(data))



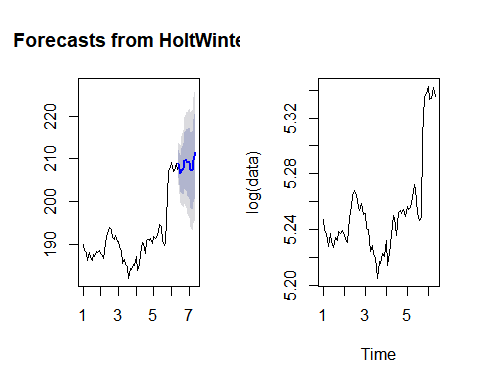
plot(forecast(model6,12))  
  
# MAPE  
# Automatic Exponential Smoothing Model  
model7<-ets(data)  
summary(model7)

## ETS(M,N,N)   
##   
## Call:  
## ets(y = data)   
##   
## Smoothing parameters:  
## alpha = 0.9999   
##   
## Initial states:  
## l = 190.0165   
##   
## sigma: 0.0115  
##   
## AIC AICc BIC   
## 378.0199 378.4133 384.5431   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.2694668 2.171911 1.450364 0.1294136 0.7535737 0.230003  
## ACF1  
## Training set 0.1710693

accuracy(model7)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.2694668 2.171911 1.450364 0.1294136 0.7535737 0.230003  
## ACF1  
## Training set 0.1710693

plot(log(data))



plot(forecast(model7,12))

