Project 2

FNU Joshua, Angela Li, Sabrina Tsai

I. Introduction

We will look at data from the New York Stock Exchange Composite Index (NYA) and IBM returns (IBM) from 1970 to 2018. The NYSE is a stock exchange based in New York City, which conducts equity trading. The NYA is a composite index of the NYSE which reflects the value of all stocks traded by the NYSE. IBM is an America-based, multinational technology company that deals broadly with technology, from artificial intelligence to computer hardware. IBM became publicly traded in 1911 and its stock trades on the NYSE. The performance of NYA will most likely have an effect on IBM, and vice versa since IBM is traded on the NYSE. We will fit a forecasting model for IBM stock using NYA.

We chose to look at data up until 2018 because we want to focus on the correlation between IBM and NYSE before the worldwide coronavirus pandemic took over. Initially looking at IBM's data, we see the largest jumps taking place around the late 1990s and early 2010s, and has been slightly decreasing over the past five years. IBM's peak price took place around 2013. The NYSE is generally exhibiting similar trends to IBM, as it has sharp dips around the early 2000s and late 2000s just like IBM. The NYSE's peaks though occurred around the turn of the century and 2007 prior to the 2008 financial crisis. The NYSE has been increasing in value ever since 2009. While IBM follows a similar trend as the NYSE, IBM appears to not be close to its peak stock price in the late 2010s, unlike the NYSE. The 2010s overall appear to show the biggest disparity between the trend of IBM's stock price and the value of the NYSE.

II. Results

Import library

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 3.6.3
```

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

```
## Warning: package 'xts' was built under R version 3.6.3
```

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

```
## Warning: package 'zoo' was built under R version 3.6.3
```

```
##
## 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 3.6.3

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

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

```
library(timeSeries)
```

```
## Warning: package 'timeSeries' was built under R version 3.6.3

## Loading required package: timeDate

## Warning: package 'timeDate' was built under R version 3.6.2

##
## Attaching package: 'timeSeries'

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

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.6.3
```

```
library(ggplot2)
library(vars)
```

```
## Warning: package 'vars' was built under R version 3.6.3

## Loading required package: MASS

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 3.6.3

## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 3.6.3

## Loading required package: urca

## Warning: package 'urca' was built under R version 3.6.3

## Loading required package: lmtest

## Warning: package 'lmtest' was built under R version 3.6.3
```

```
library(lmtest)
library(timeSeries)
library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 3.6.3
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

Modeling and Forecasting Trend, Seasonality, and Cycles

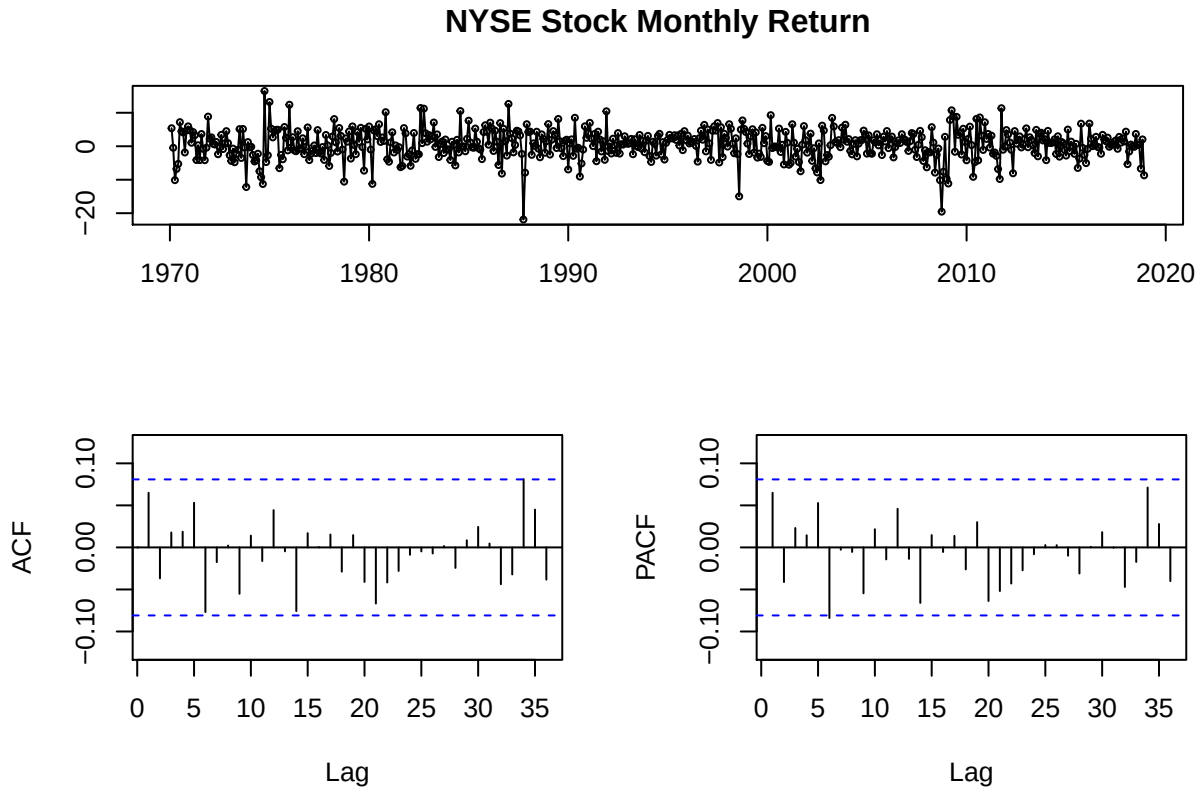
```
#use getSymbols function to get NYSE and Accenture daily stock
nyse <- getSymbols(Symbols = "^NYA", auto.assign = FALSE, from = "1970-01-01", to = "2019-12-31")[,6]
```

```
## '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.
```

```
ibm <- getSymbols(Symbols = "IBM", auto.assign = FALSE, from = "1970-01-01", to = "2019-12-31")[,6]

#convert to weekly data since we want to analyze monthly return stock data
nyse <- to.monthly(x = nyse)[,4]
ibm <- to.monthly(x = ibm)[,4]

#convert again to time series
nyse <- ts(nyse, start = 1970, frequency = 12)
ibm <- ts(ibm, start = 1970, frequency = 12)

#Testing window (Until 2019)
#calculate difference of each time series
nyse_r <- (diff(nyse) / stats::lag(nyse, -1)) * 100
ibm_r <- (diff(ibm) / stats::lag(ibm, -1)) * 100

#Training window
nyse_return <- window(nyse_r, end=c(2018,12))
ibm_return <- window(ibm_r, end=c(2018,12))
```

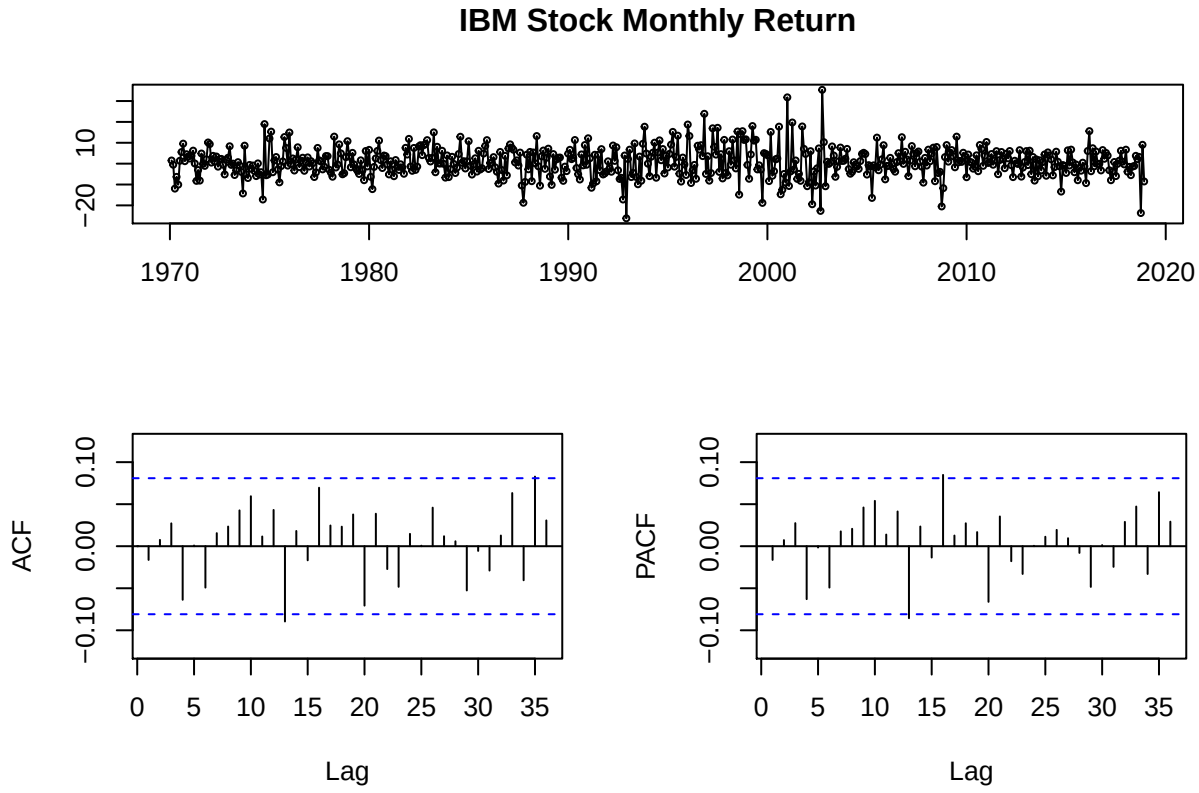
(a) Produce a time-series plot of your data including the respective ACF and PACF plots

```
tsdisplay(nyse_return, main = "NYSE Stock Monthly Return")
```



Even though it's vague, we can see that for acf and pacf lag 6 is almost statistically significant and we want to account that by fitting ARMA(6,6) model.

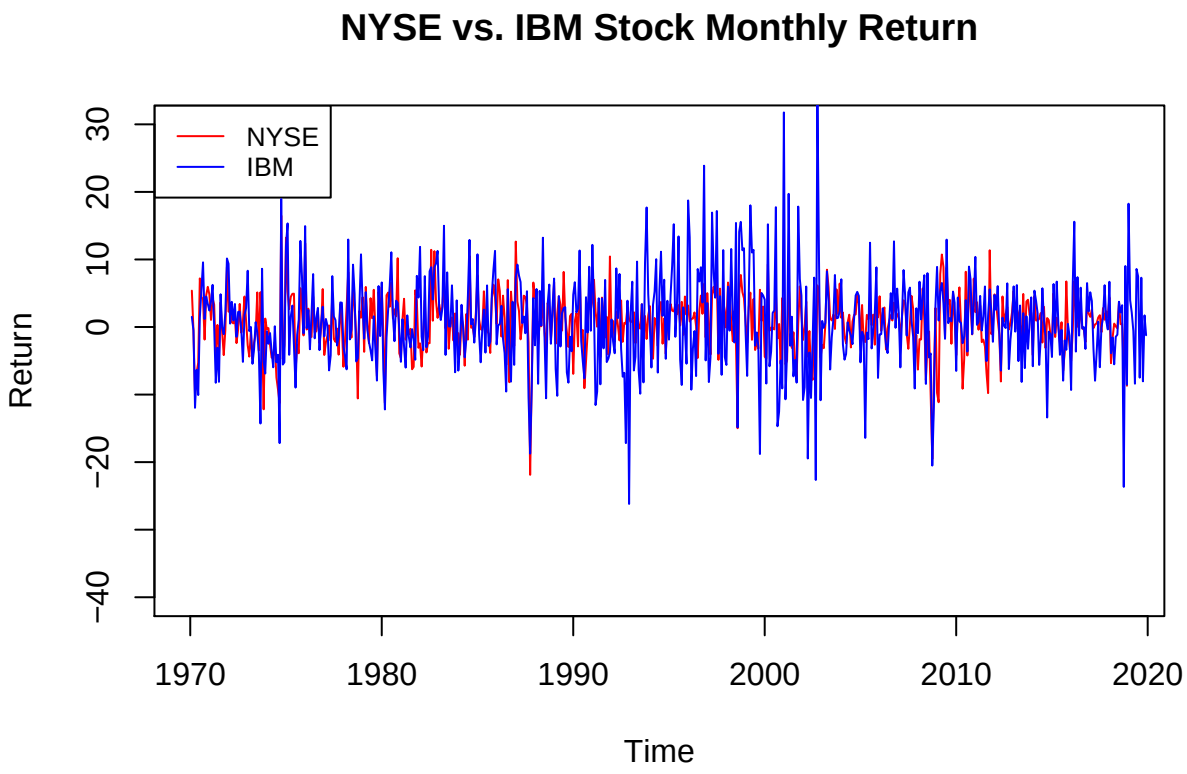
```
tsdisplay(ibm_return, main = "IBM Stock Monthly Return")
```



Statistically significant lag at 13 for both plots. Thus, can try to fit ARMA(13,13)

To get a rough idea how the two series return move contemporaneously we try to plot is alongside each other. We will also normalize both series so that we can compare between the two of them.

