STOCK MARKET PREDICTION USING ARIMA

Gowtham Krishnaswamy

library(timeSeries)

## Loading required package: timeDate

#library(tidyverse)  
#library(fPortfolio)  
library(PerformanceAnalytics)

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':  
##   
## time<-

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

##   
## Attaching package: 'PerformanceAnalytics'

## The following objects are masked from 'package:timeDate':  
##   
## kurtosis, skewness

## The following object is masked from 'package:graphics':  
##   
## legend

library(quantmod)

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

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

#library(caTools) # to split the data into test and train sets  
#library(dplyr)  
library(ggplot2)  
#library(PortfolioAnalytics)

library(tidyquant)

## Loading required package: lubridate

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

## == Need to Learn tidyquant? =============================================================================================  
## Business Science offers a 1-hour course - Learning Lab #9: Performance Analysis & Portfolio Optimization with tidyquant!  
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:xts':  
##   
## first, last

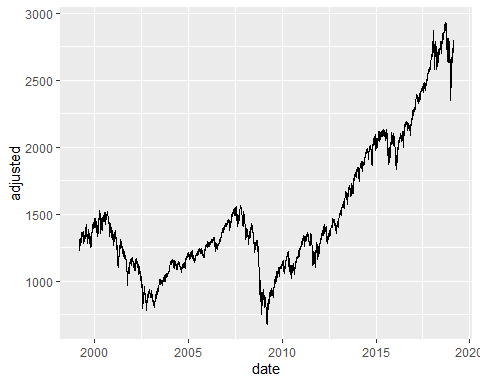
## The following objects are masked from 'package:timeSeries':  
##   
## filter, lag

## The following objects are masked from 'package:stats':  
##   
## filter, lag

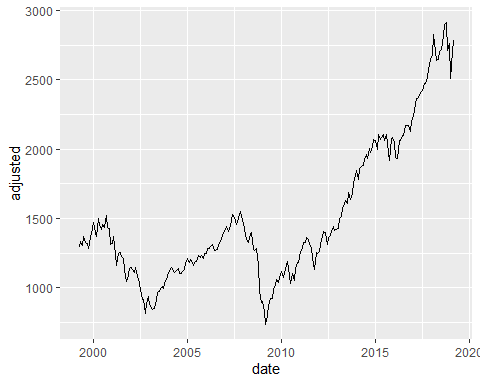
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(magrittr)  
#Load 20 years of stock data using 'GSPC'  
stock\_data <- tq\_get(c("^GSPC"), get = "stock.prices", from = "1999-03-01", to = "2019-03-01")  
  
#Calculating Monthly Returns from the raw data  
stock\_returns\_monthly <- stock\_data %>% group\_by("^GSPC") %>% tq\_transmute(select = adjusted, mutate\_fun = to\_period, period = "months")

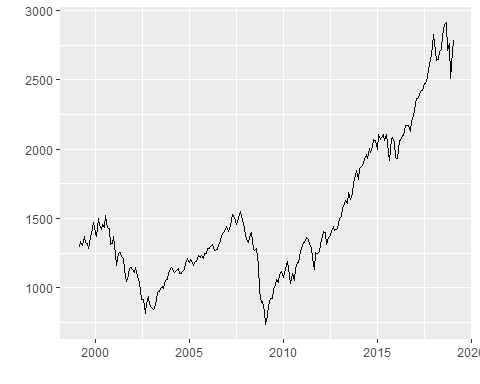
library(ggplot2)  
ggplot(data = stock\_data, aes(x = date, y = adjusted )) + geom\_line()



ggplot(data = stock\_returns\_monthly, aes(x = date, y = adjusted )) + geom\_line()



#Time Series Analysis  
stock\_ts <- ts(stock\_returns\_monthly$adjusted, start = c(1999,3), freq = 12)  
  
library(ggfortify)  
autoplot(stock\_ts)



stock\_data\_train <- ts(stock\_ts, start=c(1999, 3), end=c(2016, 12), freq=12)

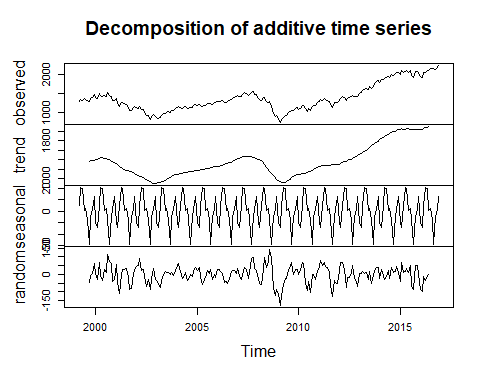
#Test for Stationarity  
  
#Box Ljung test  
Box.test(stock\_data\_train, lag = 20, type = 'Ljung-Box')

