# QF 202 Final LR

Laurent Reyes

2024-05-08

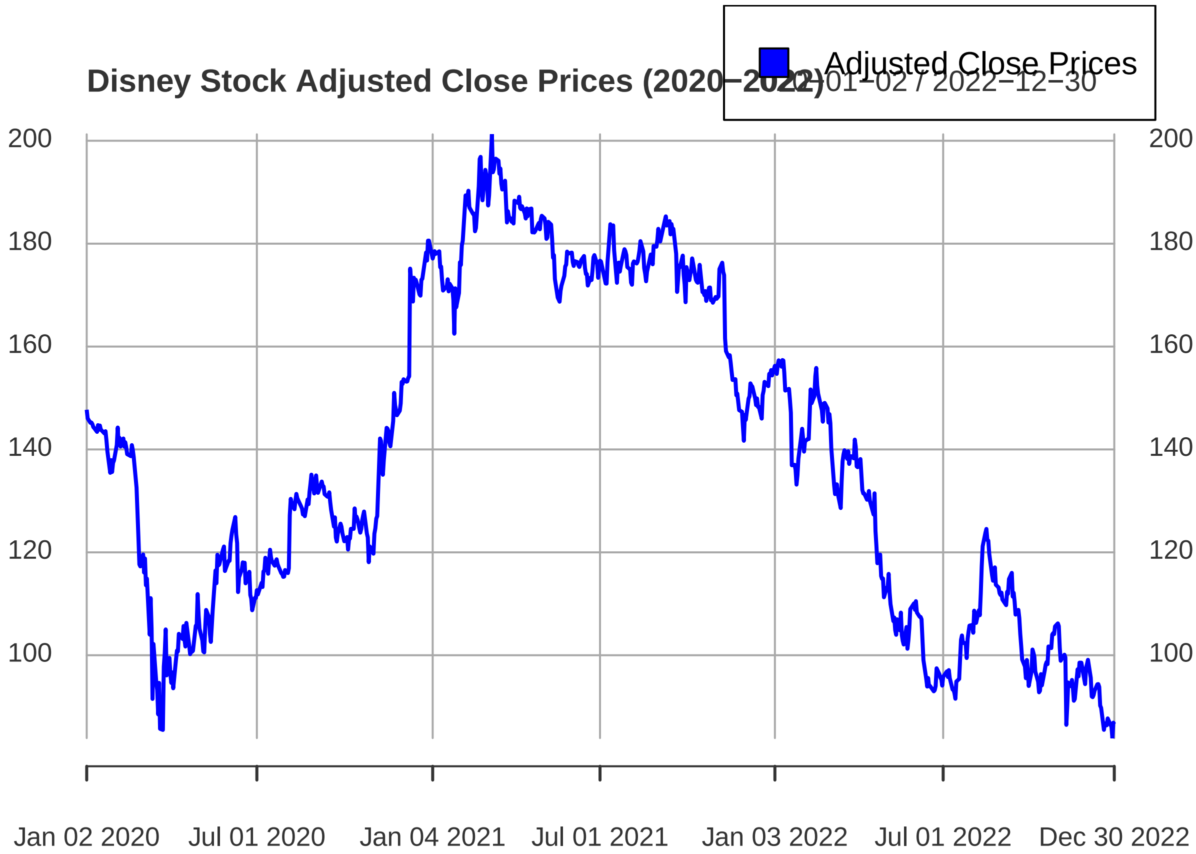
```r
library(quantmod)
library(aTSA)
library(forecast)
library(stats)
library(tseries)
library(timeDate)
library(TSA)
library(PerformanceAnalytics)
```