```
plot(nyse_return, main = "NYSE vs. IBM Stock Monthly Return", ylab = "Return", col = "red",
     type = "l", ylim = c(-40,30))
legend(x = "topleft", legend = c("NYSE","IBM"), col = c("red","blue"), lty=1:1, cex=0.8)
lines(ibm_r, col = "blue")
```



(b) Fit a model that includes, trend, seasonality and cyclical components. Make sure to discuss your model in detail.

Periodic Trend

Proposing our first fit to the model, that is, periodic since as we can see financial return looks like a random white noise and it seems to have fluctuation up and down

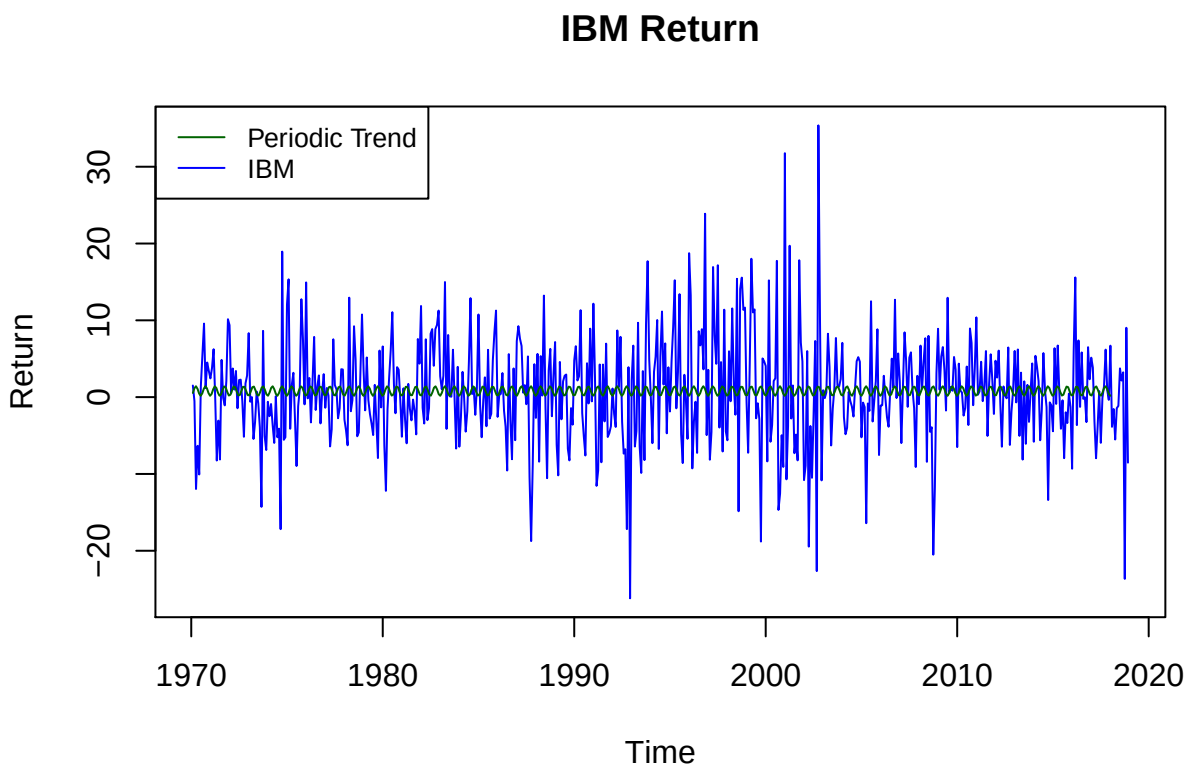
```
#construct time component
t <- seq(1970 + 1/12, 2018, length = length(nyse_return))

ibm_mod_trend <- lm(ibm_return ~ I(sin(4*pi*t)) + I(cos(4*pi*t)))
summary(ibm_mod_trend)
```

```
##
## Call:
## lm(formula = ibm_return ~ I(sin(4 * pi * t)) + I(cos(4 * pi *
##     t)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.363  -4.247  -0.175   3.942  34.425
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)         0.79512    0.29136   2.729  0.00654 **
## I(sin(4 * pi * t))  0.06849    0.41204   0.166  0.86804
## I(cos(4 * pi * t)) -0.61954    0.41205  -1.504  0.13324
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.059 on 584 degrees of freedom
## Multiple R-squared:  0.003903,   Adjusted R-squared:  0.0004918
## F-statistic: 1.144 on 2 and 584 DF,  p-value: 0.3192
```

The result is not statistically significant, independently and jointly.

```
plot(ibm_return, main = "IBM Return", col = "blue", ylab = "Return")
legend(x = "topleft", legend = c("Periodic Trend","IBM"), col = c("dark green","blue"), lty=1:1, cex=0.8
lines(ibm_mod_trend$fitted.values ~ t, col = "dark green")
```



Seasonal Component

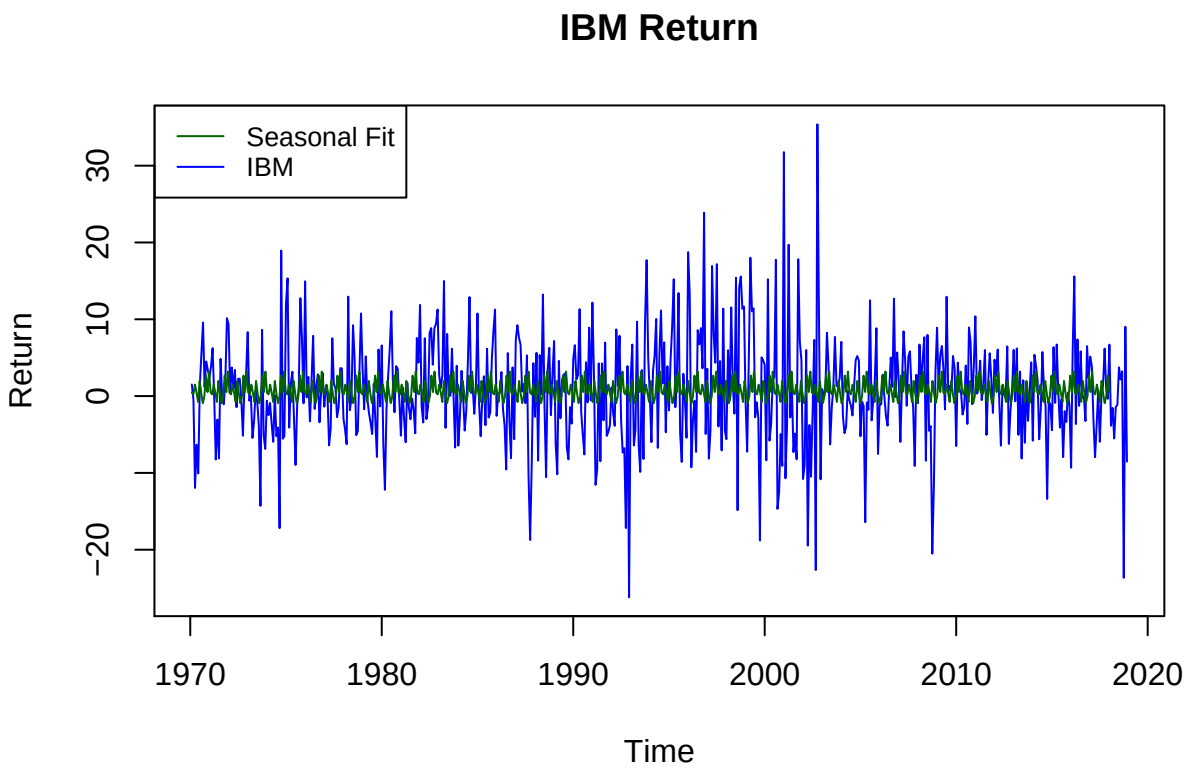
```
ibm_mod_seasonal <- tslm(ibm_return ~ season)
summary(ibm_mod_seasonal)
```

```
##
## Call:
## tslm(formula = ibm_return ~ season)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.717  -4.235  -0.222   3.869  35.412
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1934     1.0133   3.152  0.00171 **
## season2      -2.7287     1.4256  -1.914  0.05612 .
## season3      -2.9701     1.4256  -2.083  0.03766 *
## season4      -1.7065     1.4256  -1.197  0.23179
## season5      -2.7700     1.4256  -1.943  0.05251 .
## season6      -3.9607     1.4256  -2.778  0.00564 **
## season7      -1.2187     1.4256  -0.855  0.39302
## season8      -2.8422     1.4256  -1.994  0.04667 *
## season9      -4.1153     1.4256  -2.887  0.00404 **
## season10     -3.2251     1.4256  -2.262  0.02406 *
## season11     -0.5261     1.4256  -0.369  0.71225
## season12     -2.6670     1.4256  -1.871  0.06189 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.02 on 575 degrees of freedom
## Multiple R-squared:  0.03006,    Adjusted R-squared:  0.01151
## F-statistic:  1.62 on 11 and 575 DF,  p-value: 0.08908
```

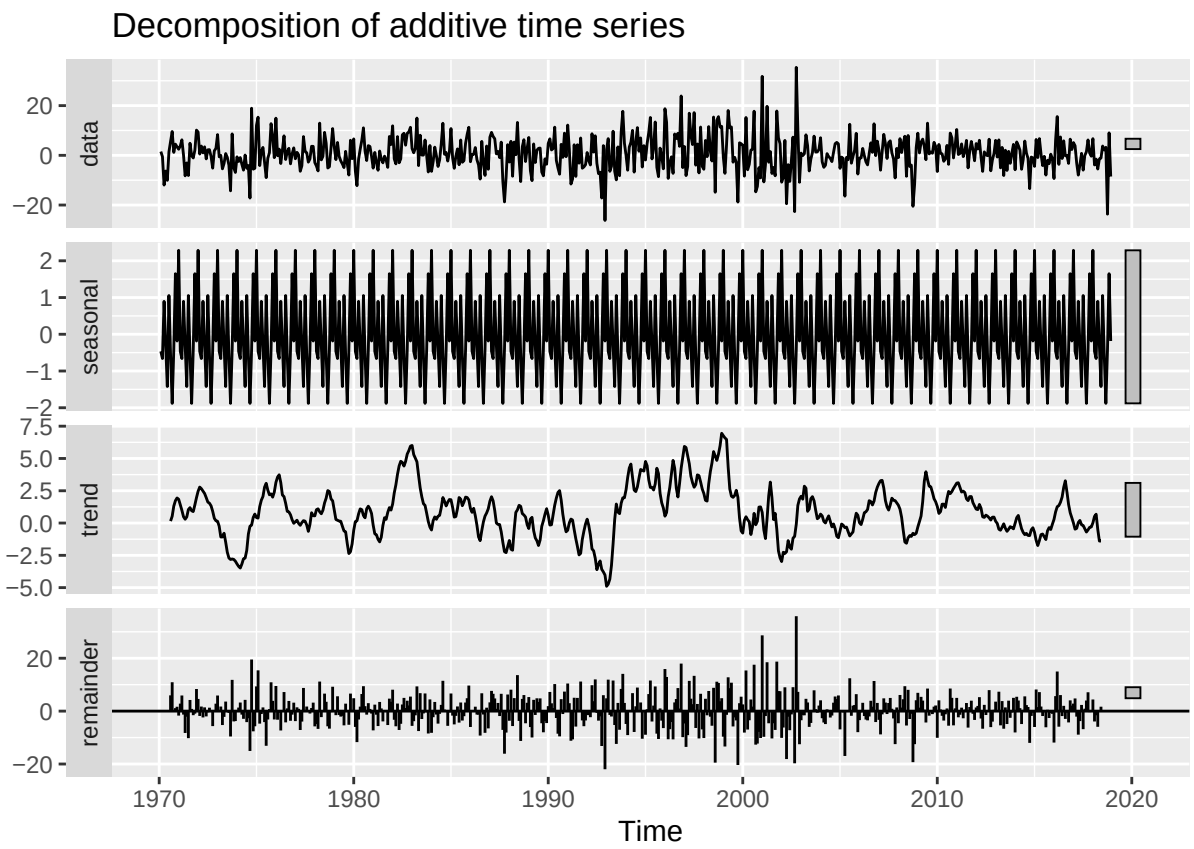
As a result, season 3, 6, 8, 9, 10 are all statistically significant with season 2, 5, 12 almost statistically significant. This indicates strong seasonality in our time series model.

```
plot(ibm_return, main = "IBM Return", col = "blue", ylab = "Return")
legend(x = "topleft", legend = c("Seasonal Fit","IBM"), col = c("dark green","blue"), lty=1:1, cex=0.8)
lines(ibm_mod_seasonal$fitted.values ~ t, col = "dark green")
```



Try to break down the components

```
ibm_dec <- decompose(ibm_return)
autoplot(ibm_dec)
```



For our types of model, the more appropriate forecasting model is the garch model where we have to consider the error term ("remainder") as a component of our forecast.