##   
## Box-Ljung test  
##   
## data: stock\_data\_train  
## X-squared = 2525.6, df = 20, p-value < 2.2e-16

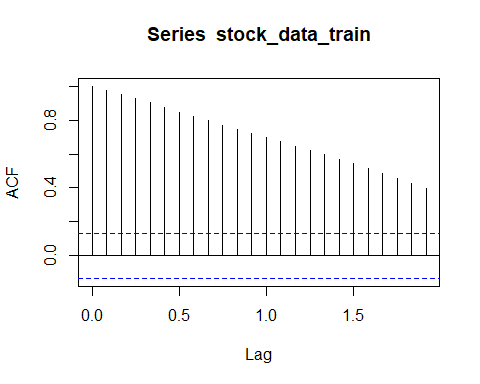
#Augmented Dickey-Fuller Test  
library(tseries)  
adf.test(stock\_data\_train)

##   
## Augmented Dickey-Fuller Test  
##   
## data: stock\_data\_train  
## Dickey-Fuller = -1.68, Lag order = 5, p-value = 0.7102  
## alternative hypothesis: stationary

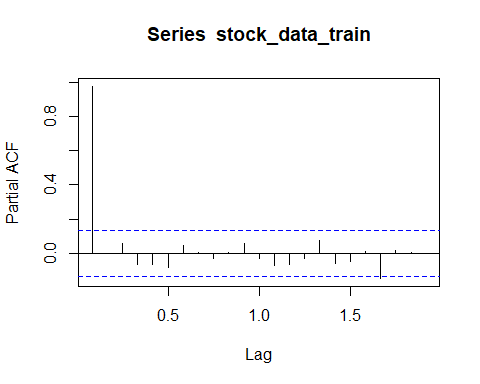
#Decomposing the time series to check Trends, Sesonality, Randomness  
stock\_decompose <- decompose(stock\_data\_train, "additive")  
plot(stock\_decompose)



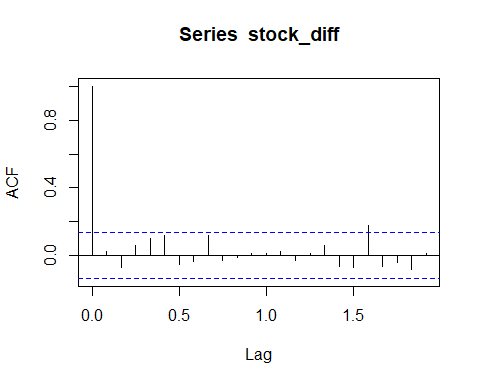
#ACF - AutoCorrelation Function  
acf(stock\_data\_train)



#PACF - Partial AutoCorrelation Function  
pacf(stock\_data\_train)



# Making the data stationary  
stock\_diff <- diff(stock\_data\_train,diff=1)  
acf(stock\_diff)



# Rechecking using the same tests  
  
#Box Ljung test  
Box.test(stock\_diff, lag = 20, type = 'Ljung-Box')

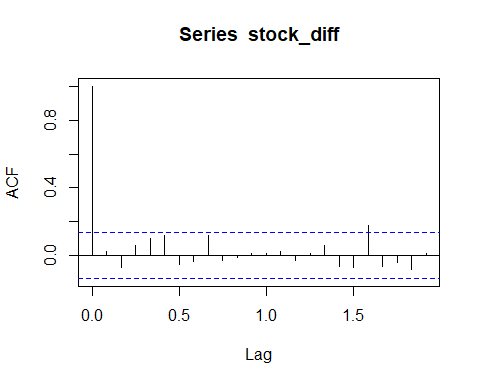
##   
## Box-Ljung test  
##   
## data: stock\_diff  
## X-squared = 23.39, df = 20, p-value = 0.2701

#Augment Dickey-Fuller test  
adf.test(stock\_diff)

