Project 2

Ian Cranfield [redacted]

Due 6/4/2020

I. Introduction

When trying to assess how well a particular stock is doing, it is natural to assume that a stocks price may depend on how well that company is doing. One variable to look at to assess a company's health is the company's revenue. It makes sense that as a company grows and revenue increases, most likely their stock price will as well. In this project, we will analyze both the quarterly revenue and the quarterly stock closing price of **Apple Inc**. We get our revenue data from macrotrends.net and our stock closing price data from Yahoo Finance. For the purposes of this project we use the past 20 years of available data to ensure we capture the trend, cycles, and seasonality. We expect for both series to move together and to exhibit much of the same dynamics. We will analyze the data to see if our assumptions are correct. We use a training set of 2005 Q2 to 2017 Q2. We leave out 12 observations ~ 3 years of data to assess our forecast.

Packages Used

```
library(rvest) ## able to webscrape the quarterly revenue
library(dplyr)
library(ggplot2)
library(forecast)
library(readr) ## for the csv file
library(strucchange) # for efp CUSUM plot
library(tidyverse)
library(tseries)
library(vars)
library(lmtest)
library(car)
library(Metrics)
library(stats)
library(moments)
library(rugarch)
```

Cleaning Quarterly AAPL Revenue Data

```
aapl_rev <- read_html("https://www.macrotrends.net/stocks/charts/AAPL/apple/revenue")
aapl.html <- html_nodes(aapl_rev, "thead+ thead th , #style-1 td")

aapl.results <- rvest::html_text(aapl.html)

Date <- aapl.results[seq(1, length(aapl.results), 2)]

`Revenue` <- aapl.results[seq(2, length(aapl.results),2)]
```

```
aapl.revenue <- data.frame(Date = Date[16:76], Revenue = `Revenue`[16:76] , stringsAsFactors = FALSE) %>%
  arrange(Date)

aapl.revenue$Date <- as.Date(aapl.revenue$Date, format = "%Y-%m-%d")

aapl.revenue$Revenue <- as.numeric(gsub("\\$|,","",aapl.revenue$Revenue ))

aapl.rev.ts <- ts(aapl.revenue$Revenue, start = 2005.25, frequency = 4)
```

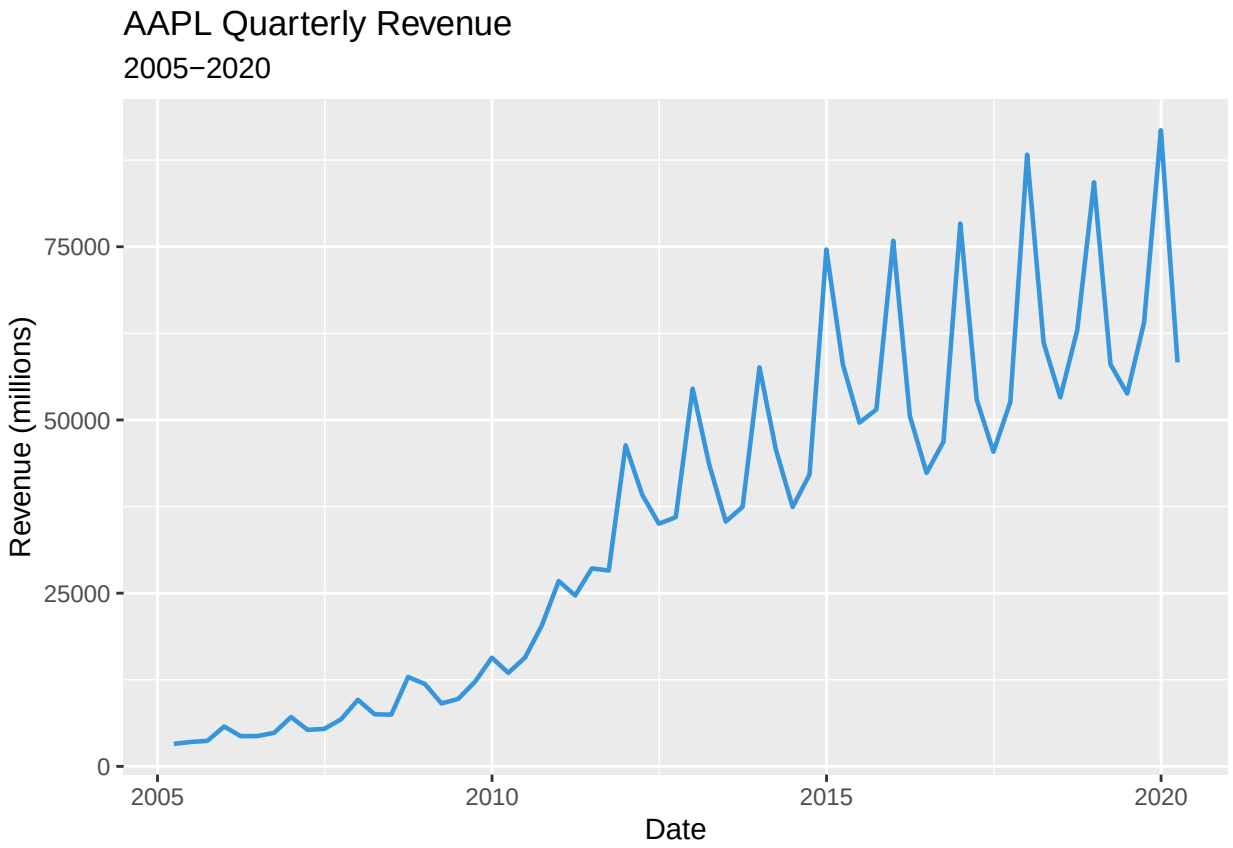
Cleaning Quarterly AAPL Stock Price Data

```
AAPL <- read_csv("AAPL.csv", col_types = cols(Date = col_date(format = "%Y-%m-%d"),
Volume = col_skip()))

aapl.monthly <- ts(AAPL$`Adj Close`, start = c(2005,2), frequency = 12)

aapl.quarterly <- ts(aggregate(aapl.monthly, nfrequency = 4)/3, start = 2005.25, frequency = 4)

aapl.stock.close <- data.frame(date=time(aapl.quarterly), close =as.matrix(aapl.quarterly))
```

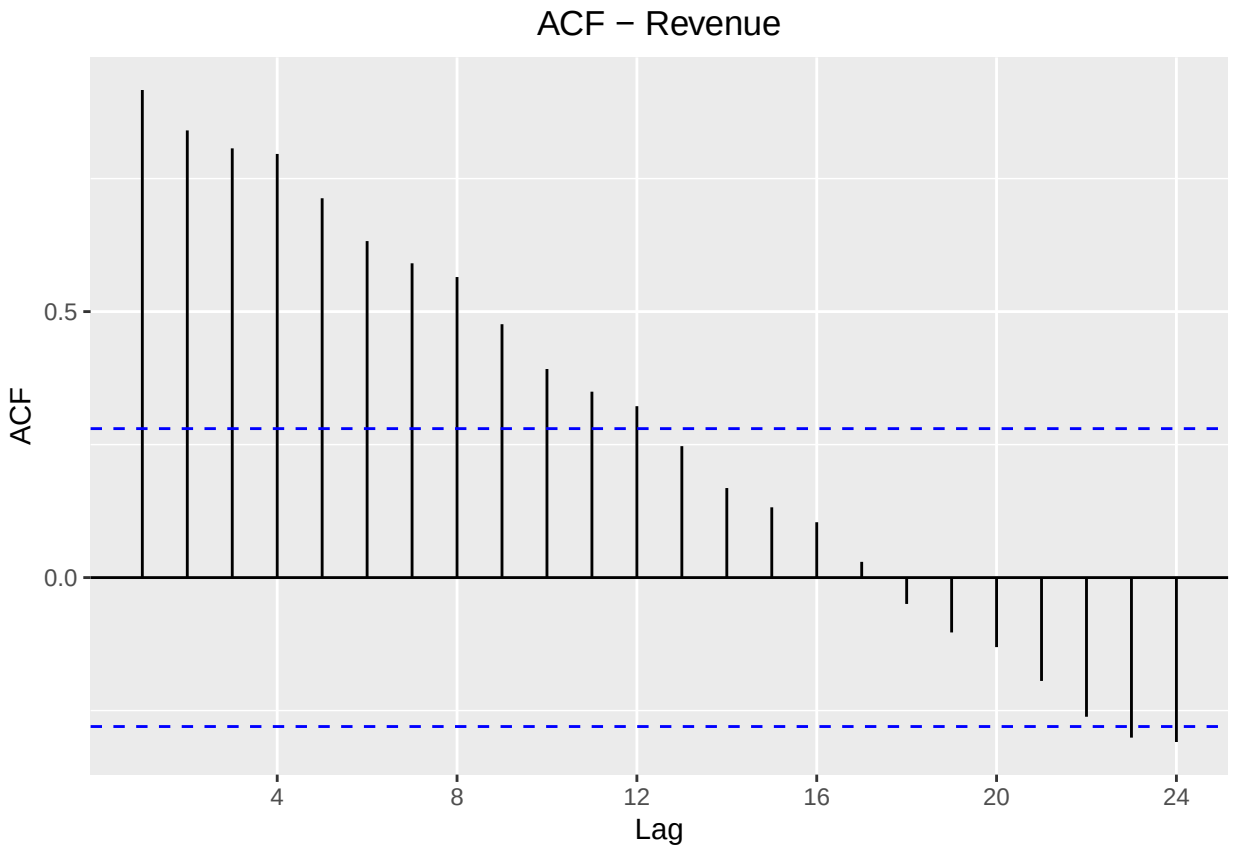
II. Results (answers and plots)

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

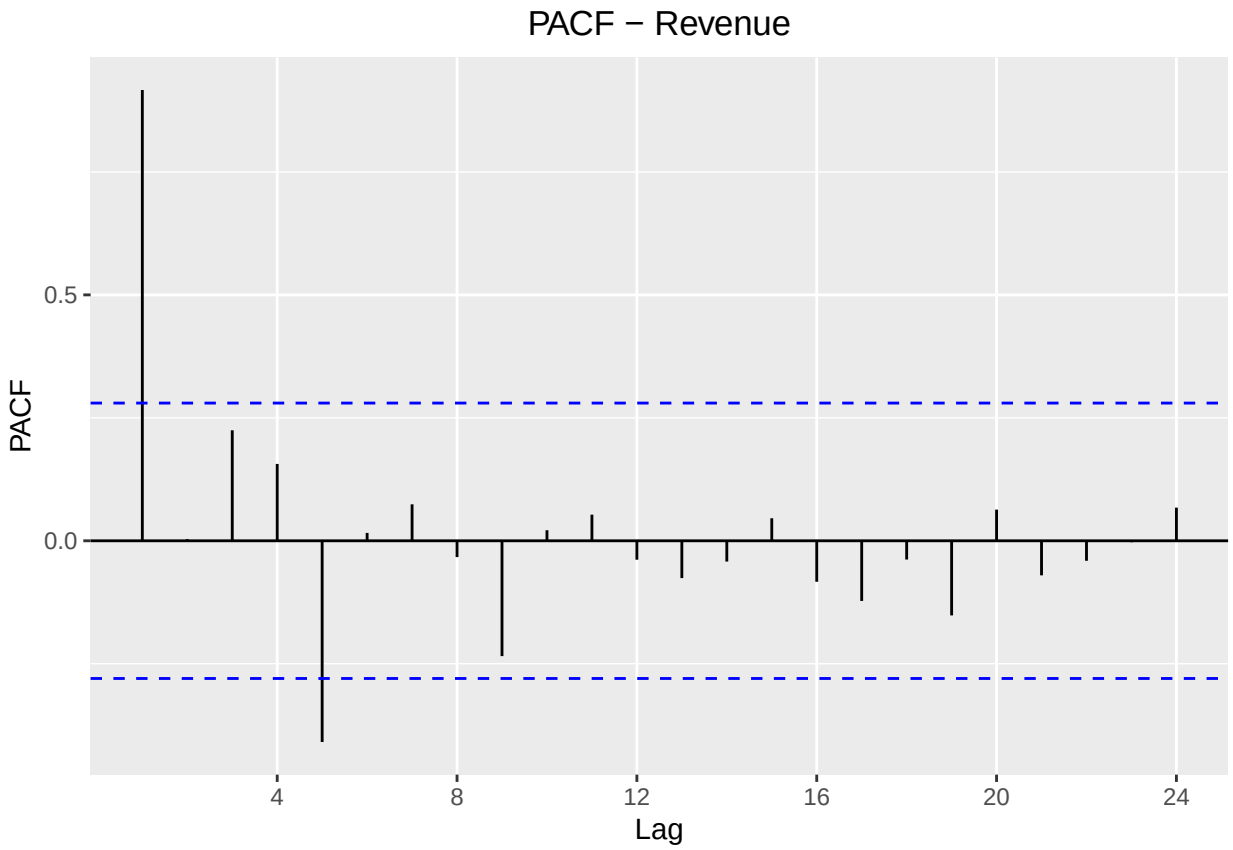
```
aapl <- cbind(aapl.revenue, Close = aapl.stock.close$close)


#Training and Test Set
aapl.train <- aapl[1:49,]
aapl.test <- aapl[50:61,]

#Times Series Variables
RevenueTS <- ts(aapl.train$Revenue, start = 2005, frequency = 4)
RevenueTS_test <- ts(aapl.test$Revenue, start = 2017.25, frequency = 4)
aapl.revenue.ts <-  ts(aapl$Revenue, start = 2005, frequency = 4)


CloseTS <- ts(aapl.train$Close, start = 2005, frequency = 4)
CloseTS_test <- ts(aapl.test$Close, start = 2017.25, frequency = 4)
aapl.close.ts <-  ts(aapl$Close, start = 2005, frequency = 4)

#Log of TS Variables
lgdata=log(RevenueTS)
lgclose=log(CloseTS)

## Quarterly Revenue AAPL
ggplot(aapl, aes(x = Date, y = Revenue)) +
  geom_line(color="#3796db", lwd = 0.8) +
  ggtitle("AAPL Quarterly Revenue", "2005-2020") +
  xlab("Date") +
  ylab("Revenue (millions)")
```



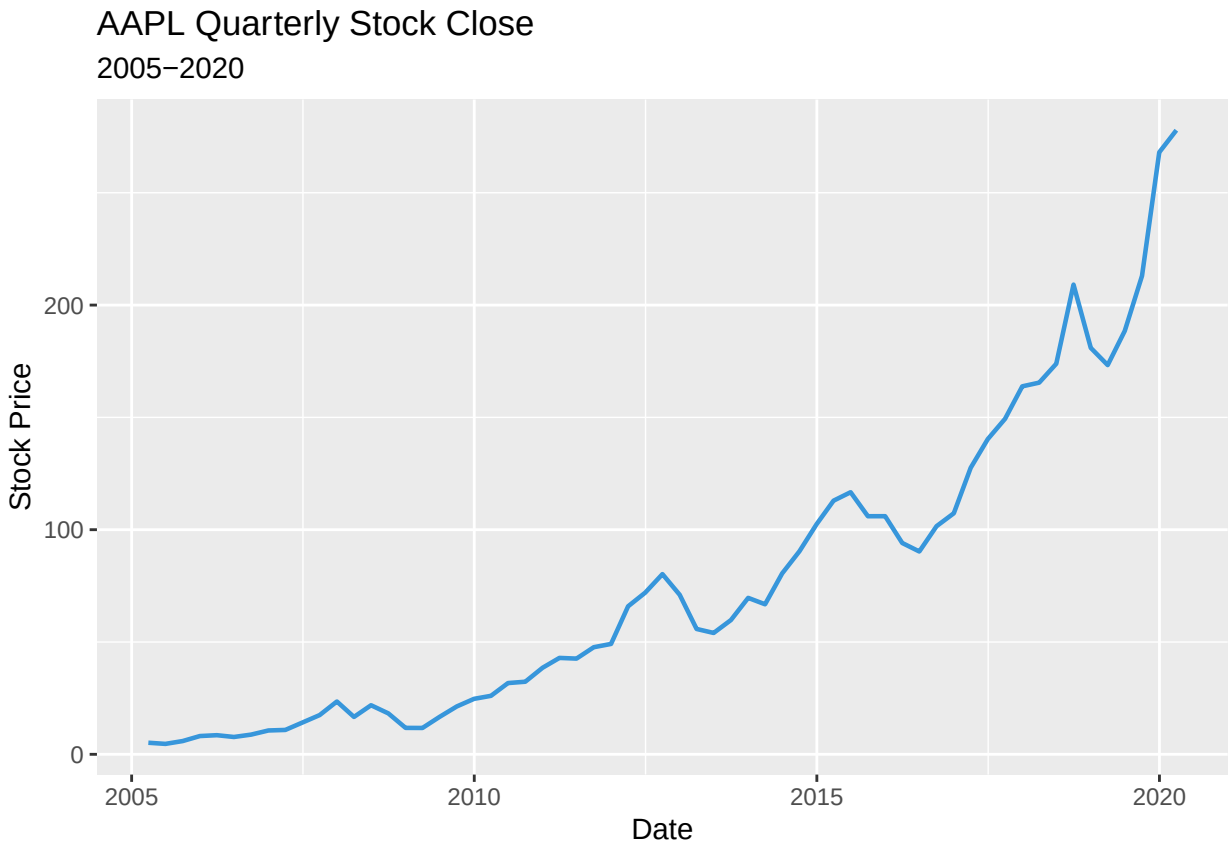

```
par(mfrow=c(2, 1))
ggAcf(lgdata, main="ACF - Revenue", lag.max=24)+
  theme(plot.title = element_text(hjust = 0.5))
```



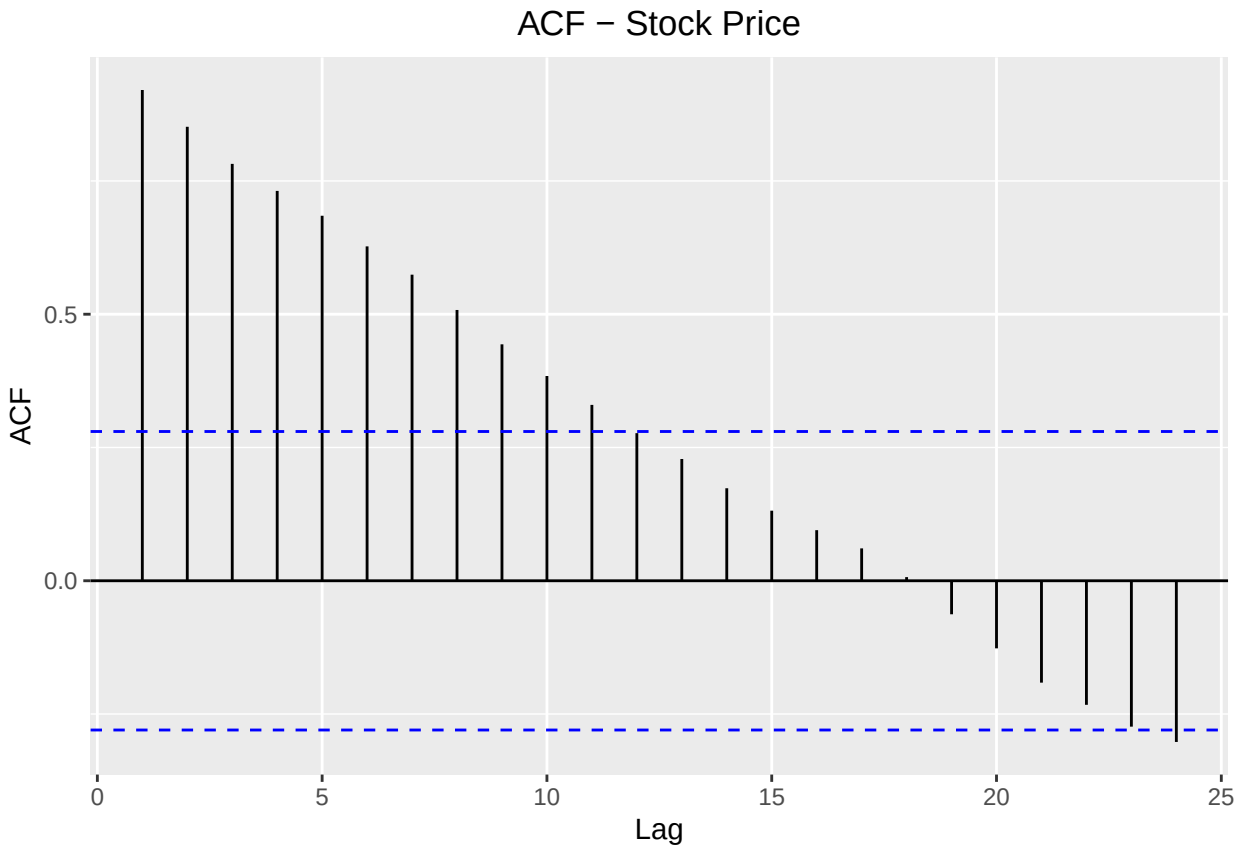

```
ggPacf(lgdata, main="PACF - Revenue", lag.max=24)+
  theme(plot.title = element_text(hjust = 0.5))  #AR 5
```



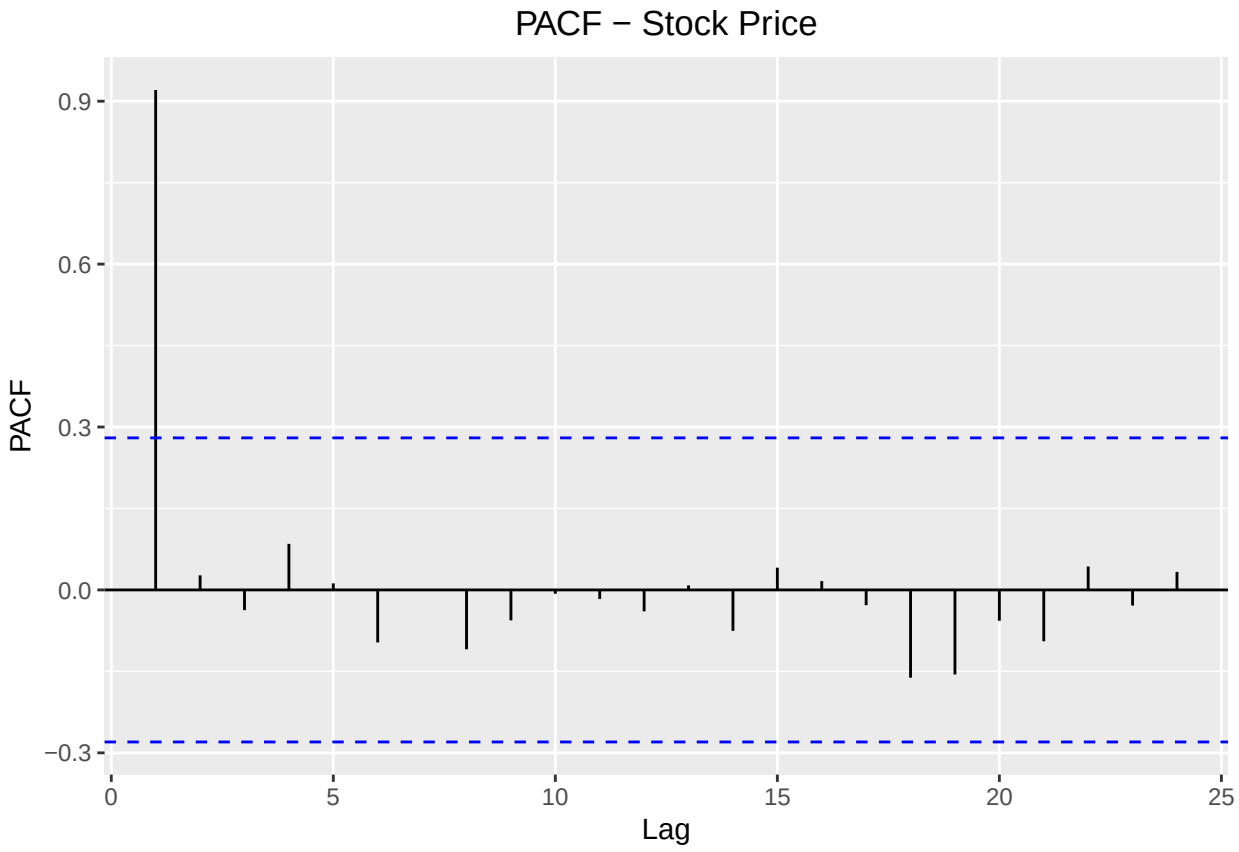

```
## AAPL Stock Close at Quarters

ggplot(aapl, aes(x = Date, y = Close)) +
  geom_line(color="#3796db", lwd = 0.8) +
  ggtitle("AAPL Quarterly Stock Close", "2005-2020") +
  xlab("Date") +
  ylab("Stock Price")
```




```
par(mfrow=c(2, 1))
ggAcf(aapl.train$Close, main="ACF - Stock Price", lag.max=24) +
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggPacf(aapl.train$Close, main="PACF - Stock Price", lag.max=24) +
  theme(plot.title = element_text(hjust = 0.5))  #AR 1
```