```
auto.arima(ibm_dec$random)
```

```
## Series: ibm_dec$random
## ARIMA(5,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.1733  -0.1618  -0.1527  -0.2498  -0.2029
## s.e.   0.0409   0.0403   0.0403   0.0402   0.0409
##
## sigma^2 estimated as 39.52:  log likelihood=-1870.7
## AIC=3753.4   AICc=3753.55   BIC=3779.52
```

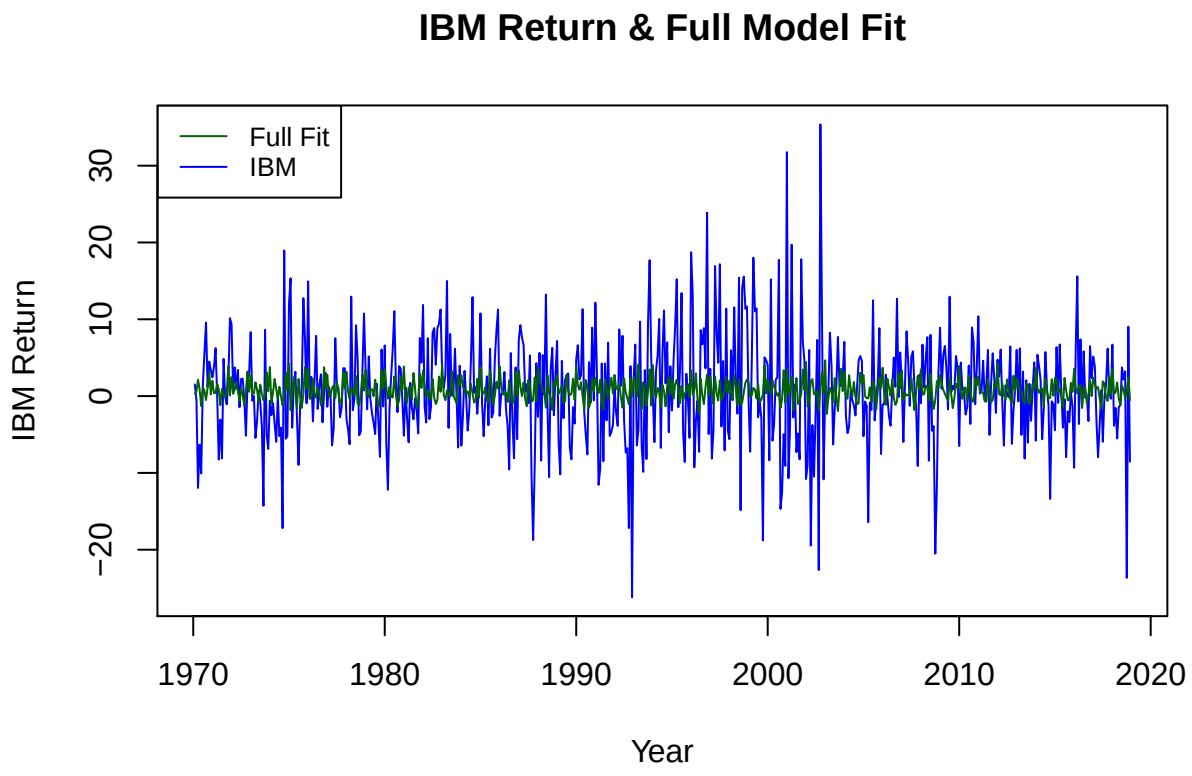
Hence we can fit it ARIMA(5,0,0) into the later model when we use GARCH.

Arma Trend, Seasonal, & Cyclical Component

```
season_m <- seasonaldummy(ibm_return)
ibm_mod_full <- Arima(ibm_return, order = c(5,0,0), seasonal=list(order=c(1,0,1), period = 12), xreg = 
summary(ibm_mod_full)
```

```
## Series: ibm_return
## Regression with ARIMA(5,0,0)(1,0,1)[12] errors
##
## Coefficients:
##           ar1     ar2     ar3      ar4     ar5     sar1    sma1  intercept
##       -0.0095  0.0124  0.0270  -0.0645  0.0093  -0.8245  0.8493     3.5096
## s.e.   0.0413  0.0412  0.0419   0.0420  0.0419   0.2904  0.2717    40.6279
##             t                       Jan     Feb      Mar     Apr      May
##       -0.0015  0.0626  -0.5689  2.6251  0.0042  -0.2494  1.0307  -0.0838
## s.e.   0.0204  0.4110   0.4110  1.4349  1.4118   1.4022  1.4609   1.4134
##           Jun     Jul      Aug      Sep      Oct     Nov
##       -1.3159  1.4448  -0.1427  -1.3739  -0.5023  2.1670
## s.e.   1.4201  1.4135   1.4621   1.4017   1.4114  1.4269
##
## sigma^2 estimated as 49.65:  log likelihood=-1967.86
## AIC=3981.73   AICc=3983.69   BIC=4082.35
##
## Training set error measures:
##                        ME    RMSE      MAE  MPE MAPE      MASE
## Training set -0.001124607 6.91285 5.197323 -Inf  Inf 0.7047393
##                      ACF1
## Training set 0.0005794261
```

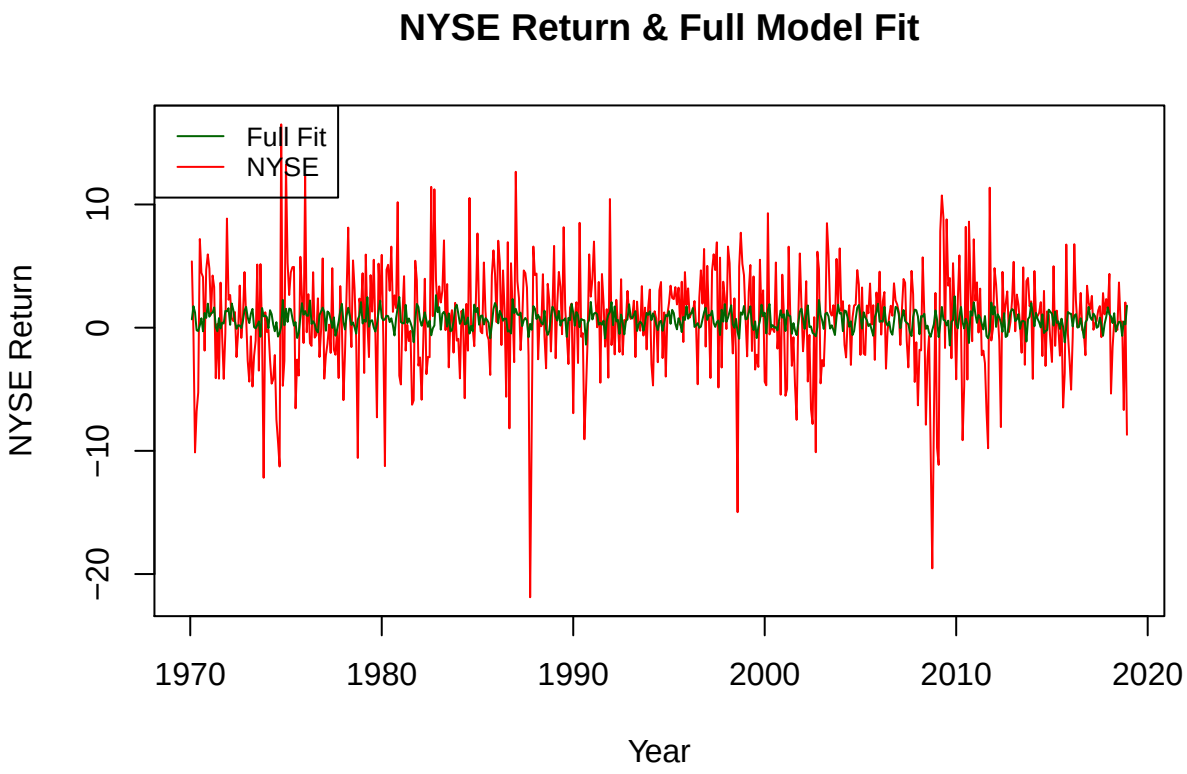
```r
plot(ibm_return, main = "IBM Return & Full Model Fit", xlab = "Year", ylab = "IBM Return", col = "blue")
legend(x = "topleft", legend = c("Full Fit","IBM"), col = c("dark green","blue"), lty=1:1, cex=0.8)
lines(ibm_mod_full$fit, col = "dark green")
```



```r
nyse_mod_full <- Arima(nyse_return, order=c(2,0,2), seasonal=list(order=c(1,0,1), period =12), xreg = cb
summary(nyse_mod_full)
```

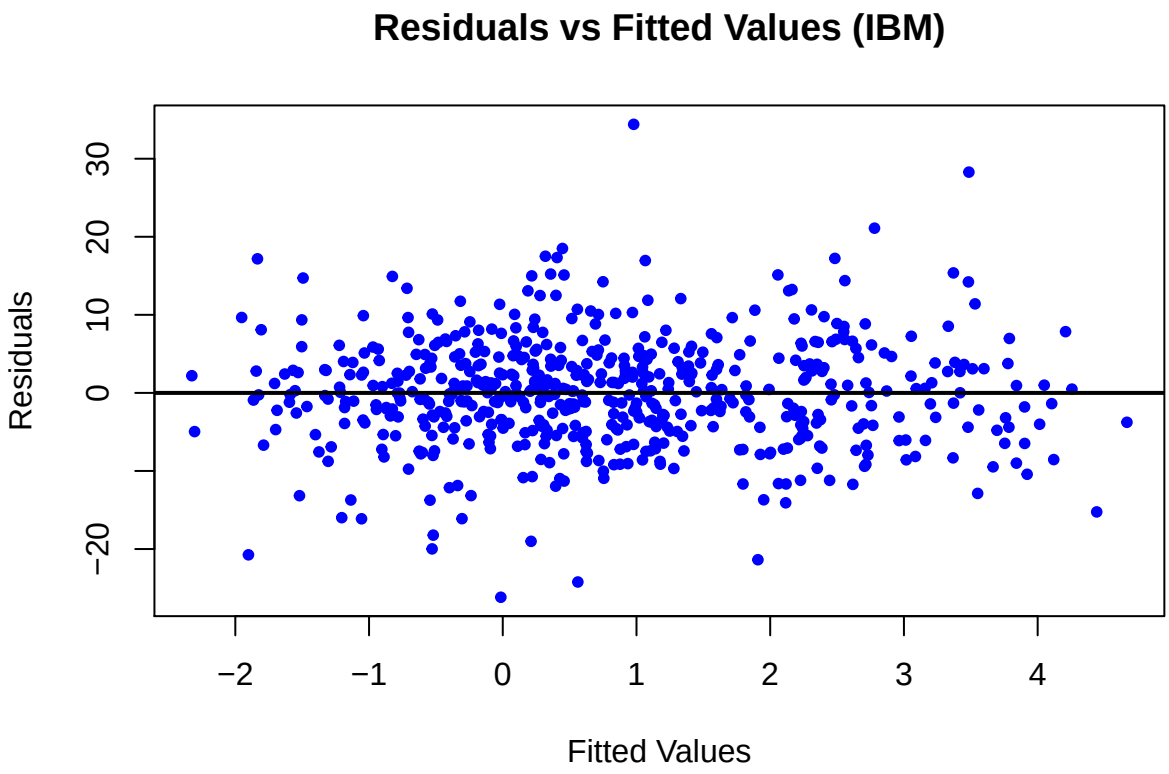
```
## Series: nyse_return
## Regression with ARIMA(2,0,2)(1,0,1)[12] errors
##
## Coefficients:
##           ar1     ar2     ma1      ma2     sar1    sma1  intercept
##       -0.1200  0.4610  0.1834  -0.4838  -0.2137  0.2428    11.1577
## s.e.   0.8694  0.5621  0.8673   0.6045   0.6897  0.6833    27.6182
##             t                       Jan      Feb      Mar      Apr
##       -0.0048  0.2652  -0.0977  -0.5139  -1.1867  -0.4691  -0.2589
## s.e.   0.0138  0.2635   0.2635   0.8640   0.9011   0.8725   0.8971
##           May      Jun      Jul      Aug      Sep      Oct      Nov
##       -1.2099  -1.4213  -0.9872  -1.5465  -2.0805  -1.0454  -0.3072
## s.e.   0.8764   0.8952   0.8764   0.8973   0.8722   0.9011   0.8590
##
## sigma^2 estimated as 19.01:  log likelihood=-1686.52
## AIC=3417.03   AICc=3418.83   BIC=3513.28
##
## Training set error measures:
##                         ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 6.155429e-05 4.280843 3.202463 87.17854 121.5478 0.7009805
##                        ACF1
## Training set -0.0007710055
```

```r
plot(nyse_return, main = "NYSE Return & Full Model Fit", xlab = "Year", ylab = "NYSE Return", col = "red
legend(x = "topleft", legend = c("Full Fit","NYSE"), col = c("dark green","red"), lty=1:1, cex=0.8)
lines(nyse_mod_full$fit, col = "dark green")
```



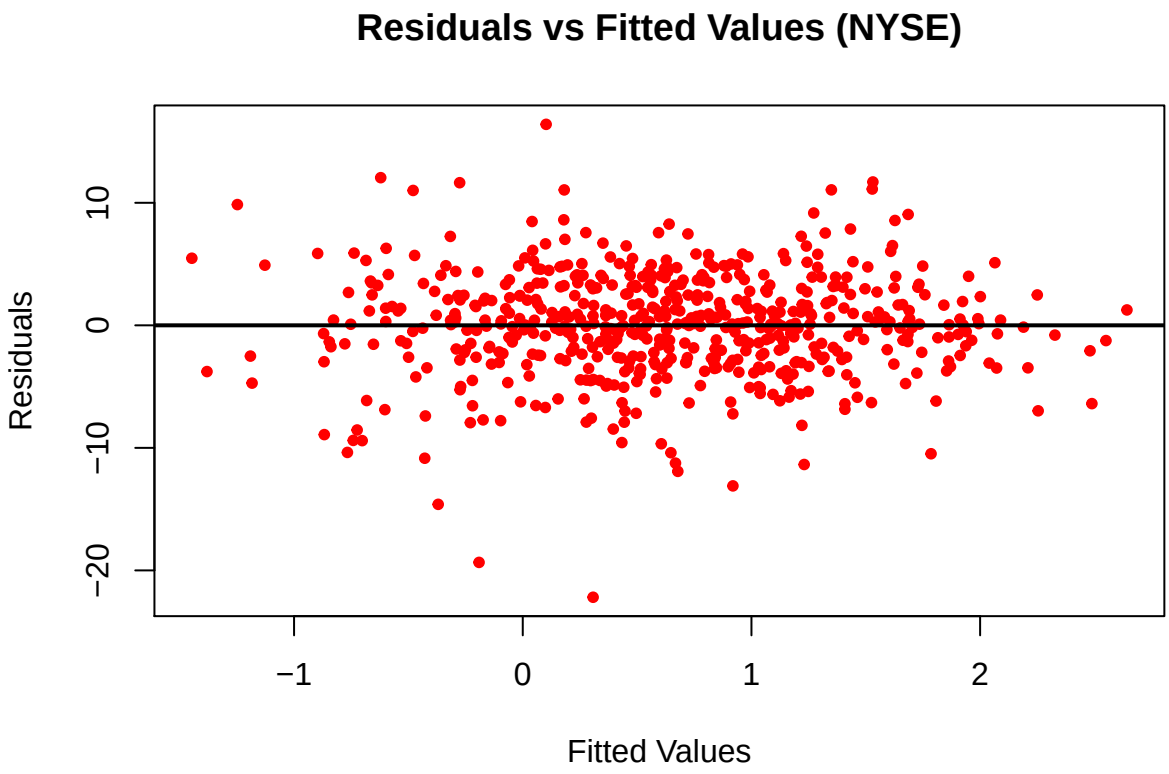
(c) Plot the respective residuals vs. fitted values and discuss your observations.