PROBLEM 1

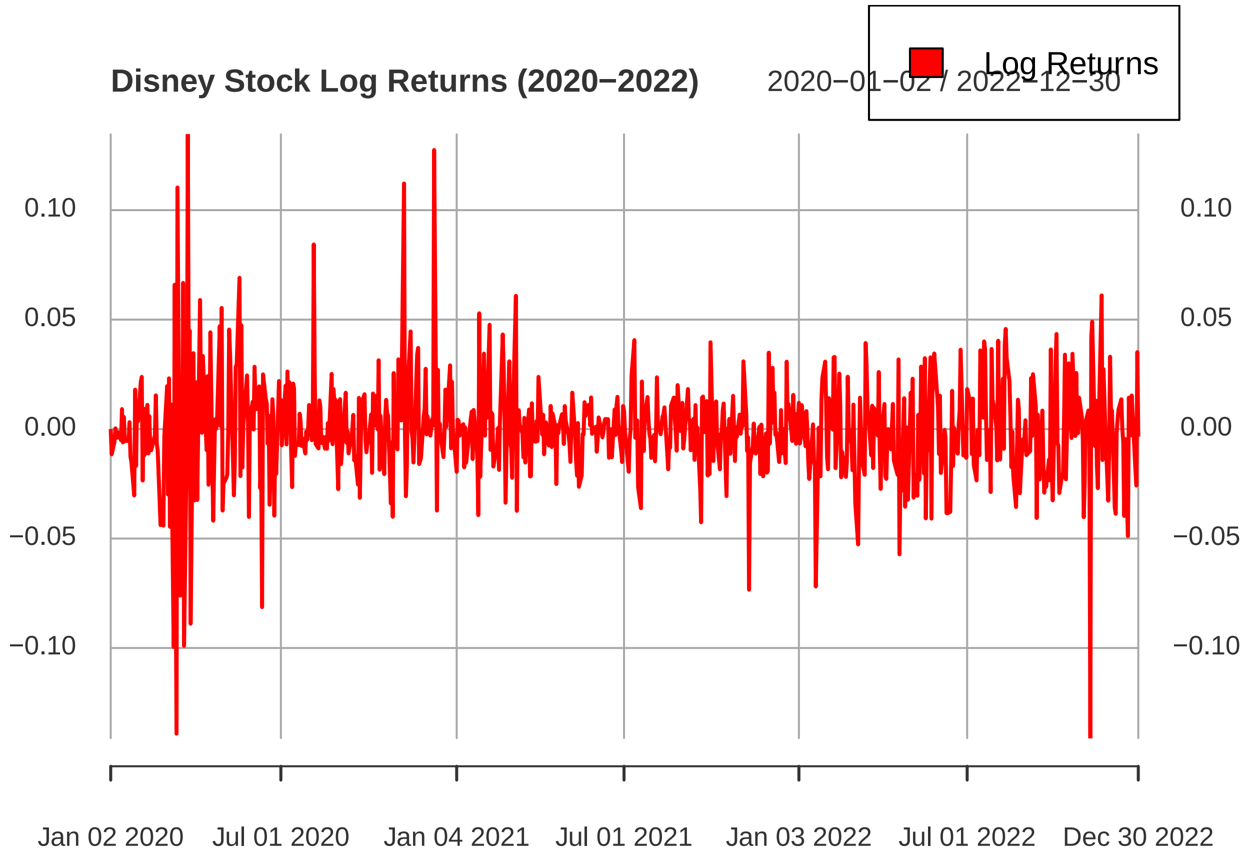
```r
getSymbols(c("DIS"), from = "2020-01-01", to = "2022-12-31")
```

```
## [1] "DIS"
```

```r
d_return <- dailyReturn(DIS$DIS.Adjusted, type = "log")

par(xpd=TRUE, mar = par()$mar + c(0, 0, 0, 3))

# First plot: Disney Stock Adjusted Close Prices
plot(Ad(DIS), main = "Disney Stock Adjusted Close Prices (2020-2022)", col = "blue")