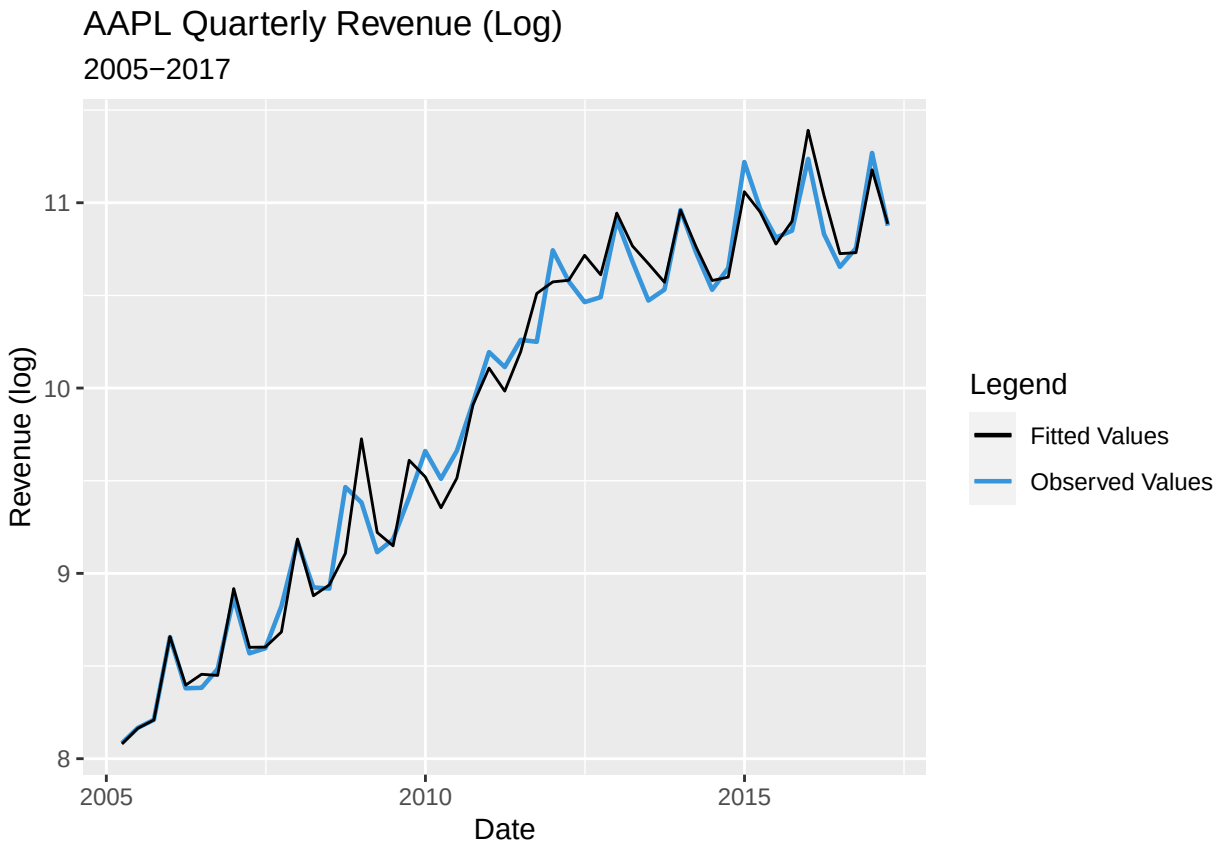
(b) Fit a model that includes, trend, seasonality and cyclical components. Make sure to discuss your model in detail.

Revenue Model

```
# Revenue Model
Rev_model=  auto.arima(lgdata)

#Stock Model
Stock_model <- auto.arima(CloseTS)


#Add Fitted and Residial variables to Training data set.
aapl.train <- aapl.train %>%
  mutate(
    Revenue_Fitted = Rev_model$fitted,
    Revenue_Residuals = Rev_model$residuals,
    Close_Fitted = Stock_model$fitted,
    Close_Residuals = Stock_model$residuals
  )
```

```
## Revenue
Rev_cols <- c("Fitted Values"="black", "Observed Values"="#3796db")

ggplot(aapl.train, aes(x = Date, y = log(Revenue))) +
  geom_line(aes(color = "Observed Values"), lwd=0.8) +
```

```
  geom_line(aes(y = Revenue_Fitted, color="Fitted Values")) +
  ggtitle("AAPL Quarterly Revenue (Log)", "2005-2017") +
  xlab("Date") +
  ylab("Revenue (log)") +
  scale_color_manual("Legend", values = Rev_cols)
```



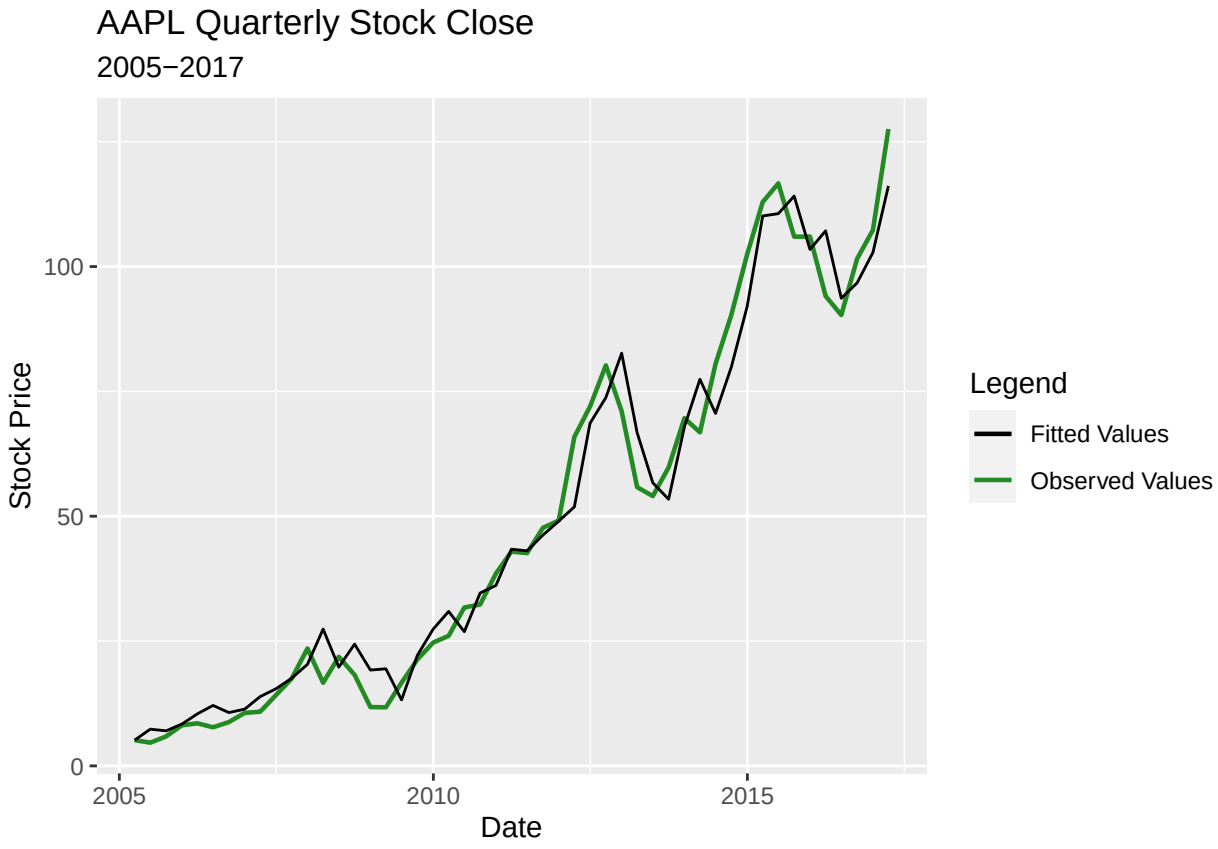

```
summary(Rev_model)
```

```
## Series: lgdata
## ARIMA(0,1,1)(0,1,1)[4]
##
## Coefficients:
##           ma1     sma1
##       -0.2887  -0.3049
## s.e.   0.1405   0.1725
##
## sigma^2 estimated as 0.01826:  log likelihood=26.43
## AIC=-46.86   AICc=-46.26   BIC=-41.5
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE      MAPE      MASE
## Training set -0.01095241 0.1250996 0.0886327 -0.1032801 0.8883947 0.3520912
##                      ACF1
## Training set -0.01561631
```

For revenue we let auto.arima choose an MA(1) S-MA(1) model with differencing and seasonal differencing. This fits better than the AR(5) model that would have been intuitivly fit due to the series PACF.

Stock Price Model

```
## Stock Price
Close_cols <- c("Fitted Values" = "black", "Observed Values" = "Forest Green")

ggplot(aapl.train, aes(x = Date, y = Close)) +
  geom_line(aes(color = "Observed Values"), lwd=0.8) +
  geom_line(aes(y = Close_Fitted, color="Fitted Values")) +
 ggtitle("AAPL Quarterly Stock Close", "2005-2017") +
  xlab("Date") +
  ylab("Stock Price") +
  scale_color_manual("Legend", values = Close_cols)
```




```
summary(Stock_model)
```

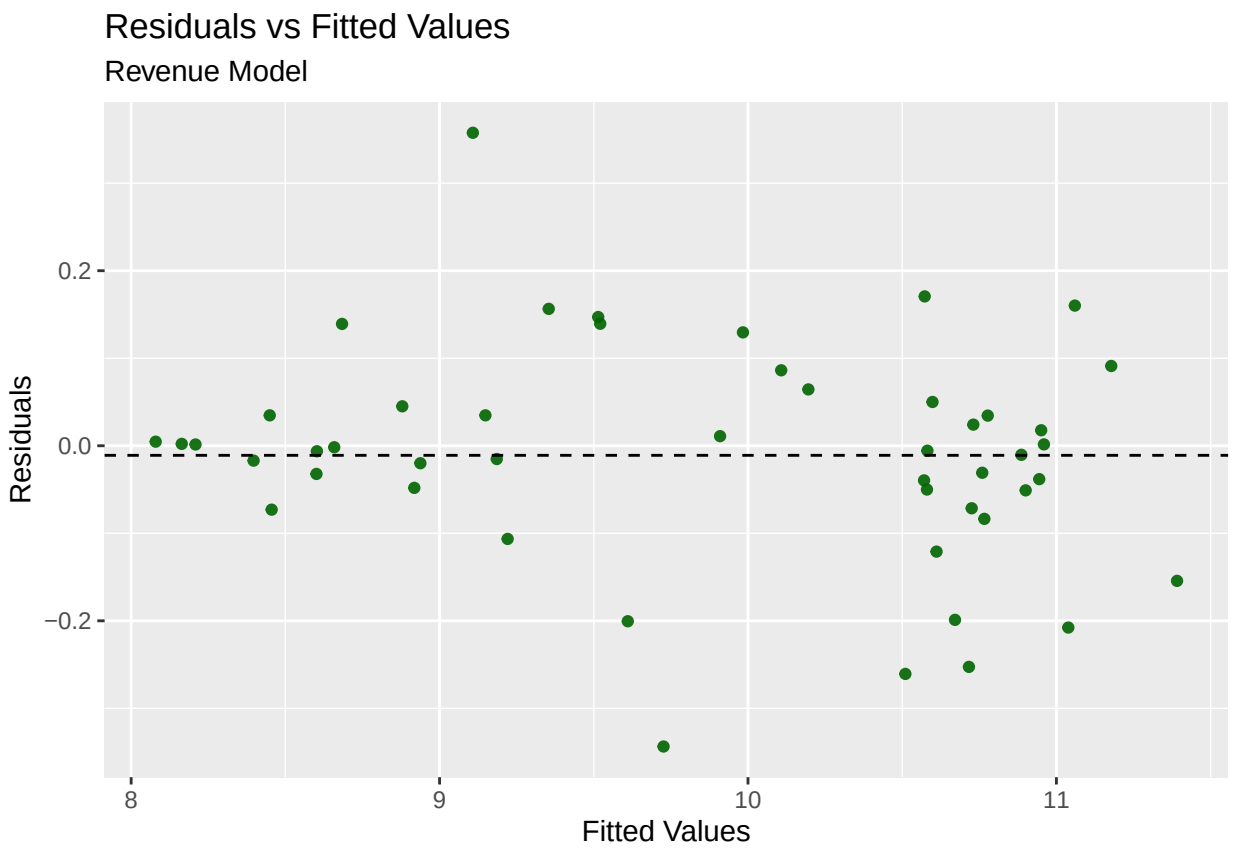
```
## Series: CloseTS
## ARIMA(0,1,0)(0,0,1)[4] with drift
##
## Coefficients:
##          sma1   drift
##       -0.4845  2.5012
## s.e.   0.1399  0.5054
##
## sigma^2 estimated as 40.58:  log likelihood=-156.5
## AIC=319   AICc=319.55   BIC=324.62
##
## Training set error measures:
##                      ME     RMSE    MAE       MPE     MAPE      MASE      ACF1
```

```
## Training set -0.1856567 6.172235 4.7728 -8.159749 15.40947 0.3327404 0.1494202
```

For the stock price we let auto.arima choose a S-MA(1) model with differencing and drift. This fits better than the AR(1) model that would have been intuitivly fit due to the series PACF.

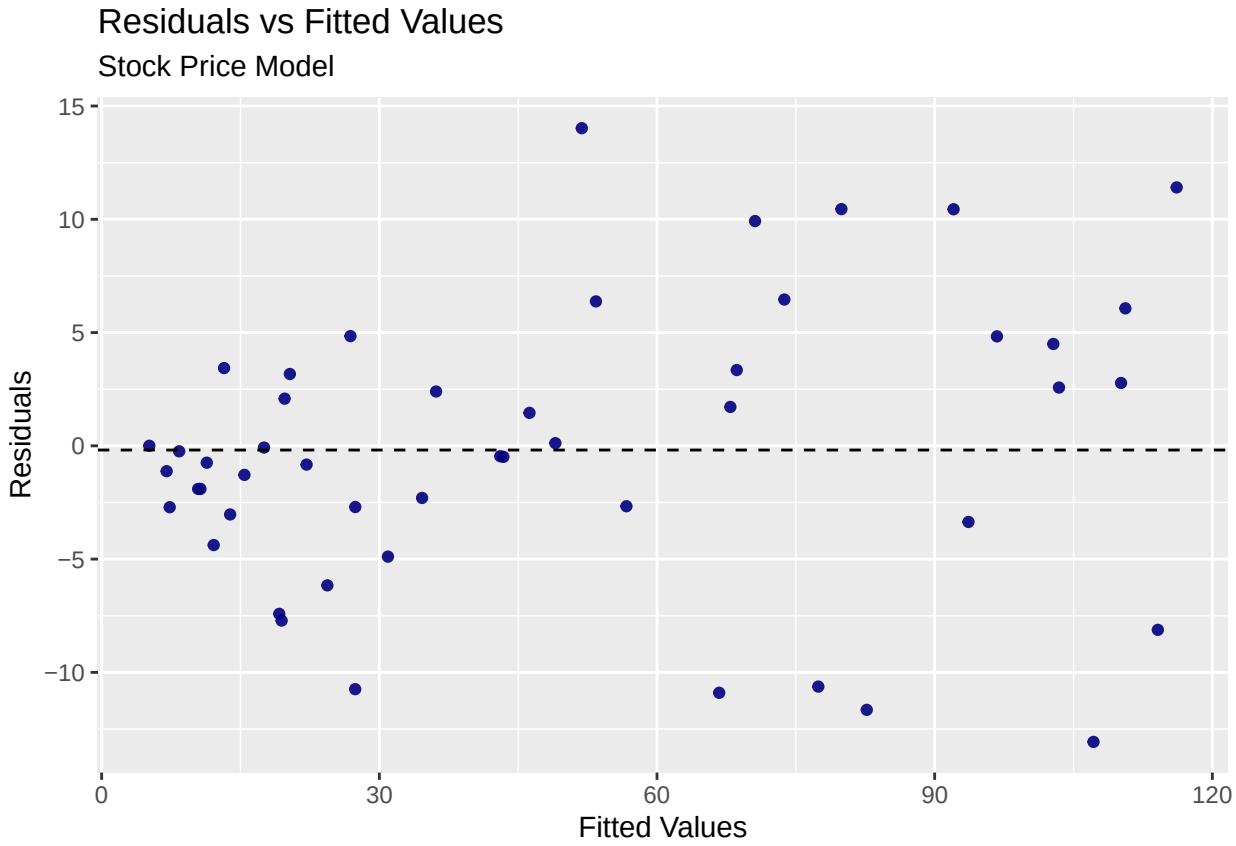
(c) Plot the respective residuals vs. fitted values and discuss your observations.

```
## Revenue
ggplot(aapl.train, aes(x = Revenue_Fitted, y = Revenue_Residuals)) +
  geom_point(color='dark green', alpha=0.9) +
  geom_hline(yintercept=mean(aapl.train$Revenue_Residuals), color='black', linetype='dashed') +
  ggtitle("Residuals vs Fitted Values", "Revenue Model") +
  xlab("Fitted Values") +
  ylab("Residuals")
```



The plot of fitted values vs. residuals shows that the residuals seem somewhat random except for a bit of clustering towards the right side of the graph. There also seems to be mostly clustering around the mean.

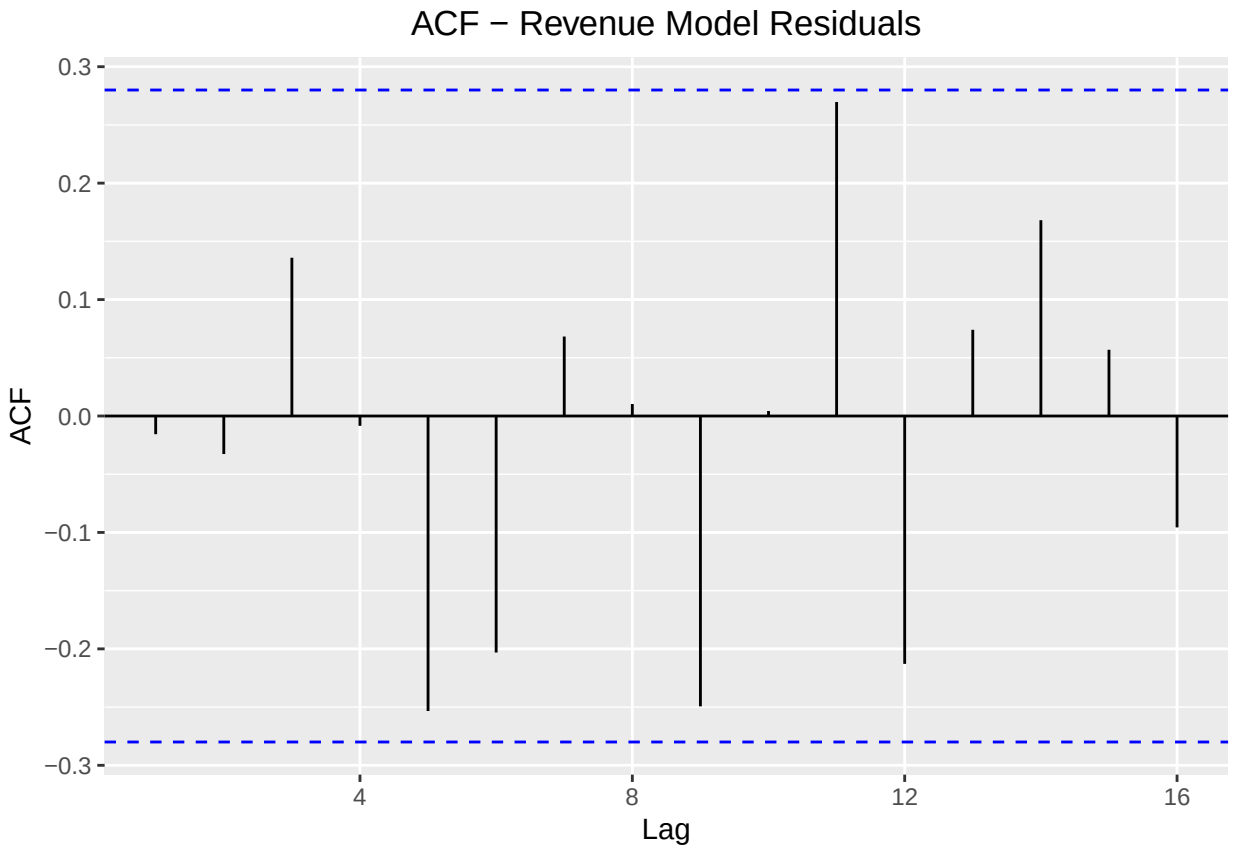
```
## Stock Price

ggplot(aapl.train, aes(x = Close_Fitted, y = Close_Residuals)) +
  geom_point(color='Navy', alpha=0.9) +
  geom_hline(yintercept=mean(aapl.train$Close_Residuals), color='black', linetype='dashed') +
  ggtitle("Residuals vs Fitted Values", "Stock Price Model") +
  xlab("Fitted Values") +
  ylab("Residuals")
```



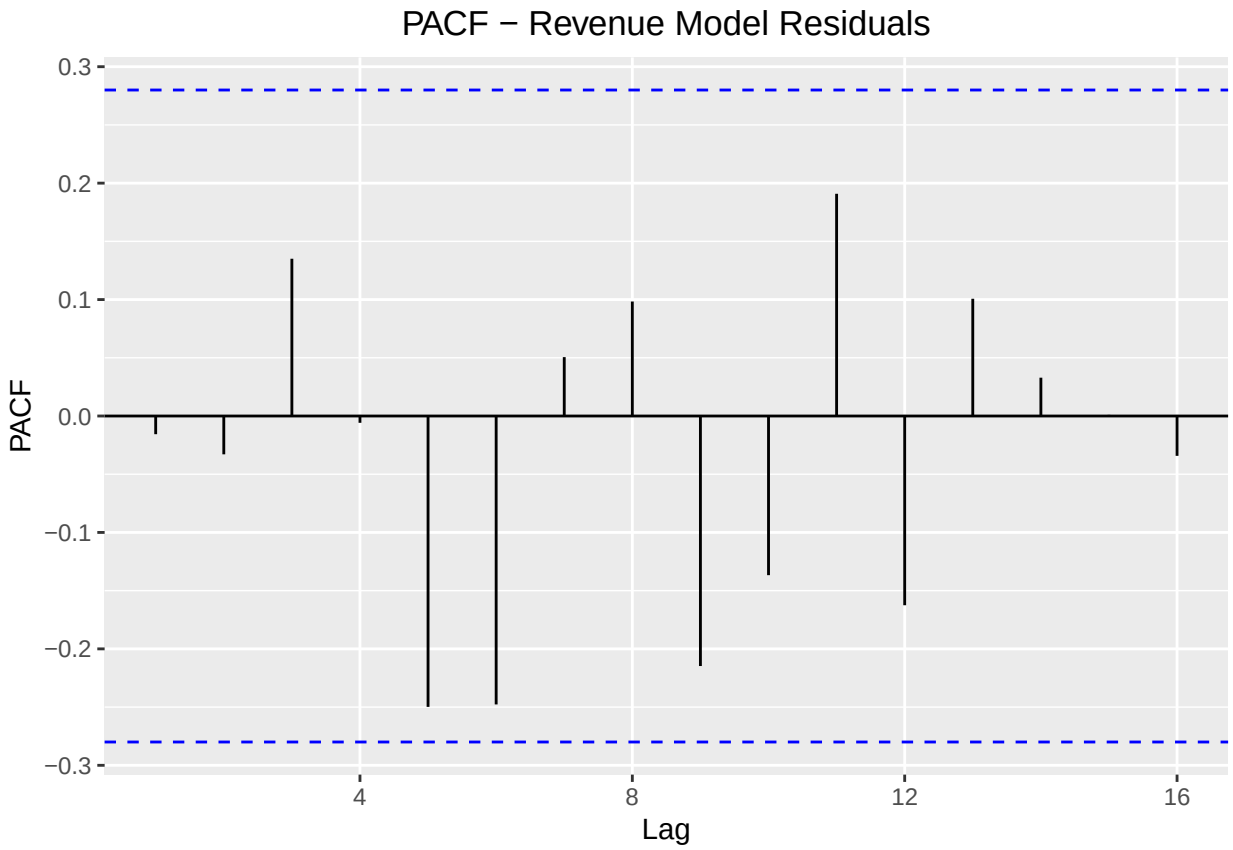
This plot of fitted values vs. residuals shows that the residuals seem quite random except for the left hand side of the graph where residuals seem to cluster around the mean. We see large dispersion as the fitted values get larger, this could indicate that the larger the amount we fit, the larger the error.