```
#IBM return
plot(ibm_mod_full$fitted, ibm_mod_full$residuals, pch=20, col="blue", lwd=1, main = "Residuals vs Fitte
abline(h=0, lwd=2, col="black")
```



We can see that the residuals are normally distributed.

```
#NYSE return
plot(nyse_mod_full$fitted, nyse_mod_full$residuals, pch=20, col="red", lwd=1, main = "Residuals vs Fitt
abline(h=0, lwd=2, col="black")
```

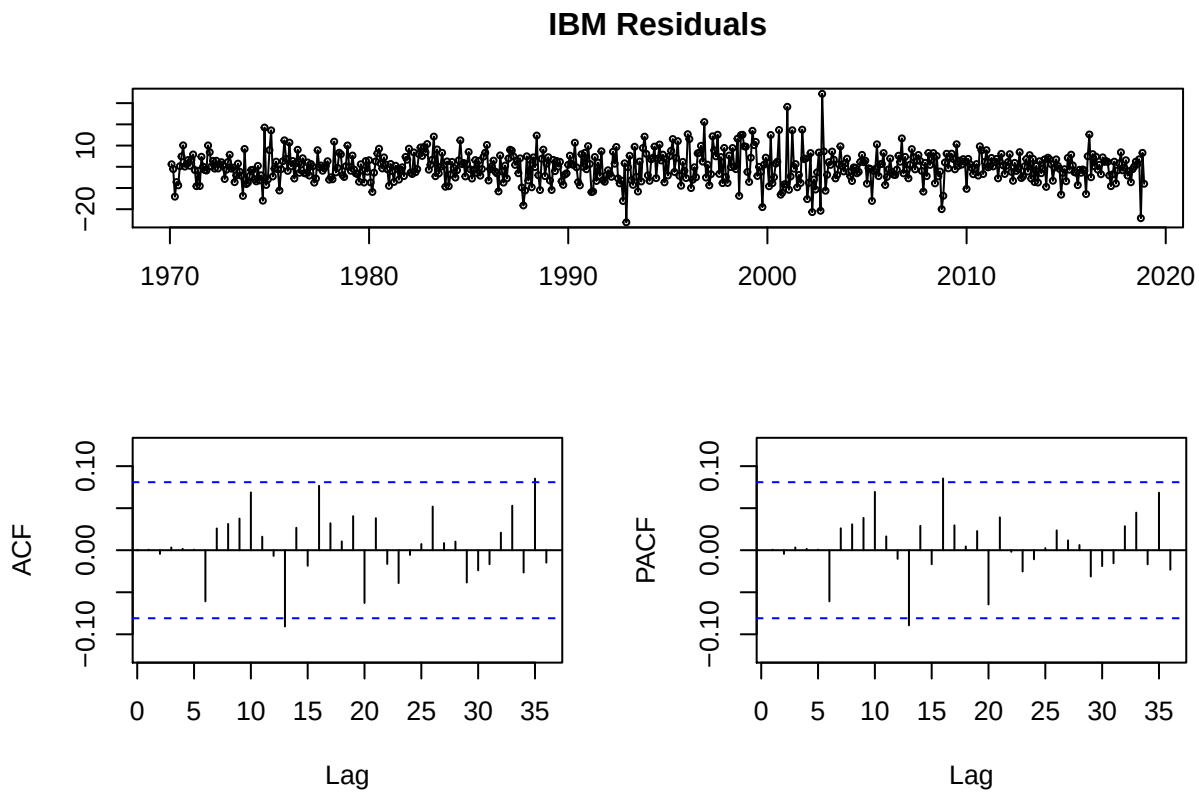


For nyse, the residuals are also normally distributed. Hence no bias as time goes on.

(e) Plot the ACF and PACF of the respective residuals and interpret the plots.

IBM Residuals

```
tsdisplay(ibm_mod_full$residuals, main = "IBM Residuals")
```



The residuals are distributed as whihte noise even though there are some spikes that are significant

We can test it using Box-Ljung test:

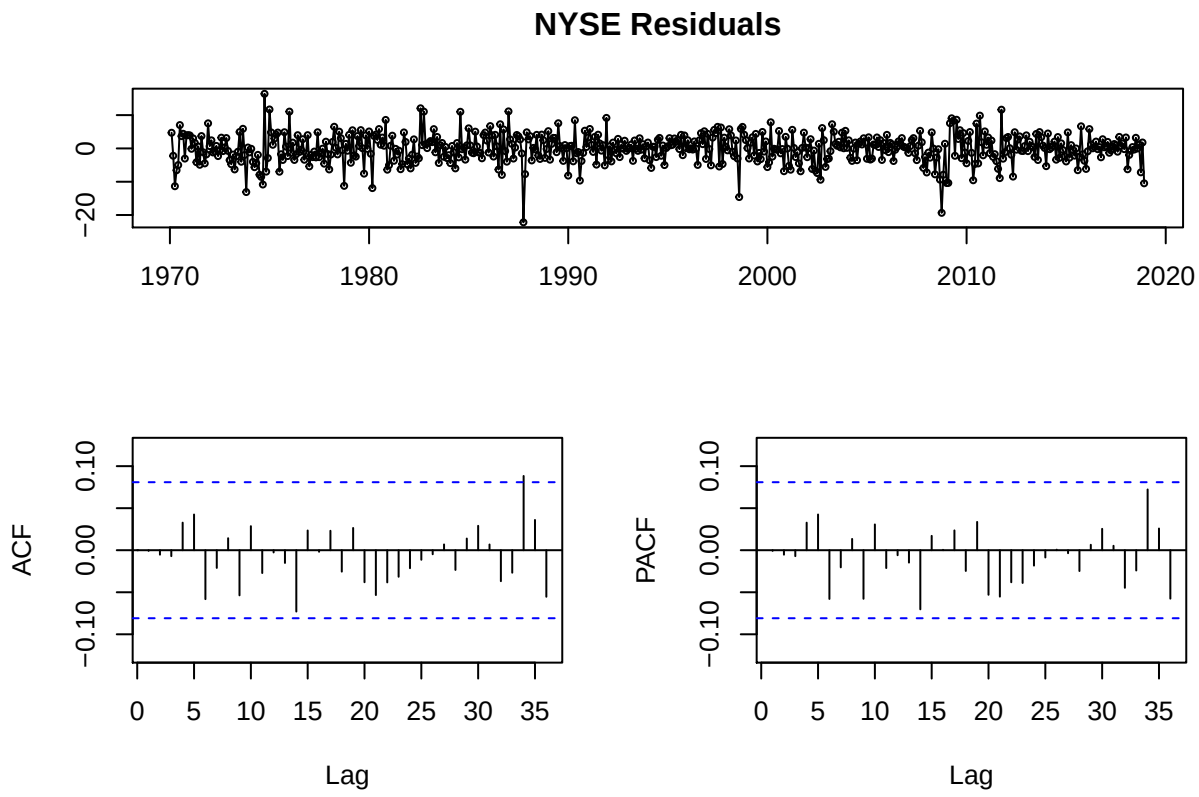
```
Box.test(ibm_mod_full$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ibm_mod_full$residuals
## X-squared = 0.00019809, df = 1, p-value = 0.9888
```

We fail to reject the null hypothesis hence there isn't enough evidence that residuals are not white noise.

NYSE Residuals

```
tsdisplay(nyse_mod_full$residuals, main = "NYSE Residuals")
```




```
Box.test(nyse_mod_full$residuals, type = "Ljung-Box")
```

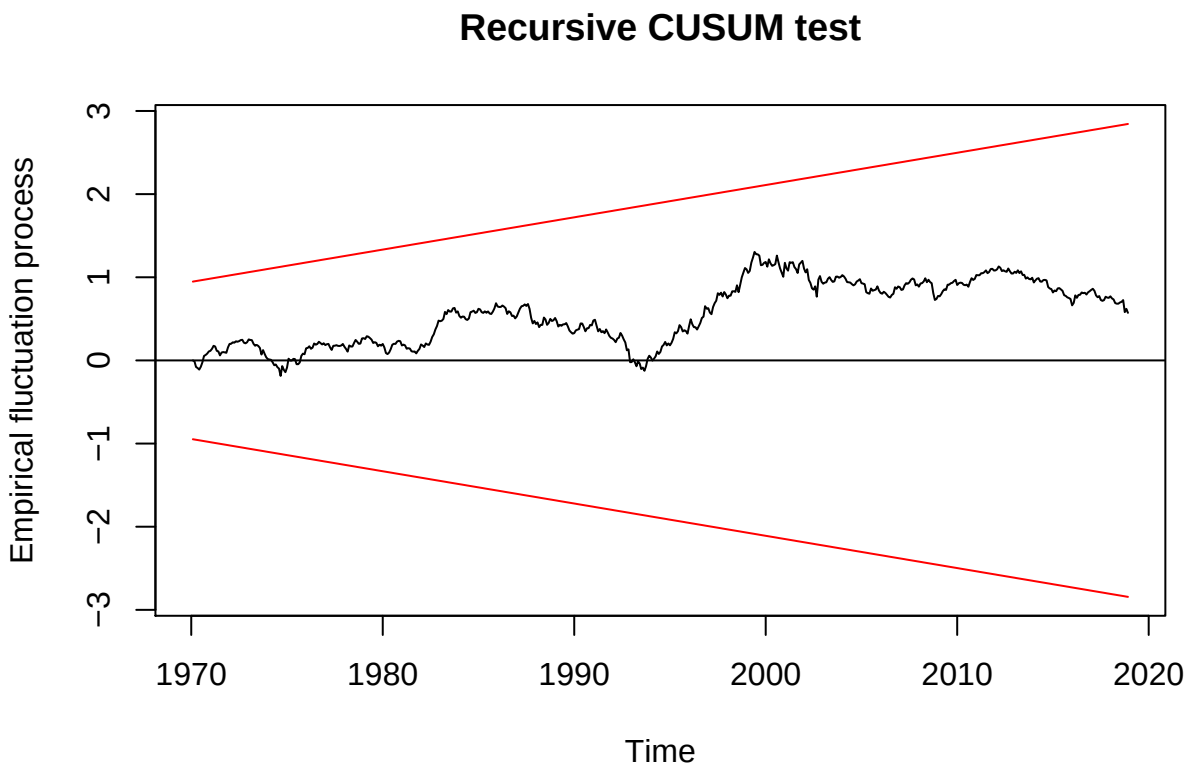
```
##
##  Box-Ljung test
##
## data:  nyse_mod_full$residuals
## X-squared = 0.00035073, df = 1, p-value = 0.9851
```

Also fail to reject the null hence we conclude that the residuals are white noise.

(f) Plot the respective CUSUM and interpret the plot.

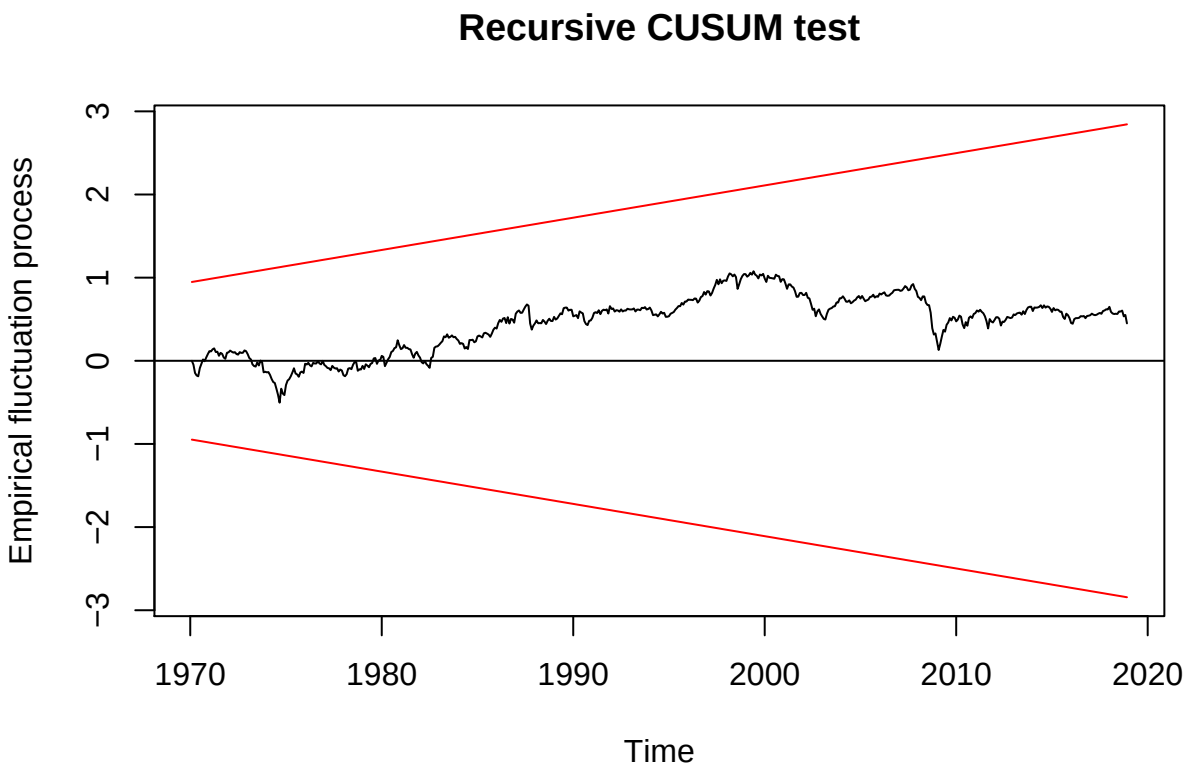
IBM CUSUM