legend("topright", legend = "Adjusted Close Prices", fill = "blue")
```

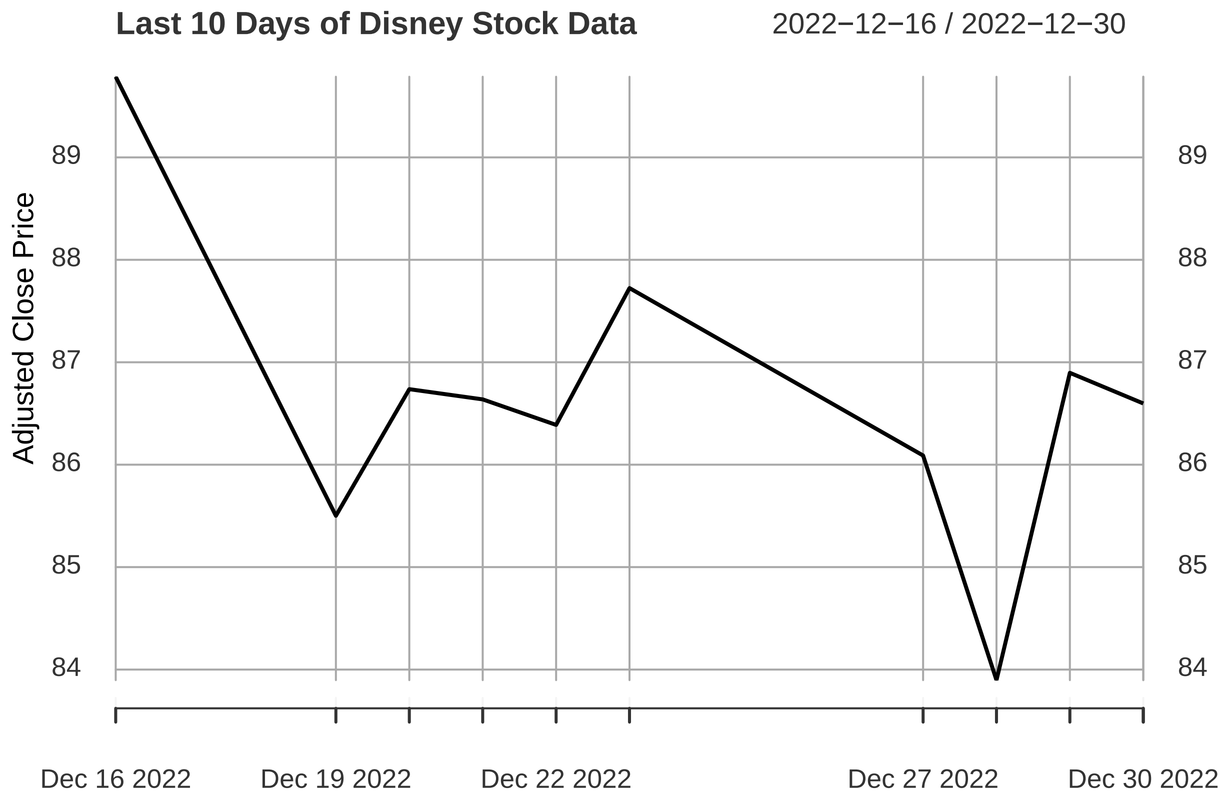



```r
# Adjusting margins for the legend
par(xpd=TRUE, mar = par()$mar + c(0, 0, 0, 3))

# Second plot: Disney Stock Log Returns
plot(d_return, main = "Disney Stock Log Returns (2020-2022)", col = "red")
legend("topright", legend = "Log Returns", fill = "red")
```