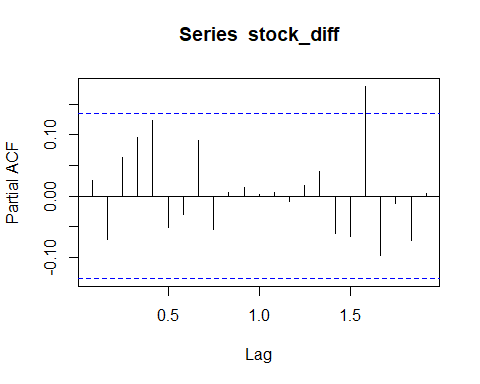
## Warning in adf.test(stock\_diff): p-value smaller than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: stock\_diff  
## Dickey-Fuller = -5.2316, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

acf(stock\_diff)



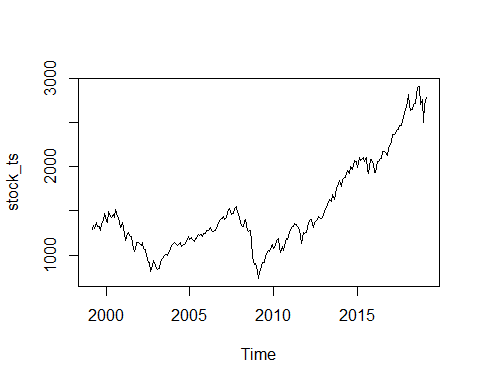
pacf(stock\_diff)



#ARIMA model  
library(forecast)

## Registered S3 methods overwritten by 'forecast':  
## method from   
## autoplot.Arima ggfortify  
## autoplot.acf ggfortify  
## autoplot.ar ggfortify  
## autoplot.bats ggfortify  
## autoplot.decomposed.ts ggfortify  
## autoplot.ets ggfortify  
## autoplot.forecast ggfortify  
## autoplot.stl ggfortify  
## autoplot.ts ggfortify  
## fitted.ar ggfortify  
## fortify.ts ggfortify  
## residuals.ar ggfortify

plot(stock\_ts)



a<-auto.arima(stock\_data\_train,D=1,trace=TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,1,2)(1,1,1)[12] : 2107.077  
## ARIMA(0,1,0)(0,1,0)[12] : 2197.109  
## ARIMA(1,1,0)(1,1,0)[12] : 2124.109  
## ARIMA(0,1,1)(0,1,1)[12] : 2110.377  
## ARIMA(2,1,2)(0,1,1)[12] : Inf  
## ARIMA(2,1,2)(1,1,0)[12] : Inf  
## ARIMA(2,1,2)(2,1,1)[12] : 2091.86  
## ARIMA(2,1,2)(2,1,0)[12] : 2107.946  
## ARIMA(2,1,2)(2,1,2)[12] : 2085.823  
## ARIMA(2,1,2)(1,1,2)[12] : Inf  
## ARIMA(1,1,2)(2,1,2)[12] : 2083.608  
## ARIMA(1,1,2)(1,1,2)[12] : 2093.012  
## ARIMA(1,1,2)(2,1,1)[12] : 2088.645  
## ARIMA(1,1,2)(1,1,1)[12] : 2111.458  
## ARIMA(0,1,2)(2,1,2)[12] : 2082.858  
## ARIMA(0,1,2)(1,1,2)[12] : 2092.926  
## ARIMA(0,1,2)(2,1,1)[12] : 2086.963  
## ARIMA(0,1,2)(1,1,1)[12] : 2108.853  
## ARIMA(0,1,1)(2,1,2)[12] : 2081.006  
## ARIMA(0,1,1)(1,1,2)[12] : 2090.801  
## ARIMA(0,1,1)(2,1,1)[12] : 2085.531  
## ARIMA(0,1,1)(1,1,1)[12] : 2106.82  
## ARIMA(0,1,0)(2,1,2)[12] : 2080.842  
## ARIMA(0,1,0)(1,1,2)[12] : 2089.193  
## ARIMA(0,1,0)(2,1,1)[12] : 2086.089  
## ARIMA(0,1,0)(1,1,1)[12] : 2106.333  
## ARIMA(1,1,0)(2,1,2)[12] : 2080.29  
## ARIMA(1,1,0)(1,1,2)[12] : 2089.444  
## ARIMA(1,1,0)(2,1,1)[12] : 2085.48  
## ARIMA(1,1,0)(1,1,1)[12] : 2107.486  
## ARIMA(2,1,0)(2,1,2)[12] : 2083.33  
## ARIMA(1,1,1)(2,1,2)[12] : 2081.504  
## ARIMA(2,1,1)(2,1,2)[12] : Inf  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(1,1,0)(2,1,2)[12] : Inf  
## ARIMA(0,1,0)(2,1,2)[12] : Inf  
## ARIMA(0,1,1)(2,1,2)[12] : Inf  
## ARIMA(1,1,1)(2,1,2)[12] : Inf  
## ARIMA(0,1,2)(2,1,2)[12] : Inf  
## ARIMA(2,1,0)(2,1,2)[12] : Inf  
## ARIMA(1,1,2)(2,1,2)[12] : Inf  
## ARIMA(1,1,0)(2,1,1)[12] : Inf  
## ARIMA(0,1,1)(2,1,1)[12] : Inf  
## ARIMA(2,1,2)(2,1,2)[12] : Inf  
## ARIMA(0,1,0)(2,1,1)[12] : Inf  
## ARIMA(0,1,2)(2,1,1)[12] : 2216.003  
##   
## Best model: ARIMA(0,1,2)(2,1,1)[12]