```
plot(efp(ibm_mod_full$residuals ~ 1))
```



NYSE CUSUM

```
plot(efp(nyse_mod_full$residuals ~ 1))
```

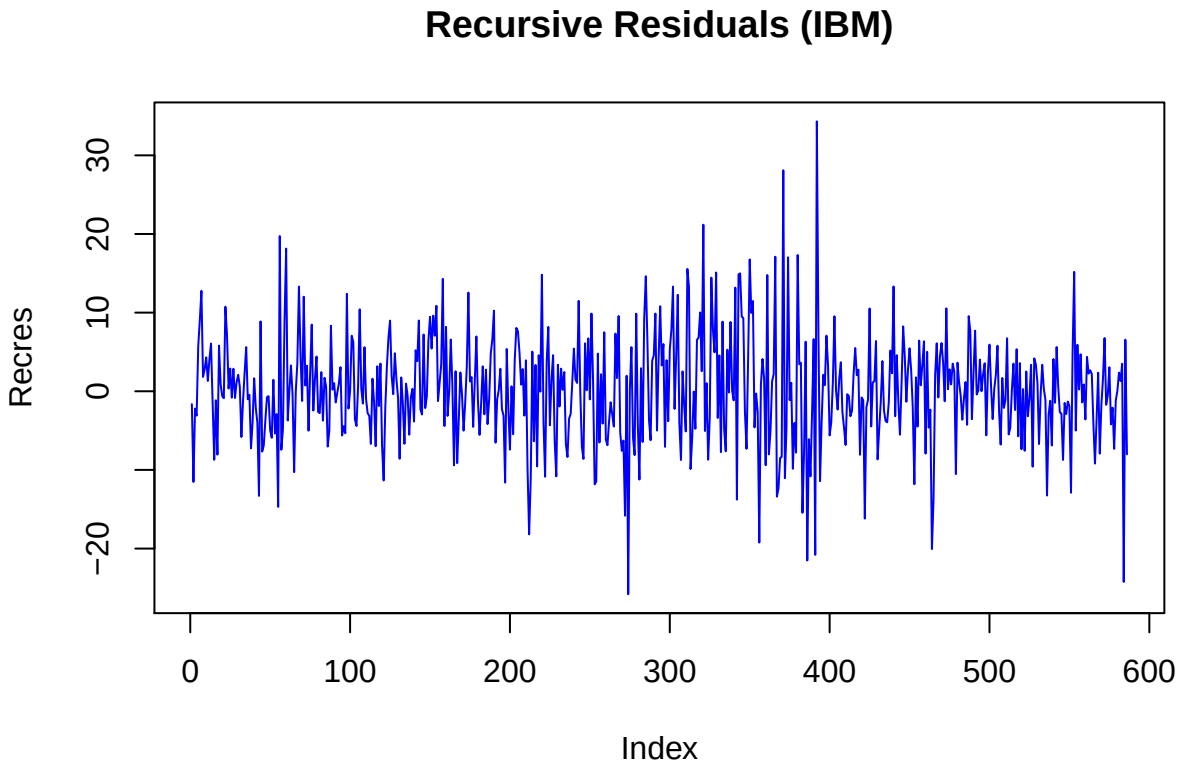


Both CUSUM plot behave inside the line. Hence in the long run, the model should be fine and fit okay.

(g) Plot the respective Recursive Residuals and interpret the plot.

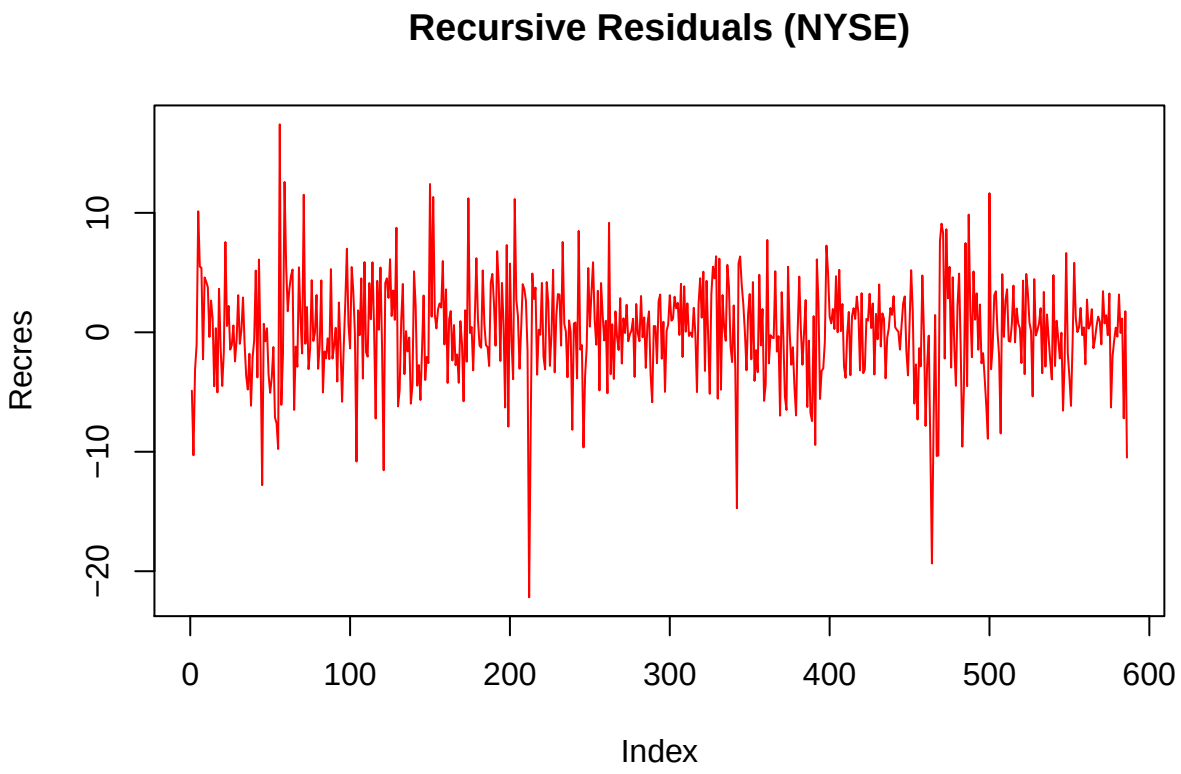
IBM Recursive Residuals

```
plot(recresid(ibm_mod_full$residuals ~ 1), main = "Recursive Residuals (IBM)", ylab = "Recres",
     pch = 16, type = "l", col = "blue")
```



NYSE Recursive Residuals

```
plot(recresid(nyse_mod_full$residuals ~ 1), main = "Recursive Residuals (NYSE)", ylab = "Recres",
     pch = 16, type = "l", col = "red")
```



Both recursive residuals plot line are normally distributed hence the model is good for the fit and it does not break in the long run.

(h) For your model, discuss the associated diagnostic statistics.

IBM diagnostic statistics

```
summary(ibm_mod_full)
```

```
## Series: ibm_return
## Regression with ARIMA(5,0,0)(1,0,1)[12] errors
##
## Coefficients:
##           ar1     ar2     ar3      ar4     ar5     sar1    sma1  intercept
##       -0.0095  0.0124  0.0270  -0.0645  0.0093  -0.8245  0.8493     3.5096
## s.e.   0.0413  0.0412  0.0419   0.0420  0.0419   0.2904  0.2717    40.6279
##             t                       Jan     Feb      Mar     Apr      May
##       -0.0015  0.0626  -0.5689  2.6251  0.0042  -0.2494  1.0307  -0.0838
## s.e.   0.0204  0.4110   0.4110  1.4349  1.4118   1.4022  1.4609   1.4134
##           Jun     Jul      Aug      Sep      Oct     Nov
##       -1.3159  1.4448  -0.1427  -1.3739  -0.5023  2.1670
## s.e.   1.4201  1.4135   1.4621   1.4017   1.4114  1.4269
##
## sigma^2 estimated as 49.65:  log likelihood=-1967.86
## AIC=3981.73   AICc=3983.69   BIC=4082.35
```

```
##
## Training set error measures:
##                       ME    RMSE      MAE  MPE MAPE      MASE
## Training set -0.001124607 6.91285 5.197323 -Inf  Inf 0.7047393
##                      ACF1
## Training set 0.0005794261
```

```
accuracy(ibm_mod_full)
```

```
##                       ME    RMSE      MAE  MPE MAPE      MASE
## Training set -0.001124607 6.91285 5.197323 -Inf  Inf 0.7047393
##                      ACF1
## Training set 0.0005794261
```

NYSE diagnostic statistics

```
summary(nyse_mod_full)
```

```
## Series: nyse_return
## Regression with ARIMA(2,0,2)(1,0,1)[12] errors
##
## Coefficients:
##           ar1     ar2     ma1      ma2     sar1    sma1  intercept
##       -0.1200  0.4610  0.1834  -0.4838  -0.2137  0.2428    11.1577
## s.e.   0.8694  0.5621  0.8673   0.6045   0.6897  0.6833    27.6182
##             t                       Jan      Feb      Mar      Apr
##       -0.0048  0.2652  -0.0977  -0.5139  -1.1867  -0.4691  -0.2589
## s.e.   0.0138  0.2635   0.2635   0.8640   0.9011   0.8725   0.8971
##           May      Jun      Jul      Aug      Sep      Oct      Nov
##       -1.2099  -1.4213  -0.9872  -1.5465  -2.0805  -1.0454  -0.3072
## s.e.   0.8764   0.8952   0.8764   0.8973   0.8722   0.9011   0.8590
##
## sigma^2 estimated as 19.01:  log likelihood=-1686.52
## AIC=3417.03   AICc=3418.83   BIC=3513.28
##
## Training set error measures:
##                        ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 6.155429e-05 4.280843 3.202463 87.17854 121.5478 0.7009805
##                       ACF1
## Training set -0.0007710055
```