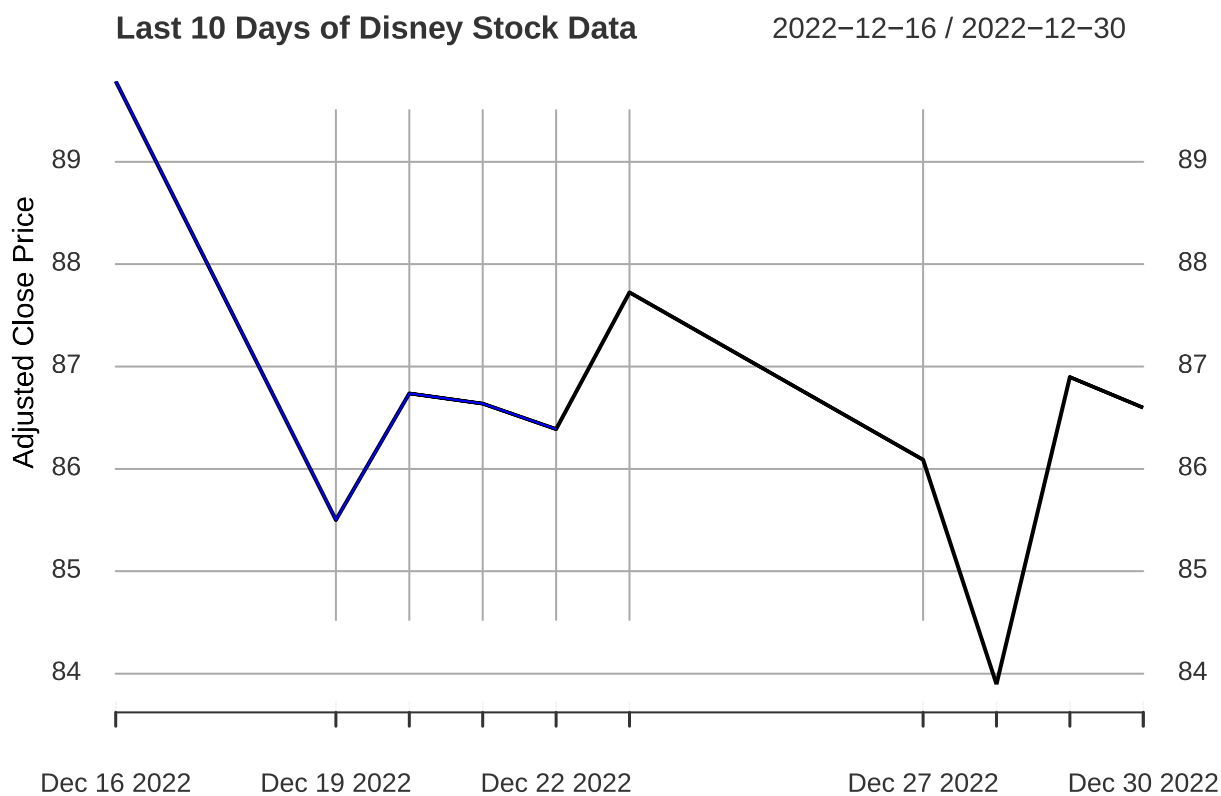



PROBLEM 2

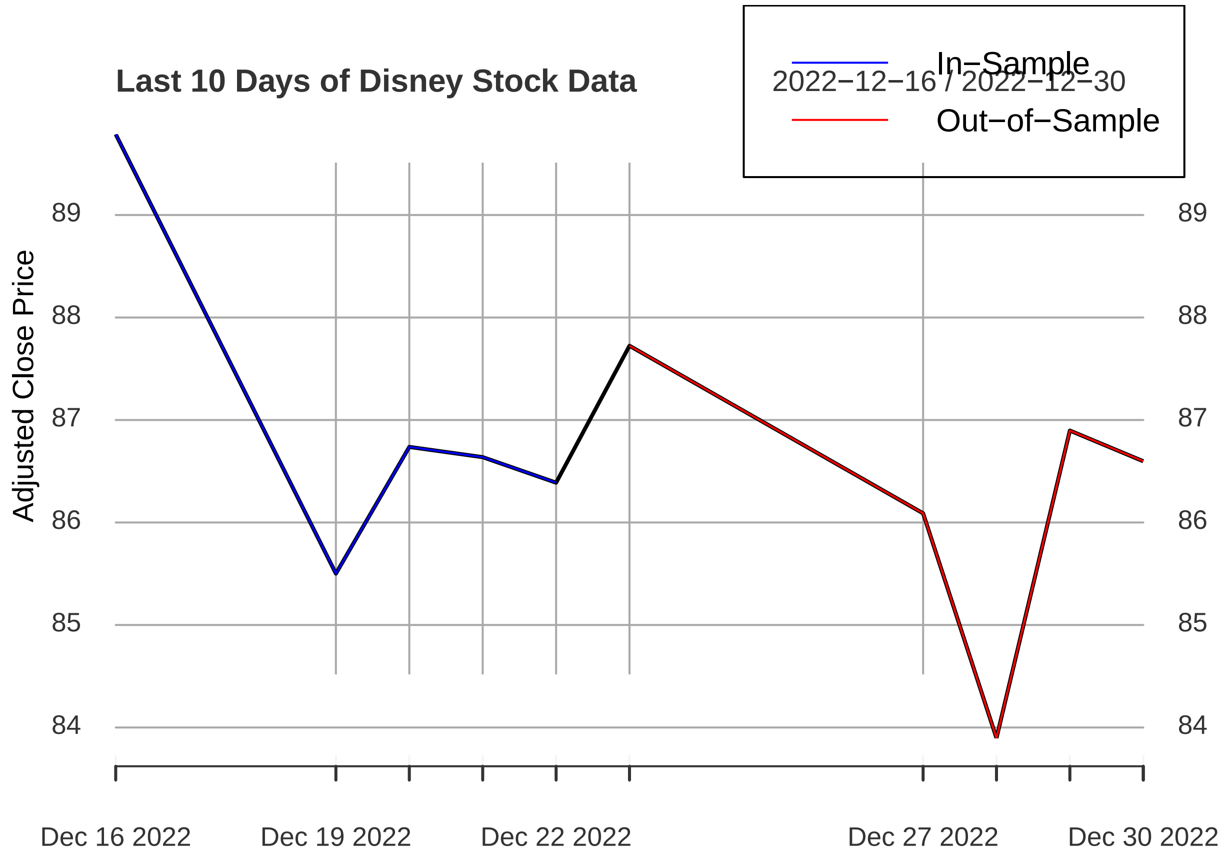
```
fin_index <- nrow(DIS)
out_data <- DIS[(fin_index - 4):fin_index, ]
in_data <- DIS[1:(fin_index - 5), ]
last_10 <- DIS[(fin_index - 9):fin_index, ]

par(mar=c(5, 4, 4, 8) + 0.1)
plot(last_10$DIS.Adjusted, type = "l", col = "black", main = "Last 10 Days of Disney Stock Data", xlab =
```




```r
lines(last_10$DIS.Adjusted[1:5], col = "blue")
```