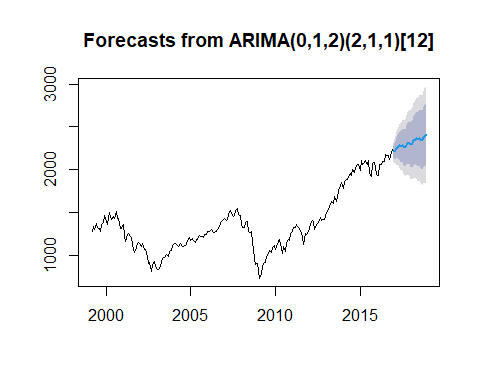
a

## Series: stock\_data\_train   
## ARIMA(0,1,2)(2,1,1)[12]   
##   
## Coefficients:  
## ma1 ma2 sar1 sar2 sma1  
## 0.0427 -0.0461 -0.0283 0.0198 -0.8873  
## s.e. 0.0726 0.0687 0.1012 0.0980 0.1028  
##   
## sigma^2 estimated as 3158: log likelihood=-1101.79  
## AIC=2215.57 AICc=2216 BIC=2235.39

model\_a <- Arima(stock\_data\_train, order = c(0,1,2),seasonal = list(order = c(2,1,1), period = 12))  
summary(model\_a)

## Series: stock\_data\_train   
## ARIMA(0,1,2)(2,1,1)[12]   
##   
## Coefficients:  
## ma1 ma2 sar1 sar2 sma1  
## 0.0427 -0.0461 -0.0283 0.0198 -0.8873  
## s.e. 0.0726 0.0687 0.1012 0.0980 0.1028  
##   
## sigma^2 estimated as 3158: log likelihood=-1101.79  
## AIC=2215.57 AICc=2216 BIC=2235.39  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 3.145023 53.77797 41.45602 0.1519446 3.223742 0.2289596  
## ACF1  
## Training set -0.0003385409

stock\_predict\_a <- forecast(model\_a, h = 24)  
plot(stock\_predict\_a)



b<-auto.arima(stock\_data\_train,trace=TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,1,2)(1,0,1)[12] with drift : Inf  
## ARIMA(0,1,0) with drift : 2299.902  
## ARIMA(1,1,0)(1,0,0)[12] with drift : 2298.588  
## ARIMA(0,1,1)(0,0,1)[12] with drift : 2303.858  
## ARIMA(0,1,0) : 2299.319  
## ARIMA(1,1,0) with drift : 2302.152  
## ARIMA(1,1,0)(2,0,0)[12] with drift : 2286.883  
## ARIMA(1,1,0)(2,0,1)[12] with drift : 2286.116  
## ARIMA(1,1,0)(1,0,1)[12] with drift : 2299.248  
## ARIMA(1,1,0)(2,0,2)[12] with drift : 2287.504  
## ARIMA(1,1,0)(1,0,2)[12] with drift : 2301.058  
## ARIMA(0,1,0)(2,0,1)[12] with drift : 2288.14  
## ARIMA(2,1,0)(2,0,1)[12] with drift : 2289.032  
## ARIMA(1,1,1)(2,0,1)[12] with drift : 2287.86  
## ARIMA(0,1,1)(2,0,1)[12] with drift : 2288.476  
## ARIMA(2,1,1)(2,0,1)[12] with drift : 2290.733  
## ARIMA(1,1,0)(2,0,1)[12] : 2285.17  
## ARIMA(1,1,0)(1,0,1)[12] : 2298.104  
## ARIMA(1,1,0)(2,0,0)[12] : 2286.391  
## ARIMA(1,1,0)(2,0,2)[12] : 2286.883  
## ARIMA(1,1,0)(1,0,0)[12] : 2297.508  
## ARIMA(1,1,0)(1,0,2)[12] : 2299.806  
## ARIMA(0,1,0)(2,0,1)[12] : 2287.871  
## ARIMA(2,1,0)(2,0,1)[12] : 2288.189  
## ARIMA(1,1,1)(2,0,1)[12] : 2287.028  
## ARIMA(0,1,1)(2,0,1)[12] : 2287.953  
## ARIMA(2,1,1)(2,0,1)[12] : 2289.821  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(1,1,0)(2,0,1)[12] : Inf  
## ARIMA(1,1,0)(2,0,1)[12] with drift : Inf  
## ARIMA(1,1,0)(2,0,0)[12] : 2311.12  
##   
## Best model: ARIMA(1,1,0)(2,0,0)[12]