```
accuracy(nyse_mod_full)
```

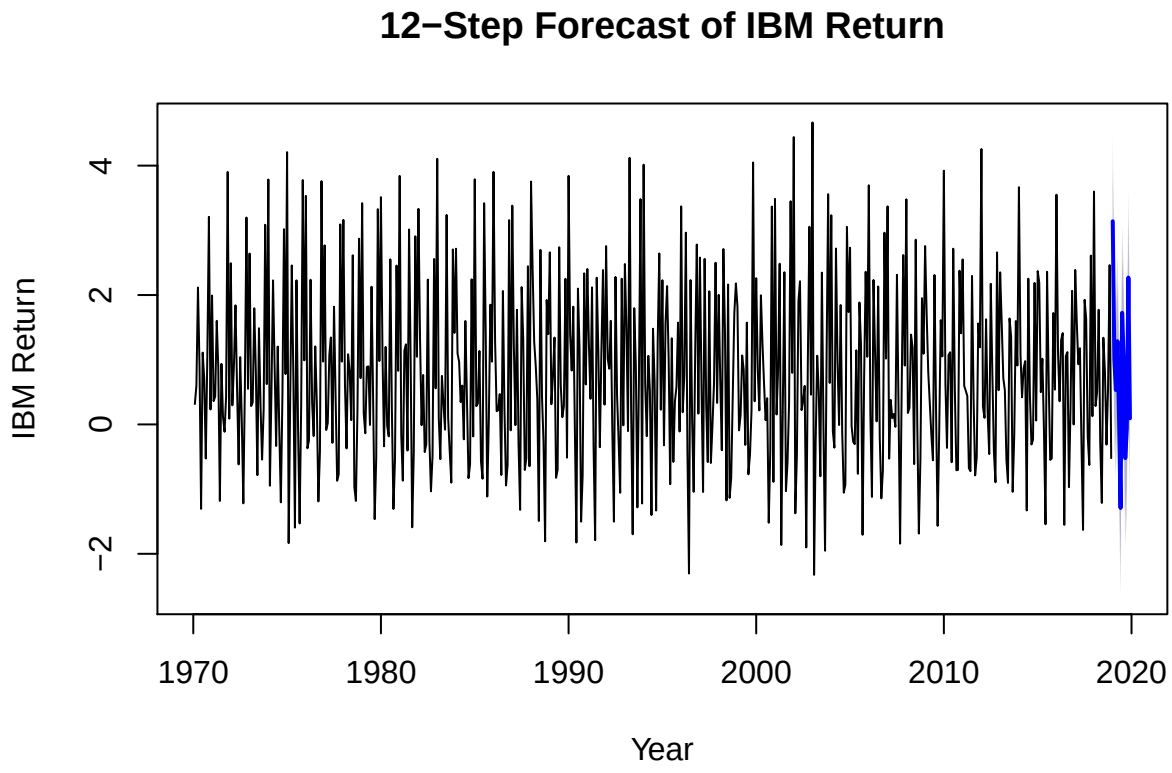
```
##                        ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 6.155429e-05 4.280843 3.202463 87.17854 121.5478 0.7009805
##                       ACF1
## Training set -0.0007710055
```

Both models perform well in terms of r-squared and signifigance of coefficients. RMSE and MAPE also look decent for fit on training data, but better comparison will come with the introduction of more models later in the report.

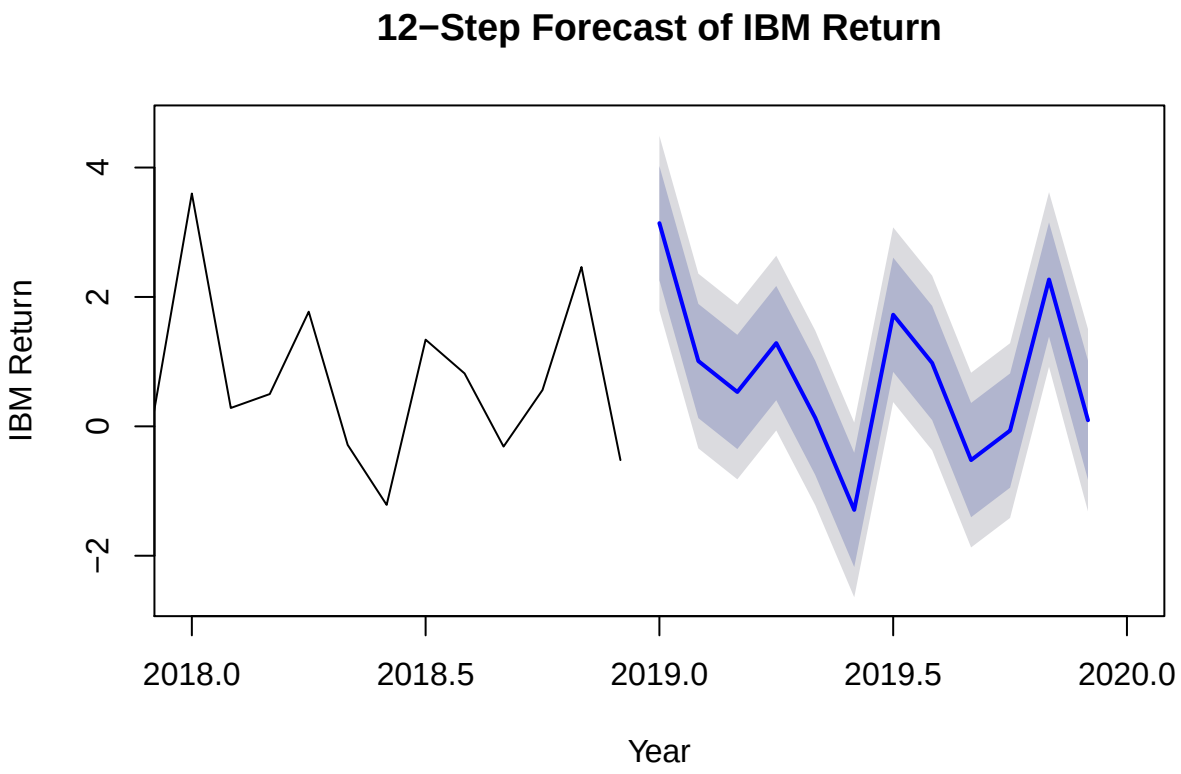
(i) Use your model to forecast 12-steps ahead. Your forecast should include the respective error bands.

IBM Forecast

```
plot(forecast(ibm_mod_full$fitted, h=12), main = "12-Step Forecast of IBM Return", ylab = "IBM Return",
```

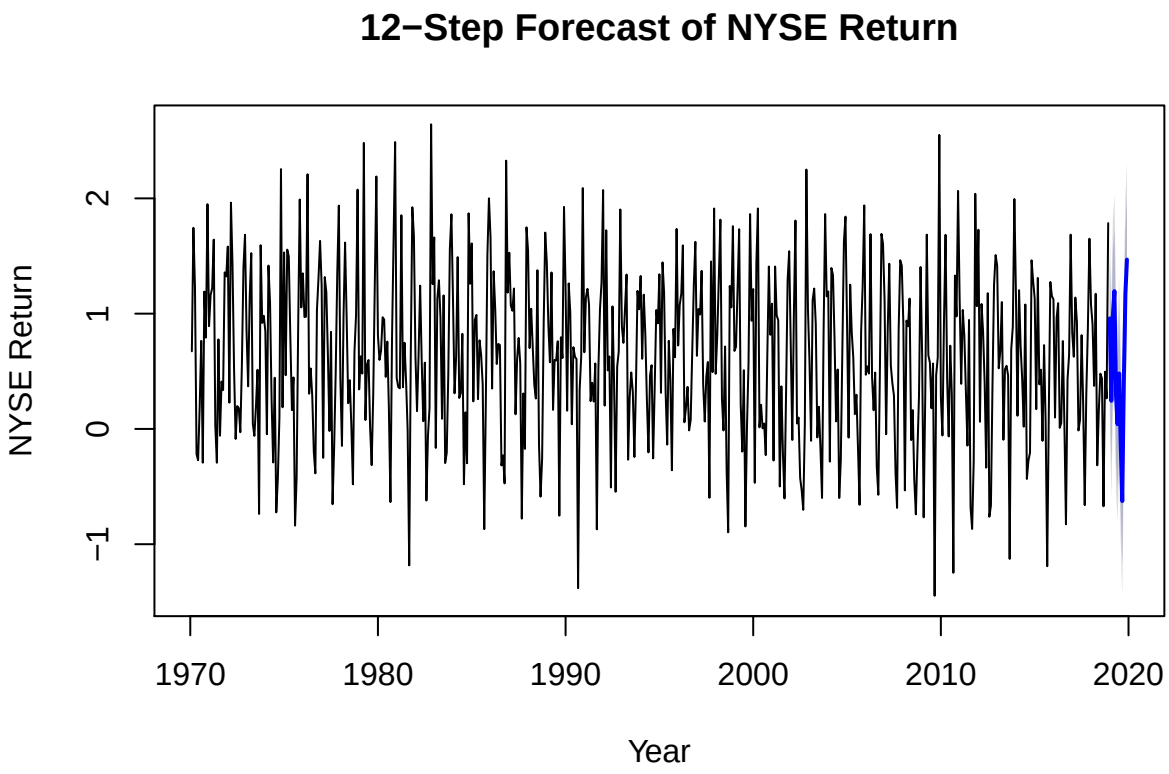



```
plot(forecast(ibm_mod_full$fitted , h=12), xlim=c(2018, 2020), main = "12-Step Forecast of IBM Return",
```

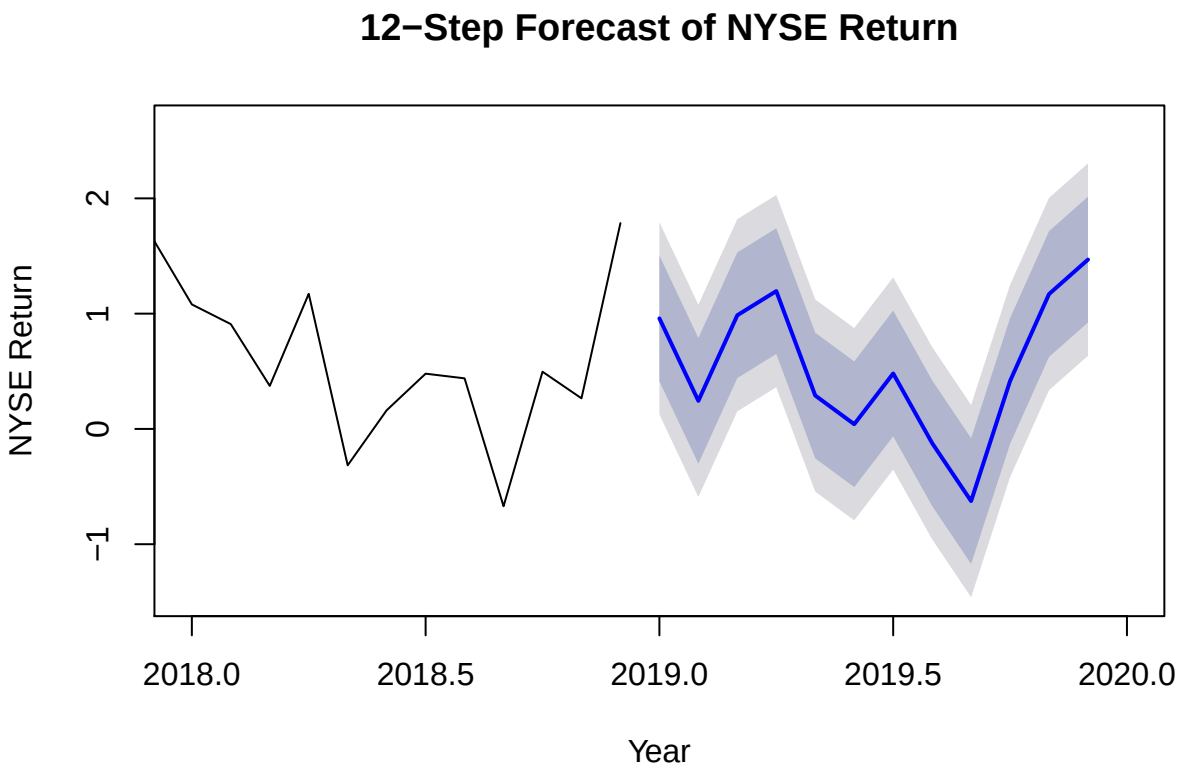


NYSE Forecast

```
plot(forecast(nyse_mod_full$fitted, h=12), main = "12-Step Forecast of NYSE Return", ylab = "NYSE Retur
```




```
plot(forecast(nyse_mod_full$fitted, h=12), xlim=c(2018, 2020), main = "12-Step Forecast of NYSE Return"
```



(j) Compare your forecast from (i) to the 12-steps ahead forecasts from ARIMA, Holt-Winters, and ETS models. Which model performs best in terms of MAPE?

IBM Model Comparison

```
# Generate ARIMA model automatically
IBM_arima <- auto.arima(ibm_return)

# Generate ets model
IBM_ets <- ets(ibm_return)

# Generate Holt-Winters model
IBM_hw <- hw(ibm_return)

# Forecast every model 12 steps ahead
ibm_arima_fcast <- IBM_arima %>% forecast(12)
ibm_ets_fcast <- IBM_ets %>% forecast(12)
ibm_hw_fcast <- IBM_hw %>% forecast(12)
ibm_own_fcast <- ibm_mod_full$fitted %>% forecast(12)

accuracy(ibm_arima_fcast, ibm_r)
```

```
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -4.622816e-16 7.054867 5.278928     -Inf      Inf 0.7158046
```

```
## Test set      1.183120e+00 7.652302 6.134990 107.6729 107.6729 0.8318837
##                      ACF1 Theil's U
## Training set -0.01630056        NA
## Test set     -0.18369191  1.121868
```

```
accuracy(ibm_ets_fcast, ibm_r)
```

```
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.001256516 7.055218 5.279184     -Inf      Inf 0.7158394
## Test set     1.184733488 7.652552 6.135259 107.6574 107.6574 0.8319202
##                      ACF1 Theil's U
## Training set -0.01630096        NA
## Test set     -0.18369191   1.12167
```

```
accuracy(ibm_hw_fcast, ibm_r)
```

```
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.140988 7.014174 5.290582      NaN      Inf 0.7173848
## Test set      2.096536 7.698219 6.280601 107.0028 107.1972 0.8516280
##                      ACF1 Theil's U
## Training set -0.01340125        NA
## Test set     -0.18700061  1.004136
```

```
accuracy(ibm_own_fcast, ibm_r)
```

```
##                        ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -0.003482088 0.680535 0.5268526 141.4880 297.2780 0.7749145
## Test set      1.204219659 7.334175 5.9884081 103.3759 108.6264 8.8079743
##                     ACF1 Theil's U
## Training set -0.1577213        NA
## Test set     -0.2223263  1.043303
```

NYSE Model Comparison

```
# Generate ARIMA model automatically
NYSE_arima <- auto.arima(nyse_return)

# Generate ETS model
NYSE_ets <- ets(nyse_return)

# Generate Holt-Winters model
NYSE_hw <- hw(nyse_return)

# Forecast every model 12 steps ahead
nyse_arima_fcast <- NYSE_arima %>% forecast(12)
nyse_ets_fcast <- NYSE_ets %>% forecast(12)
nyse_hw_fcast <- NYSE_hw %>% forecast(12)
nyse_own_fcast <- nyse_mod_full$fitted %>% forecast(12)

# Output a table of MAPE values and the respective model
accuracy(nyse_arima_fcast, nyse_r)
```

```
##                         ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.0004004473 4.339371 3.256574 92.93160 127.0592 0.7128248
## Test set      1.1759619375 3.725485 2.941146 31.33383 118.0116 0.6437815
##                    ACF1 Theil's U
## Training set  0.06603262        NA
## Test set     -0.27444305  1.148747
```

```
accuracy(nyse_ets_fcast, nyse_r)
```

```
##                         ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.0005664951 4.344089 3.253928 89.79251 128.9718 0.7122456
## Test set     1.1062436268 3.715116 2.870406 34.22189 106.9622 0.6282973
##                    ACF1 Theil's U
## Training set  0.06497122        NA
## Test set     -0.28856527   1.15164
```