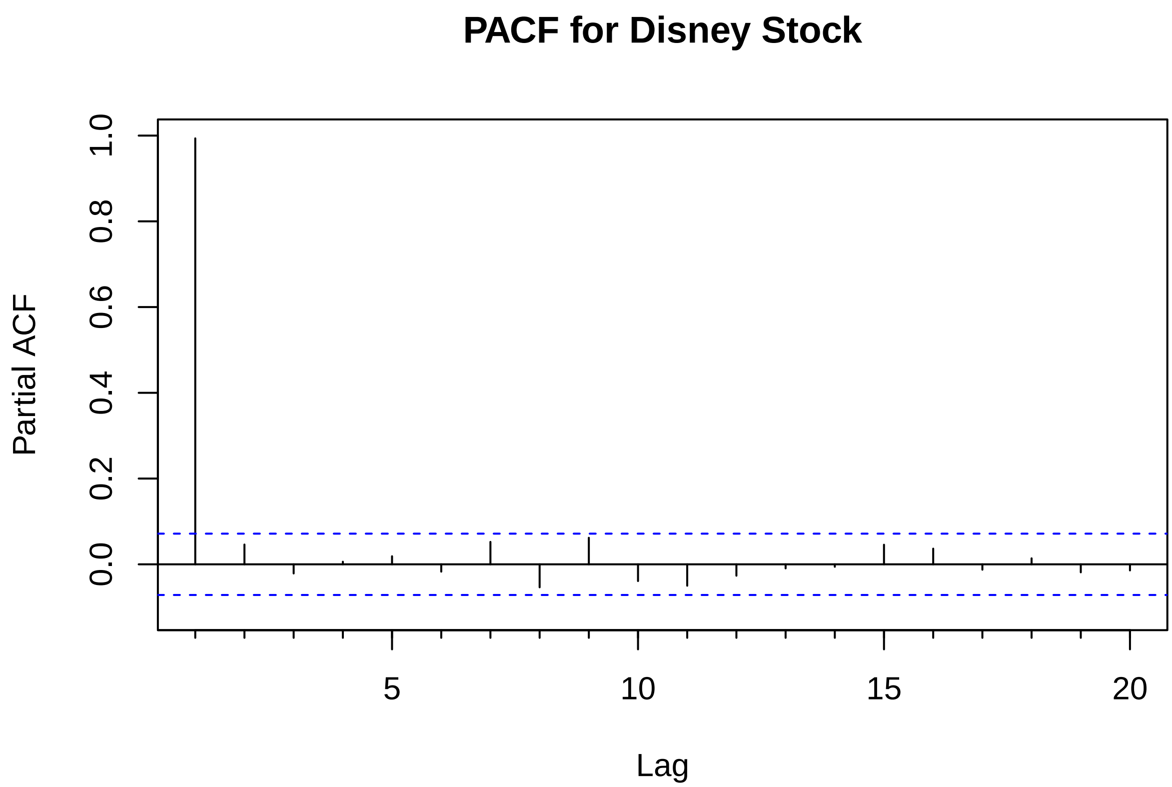



```
lines(last_10$DIS.Adjusted[6:10], col = "red")

par(xpd=TRUE, mar = par()$mar + c(0, 0, 0, 3))

legend("topright",
       legend = c("In-Sample", "Out-of-Sample"),
       col = c("blue", "red"),
       lty = 1)
```




PROBLEM 3 AR PART

```
#This step ensures I'm using the in-sample data
insmp_data <- Ad(DIS["/2022-12-22"])

# Since we are doing the AR model first, I plotted the PACF to look for
#potential recommended lag orders I can use.
Pacf(insmp_data, lag.max = 20, main = "PACF for Disney Stock")
```




```r
#Due to the only significant lag being 1, I then moved on to check if the series is stationary

#I then conducted a Dickey-Fuller test to see if the data is stationary,
#and if isn't, further steps have to be taken.
adf.test(insmp_data, alternative = "stationary")
```