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

```
## Revenue ACF & PACF

par(mfrow=c(2, 1))
ggAcf(aapl.train$Revenue_Residuals, main = "ACF - Revenue Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))
```

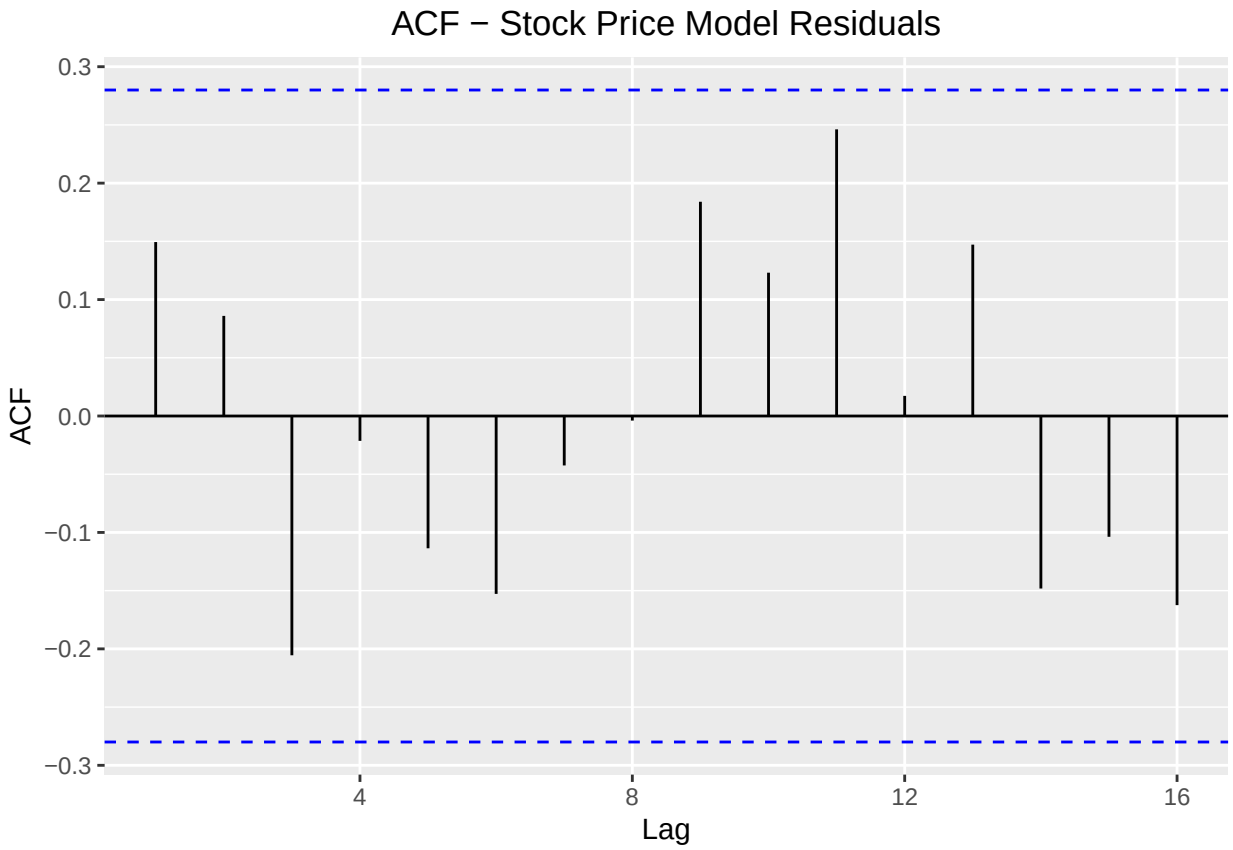



```
ggPacf(aapl.train$Revenue_Residuals, main = "PACF - Revenue Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))
```

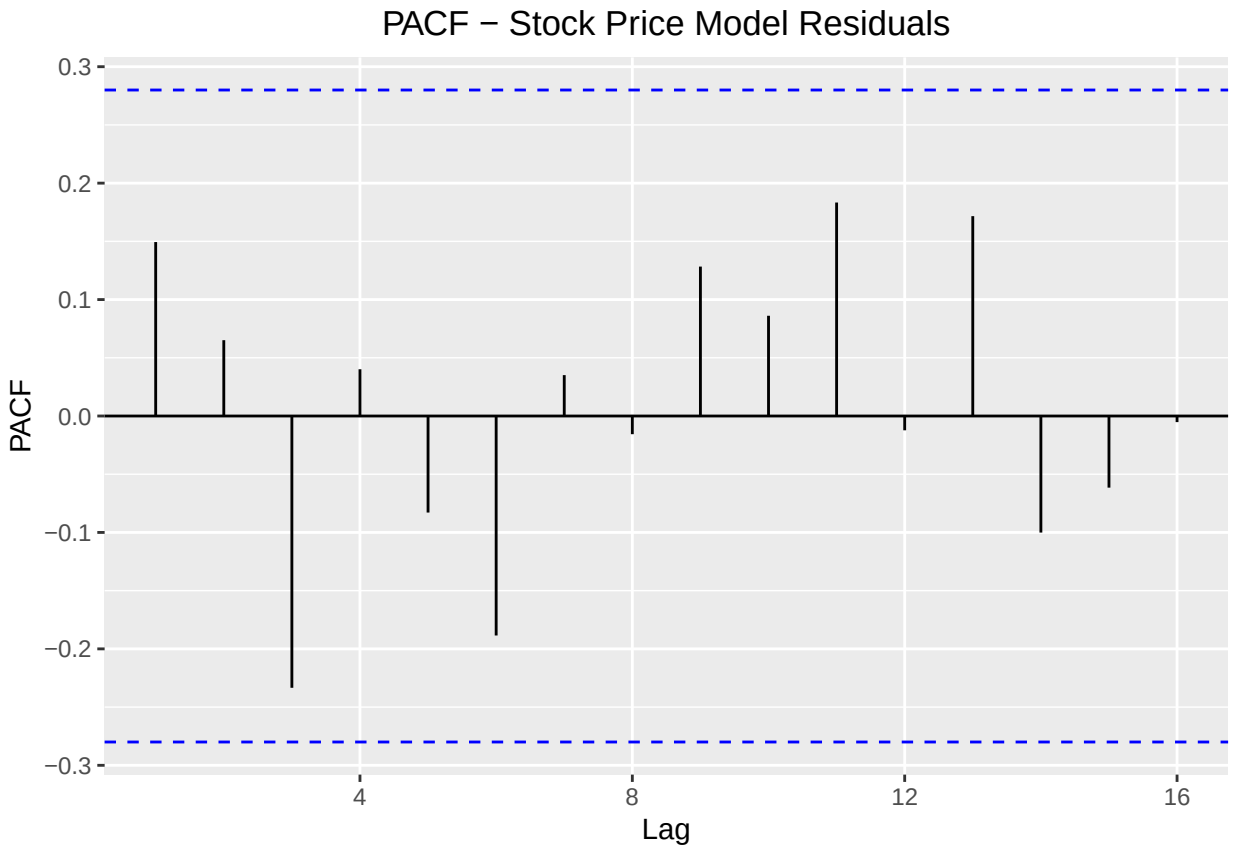


Looking at the PACF and the ACF of Revenue, it looks like all the dynamics have been wiped out, indicating that we would have achieved white noise. This indicates that the model is a pretty good fit.

```
# Stock Price ACF & PACF

par(mfrow=c(2, 1))
ggAcf(aapl.train$Close_Residuals, main = "ACF - Stock Price Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))
```



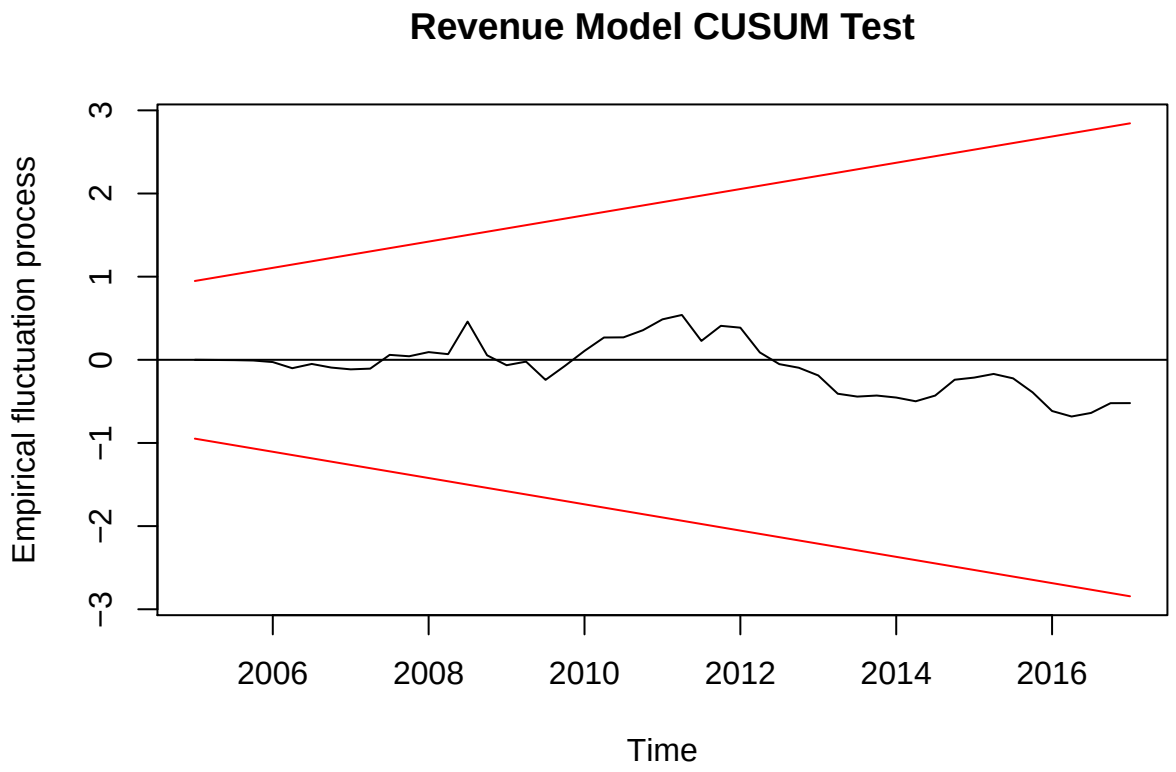

```
ggPacf(aapl.train$Close_Residuals, main = "PACF - Stock Price Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))
```



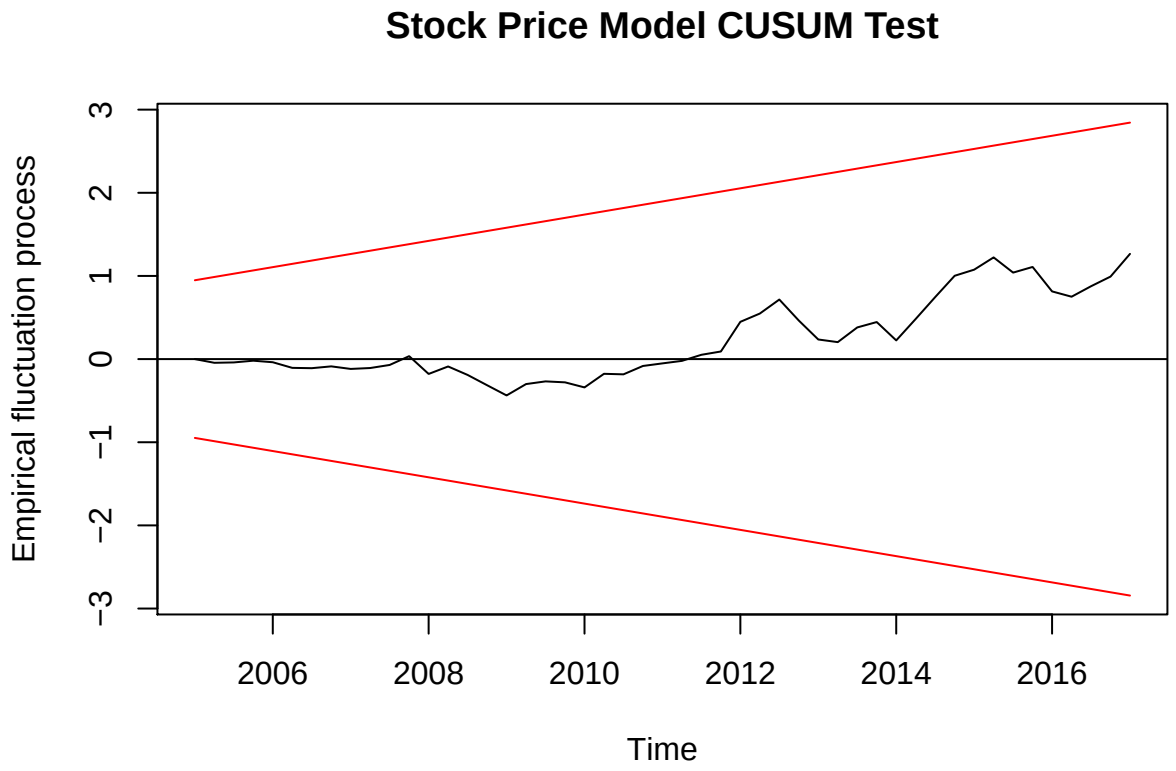
Looking at the PACF and the ACF of Stock price, it looks like almost all the dynamics have been wiped out and looks like white noise. This would lead us to believe that the model is a decent fit.

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

```
## Revenue CUSUM TEST

plot(efp(aapl.train$Revenue_Residuals~1, type = "Rec-CUSUM"),
     main = "Revenue Model CUSUM Test")
```




```
## Stock Price

plot(efp(aapl.train$Close_Residuals~1, type = "Rec-CUSUM"),
     main = "Stock Price Model CUSUM Test")
```

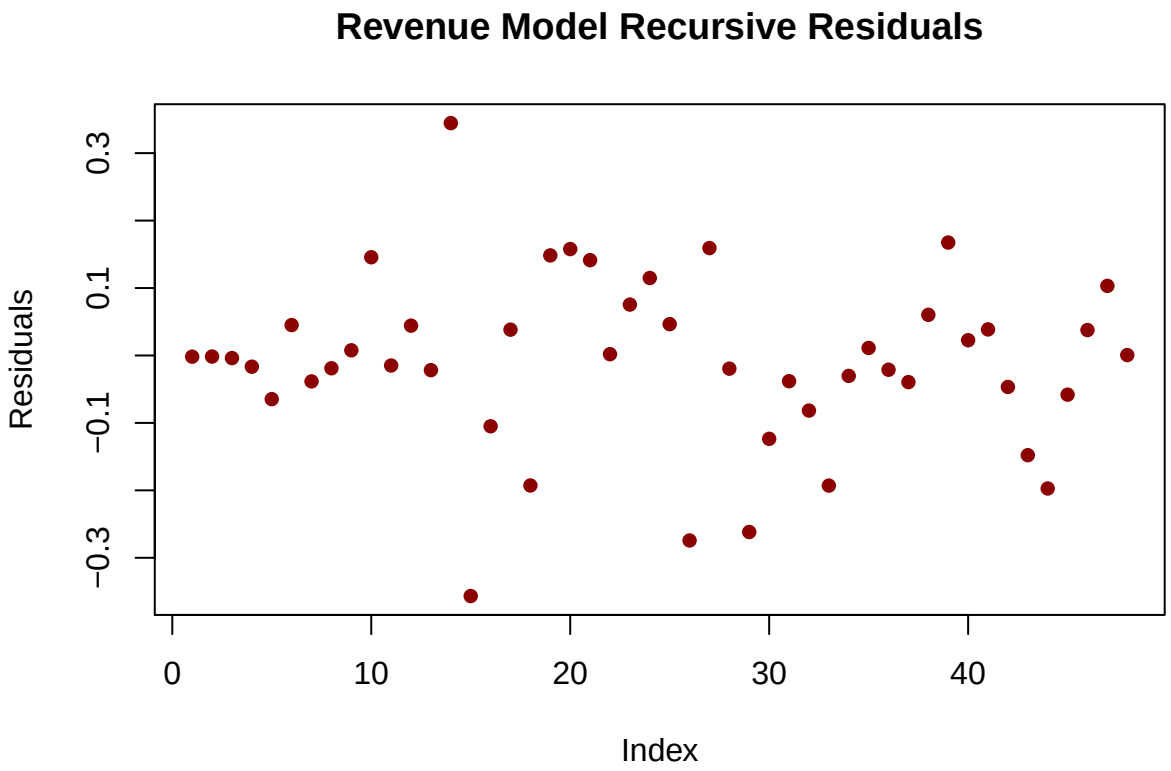


To assess for quality control of the model we plot the CUSUM plot for each model. For both models it can

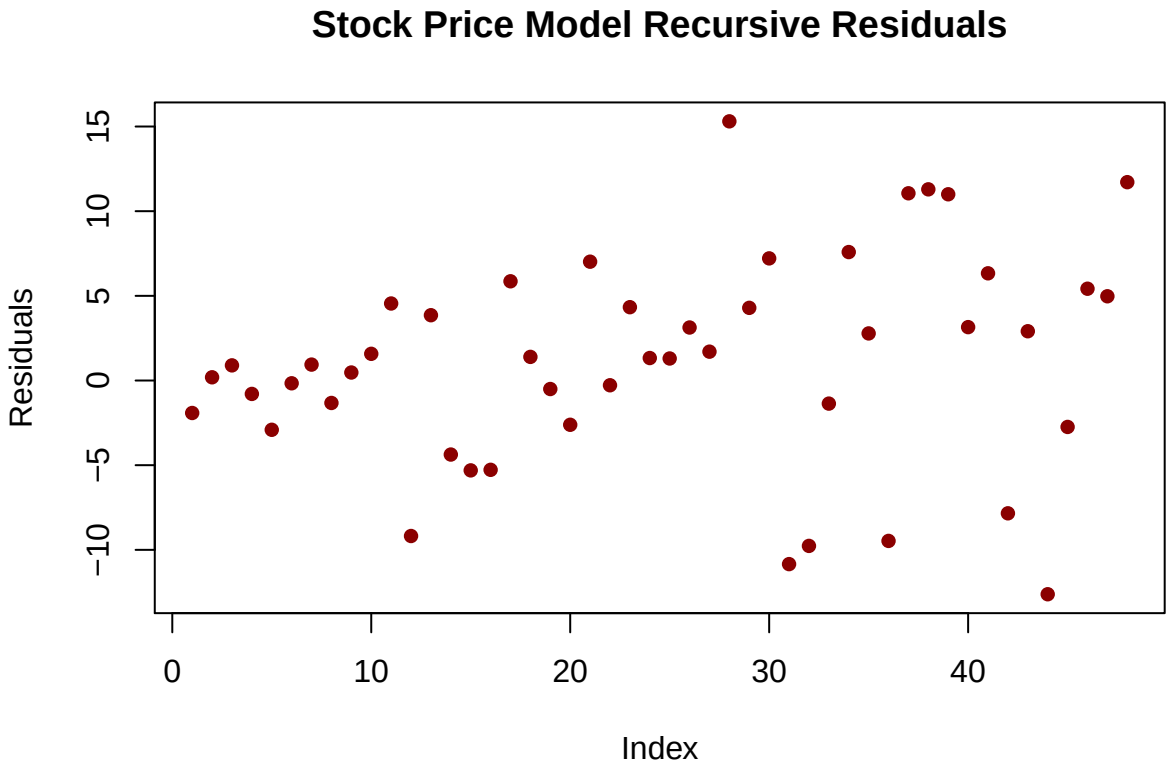
be seen that the cumulative sums do not drift outside the boundaries and stay pretty close to the mean. For revenue it can be seen that around 2012 there is some negative drift but it stays pretty close to the mean. For stock price, the cumulative sums stay around the mean, but starts to increase around 2012 and seems to be having some positive drift, however does not go outside the bands. We can conclude that the model does not break at any specific time.

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

```
## Revenue

plot(recresid(aapl.train$Revenue_Residuals~1), pch=16,
     main = "Revenue Model Recursive Residuals",
     ylab="Residuals", col = "dark red")
```




```
## Stock Price

plot(recresid(aapl.train$Close_Residuals~1), pch=16,
     main = "Stock Price Model Recursive Residuals",
     ylab="Residuals", col = "dark red")
```



Aside from a few outliters the residuals seem to be pretty randomly distributed for both stock price and revenue. There seems to be a bit of unexplained seasonality in the revenue model. But overall there seems to be no structural break in the pattern of the residuals.