```
accuracy(nyse_hw_fcast, nyse_r)
```

```
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.07313334 4.316735 3.232596  87.82787 140.1824 0.7075763
## Test set      1.65974227 3.868529 2.999678 100.08513 102.1944 0.6565934
##                    ACF1 Theil's U
## Training set  0.06207237        NA
## Test set     -0.29124192 0.7236786
```

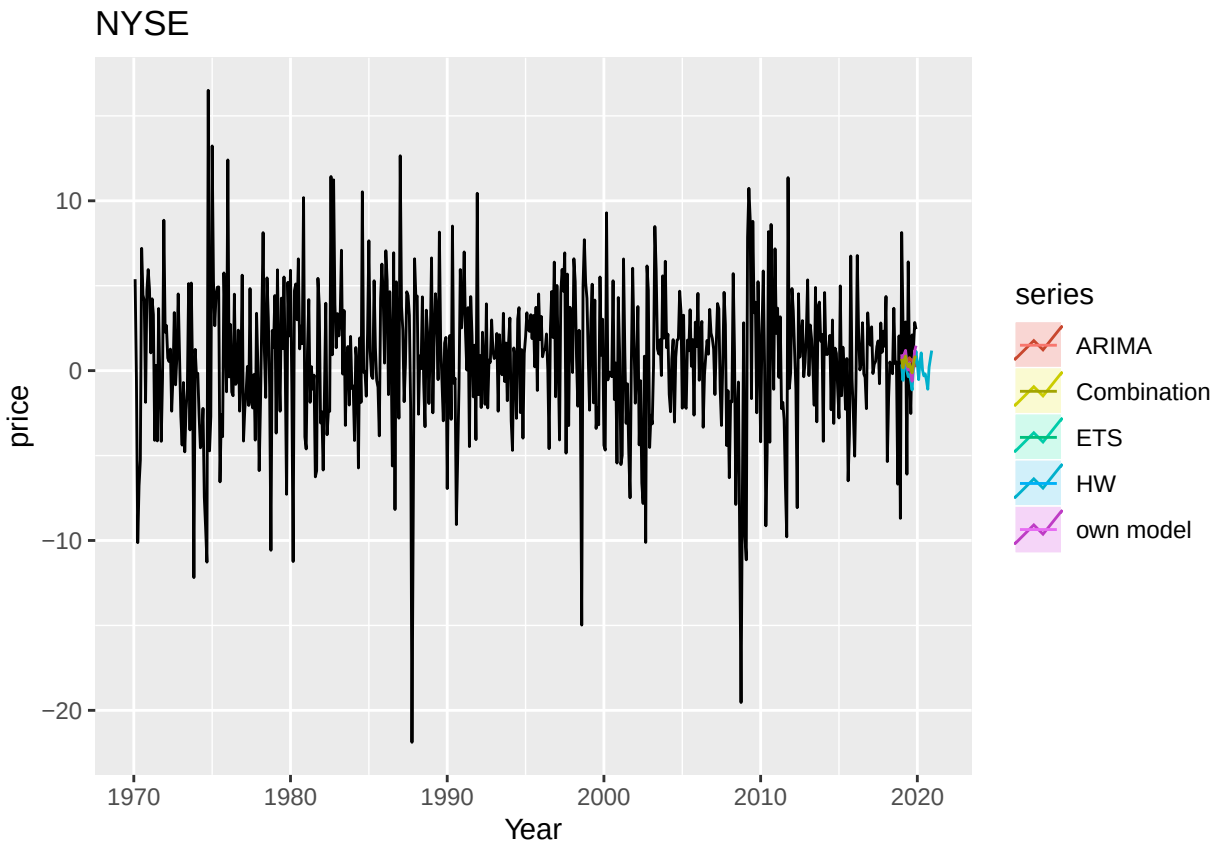
```
accuracy(nyse_own_fcast, nyse_r)
```

```
##                         ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.02893535 0.4201389 0.3340849 -11.83836 133.8958 0.8096643
## Test set      1.19124692 3.6535246 2.8098550  35.16504 103.6791 6.8097643
##                     ACF1 Theil's U
## Training set -0.2779995        NA
## Test set     -0.3252015   0.88063
```

Our model performs best in terms of MAPE

(k) Combine the four forecasts and comment on the MAPE from this forecasts vs., the individual ones.

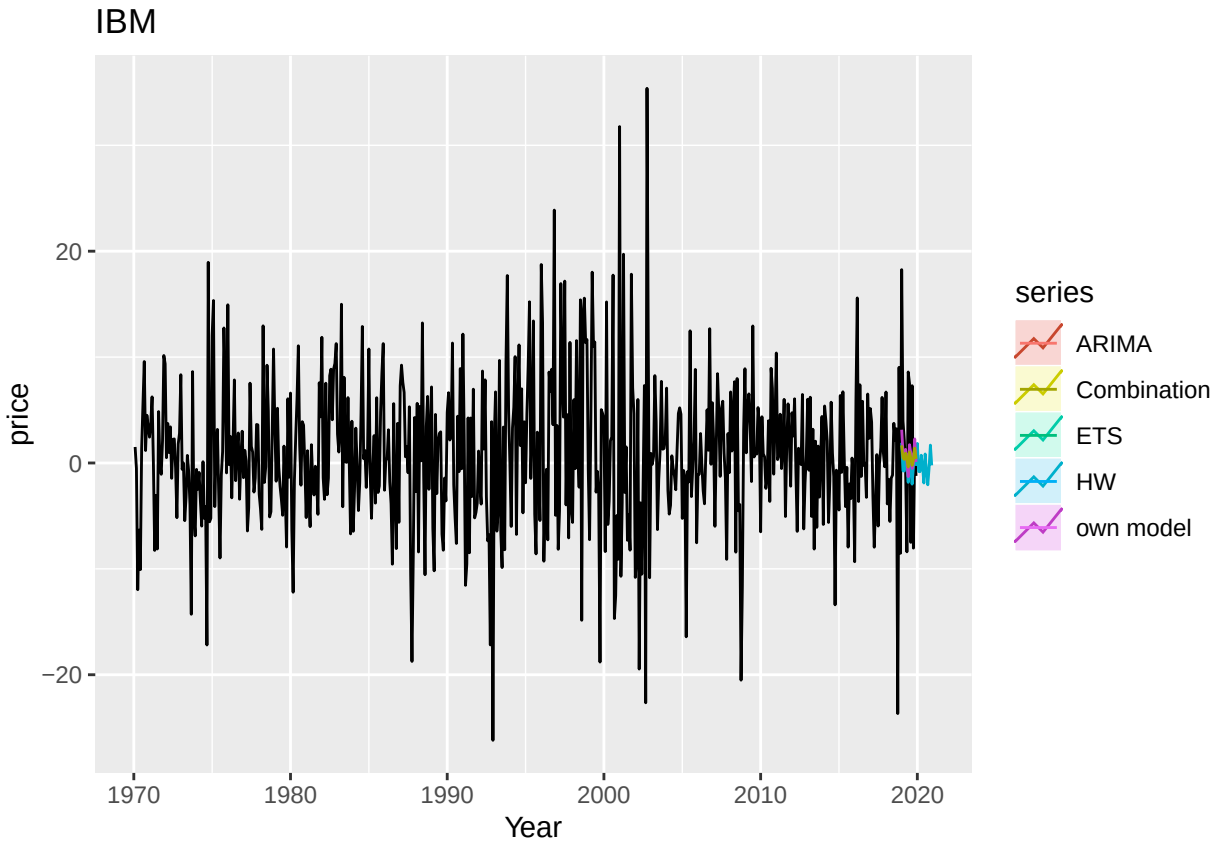
```
combination.nyse <- (nyse_ets_fcast[["mean"]]+ nyse_arima_fcast[["mean"]]+ nyse_hw_fcast[["mean"]]+ nys
autoplot(nyse_r) +
autolayer(nyse_ets_fcast, series="ETS", PI=FALSE) + autolayer(nyse_arima_fcast, series="ARIMA", PI=FALS
```




```
c(ETS = accuracy(nyse_ets_fcast, nyse_r)["Test set","MAPE"], ARIMA = accuracy(nyse_arima_fcast, nyse_r
```

```
##         ETS       ARIMA          HW Combination
##   106.96223   118.01165   102.19443    96.01206
```

```
combination.ibm <- (ibm_ets_fcast[["mean"]]+ ibm_arima_fcast[["mean"]]+ ibm_hw_fcast[["mean"]]+ ibm_own
autoplot(ibm_r) +
autolayer(ibm_ets_fcast, series="ETS", PI=FALSE) + autolayer(ibm_arima_fcast, series="ARIMA", PI=FALSE)
```




```
c(ETS = accuracy(ibm_ets_fcast, ibm_r)["Test set","MAPE"], ARIMA = accuracy(ibm_arima_fcast, ibm_r)["T
```

```
##         ETS       ARIMA          HW Combination
##    107.6574    107.6729    107.1972    106.4273
```

The combination forecast perform better in forecasting. We need to take it into account when trying to forecast.

(l) Fit an appropriate VAR model using your two variables. Make sure to show the relevant plots and discuss your results from the fit.

```
y <- cbind(nyse_r, ibm_r)
y_ts <- data.frame(y)
VARselect(y_ts[-1,], lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                1          2          3          4          5          6
## AIC(n)  6.498335   6.504898   6.510255   6.513996   6.522228   6.528615
```

```
## HQ(n)    6.515736   6.533900   6.550858   6.566199   6.586032   6.604020
## SC(n)    6.542995   6.579332   6.614463   6.647977   6.685983   6.722144
## FPE(n) 664.035198 668.408160 671.999381 674.519501 680.098067 684.459305
##                 7          8          9         10
## AIC(n)   6.540083   6.551928   6.558918   6.567550
## HQ(n)    6.627089   6.650534   6.669125   6.689358
## SC(n)    6.763385   6.805004   6.841767   6.880174
## FPE(n) 692.359418 700.616289 705.539445 711.667696
```

we can see that the lowest AIC is when p = 1 then we choose p = 1.

```
y_model1=VAR(y_ts[-1,], p = 1)
summary(y_model1)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: nyse_r, ibm_r
## Deterministic variables: const
## Sample size: 597
## Log Likelihood: -3629.05
## Roots of the characteristic polynomial:
## 0.05814 0.03918
## Call:
## VAR(y = y_ts[-1, ], p = 1)
##
##
## Estimation results for equation nyse_r:
## =======================================
## nyse_r = nyse_r.l1 + ibm_r.l1 + const
##
##           Estimate Std. Error t value Pr(>|t|)
## nyse_r.l1  0.07196    0.04906   1.467 0.143027
## ibm_r.l1  -0.01887    0.03004  -0.628 0.530267
## const      0.61392    0.17961   3.418 0.000674 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.338 on 594 degrees of freedom
## Multiple R-Squared: 0.003685,    Adjusted R-squared: 0.0003304
## F-statistic: 1.098 on 2 and 594 DF,  p-value: 0.334
##
##
## Estimation results for equation ibm_r:
## ======================================
## ibm_r = nyse_r.l1 + ibm_r.l1 + const
##
##           Estimate Std. Error t value Pr(>|t|)
## nyse_r.l1  0.08136    0.08018   1.015  0.31062
## ibm_r.l1  -0.05299    0.04909  -1.079  0.28088
## const      0.81138    0.29349   2.765  0.00588 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 7.088 on 594 degrees of freedom
## Multiple R-Squared: 0.002383,    Adjusted R-squared: -0.0009762
## F-statistic: 0.7094 on 2 and 594 DF,  p-value: 0.4924
##
##
##
## Covariance matrix of residuals:
##        nyse_r ibm_r
## nyse_r  18.82 16.90
## ibm_r   16.90 50.24
##
## Correlation matrix of residuals:
##        nyse_r  ibm_r
## nyse_r 1.0000 0.5496
## ibm_r  0.5496 1.0000
```

From the results, when trying to explain NYSE using IBM, we only see a little to no significant. When trying to explain IBM using NYSE, we do not see any statisticaly significant parameter.

(m) Compute, plot, and interpret the respective impulse response functions.