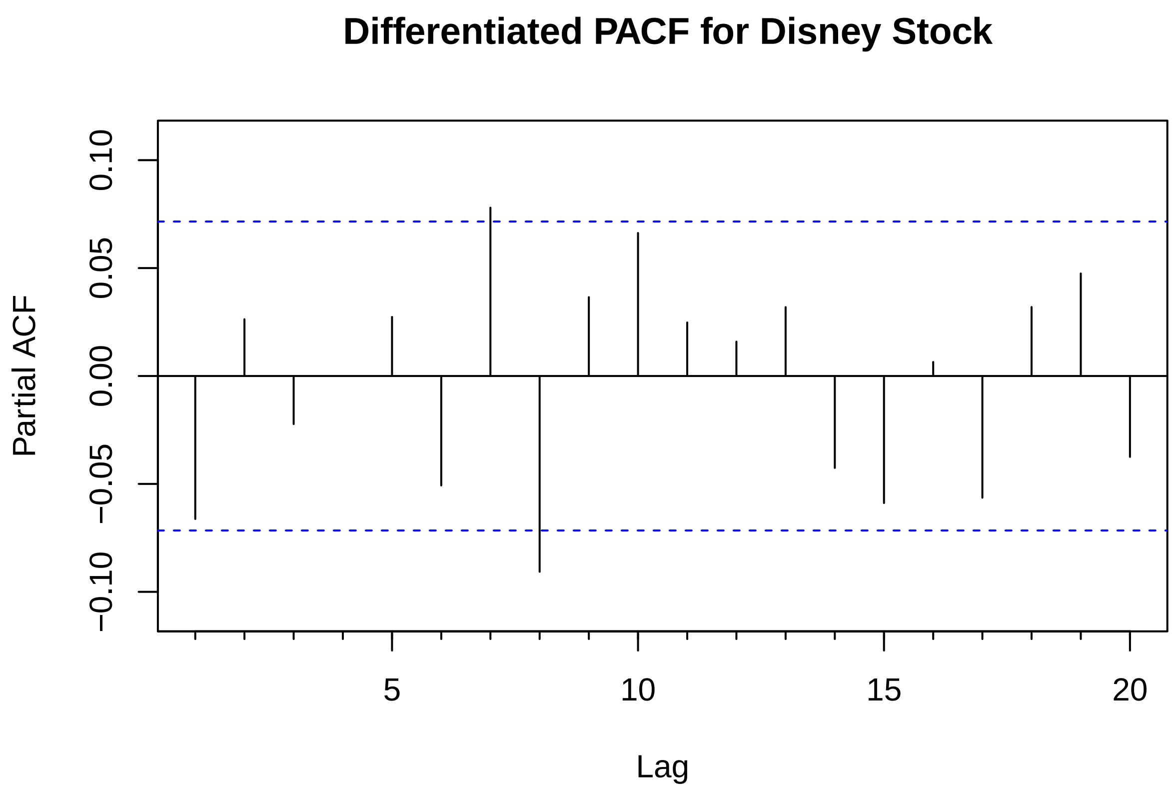
```
##
##  Augmented Dickey-Fuller Test
##
## data:  insmp_data
## Dickey-Fuller = -0.86672, Lag order = 9, p-value = 0.9557
## alternative hypothesis: stationary
```

```r
#H0: This series is non-stationary.
#H1: This series is stationary.
#With a p-value of 0.9, it's safe to conclude that we fail to reject the null
#hypothesis, thus this series is non-stationary.

#Since series is non-stationary, I have to differentiate it
#and omit the "NAs" for the next step.
insmp_diff <- diff(insmp_data)
insmp_diff <- na.omit(insmp_diff)

# Checks stationarity again
adf.test(insmp_diff, alternative = "stationary")
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  insmp_diff
## Dickey-Fuller = -8.1043, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
#H0: This series is non-stationary.
#H1: This series is stationary.
#With a p-value of 0.01 now, we can reject the null hypothesis that the series
#is non-stationary, thus suggesting that the series is stationary. This conclusion
#is also corroborated with the negative Dickey-Fuller value of -8.7537.

#Now that the series is stationary, let us redo the PACF of it to see if
#we can now find the recommended AR model
Pacf(insmp_diff, lag.max = 20, main = " Differentiated PACF for Disney Stock")
```




```r
#Looking at the new plot, I would say that the recommended order for the AR
#model would be 8.

#I then fitted the model based on AR(8) I determined from the
#new PACF using the ARIMA function.
```

```r
ar_ts <- ts(insmp_diff, frequency = 1)
ar_arima <- arima(ar_ts, order=c(8,0,0))
ar_arima
```

```
##
## Call:
## arima(x = ar_ts, order = c(8, 0, 0))
##
## Coefficients:
##           ar1     ar2      ar3     ar4     ar5      ar6     ar7      ar8
##       -0.0513  0.0194  -0.0223  0.0053  0.0200  -0.0437  0.0724  -0.0903
## s.e.   0.0363  0.0363   0.0362  0.0363  0.0362   0.0362  0.0362   0.0363
##       intercept
##         -0.0813
## s.e.     0.0997
##
## sigma^2 estimated as 8.861:  log likelihood = -1882.41,  aic = 3782.81
```

```r
#I then tested the models using the Yule-Walker and ols methods to
#compare coefficients and recommended order of each.
ar_yw <- ar(ar_ts, method = "yule-walker")
ar_ols <- ar(ar_ts, method = "ols")

ar_yw
```