b

## Series: stock\_data\_train   
## ARIMA(1,1,0)(2,0,0)[12]   
##   
## Coefficients:

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

## ar1 sar1 sar2  
## 0.0279 -0.0118 0.0705  
## s.e. 0.0019 NaN 0.0026  
##   
## sigma^2 estimated as 2962: log likelihood=-1151.46  
## AIC=2310.93 AICc=2311.12 BIC=2324.37

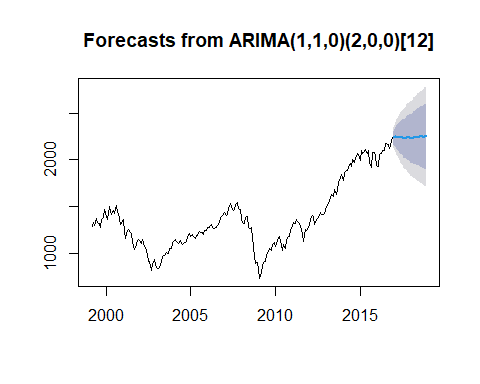
model\_b <- Arima(stock\_data\_train, order = c(1,1,0),seasonal = list(order = c(2,0,0), period = 12))  
summary(model\_b)

## Series: stock\_data\_train   
## ARIMA(1,1,0)(2,0,0)[12]   
##   
## Coefficients:

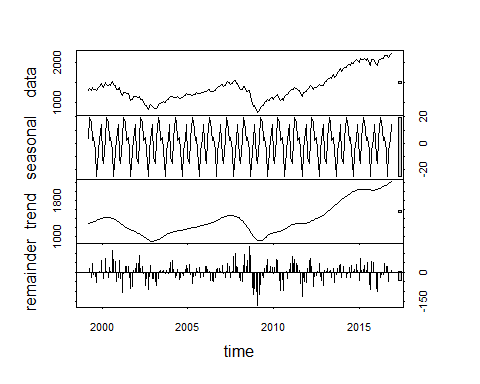
## Warning in sqrt(diag(x$var.coef)): NaNs produced

## ar1 sar1 sar2  
## 0.0279 -0.0118 0.0705  
## s.e. 0.0019 NaN 0.0026  
##   
## sigma^2 estimated as 2962: log likelihood=-1151.46  
## AIC=2310.93 AICc=2311.12 BIC=2324.37  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 4.143009 53.91145 42.25847 0.1571608 3.270065 0.2333915  
## ACF1  
## Training set 0.001832111

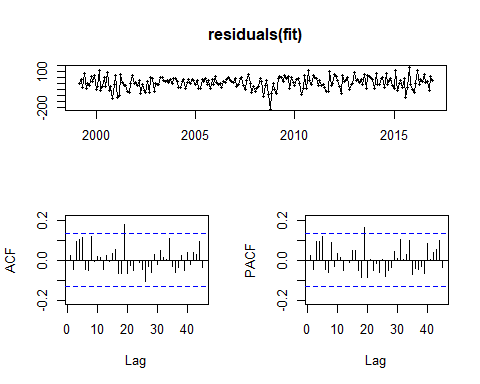
stock\_predict\_b <- forecast(model\_b, h = 24)  
plot(stock\_predict\_b)



stock\_decompose\_stl <- stl(stock\_data\_train, s.window = "periodic")  
plot(stock\_decompose\_stl)  
stock\_decompose\_stl\_deseasonal <- seasadj(stock\_decompose\_stl)  
plot(stock\_decompose\_stl)



fit <- auto.arima(stock\_decompose\_stl\_deseasonal, seasonal = FALSE)  
tsdisplay(residuals(fit), lag.max = 45)



test\_data <- ts(stock\_ts, start=c(2017, 1), end=c(2019, 3), freq=12)  
  