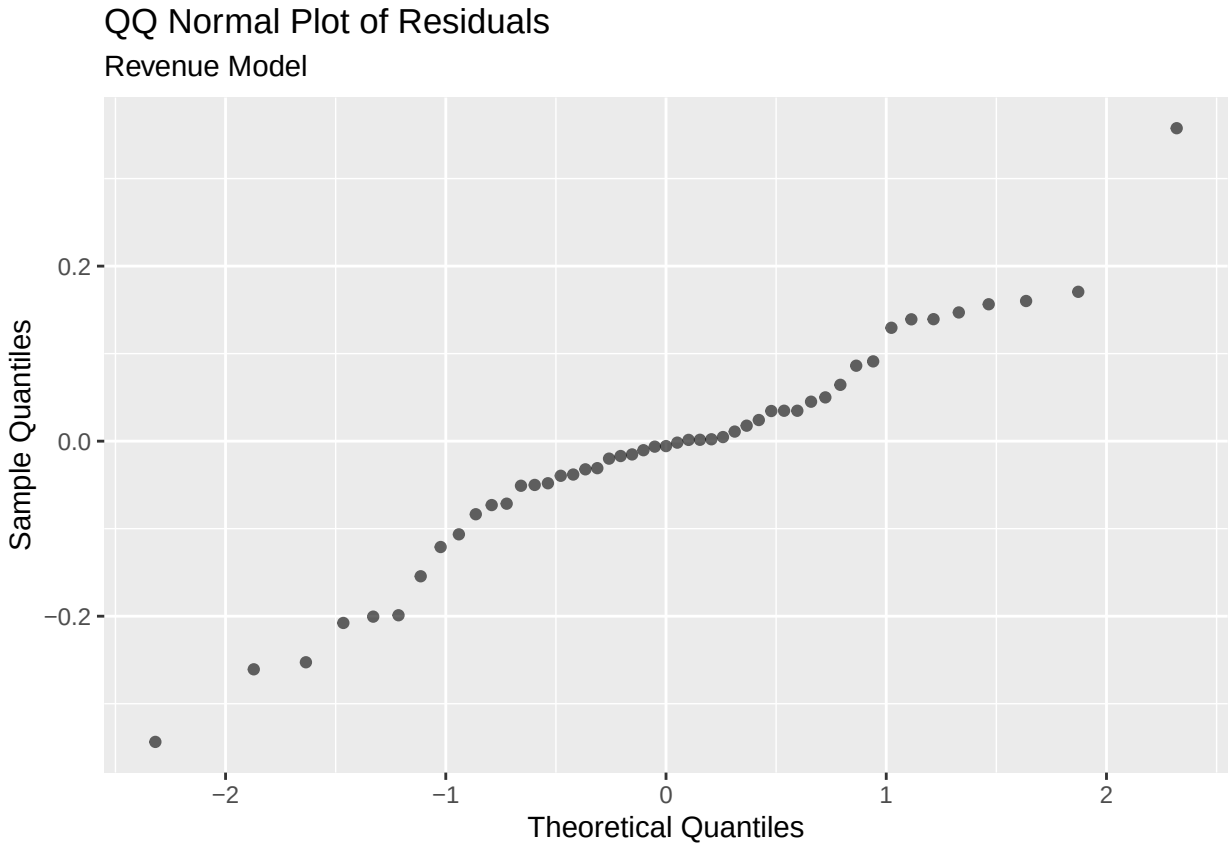
(h) For your model, discuss the associated diagnostic statistics.

```
# Revenue
coeftest(Rev_model)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ma1  -0.28873    0.14048 -2.0554  0.03984 *
## sma1 -0.30494    0.17252 -1.7675  0.07714 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the coefficient test, we note that only one of the estimated parameters is statistically significant. The MA(1) term. The revenue time series seems to be a short-term dynamics process.

```
# QQ plot of Residuals for Revenue
ggplot(aapl.train) +
  geom_qq(aes(sample=Revenue_Residuals), color='black', alpha=0.6) +
  ggtitle('QQ Normal Plot of Residuals', 'Revenue Model') +
  ylab('Sample Quantiles') +
  xlab('Theoretical Quantiles')
```




```
print(paste("Skewness:", skewness(aapl.train$Revenue_Residuals)))
```

```
## [1] "Skewness: -0.120258044271048"
```

```
print(paste("Kurtosis:", kurtosis(aapl.train$Revenue_Residuals)))
```

```
## [1] "Kurtosis: 4.04272829822086"
```

Looking at the QQ plot of the residuals, they are not completely normally distributed, there is a slight left skew of -0.12 and a Kurtosis of 4.04. This confirms what was seen in the residual vs. fitted plots, that the ARIMA model could be improved to account for extreme values.

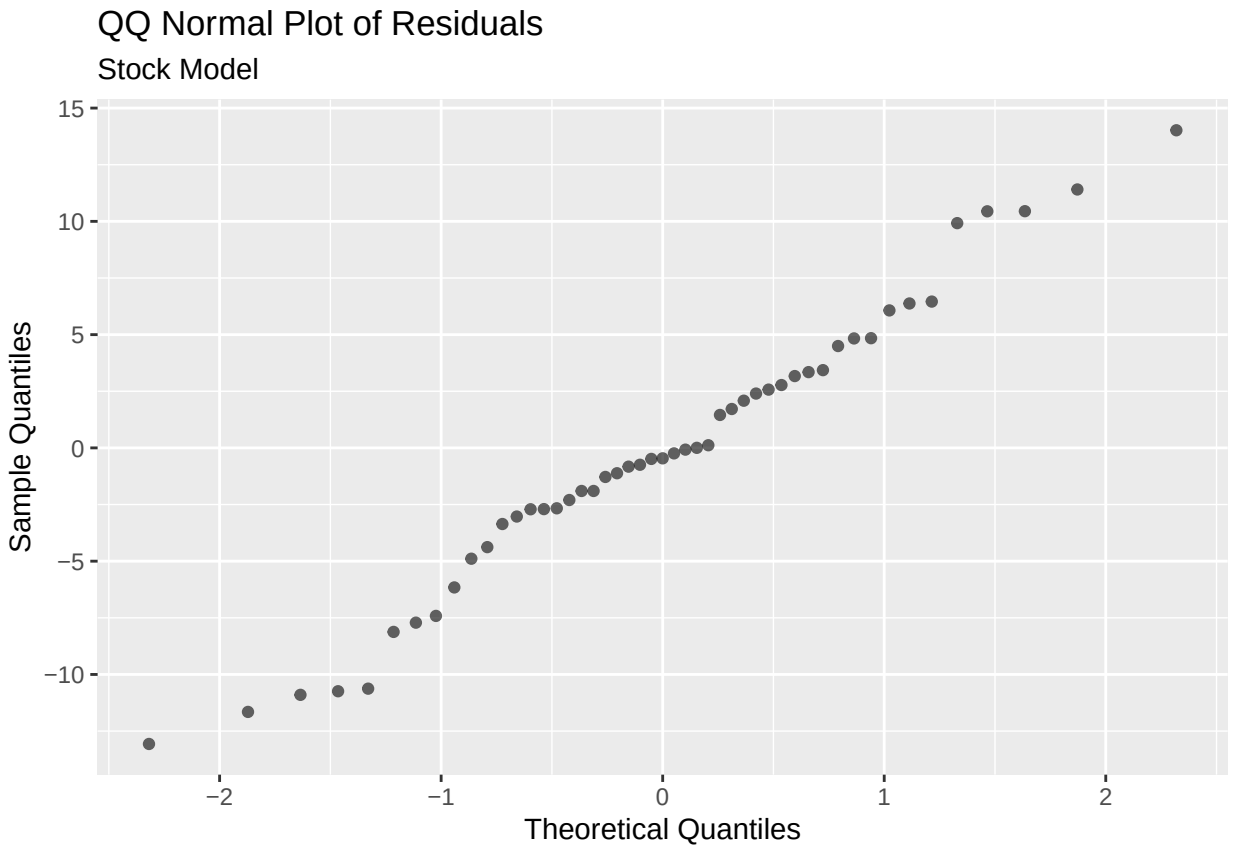
```
# Revenue
coeftest(Stock_model)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## sma1  -0.48449    0.13988 -3.4636  0.000533 ***
## drift  2.50117    0.50535  4.9494 7.445e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the coefficient test for the stock model, we see that the two estimated parameters are statistically significant. Those being the S-MA(1) and drift terms. The stock price time series is also a short-term dynamics process.

```
# QQ plot of Residuals for Stock Price
ggplot(aapl.train) +
  geom_qq(aes(sample=Close_Residuals), color='black', alpha=0.6) +
```

```
  ggtitle('QQ Normal Plot of Residuals', 'Stock Model') +
  ylab('Sample Quantiles') +
  xlab('Theoretical Quantiles')
```




```
print(paste("Skewness:", skewness(aapl.train$Close_Residuals)))
```

```
## [1] "Skewness: 0.021061237754844"
```

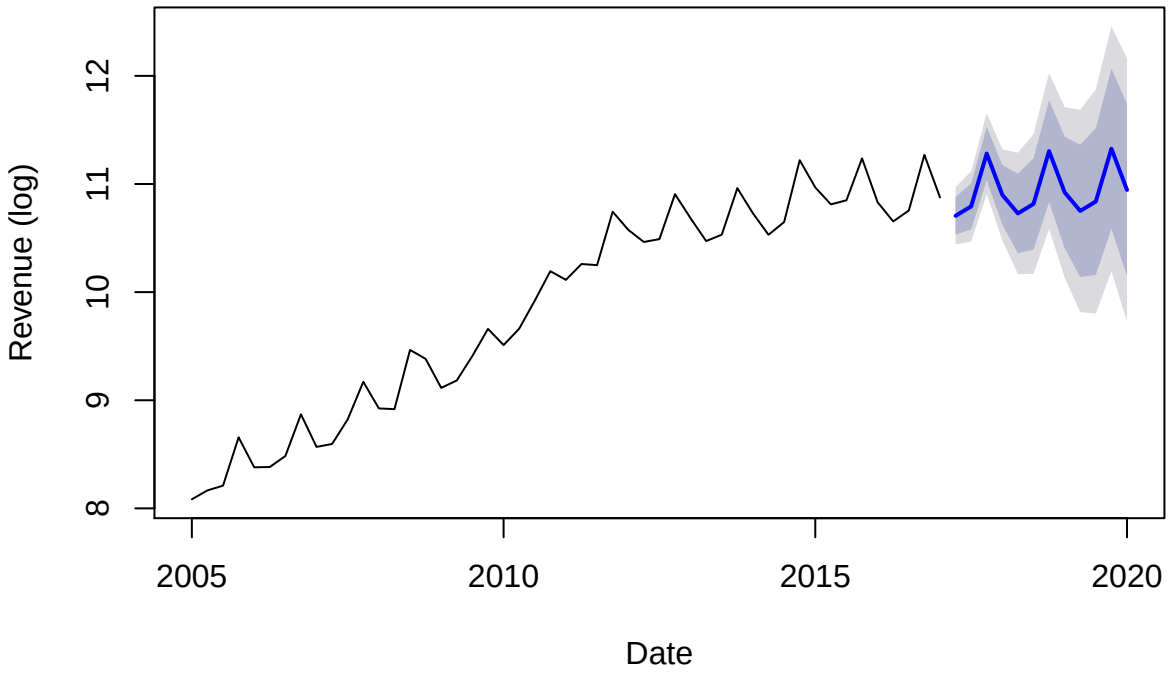
```
print(paste("Kurtosis:", kurtosis(aapl.train$Close_Residuals)))
```

```
## [1] "Kurtosis: 2.77764386031259"
```

Looking at the QQ plot of the residuals for stock prices, they are also not completly normally distributed, there is a small right skew of 0.02 which is less than the Revenue plot and a Kurtosis of 2.77. This also confirms what was seen in the residual vs. fitted plots. The ARIMA model could be improved to account for extreme values.

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

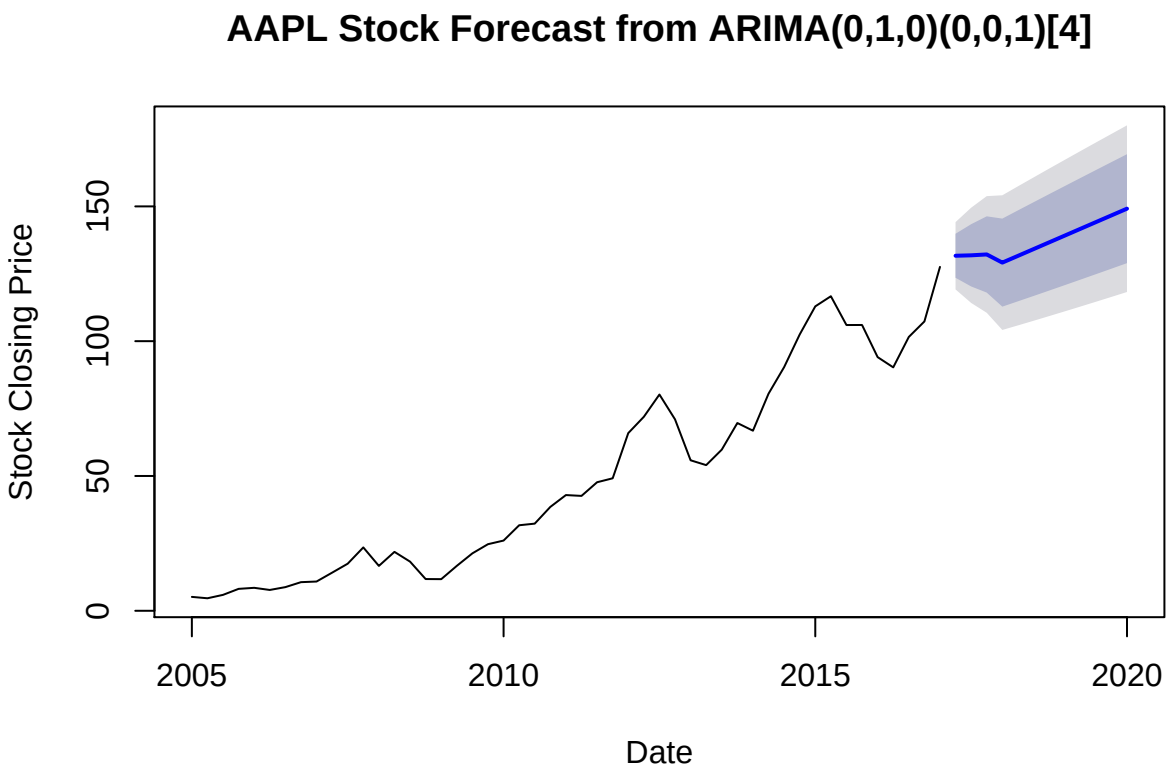
```
#Revenue Forecast
rev_forecasts=forecast(Rev_model, h = 12)
plot(rev_forecasts, xlab = "Date", ylab = "Revenue (log)",
     main = "AAPL Revenue Forecast from ARIMA(0,1,1)(0,1,1)[4]")
```




```
forecast::accuracy(rev_forecasts,  log(aapl.test$Revenue)) ### test set MAPE 0.9777
```

```
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set -0.01095241 0.1250996 0.08863273 -0.1032801 0.8883947 0.3977920
## Test set      0.10807869 0.1285601 0.10807869  0.9775316 0.9775316 0.4850673
##                    ACF1
## Training set -0.01561631
## Test set              NA
```

```
#Stock Forecast
stock_forecasts=forecast(Stock_model, h =  12)
plot(stock_forecasts,  xlab = "Date", ylab = "Stock Closing Price",
     main = "AAPL Stock Forecast from ARIMA(0,1,0)(0,0,1)[4]")
```



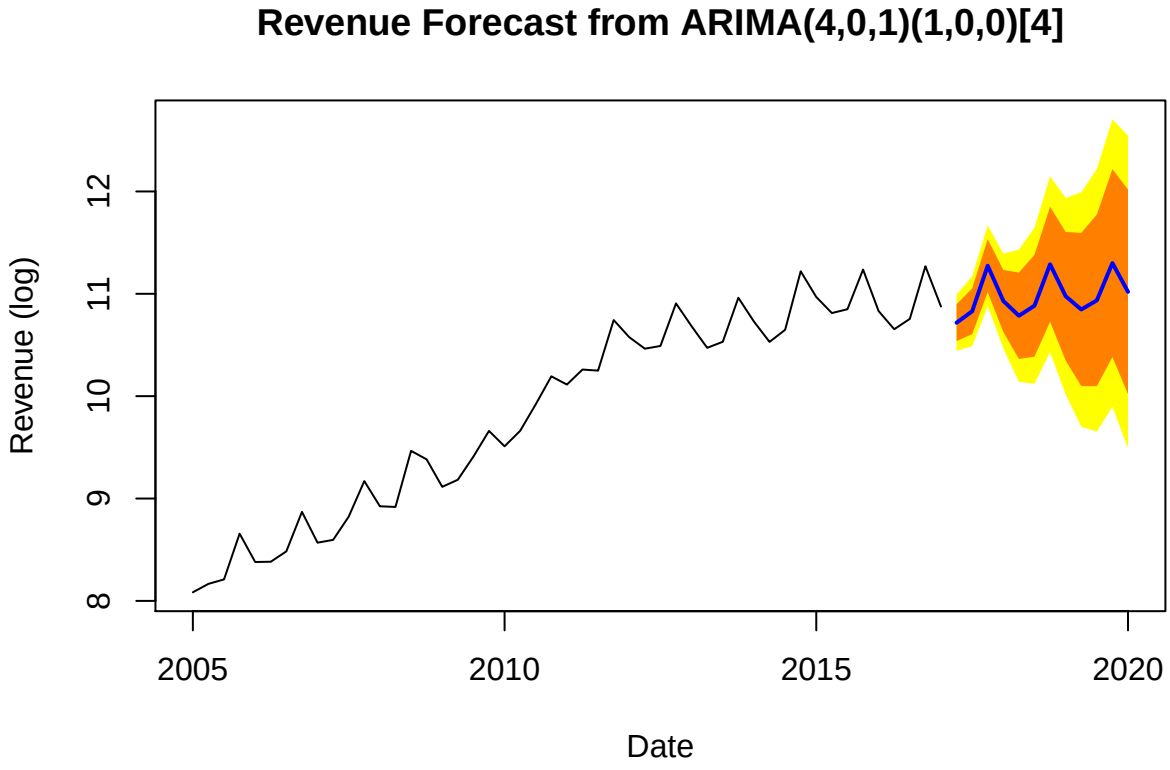

```
forecast::accuracy(stock_forecasts,  aapl.test$Close) ### test set MAPE 26.02
```

```
##                       ME      RMSE      MAE       MPE     MAPE      MASE
## Training set -0.1856567  6.172235  4.77280 -8.159749 15.40947 0.8471215
## Test set     54.6244601 65.482854 54.62446 26.020846 26.02085 9.6952646
##                   ACF1
## Training set 0.1494202
## Test set             NA
```

(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?

ARIMA

```
#ARIMA for Revenue
# Choose AR5 based off PACF, but AR(4) had a lower AIC and BIC

ARIMA.rev = Arima(lgdata,order=c(4,0,1),seasonal=list(order=c(1,0,0)))
plot(forecast(ARIMA.rev,h=12),shadecols="oldstyle", xlab = "Date",
     ylab = "Revenue (log)", main = "Revenue Forecast from ARIMA(4,0,1)(1,0,0)[4]")
```




```
summary(ARIMA.rev)
```

```
## Series: lgdata
## ARIMA(4,0,1)(1,0,0)[4] with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ma1    sar1      mean
##       0.2340  0.5541  0.2987  -0.0922  0.5042  0.9014   11.9156
## s.e.  0.0596  0.0935  0.0287   0.1387  0.0630  0.0003  103.5326
##
## sigma^2 estimated as 0.01964:  log likelihood=26.97
## AIC=-37.94   AICc=-34.34   BIC=-22.8
##
## Training set error measures:
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set 0.003604017 0.1297352 0.09613827 0.04063943 0.9752364 0.3819067
##                      ACF1
## Training set 0.004785123
```

```
#MAPE
forecast::accuracy(forecast(ARIMA.rev,h=12), log(aapl.test$Revenue)) ### test set MAPE 0.695
```

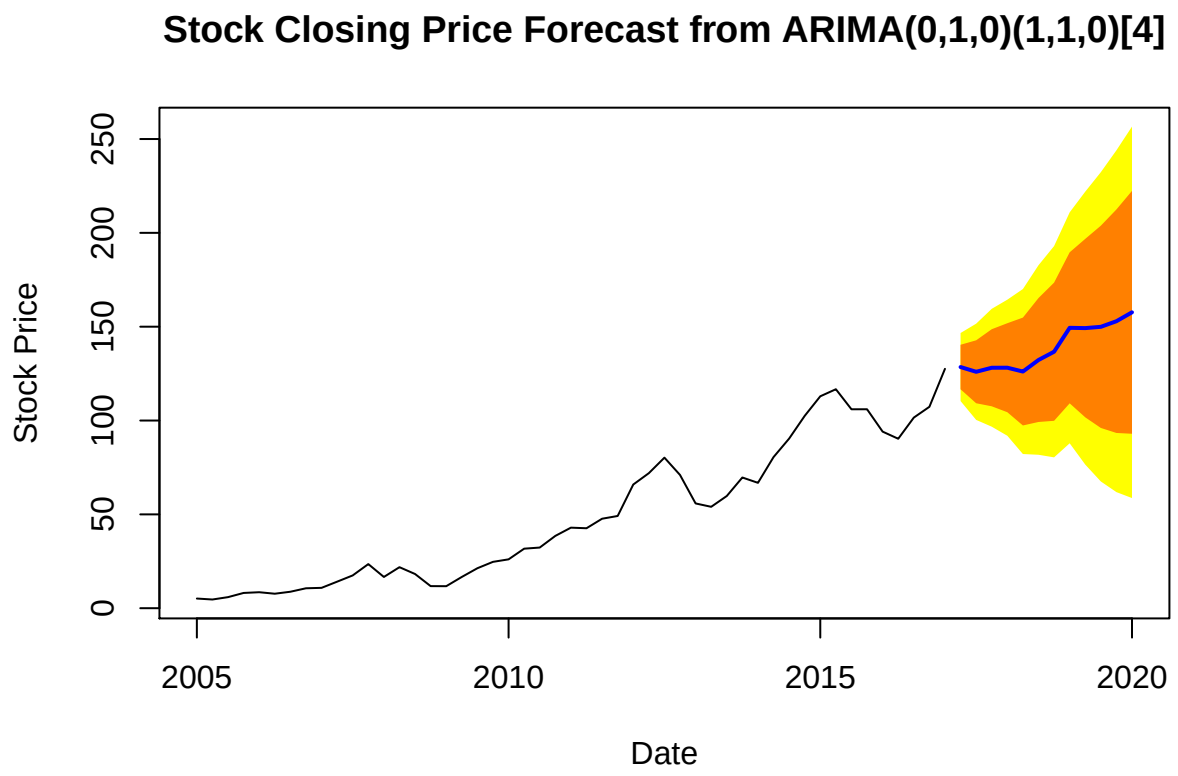
```
##                       ME       RMSE        MAE        MPE      MAPE      MASE
## Training set 0.003604017 0.12973523 0.09613827 0.04063943 0.9752364 0.4314776
## Test set     0.068358862 0.09157076 0.07730276 0.61388882 0.6953983 0.3469421
##                      ACF1
## Training set 0.004785123
## Test set              NA
```

```
#ARIMA for Stock Price
```

```r
# Choose S-AR1 based off PACF, including  drift lowered the AIC and BIC

ARIMA.stock = Arima(CloseTS,order=c(0,1,0),include.drift=TRUE,seasonal=list(order=c(1,1,0)))
```

```
## Warning in Arima(CloseTS, order = c(0, 1, 0), include.drift = TRUE, seasonal
## = list(order = c(1, : No drift term fitted as the order of difference is 2 or
## more.
```

```r
plot(forecast(ARIMA.stock,h=12) , shadecols="oldstyle", xlab = "Date",
     ylab = "Stock Price", main = "Stock Closing Price Forecast from ARIMA(0,1,0)(1,1,0)[4]")
```




```r
summary(ARIMA.stock)
```

```
## Series: CloseTS
## ARIMA(0,1,0)(1,1,0)[4]
##
## Coefficients:
##          sar1
##       -0.6288
## s.e.   0.1249
##
## sigma^2 estimated as 85.55:  log likelihood=-160.81
## AIC=325.63   AICc=325.92   BIC=329.2
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.3341459 8.664725 6.061268 -0.2239617 13.08718 0.4225673
##                    ACF1
## Training set 0.2422916
```

```r
coeftest(ARIMA.stock)
```