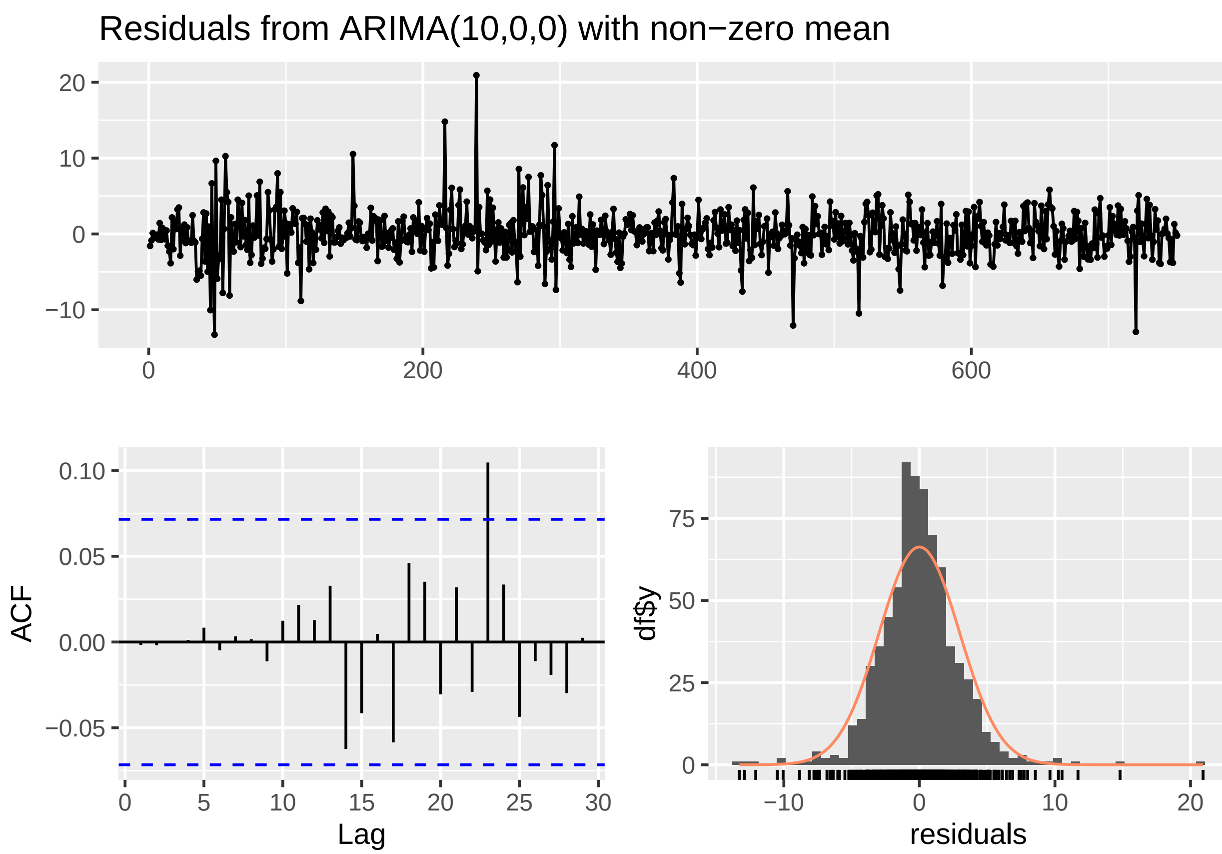
```
##
## Call:
## ar(x = ar_ts, method = "yule-walker")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -0.0506   0.0228  -0.0255   0.0075   0.0184  -0.0432   0.0733  -0.0899
##       9       10
##  0.0397   0.0663
##
## Order selected 10  sigma^2 estimated as  8.943
```

```r
ar_ols
```

```
##
## Call:
## ar(x = ar_ts, method = "ols")
##
## Coefficients:
##       1
## -0.0662
##
## Intercept: 0.002168 (0.1099)
##
## Order selected 1  sigma^2 estimated as  9.04
```

```r
#Looking at the results, the difference in the two are pretty stark, with the
#yule-walker recommending an order of 10, while the ols recommending an order of 1.
#Because of this difference, I moved on to using the AIC criterion to
#finding the recommended order

#Sets up variables that will be utilized soon.
best_aic <- Inf
best_model <- NULL
best_order <- NULL

# I cycled AR(1) - AR(10) to see which in this range is the best order
for (p in 1:10) {
    # This fits the ARIMA model while using try to catch errors
    fit <- try(Arima(insmp_diff, order=c(p,0,0)), silent=TRUE)

    # Check if the fit was successful
    if (!inherits(fit, "try-error")) {
        model_aic <- AIC(fit)
        if (model_aic < best_aic) {
            best_aic <- model_aic
            best_model <- fit
            best_order <- p
        }
    }
}

#This gives results
if (!is.null(best_model)) {
    cat("Best ARIMA model order is ARIMA(", best_order,",0,0) with AIC:", best_aic, "\n")
    print(summary(best_model))
    checkresiduals(best_model)
} else {
    cat("No recommended model was found.\n")
}
```

```
## Best ARIMA model order is ARIMA( 10 ,0,0) with AIC: 3784.555
## Series: insmp_diff
## ARIMA(10,0,0) with non-zero mean
##
## Coefficients:
##           ar1     ar2      ar3     ar4     ar5      ar6     ar7      ar8
##       -0.0504  0.0227  -0.0254  0.0074  0.0184  -0.0432  0.0730  -0.0894
## s.e.   0.0364  0.0364   0.0363  0.0362  0.0361   0.0361  0.0361   0.0362
##          ar9    ar10     mean
##       0.0397  0.0656  -0.0824
## s.e.  0.0363  0.0363   0.1103
##
## sigma^2 = 8.942:  log likelihood = -1880.28
## AIC=3784.55   AICc=3784.98   BIC=3840
##
## Training set error measures:
##                        ME     RMSE     MAE MPE MAPE      MASE         ACF1
## Training set 0.0002812029 2.968248 2.10485 Inf  Inf 0.6766825 -0.001735798
```