```
irf(y_model1)
```

```
##
## Impulse response coefficients
## $nyse_r
##             nyse_r        ibm_r
##  [1,] 4.337709e+00 3.895718e+00
##  [2,] 2.386233e-01 1.464958e-01
##  [3,] 1.440637e-02 1.165219e-02
##  [4,] 8.167806e-04 5.546959e-04
##  [5,] 4.830661e-05 3.706216e-05
##  [6,] 2.776690e-06 1.966443e-06
##  [7,] 1.626980e-07 1.217174e-07
##  [8,] 9.410620e-09 6.787770e-09
##  [9,] 5.490836e-10 4.059907e-10
## [10,] 3.184992e-11 2.316157e-11
## [11,] 1.854797e-12 1.364068e-12
##
## $ibm_r
##              nyse_r         ibm_r
##  [1,]  0.000000e+00  5.921707e+00
##  [2,] -1.117202e-01 -3.137780e-01
##  [3,] -2.119047e-03  7.536702e-03
##  [4,] -2.946655e-04 -5.717611e-04
##  [5,] -1.041575e-05  6.321999e-06
##  [6,] -8.687397e-07 -1.182427e-06
##  [7,] -4.020244e-08 -8.027581e-09
```

```
##  [8,] -2.741325e-09 -2.845556e-09
##  [9,] -1.435678e-10 -7.225791e-11
## [10,] -8.967217e-12 -7.852063e-12
## [11,] -4.970993e-13 -3.135204e-13
##
##
## Lower Band, CI= 0.95
## $nyse_r
##              nyse_r         ibm_r
##  [1,]  3.939708e+00  3.309994e+00
##  [2,] -6.379503e-02 -4.838143e-01
##  [3,] -1.073824e-02 -1.515389e-02
##  [4,] -7.313343e-04 -3.922126e-03
##  [5,] -1.454296e-05 -6.361025e-05
##  [6,] -8.052499e-06 -4.384505e-05
##  [7,] -5.439905e-08 -4.148189e-07
##  [8,] -6.472008e-08 -4.426156e-07
##  [9,] -4.239887e-10 -4.722913e-09
## [10,] -5.585334e-10 -4.366228e-09
## [11,] -5.389881e-13 -2.913966e-11
##
## $ibm_r
##              nyse_r         ibm_r
##  [1,]  0.000000e+00  5.421469e+00
##  [2,] -4.442380e-01 -7.700745e-01
##  [3,] -3.744193e-02 -2.898056e-02
##  [4,] -4.222457e-03 -9.245875e-03
##  [5,] -3.488818e-04 -4.311045e-04
##  [6,] -5.456846e-05 -1.387901e-04
##  [7,] -7.264310e-06 -5.068706e-06
##  [8,] -1.196454e-06 -3.371197e-06
##  [9,] -1.739969e-07 -9.040632e-08
## [10,] -2.812017e-08 -8.681629e-08
## [11,] -4.306507e-09 -2.637447e-09
##
##
## Upper Band, CI= 0.95
## $nyse_r
##             nyse_r        ibm_r
##  [1,] 4.683708e+00 4.407712e+00
##  [2,] 6.346157e-01 7.391420e-01
##  [3,] 9.888390e-02 8.755924e-02
##  [4,] 1.559460e-02 1.209288e-02
##  [5,] 2.455281e-03 1.695522e-03
##  [6,] 3.902145e-04 2.499389e-04
##  [7,] 6.138885e-05 4.061568e-05
##  [8,] 9.702706e-06 6.234530e-06
##  [9,] 1.534413e-06 1.009885e-06
## [10,] 2.438193e-07 1.579297e-07
## [11,] 3.878322e-08 2.558862e-08
##
## $ibm_r
##             nyse_r        ibm_r
##  [1,] 0.000000e+00 6.345484e+00
```

```
##  [2,] 1.921495e-01 2.176254e-01
##  [3,] 1.634088e-02 8.053358e-02
##  [4,] 2.202556e-03 1.689331e-03
##  [5,] 2.692040e-04 1.179277e-03
##  [6,] 3.590232e-05 3.448941e-05
##  [7,] 3.809339e-06 1.770177e-05
##  [8,] 5.880889e-07 4.641092e-07
##  [9,] 6.011461e-08 2.769028e-07
## [10,] 9.758048e-09 6.185948e-09
## [11,] 1.139920e-09 4.744262e-09
```

The NYSE and IBM Index turn decays every year.

(n) Perform a Granger-Causality test on your variables and discuss your results from the test.

Granger causality test

```
#H0: IBM does not cause NYSE (do not reject)
grangertest(nyse_r ~ ibm_r, order = 1)
```

```
## Granger causality test
##
## Model 1: nyse_r ~ Lags(nyse_r, 1:1) + Lags(ibm_r, 1:1)
## Model 2: nyse_r ~ Lags(nyse_r, 1:1)
##   Res.Df Df     F Pr(>F)
## 1    595
## 2    596 -1 0.385 0.5352
```

we fail to reject the null hence we cannot predict NYSE return using IBM return.

```
#H0: NYSE does not cause IBM (do not reject)
grangertest(ibm_r ~ nyse_r, order = 1)
```

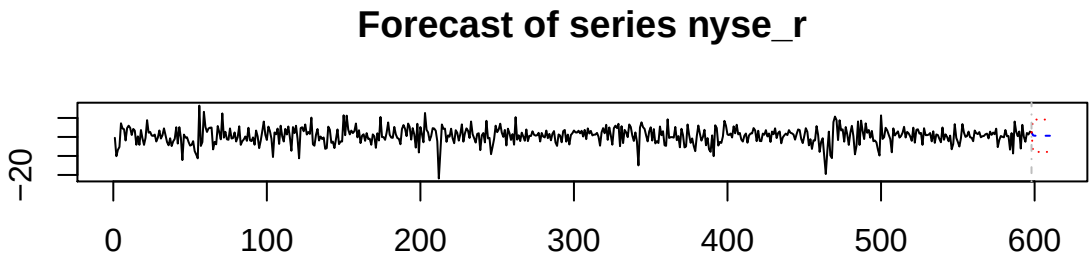
```
## Granger causality test
##
## Model 1: ibm_r ~ Lags(ibm_r, 1:1) + Lags(nyse_r, 1:1)
## Model 2: ibm_r ~ Lags(ibm_r, 1:1)
##   Res.Df Df      F Pr(>F)
## 1    595
## 2    596 -1 1.0097 0.3154
```

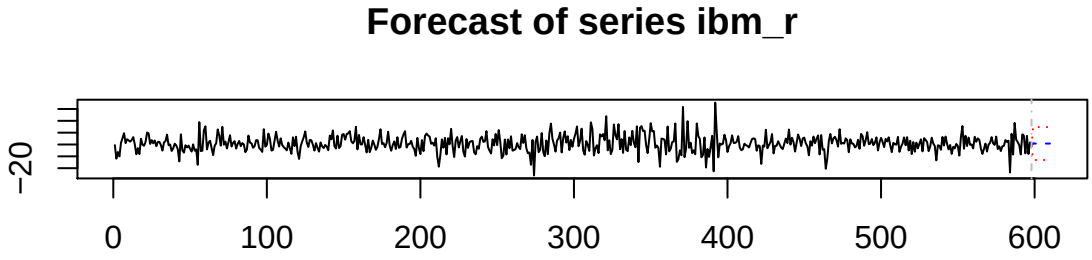
we fail to reject the null hence we cannot predict IBM return using NYSE return.

For this particular stock market return case, it is somehow logical since we cannot really predict stock market with just one component. There are a lot of little variable that can contribute to the stock market which is why if we only account for one stock to predict another, it would not work.

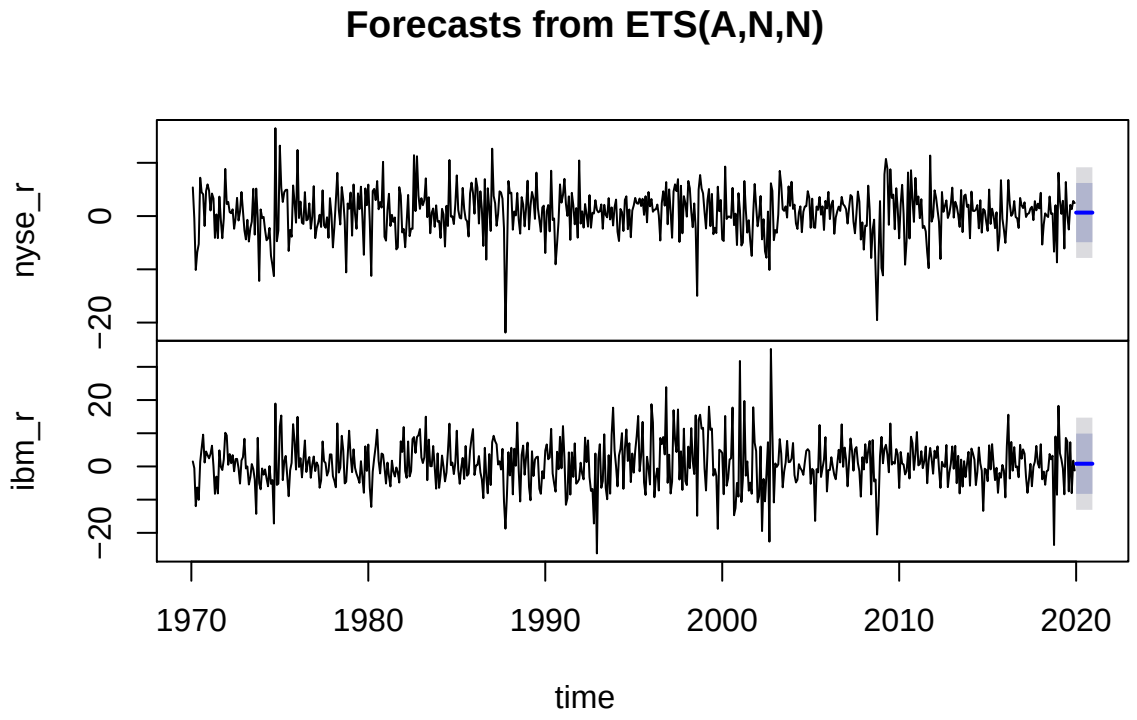
(o) Use your VAR model to forecast 12-steps ahead. Your forecast should include the respective error bands. Comment on the differences between the VAR forecast and the other ones obtained using the different methods.

```
var.predict <- predict(object=y_model1, n.ahead=12)
plot(var.predict)
```






```
plot(forecast(y, h=12))
```



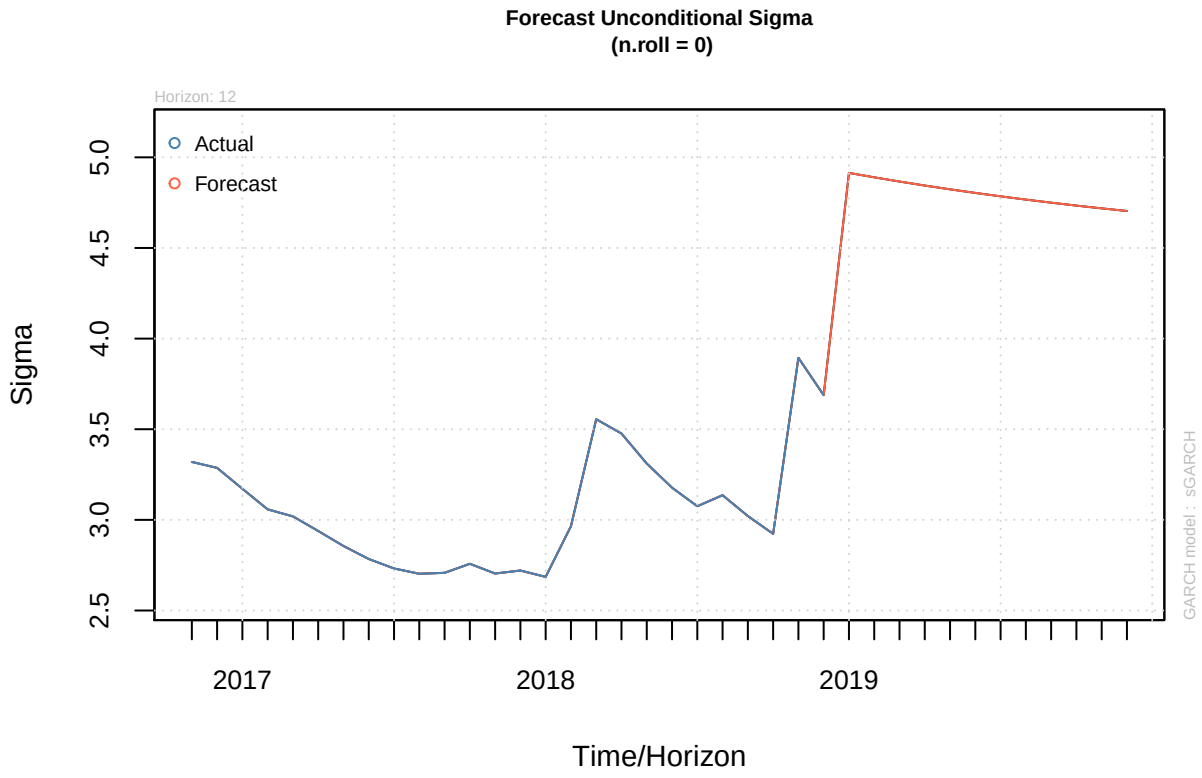
(p) Fit a GARCH model to the residuals from your favorite model, and produce a new 12-steps ahead forecast, including one for the variance.

We use our NYSE index as our favorite model for the GARCH Model

```
#load favorite model residuals
nyse_resid <- NYSE_arima$residuals

# load specification for garch model
nyse_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),mean.model = nyse_mod_full,
  distribution.model = "sstd")
```

```
## Warning: unidentified option(s) in mean.model:
##  coef sigma2 var.coef mask loglik aic arma residuals call series code n.cond nobs model aicc bic xreg
```

```
# Fit garch model for nyse data
nyse_garch <- ugarchfit(spec = nyse_spec, data = nyse_return)

# Forecast garch model
nyse_garch_fcast <- ugarchforecast(nyse_garch, n.ahead = 12, n.roll = 0, out.sample = 0)

plot(nyse_garch_fcast, which = 3)
```



III. Conclusions and Future Work

NYSE stock does not fully explained by IBM stock and it also works the other way around. One of the reason for is is that there are a lot of macroeconomics variable that might affect the fluctuation of both stocks, making it unpredictable. We have seen that stock market return cannot be easily predict with just fitting trend seasonality and cycle. One of the better ways to forecast it is using the GARCH model which solely analyze the remainder white noise part of the time series.

IV. References

from 'quantmod' library using the getSymbols function to get the financial data. Most likely the data came from yahoo finance. IBM: https://finance.yahoo.com/quote/IBM?p=IBM&.tsrc=fin-srch-v1 NYSE: https://finance.yahoo.com/quote/%5ENYA?p=%5ENYA&.tsrc=fin-srch

V. R Source Code

In the R Code