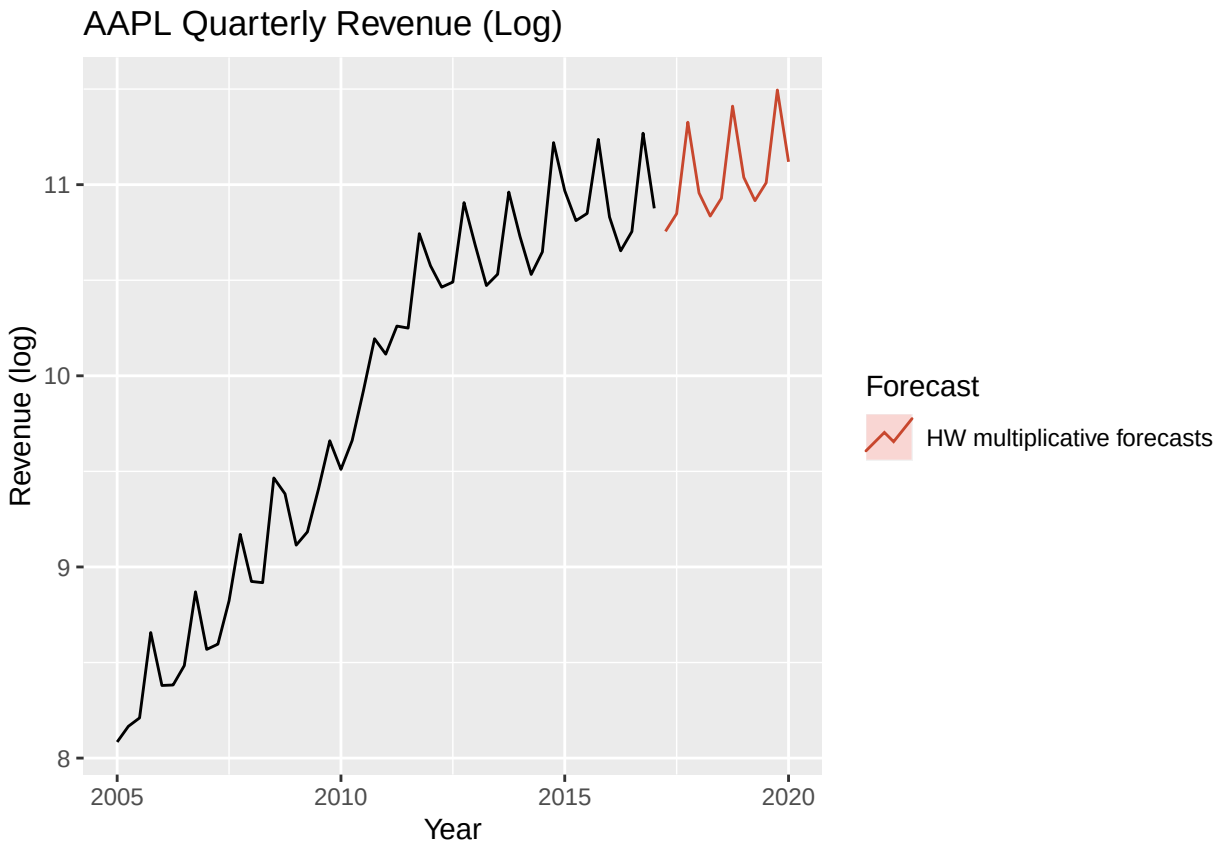
```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## sar1 -0.62880    0.12494 -5.0326 4.838e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#MAPE
forecast::accuracy(forecast(ARIMA.stock,h = 12), aapl.test$Close) ### test set MAPE 25.62
```

```
##                      ME      RMSE       MAE        MPE     MAPE     MASE
## Training set  0.3341459  8.664725  6.061268 -0.2239617 13.08718 1.075811
## Test set     53.2130737 62.766941 53.213074 25.6265022 25.62650 9.444758
##                   ACF1
## Training set 0.2422916
## Test set            NA
```

Holt-Winters

```r
#Holt-Winters for Revenue

HW.rev <- hw(lgdata,seasonal="multiplicative", h =12)

autoplot(lgdata) +
  autolayer(HW.rev, series="HW multiplicative forecasts",
    PI=FALSE) +
  xlab("Year") +
  ylab("Revenue (log)") +
  ggtitle("AAPL Quarterly Revenue (Log)") +
  guides(colour=guide_legend(title="Forecast"))
```



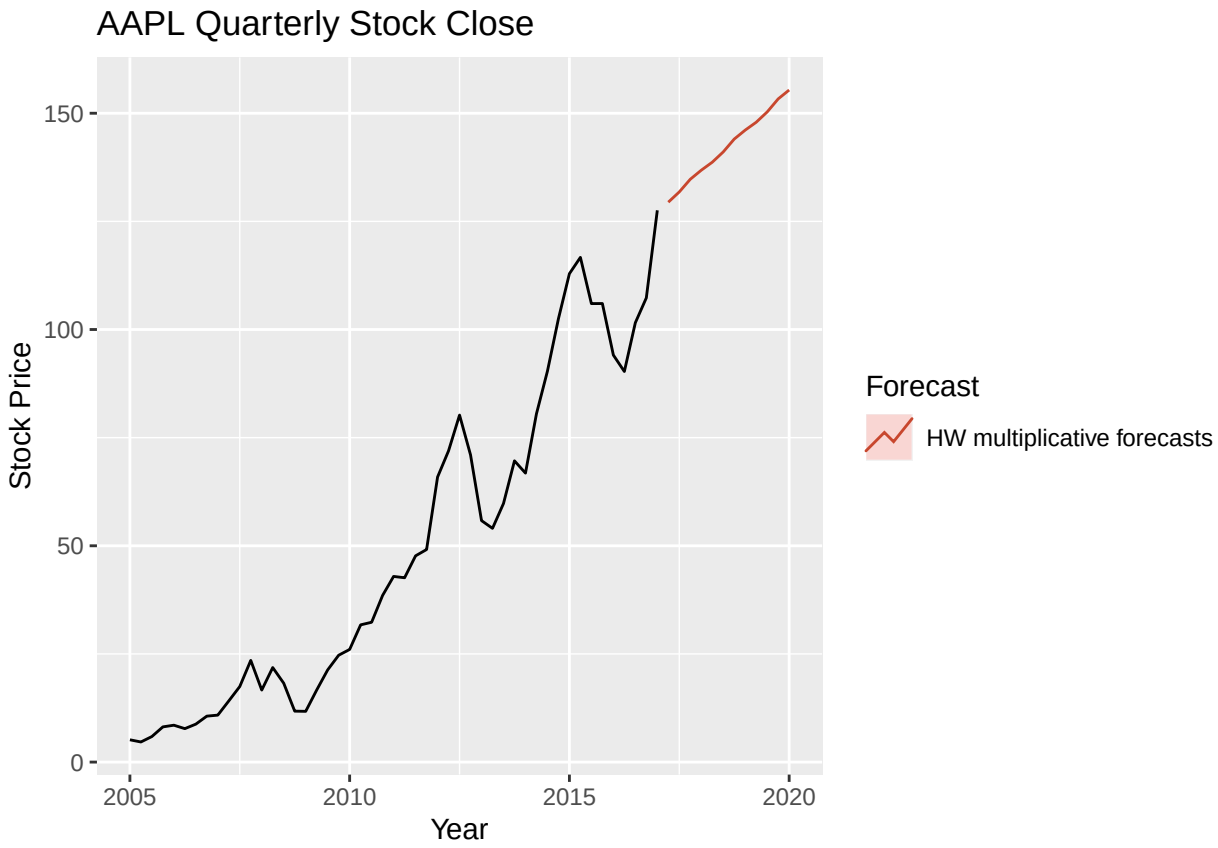

```
summary(HW.rev)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = lgdata, h = 12, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.5586
##     beta  = 0.073
##     gamma = 0.4414
##
##   Initial states:
##     l = 8.112
##     b = 0.0634
##     s = 1.028 0.9915 0.9906 0.9899
##
##   sigma:  0.0132
##
##        AIC       AICc        BIC
## -0.3339071  4.2814775 16.6924756
##
## Error measures:
```

```
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set -0.01223009 0.1193884 0.08814521 -0.1102149 0.8832801 0.3501545
##                     ACF1
## Training set 0.01074813
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2017 Q2       10.75560 10.57396 10.93723 10.47781 11.03338
## 2017 Q3       10.84821 10.63163 11.06478 10.51698 11.17943
## 2017 Q4       11.32663 11.06362 11.58963 10.92440 11.72886
## 2018 Q1       10.95669 10.66478 11.24860 10.51026 11.40313
## 2018 Q2       10.83581 10.46336 11.20826 10.26620 11.40542
## 2018 Q3       10.92896 10.51706 11.34086 10.29901 11.55891
## 2018 Q4       11.41079 10.94062 11.88096 10.69172 12.12986
## 2019 Q1       11.03796 10.54234 11.53358 10.27998 11.79594
## 2019 Q2       10.91626 10.34723 11.48530 10.04600 11.78653
## 2019 Q3       11.00996 10.39443 11.62550 10.06859 11.95134
## 2019 Q4       11.49521 10.80715 12.18327 10.44291 12.54751
## 2020 Q1       11.11947 10.40822 11.83073 10.03171 12.20724
```

```
# MAPE = 0.883
```

```
#Holt-Winters for Stock Price

HW.close <- hw(CloseTS,seasonal="multiplicative", h = 12)


autoplot(CloseTS) +
  autolayer(HW.close, series="HW multiplicative forecasts",
    PI=FALSE) +
  xlab("Year") +
  ylab("Stock Price") +
  ggtitle("AAPL Quarterly Stock Close") +
  guides(colour=guide_legend(title="Forecast"))
```




```
summary(HW.close)
```

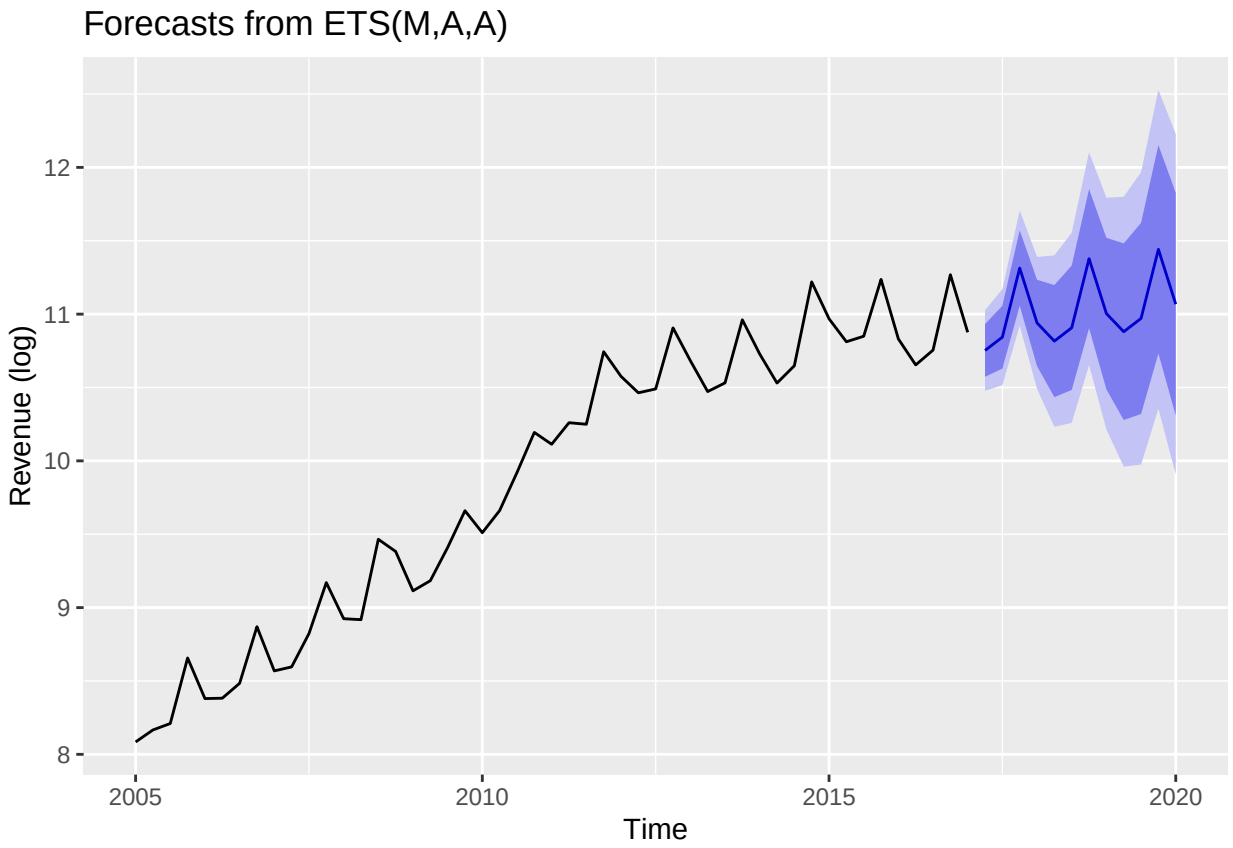
```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = CloseTS, h = 12, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.0249
##     gamma = 1e-04
##
##   Initial states:
##     l = 3.2163
##     b = 0.7013
##     s = 1.0028 0.9982 0.9977 1.0012
##
##   sigma:  0.183
##
##      AIC     AICc      BIC
## 374.3692 378.9846 391.3956
##
## Error measures:
```

```
##                     ME     RMSE      MAE      MPE    MAPE      MASE    ACF1
## Training set 1.320754 6.994916 5.214971 1.622708 13.5014 0.3635669 0.25033
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80      Lo 95    Hi 95
## 2017 Q2       129.4341 99.08227 159.7859  83.014999 175.8532
## 2017 Q3       131.8064 87.57769 176.0351  64.164423 199.4484
## 2017 Q4       134.7345 78.69847 190.7705  49.034786 220.4342
## 2018 Q1       136.8263 70.32190 203.3308  35.116576 238.5361
## 2018 Q2       138.6744 62.39450 214.9543  22.014357 255.3345
## 2018 Q3       141.0511 55.01235 227.0899   9.466190 272.6360
## 2018 Q4       144.0218 47.95387 240.0896  -2.901398 290.9449
## 2019 Q1       146.0980 40.61184 251.5842 -15.229172 307.4252
## 2019 Q2       147.9148 33.20629 262.6233 -27.516732 323.3464
## 2019 Q3       150.2959 25.87107 274.7207 -39.995448 340.5872
## 2019 Q4       153.3091 18.48485 288.1333 -52.886796 359.5050
## 2020 Q1       155.3698 10.80860 299.9310 -65.717466 376.4570
```

```
#MAPE = 13.50
```

ETS

```
#ETS for Revenue
ets.rev <- ets(lgdata)

ets.rev.forecast <- forecast(ets.rev, h = 12)

ets.rev %>% forecast(h=12) %>%
  autoplot() +
  ylab("Revenue (log)")
```



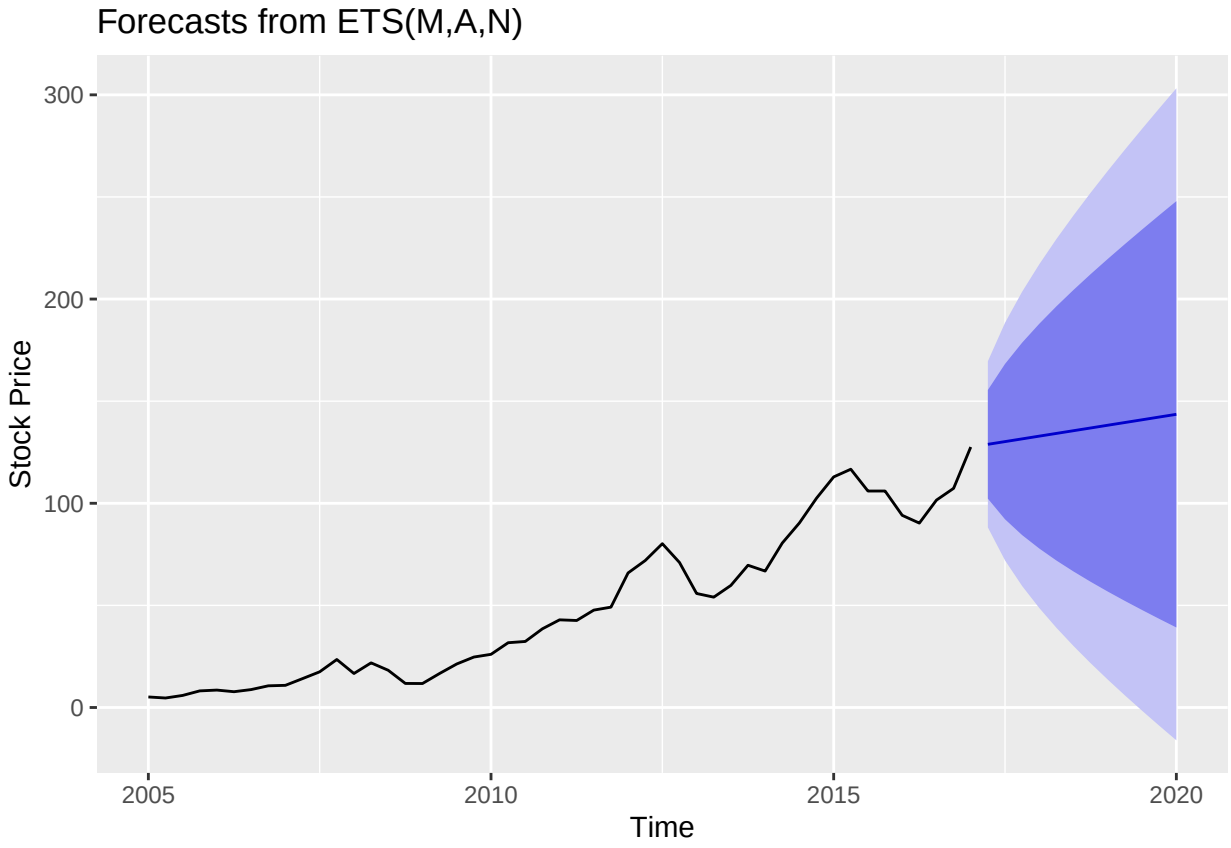

```
summary(ets.rev)
```

```
## ETS(M,A,A)
##
## Call:
##  ets(y = lgdata)
##
##   Smoothing parameters:
##     alpha = 0.5482
##     beta  = 0.089
##     gamma = 0.4518
##
##   Initial states:
##     l = 8.1155
##     b = 0.0654
##     s = 0.2372 -0.0816 -0.0801 -0.0755
##
##   sigma:  0.013
##
##       AIC      AICc       BIC
## -1.507314  3.108071 15.519069
##
## Training set error measures:
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set -0.01131898 0.1179501 0.08633936 -0.1042718 0.8655926 0.3429808
##                     ACF1
## Training set 0.01781037
```

```
forecast::accuracy(ets.rev.forecast, log(aapl.rev.ts)) ### test set MAPE is 2.19
```

```
##                       ME      RMSE        MAE         MPE      MAPE      MASE
## Training set -0.01131898 0.1179501 0.08633936 -0.1042718 0.8655926 0.3429808
## Test set      0.01614665 0.2852470 0.24433887  0.1139398 2.1970131 0.9706297
##                    ACF1 Theil's U
## Training set  0.01781037        NA
## Test set     -0.26241860  1.055369
```

```
#ETS for Stock Price

ets.stock <- ets(CloseTS)

ets.stock.forecast <- forecast(ets.stock, h = 12)

ets.stock %>% forecast(h=12) %>%
  autoplot() +
  ylab("Stock Price")
```




```
summary(ets.stock)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = CloseTS)
##
##   Smoothing parameters:
##     alpha = 0.9987
```

```
##     beta  = 1e-04
##
##   Initial states:
##     l = 3.704
##     b = 1.33
##
##   sigma:  0.1611
##
##      AIC     AICc      BIC
## 360.8782 362.2736 370.3373
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 1.197506 6.963354 5.162642 -0.835211 13.12306 0.3599187 0.2605109
```

```
forecast::accuracy(ets.stock.forecast,aapl.close.ts) ### test set MAPE is 26.50
```

```
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set  1.197506  6.963354  5.162642 -0.835211 13.12306 0.3599187
## Test set     55.744310 67.208163 55.744310 26.503605 26.50361 3.8862694
##                   ACF1 Theil's U
## Training set 0.2605109        NA
## Test set     0.5804522  2.712087
```

```
#Table of MAPE for each model and its respective series
MAPE = data.frame(Revenue=c(0.695, 0.883 , 2.19 , 0.97777),
                  Stock=c( 25.62, 13.50, 26.50 , 26.02))

rownames(MAPE) <- c("ARIMA", "Holt-Winters", "ETS", "Our Model")
print(MAPE)
```

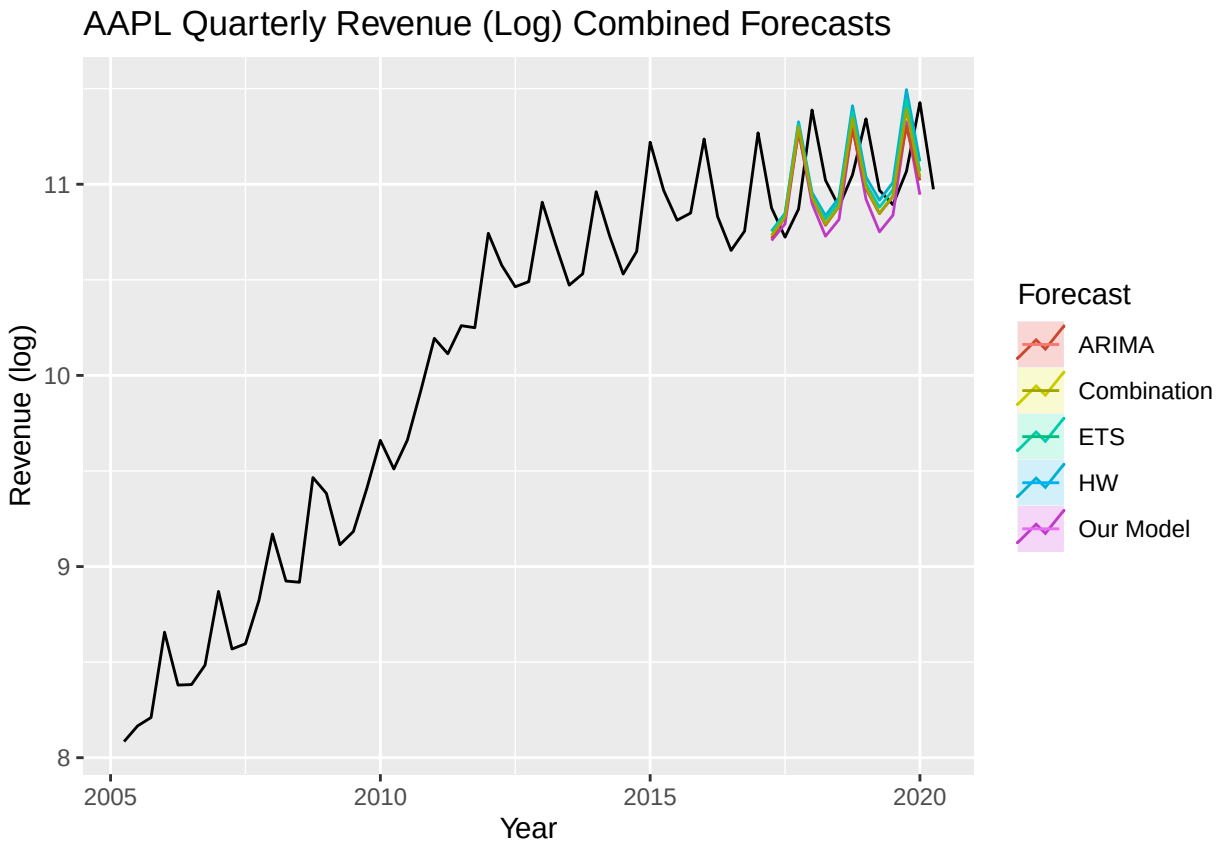
```
##              Revenue Stock
## ARIMA        0.69500 25.62
## Holt-Winters 0.88300 13.50
## ETS          2.19000 26.50
## Our Model    0.97777 26.02
```

The table above displays the MAPE of each model for Revenue and Stock. Comparing all 4 models in terms of MAPE, the ARIMA model has the lowest MAPE of 0.695 for Revenue and the Holt Winters model has the lowest MAPE of 13.50 for stock price.