#Mean method  
accuracy(meanf(stock\_data\_train, h = 24), test\_data)

## ME RMSE MAE MPE MAPE MASE  
## Training set -1.024004e-13 354.90695 274.58527 -6.32150434 20.492983 1.5165214  
## Test set 2.913118e+00 71.52746 60.27488 -0.05707497 4.371263 0.3328953  
## ACF1 Theil's U  
## Training set 0.9744358 NA  
## Test set 0.4805775 1.070113

#Naive Method  
accuracy(rwf(stock\_data\_train, h = 24), test\_data)

## ME RMSE MAE MPE MAPE MASE  
## Training set 4.471644 54.19181 42.64432 0.1661351 3.30603 0.2355226  
## Test set -860.008412 862.97286 860.00841 -62.8093593 62.80936 4.7497857  
## ACF1 Theil's U  
## Training set 0.02483767 NA  
## Test set 0.48057753 13.16037

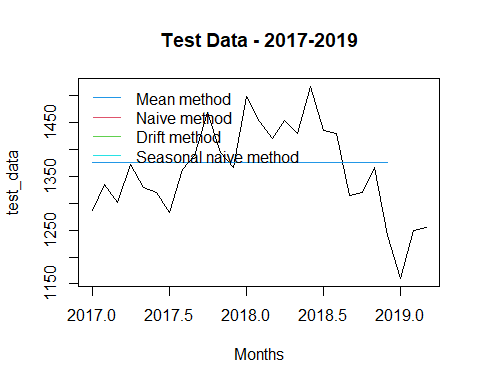
accuracy(rwf(stock\_data\_train, drift = TRUE, h = 24), test\_data)

## ME RMSE MAE MPE MAPE MASE  
## Training set -4.803936e-14 54.00701 42.0643 -0.1793955 3.267652 0.2323191  
## Test set -9.159040e+02 918.63294 915.9040 -66.8476790 66.847679 5.0584942  
## ACF1 Theil's U  
## Training set 0.02483767 NA  
## Test set 0.42454959 14.02732

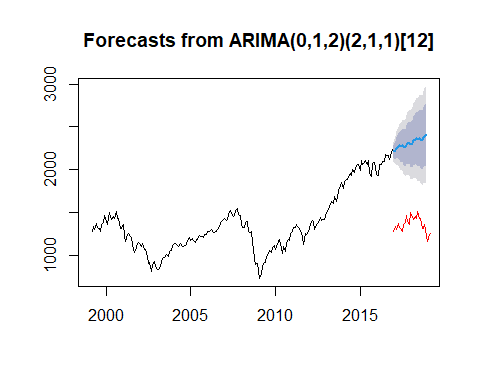
accuracy(snaive(stock\_data\_train, h=24), test\_data)

## ME RMSE MAE MPE MAPE MASE  
## Training set 44.85158 218.1848 181.0626 0.9490803 14.69415 1.000000  
## Test set -727.00585 738.1897 727.0058 -53.2107743 53.21077 4.015219  
## ACF1 Theil's U  
## Training set 0.9354494 NA  
## Test set 0.4011467 11.36451

plot(test\_data, main="Test Data - 2017-2019", xlab = "Months" )  
  
train.mean <- meanf(stock\_data\_train, h = 24)$mean  
train.naive <- rwf(stock\_data\_train, h = 24)$mean   
train.drift <- rwf(stock\_data\_train, drift = TRUE, h = 24)$mean  
train.seas <- snaive(stock\_data\_train, h = 24)$mean  
  
 lines(train.mean, col=4)  
 lines(train.naive, col=2)  
 lines(train.drift, col=3)  
 lines(train.seas, col=5)  
   
 legend("topleft", lty=1, col=c(4,2,3,5), legend=c("Mean method","Naive method","Drift method", "Seasonal naive method"),bty="n")



plot(forecast(auto.arima(stock\_data\_train , D = 1), h = 24))  
  
lines(test\_data, col = "red")



ggplot(data = stock\_returns\_monthly, aes(x = date, y = adjusted )) + geom\_line()