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

```
#Combined Forecasts for Revenue

Our.Rev.forecast <- forecast(Rev_model, h = 12)

ARIMA.rev.forecast <- forecast(ARIMA.rev, h = 12)

ets.rev.forecast <- forecast(ets.rev, h = 12)

combination_rev <- (ets.rev.forecast[["mean"]] +
                      ARIMA.rev.forecast[["mean"]] +
                      Our.Rev.forecast[["mean"]] + HW.rev[["mean"]]) / 4
```

```
autoplot(log(aapl.rev.ts)) +
  autolayer(Our.Rev.forecast, series = "Our Model", PI = F) +
  autolayer(ARIMA.rev.forecast, series = "ARIMA", PI = F) +
  autolayer(HW.rev, series = "HW", PI = F) +
  autolayer(ets.rev.forecast, series = "ETS", PI = F) +
  autolayer(combination_rev, series = "Combination") +
  xlab("Year") +
  ylab("Revenue (log)") +
  ggtitle("AAPL Quarterly Revenue (Log) Combined Forecasts") +
  guides(colour=guide_legend(title="Forecast"))
```




```
forecast::accuracy(Our.Rev.forecast, log(aapl.rev.ts)) ### Test Set MAPE 2.39
```

```
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set -0.01095241 0.1250996 0.08863273 -0.1032801 0.8883947 0.3520912
## Test set      0.09995391 0.3060177 0.26575958  0.8722261 2.3855985 1.0557229
##                     ACF1 Theil's U
## Training set -0.01561631        NA
## Test set     -0.24556564  1.120092
```

```
forecast::accuracy(ARIMA.rev.forecast, log(aapl.rev.ts)) ### Test Set MAPE 2.07
```

```
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set 0.003604017 0.1297352 0.09613827 0.04063943 0.9752364 0.3819067
## Test set     0.060234084 0.2729042 0.23107030 0.51294074 2.0743168 0.9179207
##                     ACF1 Theil's U
```

```
## Training set  0.004785123        NA
## Test set     -0.232887139  1.000474
```

```
forecast::accuracy(ets.rev.forecast, log(aapl.rev.ts))  ### Test Set MAPE 2.19
```

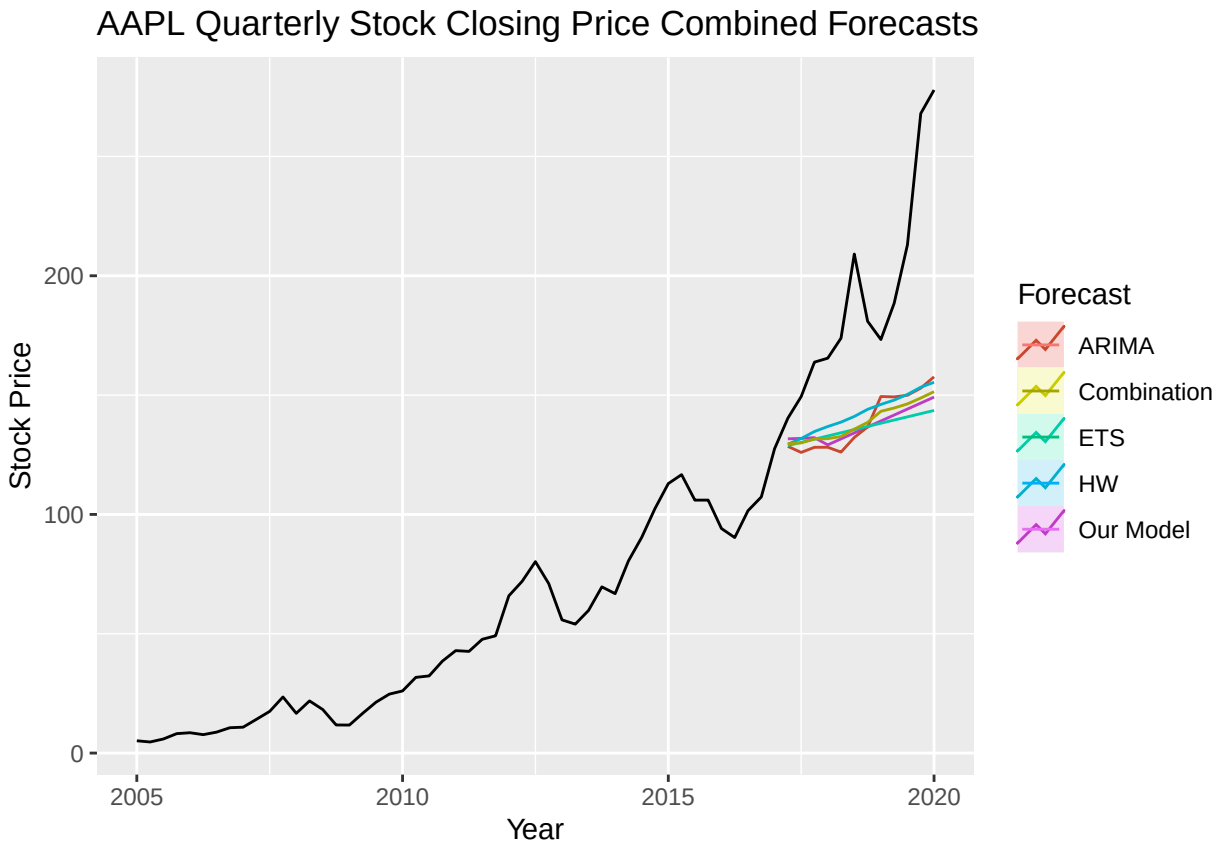
```
##                       ME      RMSE        MAE        MPE      MAPE      MASE
## Training set -0.01131898 0.1179501 0.08633936 -0.1042718 0.8655926 0.3429808
## Test set      0.01614665 0.2852470 0.24433887  0.1139398 2.1970131 0.9706297
##                     ACF1 Theil's U
## Training set  0.01781037        NA
## Test set     -0.26241860  1.055369
```

```
forecast::accuracy(combination_rev, log(aapl.rev.ts))  ### Test Set MAPE 2.26
```

```
##                   ME      RMSE       MAE       MPE     MAPE       ACF1 Theil's U
## Test set 0.04134229 0.2835946 0.2413375 0.3422109 2.167922 -0.2553067  1.044636
```

Comparing the MAPE of the test sets, it looks like the ARIMA forecast and the ETS forecast model has a lower MAPE than combination.

```
#Combined Forecasts for Stock

Our.Stock.forecast <- forecast(Stock_model, h = 12)

ARIMA.stock.forecast <- forecast(ARIMA.stock, h = 12)

ets.stock.forecast <- forecast(ets.stock, h = 12)


combination_stock <- (ets.stock.forecast[["mean"]] +
                        ARIMA.stock.forecast[["mean"]] +
                        Our.Stock.forecast[["mean"]] + HW.close[["mean"]]) / 4

autoplot(aapl.close.ts) +
  autolayer(Our.Stock.forecast, series = "Our Model", PI = F) +
  autolayer(ARIMA.stock.forecast, series = "ARIMA", PI = F) +
  autolayer(HW.close, series = "HW", PI = F) +
  autolayer(ets.stock.forecast, series = "ETS", PI = F) +
  autolayer(combination_stock, series = "Combination") +
  xlab("Year") +
  ylab("Stock Price") +
  ggtitle("AAPL Quarterly Stock Closing Price Combined Forecasts") +
  guides(colour = guide_legend(title = "Forecast"))
```




```
forecast::accuracy(Our.Stock.forecast, aapl.close.ts) ### Test Set MAPE 26.02085
```

```
##                       ME      RMSE      MAE       MPE     MAPE      MASE
## Training set -0.1856567  6.172235  4.77280 -8.159749 15.40947 0.3327404
## Test set     54.6244601 65.482854 54.62446 26.020846 26.02085 3.8081980
##                   ACF1 Theil's U
## Training set 0.1494202        NA
## Test set     0.5684871  2.661301
```

```
forecast::accuracy(ARIMA.stock.forecast, aapl.close.ts)  ### Test Set MAPE 25.62
```

```
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set  0.3341459  8.664725  6.061268 -0.2239617 13.08718 0.4225673
## Test set     53.2130737 62.766941 53.213074 25.6265022 25.62650 3.7098018
##                   ACF1 Theil's U
## Training set 0.2422916        NA
## Test set     0.5316961  2.584017
```

```
forecast::accuracy(ets.stock.forecast, aapl.close.ts) ### Test Set MAPE 26.50
```

```
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set  1.197506  6.963354  5.162642 -0.835211 13.12306 0.3599187
## Test set     55.744310 67.208163 55.744310 26.503605 26.50361 3.8862694
##                   ACF1 Theil's U
## Training set 0.2605109        NA
## Test set     0.5804522  2.712087
```

```
forecast::accuracy(combination_stock, aapl.close.ts)  ### Test Set MAPE 19.88
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
```

```
## Test set 53.26966 63.90824 53.26966 25.39802 25.39802 0.5613064   2.59287
```

Comparing the MAPE of the test sets, the combination forecast has the lowest value than the ARIMA and ETS forecast. By looking at the actual data, and the forecast we can see it did not 100% accurately forecast. This can be captured by doing more estimation methods and possibly a GARCH model to estimate the volatility.

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

```
#FOR VAR MODEL, Make TS variables using Diff(LOGS)

l.d.rev = diff(lgdata) %>% na.remove()
l.d.close = diff(lgclose) %>% na.remove()

#Data frame of two new ts variables
var_appl <- data.frame(cbind(l.d.rev, l.d.close))

#Use VARselect to choose a lag
VARselect(var_appl, lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      4      4      5
##
## $criteria
##                  1            2            3             4             5
## AIC(n) -6.160325031 -6.312951399 -6.626413924 -7.1828796856 -7.2160876744
## HQ(n)  -6.068329140 -6.159624915 -6.411756847 -6.9068920152 -6.8787694106
## SC(n)  -5.901758795 -5.882007672 -6.023092708 -6.4071809783 -6.2680114766
## FPE(n)  0.002112958  0.001818248  0.001336227  0.0007735312  0.0007601342
##                    6            7            8             9            10
## AIC(n) -7.1392788698 -7.0143923731 -6.889028902 -6.835995742 -6.782583840
## HQ(n)  -6.7406300126 -6.5544129224 -6.367718857 -6.253355105 -6.138612609
## SC(n)  -6.0188251816 -5.7215611944 -5.423820232 -5.198409582 -4.972620190
## FPE(n)  0.0008402591  0.0009841381  0.001167511  0.001308634  0.001496746
```

```
#From VAR Select we choose a p = 5
var <- VAR(var_appl, p=5)
summary(var)
```

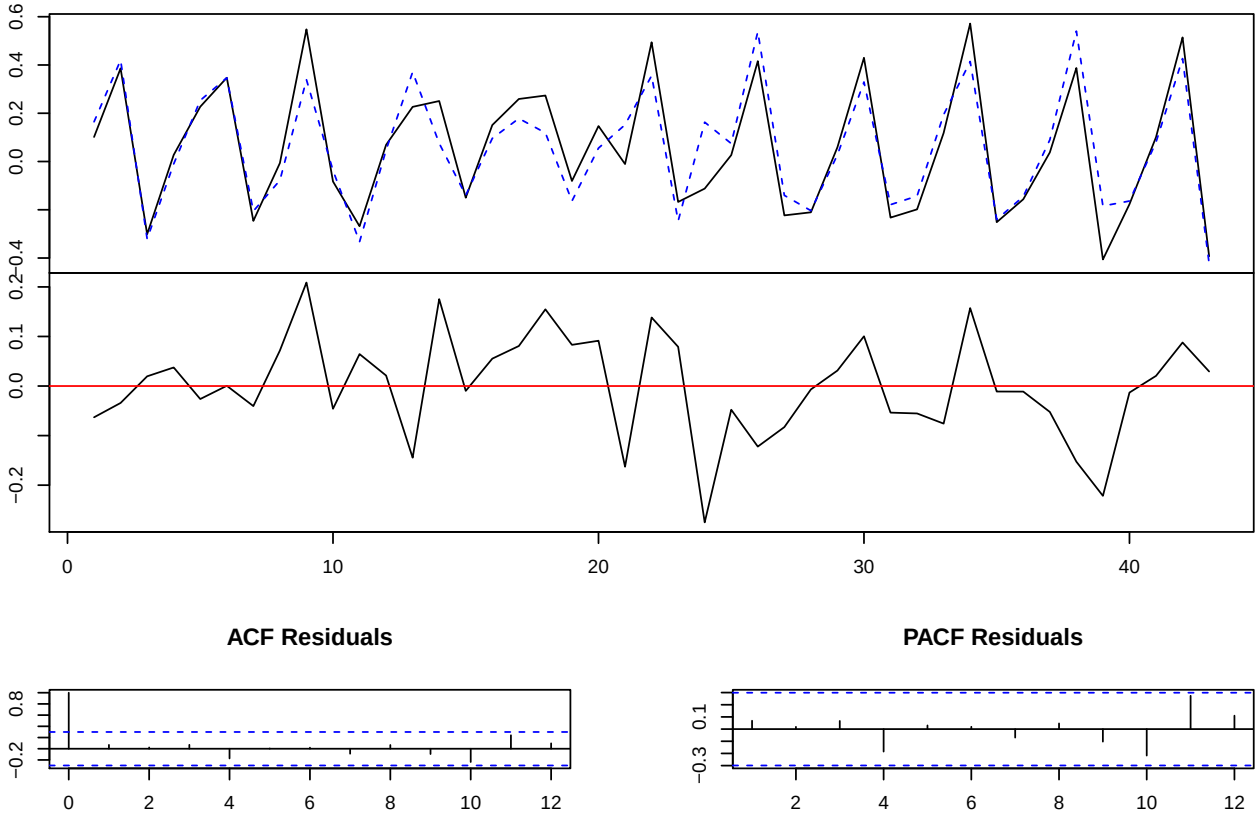
```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: l.d.rev, l.d.close
## Deterministic variables: const
## Sample size: 43
## Log Likelihood: 60.218
## Roots of the characteristic polynomial:
## 0.9837 0.9837 0.9373 0.8396 0.8396 0.8325 0.6991 0.6991 0.6113 0.6113
## Call:
## VAR(y = var_appl, p = 5)
##
##
## Estimation results for equation l.d.rev:
## ========================================
```

```
## l.d.rev = l.d.rev.l1 + l.d.close.l1 + l.d.rev.l2 + l.d.close.l2 + l.d.rev.l3 + l.d.close.l3 + l.d.rev
##
##              Estimate Std. Error t value Pr(>|t|)
## l.d.rev.l1   -0.44345    0.17065  -2.599   0.0140 *
## l.d.close.l1  0.30501    0.12206   2.499   0.0178 *
## l.d.rev.l2   -0.24889    0.13516  -1.841   0.0748 .
## l.d.close.l2 -0.03305    0.13081  -0.253   0.8022
## l.d.rev.l3   -0.21383    0.13885  -1.540   0.1334
## l.d.close.l3  0.30828    0.12238   2.519   0.0170 *
## l.d.rev.l4    0.74715    0.14362   5.202  1.1e-05 ***
## l.d.close.l4  0.12335    0.13588   0.908   0.3708
## l.d.rev.l5    0.23438    0.16137   1.452   0.1561
## l.d.close.l5 -0.22300    0.12960  -1.721   0.0950 .
## const         0.02006    0.03052   0.657   0.5157
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1184 on 32 degrees of freedom
## Multiple R-Squared: 0.8548,  Adjusted R-squared: 0.8094
## F-statistic: 18.83 on 10 and 32 DF,  p-value: 1.054e-10
##
##
## Estimation results for equation l.d.close:
## ==========================================
## l.d.close = l.d.rev.l1 + l.d.close.l1 + l.d.rev.l2 + l.d.close.l2 + l.d.rev.l3 + l.d.close.l3 + l.d.
##
##              Estimate Std. Error t value Pr(>|t|)
## l.d.rev.l1    0.07404    0.24101   0.307   0.7607
## l.d.close.l1  0.06949    0.17238   0.403   0.6895
## l.d.rev.l2    0.11157    0.19089   0.584   0.5630
## l.d.close.l2 -0.01221    0.18474  -0.066   0.9477
## l.d.rev.l3    0.15773    0.19610   0.804   0.4271
## l.d.close.l3 -0.11530    0.17283  -0.667   0.5095
## l.d.rev.l4    0.17557    0.20284   0.866   0.3932
## l.d.close.l4 -0.22260    0.19190  -1.160   0.2547
## l.d.rev.l5   -0.15776    0.22791  -0.692   0.4938
## l.d.close.l5 -0.26463    0.18304  -1.446   0.1580
## const         0.08464    0.04311   1.963   0.0584 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1673 on 32 degrees of freedom
## Multiple R-Squared: 0.2246,  Adjusted R-squared: -0.01768
## F-statistic: 0.927 on 10 and 32 DF,  p-value: 0.5218
##
##
##
## Covariance matrix of residuals:
##            l.d.rev l.d.close
## l.d.rev   0.014029  0.004057
## l.d.close 0.004057  0.027982
##
```

```
## Correlation matrix of residuals:
##           l.d.rev l.d.close
## l.d.rev    1.0000    0.2048
## l.d.close  0.2048    1.0000
```

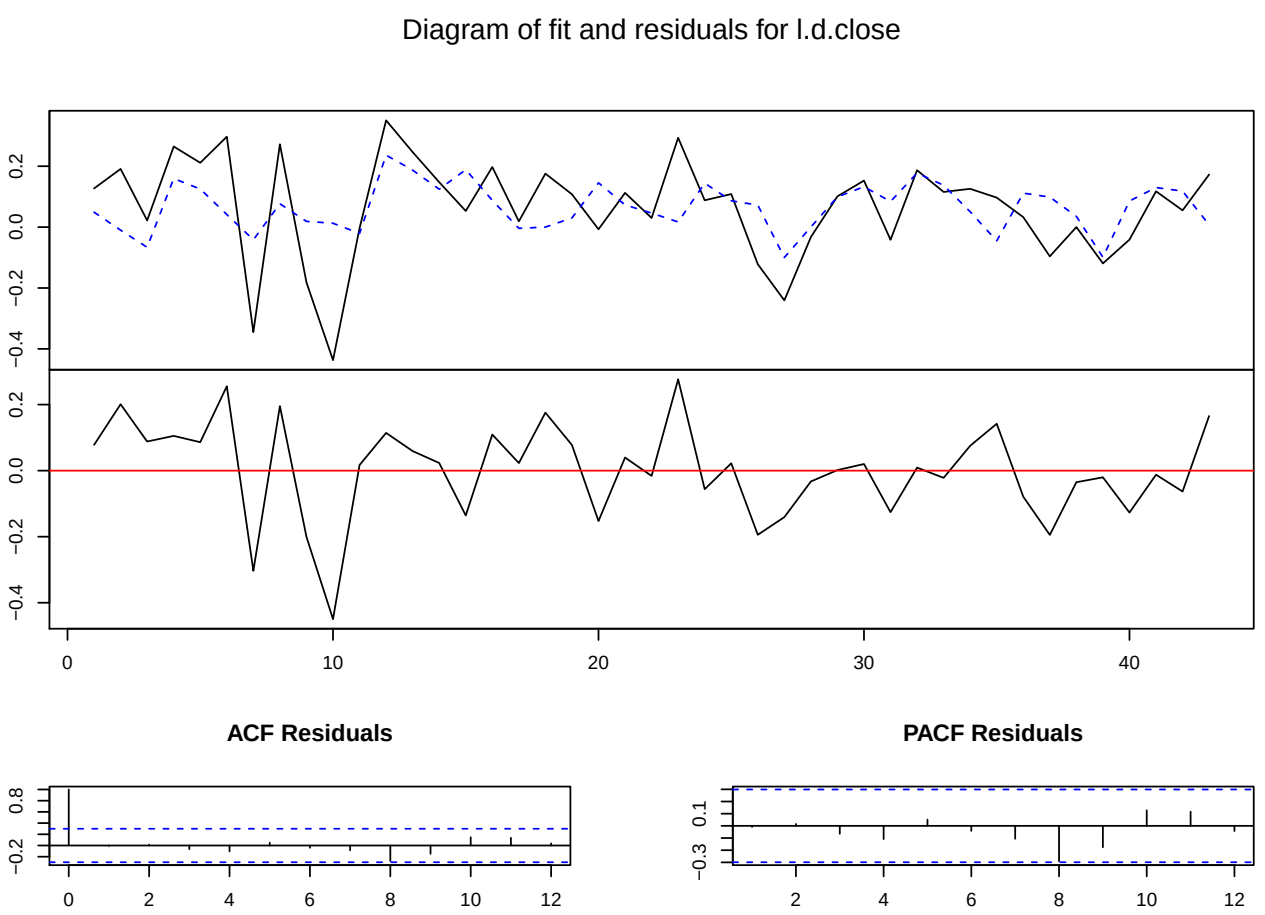
```
#Plot of fit and residuals
plot(var)
```

Diagram of fit and residuals for l.d.rev

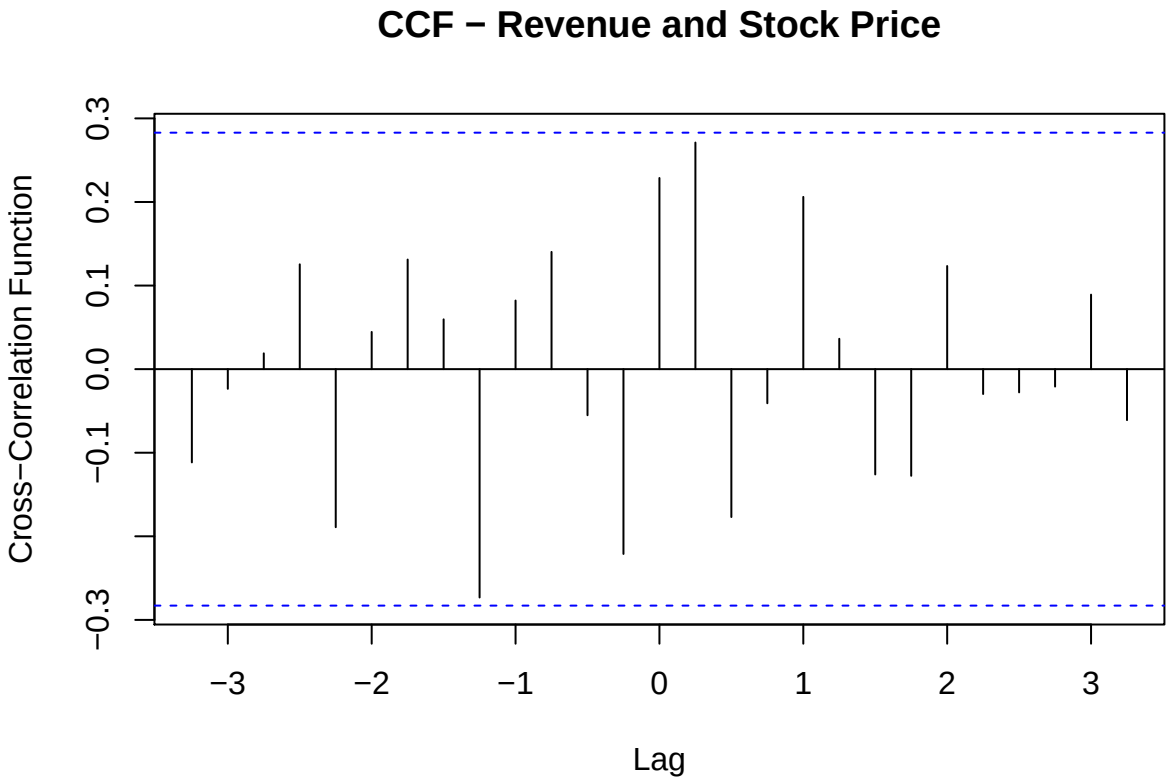


ACF Residuals

PACF Residuals




```
# CCF of Diff(log(data))
ccf(l.d.rev, l.d.close,ylab="Cross-Correlation Function",
    main="CCF - Revenue and Stock Price")
```

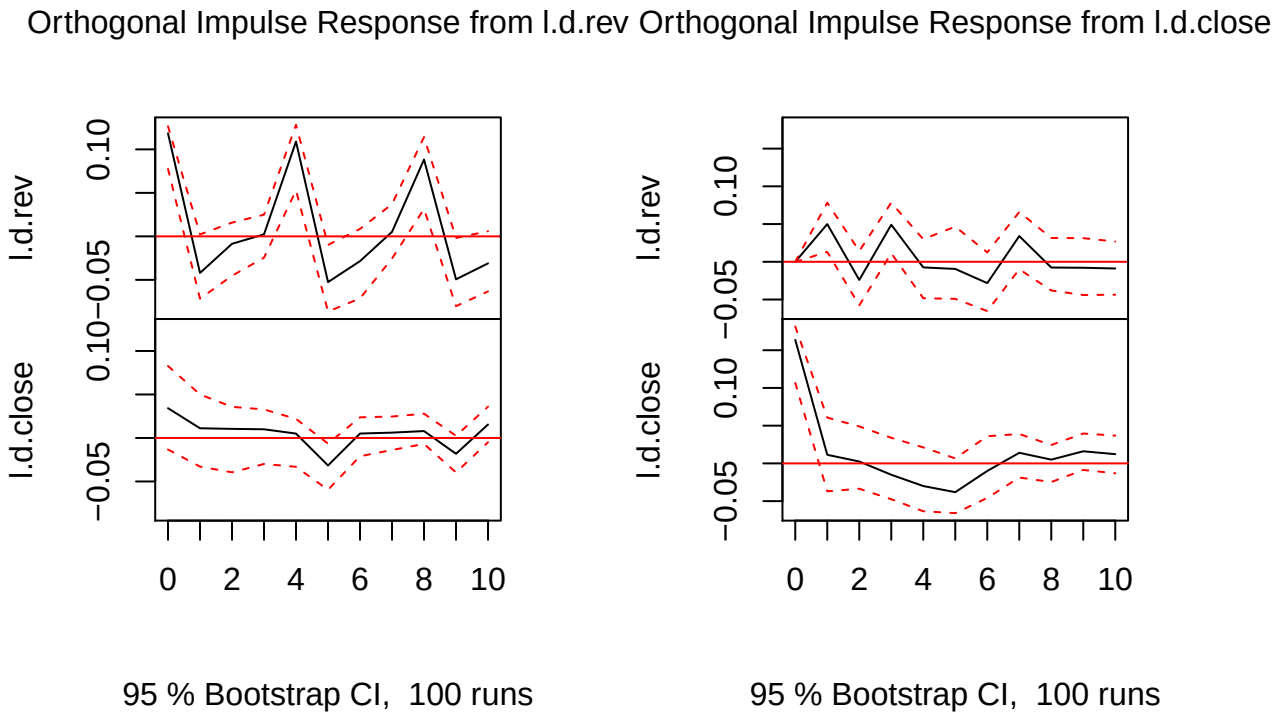


From the summary of the VAR model, we see that when Stock returns is statistically significant when we regress it on AAPL's revenue.

The plot of the VAR model for residuals show a mean reverting series and the ACF and PACF of the residuals look like white noise. It looks like Revenue leads to a above average value for one quarter later .

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

```
plot(irf(var))
```



Examining the impulse response function it would seem that a shock from revenue on stock returns increases the volatility. A shock from returns seems to not effect revenue all that much.

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

```
# H0: Revenue does not Granger cause Stock
grangertest(l.d.close ~ l.d.rev, order=4)
```

```
## Granger causality test
##
## Model 1: l.d.close ~ Lags(l.d.close, 1:4) + Lags(l.d.rev, 1:4)
## Model 2: l.d.close ~ Lags(l.d.close, 1:4)
##   Res.Df Df      F Pr(>F)
## 1     35
## 2     39 -4 0.9056 0.4713
```

```
# H0: Stock does not Granger cause Revenue
grangertest(l.d.rev ~ l.d.close, order=4)
```

```
## Granger causality test
##
## Model 1: l.d.rev ~ Lags(l.d.rev, 1:4) + Lags(l.d.close, 1:4)
## Model 2: l.d.rev ~ Lags(l.d.rev, 1:4)
##   Res.Df Df      F  Pr(>F)
## 1     35
## 2     39 -4 3.7978 0.01146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Returns do Granger cuase Revenue
```

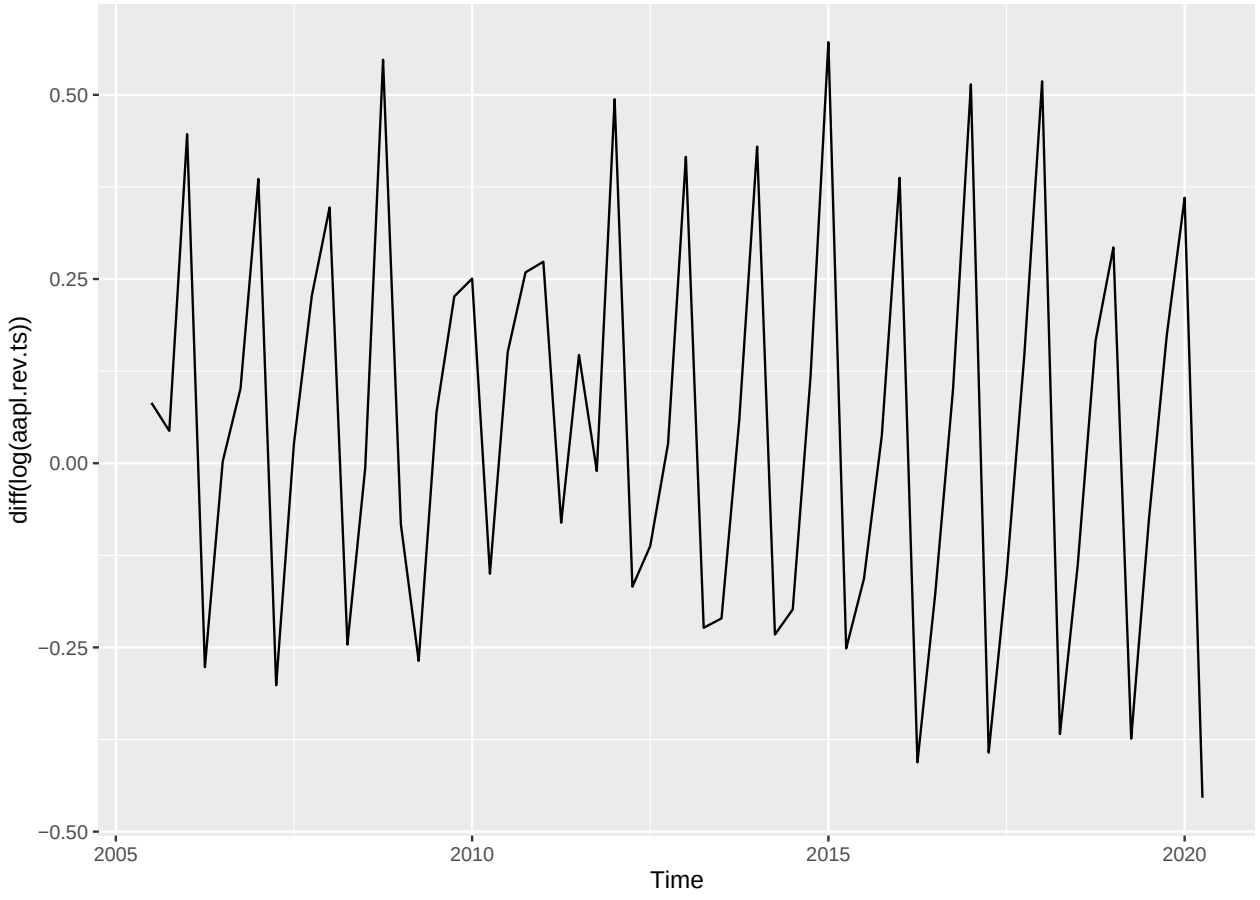
The Granger-Causality test of revenue on stock returns gives us a p-value of 0.47, which is larger than 0.05 so

we fail to reject the null hypothesis. We conclude that revenue does not Granger-Cause returns.

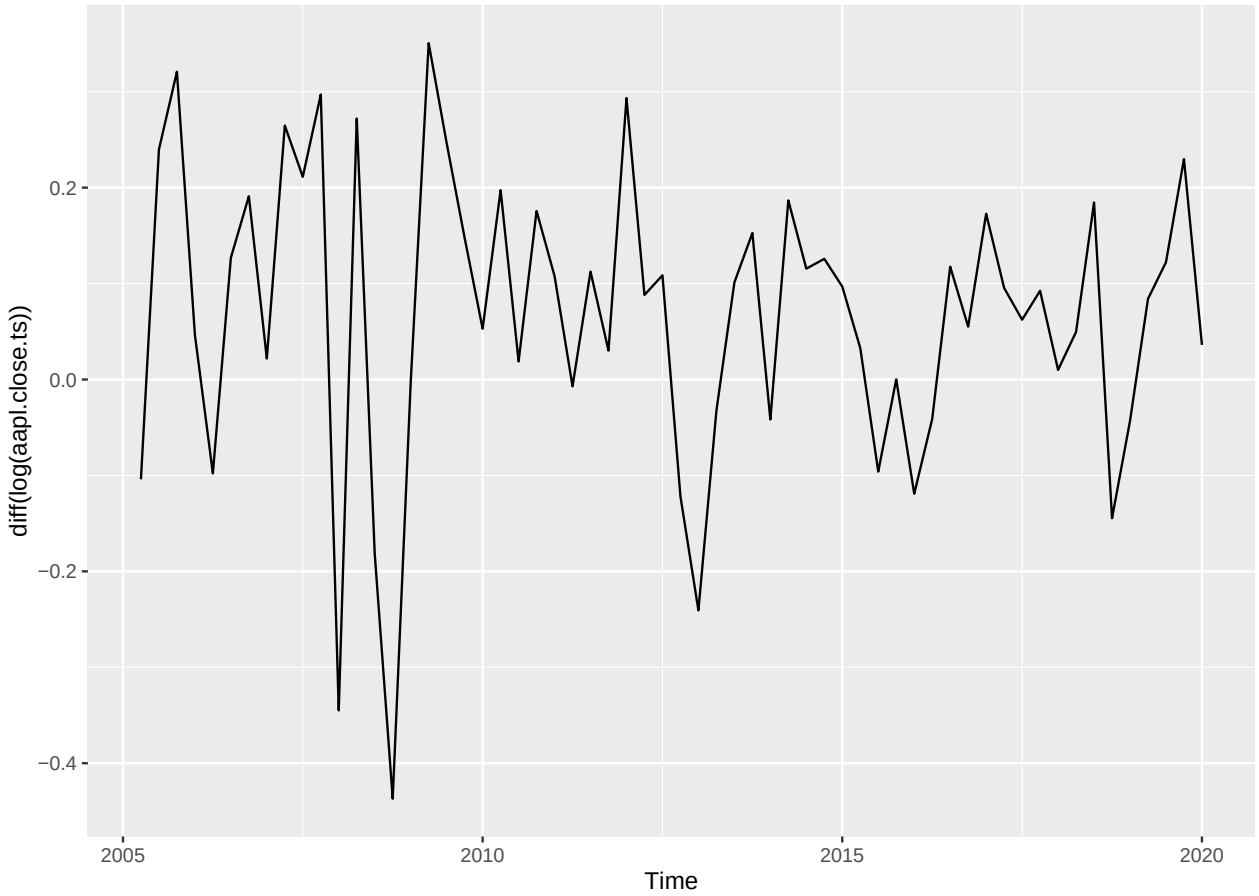
The Granger-Causality test of returns on revenue gives us a p-value of 0.01, which is less than 0.05 so we reject the null hypothesis. We conclude that returns Granger-Cause revenue.

(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.

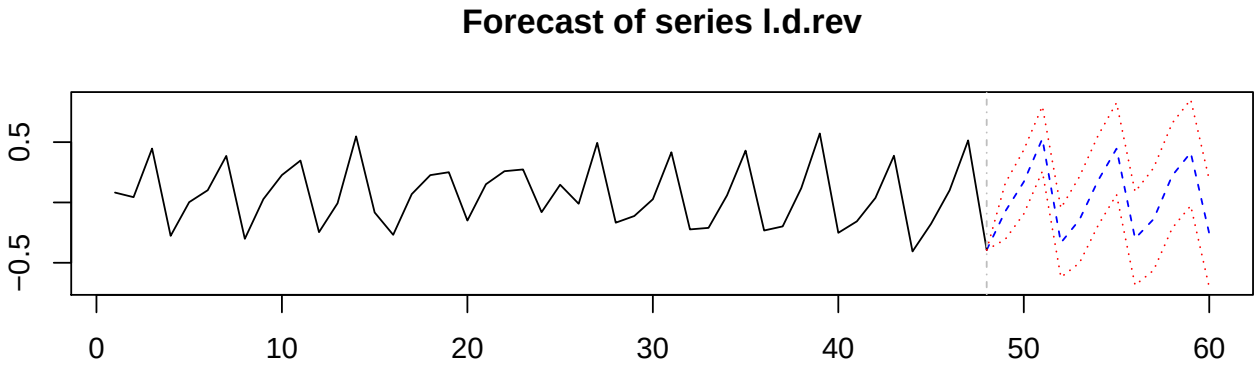
```
#Forecast for VAR model and Plot of Forecasts

## Plot of the difference of logs to compare the actual data vs forecast
autoplot(diff(log(aapl.rev.ts)))
```

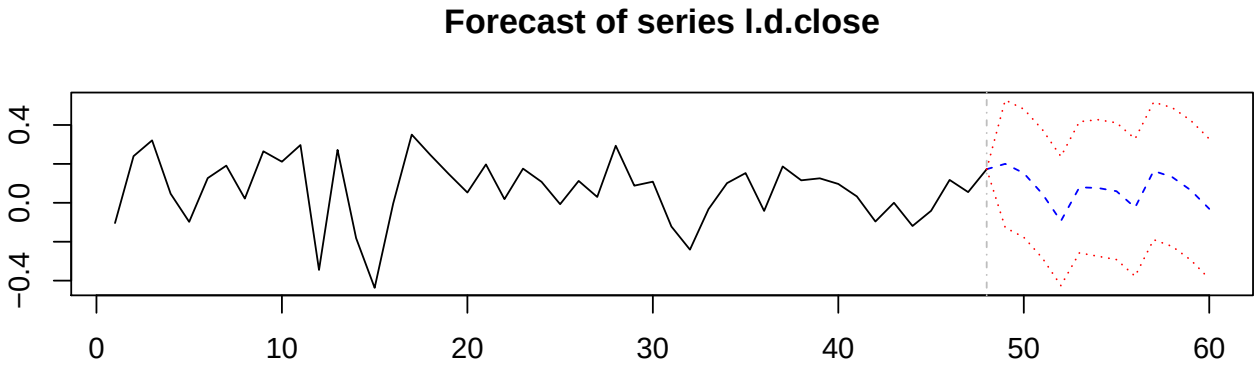



```
autoplot(diff(log(aapl.close.ts)))
```




```
var_forecasts = predict(object = var, n.ahead =  12)
plot(var_forecasts)
```






```
#Take diff(log()) of each
d.l.rev.test = diff(log(RevenueTS_test)) %>% na.remove()
d.l.close.test = diff(log(CloseTS_test)) %>% na.remove()

#Create Data frame of new variables
TEST = data.frame(d.l.rev.test,d.l.close.test)

forecast::accuracy(var_forecasts$fcst$l.d.rev[1:12], TEST$d.l.rev.test )
```

```
##                    ME      RMSE       MAE      MPE     MAPE       ACF1 Theil's U
## Test set -0.06548153 0.4962889 0.4040131 79.53311 159.4432 -0.2278183  1.099578
```

```
forecast::accuracy(var_forecasts$fcst$l.d.close[1:12], TEST$d.l.close.test )
```

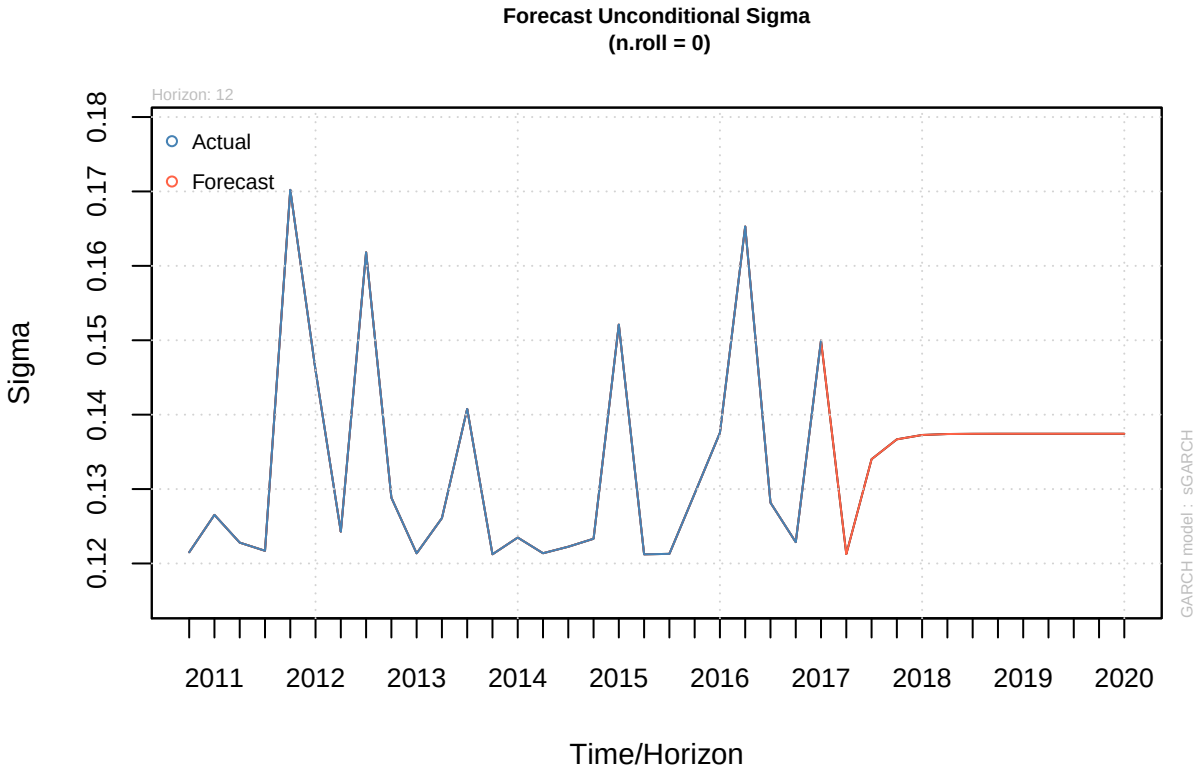
```
##                    ME      RMSE        MAE      MPE    MAPE        ACF1
## Test set -0.01585286 0.1122193 0.09835506 13.15173 151.846 -0.02275947
##          Theil's U
## Test set  2.402439
```

The Var forecasts have the largest MAPE for the test sets of 159.44 for Revenue and 151.85 for Returns. This would indicate that out of all the forecasts, it is the worst one for forecasting these two series.

(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.

Revenue ARIMA Residuals

```
#For Revenue, we choose the ARIMA model since it has the lowest MAPE

#Model Specification
model=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
distribution.model = "sstd")

#Fit the model to residuals of model
modelfit=ugarchfit(spec=model,data=ARIMA.rev$residuals)

#Forecast model
modelfor = ugarchforecast(modelfit, data = NULL, n.ahead = 12,
                          n.roll = 0, out.sample = 0)
plot(modelfor, which=3)
```

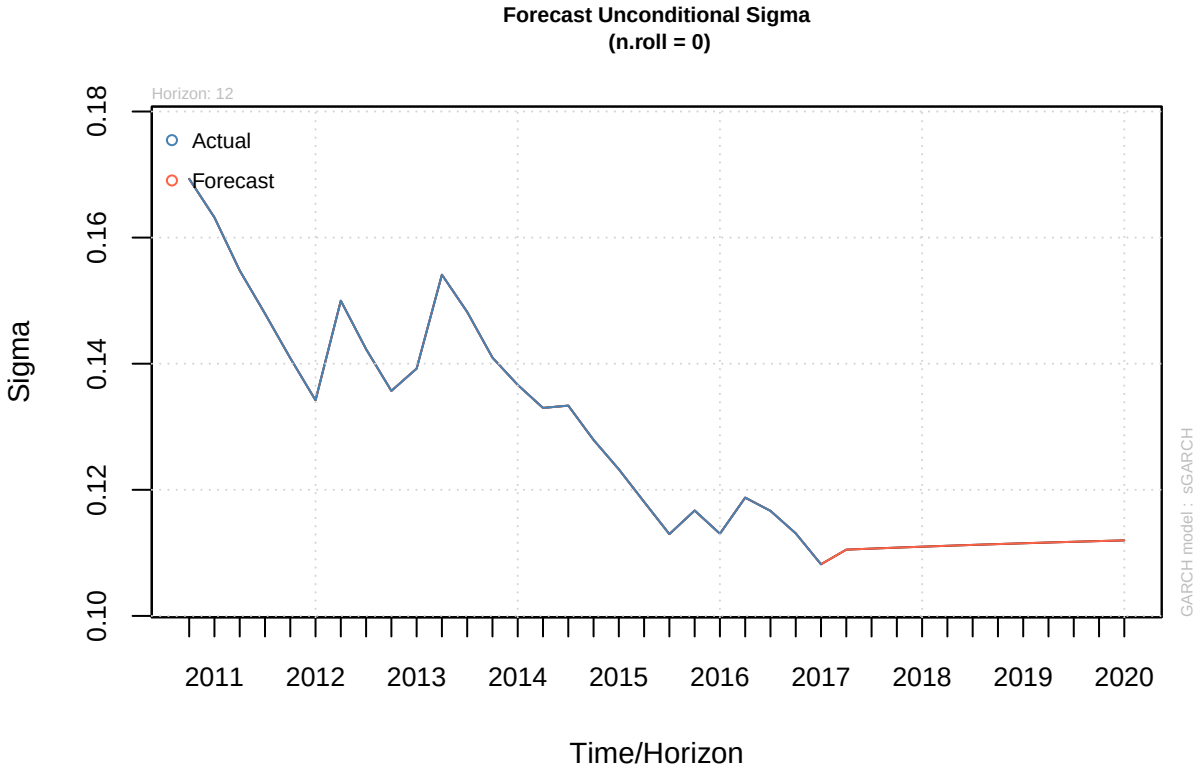


Here we fit a GARCH model on our best model for AAPL Returns(log) residuals. Our best model was ARIMA and we can see that the volatility forecast has some variance in the first few quarters and then mean reverts. The standard deviation is forecasted at a low value.

Holt Winters Residuals

```
#For Stock, we choose the Holt Winters model since it has the lowest MAPE

#Model Specification
model2=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
distribution.model = "sstd")
```

```
#Fit the model to residuals of model
modelfit2=ugarchfit(spec=model2, data=HW.close$residuals)

#Forecast model
modelfor2 = ugarchforecast(modelfit2, data = NULL, n.ahead = 12,
                           n.roll = 0, out.sample = 0)
plot(modelfor2, which=3)
```



Here we fit a GARCH(1,1) on our best model for forecasting AAPL Stock price which was Holt Winters. We can see from this forecast of volatility the standard deviation remains low which means AAPL is a less risky investment. So it will be safe to invest and expect some returns without too much volatility.

III. Conclusions and Future Work

The finding that returns cause revenue was certainly against our expectations. As said in at the beginning of our project, we expected that revenue would increase the stock price as investors will be more eager to buy into a company that is doing well. However we discovered the opposite. It could be the case that as returns increase APPLE is able to expand more in order to increase revenue. Ultimately forecasting the volatility was problematic, even with the GARCH model, this is to be expected with highly volatile series. Perhaps future methods for forecasting volatility would serve us better for future predictions.

For future work, we would like to use a higher frequency for the data. Since the revenues were given as quarterly, we had to make stock prices on a quarterly basis so our analysis would be abele to match on similar time series horizons. Using higher frequency data, we would able to fit more appropriate models and our forecasts would be more robusts given us lower MAPE values.

IV. References

Quarterly AAPL Revenue Web Scrape from https://www.macrotrends.net/stocks/charts/AAPL/apple/revenue .

Monthly Stock Data for AAPL from https://finance.yahoo.com/quote/AAPL/history?period1=1104537600&period2=1590624000&interval=1mo&filter=history&frequency=1mo .

V. R Source Code

```
#### load libraries
library(rvest) ## able to webscrape the quarterly revenue
library(dplyr)
library(ggplot2)
library(forecast)
library(readr) ## for the csv file
library(strucchange) # for efp CUSUM plot
library(tidyverse)
library(tseries)
library(vars)
library(lmtest)
library(car)
library(Metrics)
library(stats)
library(moments)
library(rugarch)

##### CLEANING AND WEB SCRAPING QUARTERLY REVENUE ####
aapl_rev <- read_html("https://www.macrotrends.net/stocks/charts/AAPL/apple/revenue")
aapl.html <- html_nodes(aapl_rev, "thead+ thead th , #style-1 td")

aapl.results <- rvest::html_text(aapl.html)

Date <- aapl.results[seq(1, length(aapl.results), 2)]

`Revenue` <- aapl.results[seq(2, length(aapl.results),2)]

aapl.revenue <- data.frame(Date = Date[16:76], Revenue = `Revenue`[16:76] , stringsAsFactors = FALSE) %
  arrange(Date)

aapl.revenue$Date <- as.Date(aapl.revenue$Date, format = "%Y-%m-%d")

aapl.revenue$Revenue <- as.numeric(gsub("\\$|,","",aapl.revenue$Revenue ))

aapl.rev.ts <- ts(aapl.revenue$Revenue, start = 2005.25, frequency = 4)

##### CLEANING STOCK DATA ####
AAPL <- read_csv("AAPL.csv", col_types = cols(Date = col_date(format = "%Y-%m-%d"),
Volume = col_skip()))

#ggplot(AAPL, aes(x=Date)) + geom_line(aes(y=`Adj Close`))

aapl.monthly <- ts(AAPL$`Adj Close`, start = c(2005,2), frequency = 12)
```

```r
aapl.quarterly <- ts(aggregate(aapl.monthly, nfrequency = 4)/3, start = 2005.25, frequency = 4)

aapl.stock.close <- data.frame(date=time(aapl.quarterly), close =as.matrix(aapl.quarterly))

#autoplot(aapl.quarterly) + ggtitle("AAPL Quarterly Stock Price", "2005-2020") + xlab("Date") + ylab("C

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

aapl <- cbind(aapl.revenue, Close = aapl.stock.close$close)

#Training and Test Set
aapl.train <- aapl[1:49,]
aapl.test <- aapl[50:61,]

#Times Series Variables
RevenueTS <- ts(aapl.train$Revenue, start = 2005, frequency = 4)
RevenueTS_test <- ts(aapl.test$Revenue, start = 2017.25, frequency = 4)
aapl.revenue.ts <-  ts(aapl$Revenue, start = 2005, frequency = 4)


CloseTS <- ts(aapl.train$Close, start = 2005, frequency = 4)
CloseTS_test <- ts(aapl.test$Close, start = 2017.25, frequency = 4)
aapl.close.ts <-  ts(aapl$Close, start = 2005, frequency = 4)

#Log of TS Variables
lgdata=log(RevenueTS)
lgclose=log(CloseTS)

## Quarterly Revenue AAPL
ggplot(aapl, aes(x = Date, y = Revenue)) +
  geom_line(color="#3796db", lwd = 0.8) +
  ggtitle("AAPL Quarterly Revenue", "2005-2020") +
  xlab("Date") +
  ylab("Revenue (millions)")

par(mfrow=c(2, 1))
ggAcf(lgdata, main="ACF - Revenue", lag.max=24)+
  theme(plot.title = element_text(hjust = 0.5))
ggPacf(lgdata, main="PACF - Revenue", lag.max=24)+
  theme(plot.title = element_text(hjust = 0.5))  #AR 5

## AAPL Stock Close at Quarters

ggplot(aapl, aes(x = Date, y = Close)) +
  geom_line(color="#3796db", lwd = 0.8) +
  ggtitle("AAPL Quarterly Stock Close", "2005-2020") +
  xlab("Date") +
  ylab("Stock Price")

par(mfrow=c(2, 1))
ggAcf(aapl.train$Close, main="ACF - Stock Price", lag.max=24) +
  theme(plot.title = element_text(hjust = 0.5))
ggPacf(aapl.train$Close, main="PACF - Stock Price", lag.max=24) +
```

```
  theme(plot.title = element_text(hjust = 0.5))  #AR 1

#### (b) Fit a model that includes, trend, seasonality and cyclical components. Make sure to discuss yo

##### Revenue Model
# Revenue Model
Rev_model=  auto.arima(lgdata)

#Stock Model
Stock_model <- auto.arima(CloseTS)

#Add Fitted and Residial variables to Training data set.
aapl.train <- aapl.train %>%
  mutate(
    Revenue_Fitted = Rev_model$fitted,
    Revenue_Residuals = Rev_model$residuals,
    Close_Fitted = Stock_model$fitted,
    Close_Residuals = Stock_model$residuals
  )

## Revenue
Rev_cols <- c("Fitted Values"="black", "Observed Values"="#3796db")

ggplot(aapl.train, aes(x = Date, y = log(Revenue))) +
  geom_line(aes(color = "Observed Values"), lwd=0.8) +
  geom_line(aes(y = Revenue_Fitted, color="Fitted Values")) +
  ggtitle("AAPL Quarterly Revenue (Log)", "2005-2017") +
  xlab("Date") +
  ylab("Revenue (log)") +
  scale_color_manual("Legend", values = Rev_cols)

summary(Rev_model)

##### Stock Price Model

## Stock Price
Close_cols <- c("Fitted Values" = "black", "Observed Values" = "Forest Green")

ggplot(aapl.train, aes(x = Date, y = Close)) +
  geom_line(aes(color = "Observed Values"), lwd=0.8) +
  geom_line(aes(y = Close_Fitted, color="Fitted Values")) +
 ggtitle("AAPL Quarterly Stock Close", "2005-2017") +
  xlab("Date") +
  ylab("Stock Price") +
  scale_color_manual("Legend", values = Close_cols)

summary(Stock_model)

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

## Revenue
ggplot(aapl.train, aes(x = Revenue_Fitted, y = Revenue_Residuals)) +
  geom_point(color='dark green', alpha=0.9) +
```

```
  geom_hline(yintercept=mean(aapl.train$Revenue_Residuals),
             color='black', linetype='dashed') +
  ggtitle("Residuals vs Fitted Values", "Revenue Model") +
  xlab("Fitted Values") +
  ylab("Residuals")

## Stock Price

ggplot(aapl.train, aes(x = Close_Fitted, y = Close_Residuals)) +
  geom_point(color='Navy', alpha=0.9) +
  geom_hline(yintercept=mean(aapl.train$Close_Residuals),
             color='black', linetype='dashed') +
  ggtitle("Residuals vs Fitted Values", "Stock Price Model") +
  xlab("Fitted Values") +
  ylab("Residuals")

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

## Revenue ACF & PACF

par(mfrow=c(2, 1))
ggAcf(aapl.train$Revenue_Residuals, main = "ACF - Revenue Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

ggPacf(aapl.train$Revenue_Residuals, main = "PACF - Revenue Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

# Stock Price ACF & PACF

par(mfrow=c(2, 1))
ggAcf(aapl.train$Close_Residuals, main = "ACF - Stock Price Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

ggPacf(aapl.train$Close_Residuals, main = "PACF - Stock Price Model Residuals") +
  theme(plot.title = element_text(hjust = 0.5))

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

## Revenue CUSUM TEST

plot(efp(aapl.train$Revenue_Residuals~1, type = "Rec-CUSUM"),
     main = "Revenue Model CUSUM Test")

## Stock Price

plot(efp(aapl.train$Close_Residuals~1, type = "Rec-CUSUM"),
     main = "Stock Price Model CUSUM Test")

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

## Revenue

plot(recresid(aapl.train$Revenue_Residuals~1), pch=16,
```

```
     main = "Revenue Model Recursive Residuals",
     ylab="Residuals", col = "dark red")

## Stock Price

plot(recresid(aapl.train$Close_Residuals~1), pch=16,
     main = "Stock Price Model Recursive Residuals",
     ylab="Residuals", col = "dark red")

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

# Revenue
coeftest(Rev_model)

# QQ plot of Residuals for Revenue
ggplot(aapl.train) +
  geom_qq(aes(sample=Revenue_Residuals), color='black', alpha=0.6) +
  ggtitle('QQ Normal Plot of Residuals', 'Revenue Model') +
  ylab('Sample Quantiles') +
  xlab('Theoretical Quantiles')

print(paste("Skewness:", skewness(aapl.train$Revenue_Residuals)))
print(paste("Kurtosis:", kurtosis(aapl.train$Revenue_Residuals)))

# Revenue
coeftest(Stock_model)

# QQ plot of Residuals for Stock Price
ggplot(aapl.train) +
  geom_qq(aes(sample=Close_Residuals), color='black', alpha=0.6) +
  ggtitle('QQ Normal Plot of Residuals', 'Stock Model') +
  ylab('Sample Quantiles') +
  xlab('Theoretical Quantiles')
print(paste("Skewness:", skewness(aapl.train$Close_Residuals)))
print(paste("Kurtosis:", kurtosis(aapl.train$Close_Residuals)))

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

#Revenue Forecast
rev_forecasts=forecast(Rev_model, h = 12)
plot(rev_forecasts, xlab = "Date", ylab = "Revenue (log)",
     main = "AAPL Revenue Forecast from ARIMA(0,1,1)(0,1,1)[4]")

forecast::accuracy(rev_forecasts,  log(aapl.test$Revenue)) ### test set MAPE 0.97777

#Stock Forecast
stock_forecasts=forecast(Stock_model, h =  12)
plot(stock_forecasts,  xlab = "Date", ylab = "Stock Closing Price",
     main = "AAPL Stock Forecast from ARIMA(0,1,0)(0,0,1)[4]")

forecast::accuracy(stock_forecasts,  aapl.test$Close) ### test set MAPE 26.02
```

```
#### (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?

##### ARIMA

#ARIMA for Revenue
# Choose AR5 based off PACF, but AR(4) had a lower AIC and BIC

ARIMA.rev = Arima(lgdata,order=c(4,0,1),seasonal=list(order=c(1,0,0)))
plot(forecast(ARIMA.rev,h=12),shadecols="oldstyle", xlab = "Date",
     ylab = "Revenue (log)", main = "Revenue Forecast from ARIMA(4,0,1)(1,0,0)[4]")

summary(ARIMA.rev)

#MAPE
forecast::accuracy(forecast(ARIMA.rev,h=12), log(aapl.test$Revenue)) ### test set MAPE 0.695

#ARIMA for Stock Price

# Choose S-AR1 based off PACF, including  drift lowered the AIC and BIC

ARIMA.stock = Arima(CloseTS,order=c(0,1,0),include.drift=TRUE,seasonal=list(order=c(1,1,0)))
plot(forecast(ARIMA.stock,h=12) , shadecols="oldstyle", xlab = "Date",
     ylab = "Stock Price", main = "Stock Closing Price Forecast from ARIMA(0,1,0)(1,1,0)[4]")

summary(ARIMA.stock)
coeftest(ARIMA.stock)

#MAPE
forecast::accuracy(forecast(ARIMA.stock,h = 12), aapl.test$Close) ### test set MAPE 25.62

##### Holt-Winters

#Holt-Winters for Revenue

HW.rev <- hw(lgdata,seasonal="multiplicative", h =12)

autoplot(lgdata) +
  autolayer(HW.rev, series="HW multiplicative forecasts",
    PI=FALSE) +
  xlab("Year") +
  ylab("Revenue (log)") +
  ggtitle("AAPL Quarterly Revenue (Log)") +
  guides(colour=guide_legend(title="Forecast"))

summary(HW.rev)

# MAPE = 0.883

#Holt-Winters for Stock Price

HW.close <- hw(CloseTS,seasonal="multiplicative", h = 12)
```

```r
autoplot(CloseTS) +
  autolayer(HW.close, series="HW multiplicative forecasts",
    PI=FALSE) +
  xlab("Year") +
  ylab("Stock Price") +
  ggtitle("AAPL Quarterly Stock Close") +
  guides(colour=guide_legend(title="Forecast"))

summary(HW.close)

#MAPE = 13.50

##### ETS

#ETS for Revenue
ets.rev <- ets(lgdata)

ets.rev.forecast <- forecast(ets.rev, h = 12)

ets.rev %>% forecast(h=12) %>%
  autoplot() +
  ylab("Revenue (log)")

summary(ets.rev)

forecast::accuracy(ets.rev.forecast, log(aapl.rev.ts)) ### test set MAPE is 2.19

#ETS for Stock Price

ets.stock <- ets(CloseTS)

ets.stock.forecast <- forecast(ets.stock, h = 12)

ets.stock %>% forecast(h=12) %>%
  autoplot() +
  ylab("Stock Price")

summary(ets.stock)

forecast::accuracy(ets.stock.forecast,aapl.close.ts) ### test set MAPE is 26.50

#Table of MAPE for each model and its respective series
MAPE = data.frame(Revenue=c(0.695, 0.883 , 2.19 , 0.97777),
                  Stock=c( 25.62, 13.50, 26.50 , 26.02))
rownames(MAPE) <- c("ARIMA", "Holt-Winters", "ETS", "Our Model")
print(MAPE)

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

#Combined Forecasts for Revenue

Our.Rev.forecast <- forecast(Rev_model, h = 12)
```

```r
ARIMA.rev.forecast <- forecast(ARIMA.rev, h = 12)

ets.rev.forecast <- forecast(ets.rev, h = 12)

combination_rev <- (ets.rev.forecast[["mean"]] +
                      ARIMA.rev.forecast[["mean"]] +
                      Our.Rev.forecast[["mean"]] + HW.rev[["mean"]]) / 4

autoplot(log(aapl.rev.ts)) +
  autolayer(Our.Rev.forecast, series = "Our Model", PI = F) +
  autolayer(ARIMA.rev.forecast, series = "ARIMA", PI = F) +
  autolayer(HW.rev, series = "HW", PI = F) +
  autolayer(ets.rev.forecast, series = "ETS", PI = F) +
  autolayer(combination_rev, series = "Combination") +
  xlab("Year") +
  ylab("Revenue (log)") +
  ggtitle("AAPL Quarterly Revenue (Log) Combined Forecasts") +
  guides(colour=guide_legend(title="Forecast"))

forecast::accuracy(Our.Rev.forecast, log(aapl.rev.ts)) ### Test Set MAPE 2.39

forecast::accuracy(ARIMA.rev.forecast, log(aapl.rev.ts)) ### Test Set MAPE 2.07

forecast::accuracy(ets.rev.forecast, log(aapl.rev.ts))  ### Test Set MAPE 2.19

forecast::accuracy(combination_rev, log(aapl.rev.ts))  ### Test Set MAPE 2.26

#Comparing the MAPE of the test sets, it looks like the ARIMA forecast and the ETS forecast model has a

#Combined Forecasts for Stock

Our.Stock.forecast <- forecast(Stock_model, h = 12)

ARIMA.stock.forecast <- forecast(ARIMA.stock, h = 12)

ets.stock.forecast <- forecast(ets.stock, h = 12)

combination_stock <- (ets.stock.forecast[["mean"]] +
                        ARIMA.stock.forecast[["mean"]] +
                        Our.Stock.forecast[["mean"]] + HW.close[["mean"]]) / 4

autoplot(aapl.close.ts) +
  autolayer(Our.Stock.forecast, series = "Our Model", PI = F) +
  autolayer(ARIMA.stock.forecast, series = "ARIMA", PI = F) +
  autolayer(HW.close, series = "HW", PI = F) +
  autolayer(ets.stock.forecast, series = "ETS", PI = F) +
  autolayer(combination_stock, series = "Combination") +
  xlab("Year") +
  ylab("Stock Price") +
  ggtitle("AAPL Quarterly Stock Closing Price Combined Forecasts") +
  guides(colour = guide_legend(title = "Forecast"))

forecast::accuracy(Our.Stock.forecast, aapl.close.ts) ### Test Set MAPE 26.02085
```

```r
forecast::accuracy(ARIMA.stock.forecast, aapl.close.ts)  ### Test Set MAPE 25.62

forecast::accuracy(ets.stock.forecast, aapl.close.ts) ### Test Set MAPE 26.50

forecast::accuracy(combination_stock, aapl.close.ts)  ### Test Set MAPE 19.88

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

#FOR VAR MODEL, Make TS variables using Diff(LOGS)

l.d.rev = diff(lgdata) %>% na.remove()
l.d.close = diff(lgclose) %>% na.remove()

#Data frame of two new ts variables
var_appl <- data.frame(cbind(l.d.rev, l.d.close))

#Use VARselect to choose a lag
VARselect(var_appl, lag.max = 10)

#From VAR Select we choose a p = 5
var <- VAR(var_appl, p=5)
summary(var)

#Plot of fit and residuals
plot(var)

# CCF of Diff(log(data))
ccf(l.d.rev, l.d.close,ylab="Cross-Correlation Function",
    main="CCF - Revenue and Stock Price")

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

plot(irf(var))

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

# H0: Revenue does not Granger cause Stock
grangertest(l.d.close ~ l.d.rev, order=4)

# H0: Stock does not Granger cause Revenue
grangertest(l.d.rev ~ l.d.close, order=4)

#Returns do Granger cuase Revenue

#### (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 me

#Forecast for VAR model and Plot of Forecasts

## Plot of the difference of logs to compare the actual data vs forecast
autoplot(diff(log(aapl.rev.ts)))
```

```r
autoplot(diff(log(aapl.close.ts)))

var_forecasts = predict(object = var, n.ahead =  12)
plot(var_forecasts)

#Take diff(log()) of each
d.l.rev.test = diff(log(RevenueTS_test)) %>% na.remove()
d.l.close.test = diff(log(CloseTS_test)) %>% na.remove()

#Create Data frame of new variables
TEST = data.frame(d.l.rev.test,d.l.close.test)

forecast::accuracy(var_forecasts$fcst$l.d.rev[1:12], TEST$d.l.rev.test )

forecast::accuracy(var_forecasts$fcst$l.d.close[1:12], TEST$d.l.close.test )

#### (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.

##### Revenue ARIMA Residuals

#For Revenue, we choose the ARIMA model since it has the lowest MAPE

#Model Specification
model=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
distribution.model = "sstd")

#Fit the model to residuals of model
modelfit=ugarchfit(spec=model,data=ARIMA.rev$residuals)

#Forecast model
modelfor = ugarchforecast(modelfit, data = NULL, n.ahead = 12, n.roll = 0, out.sample = 0)
plot(modelfor, which=3)

##### Holt Winters Residuals

#For Stock, we choose the Holt Winters model since it has the lowest MAPE

#Model Specification
model2=ugarchspec(
variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
distribution.model = "sstd")

#Fit the model to residuals of model
modelfit2=ugarchfit(spec=model2, data=HW.close$residuals)

#Forecast model
modelfor2 = ugarchforecast(modelfit2, data = NULL, n.ahead = 12, n.roll = 0, out.sample = 0)
plot(modelfor2, which=3)
```