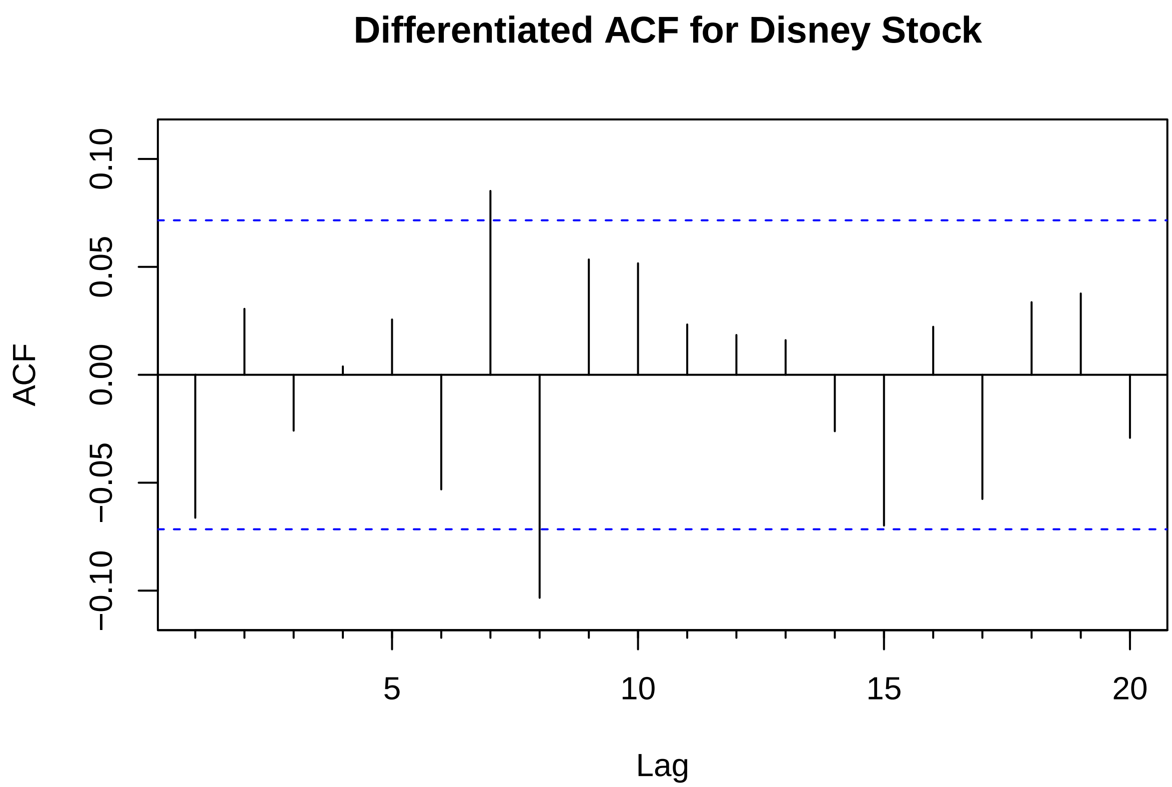



```
##
## 	Ljung-Box test
##
## data:  Residuals from ARIMA(10,0,0) with non-zero mean
## Q* = 1.6087, df = 3, p-value = 0.6574
##
## Model df: 10.   Total lags used: 13
```

After conducting the yule-walker, ols, and AIC criterion test on the AR model of the series, I can conclude that the recommended AR model would be AR(10), due to the yule-walker and AIC criterion test producing comparable coefficients, the same sigma^2, and same recommended order.

PROBLEM 3 MA PART

```
#Due to using the stationarity test before, I already knew the series had to be
#differentiated, so I used the differentiated data here when calculating the ACF to save time.
Acf(insmp_diff, lag.max = 20, main = " Differentiated ACF for Disney Stock")
```




```r
#Looking at the graph, the areas of interest to me are at lag 7, 8, and 15.

arima(insmp_diff, order = c(0,0,7))
```

```
##
## Call:
## arima(x = insmp_diff, order = c(0, 0, 7))
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5      ma6     ma7  intercept
##       -0.0432  0.0099  -0.0322  0.0129  0.0113  -0.0478  0.0891    -0.0823
## s.e.   0.0371  0.0372   0.0376  0.0410  0.0349   0.0372  0.0397     0.1092
##
## sigma^2 estimated as 8.944:  log likelihood = -1885.88,  aic = 3787.76
```

```r
arima(insmp_diff, order = c(0,0,8))
```

```
##
## Call:
## arima(x = insmp_diff, order = c(0, 0, 8))
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5      ma6     ma7      ma8
##       -0.0423  0.0305  -0.0293  0.0171  0.0190  -0.0594  0.0835  -0.1038
## s.e.   0.0360  0.0370   0.0370  0.0370  0.0337   0.0363  0.0390   0.0382
```

```
##       intercept
##         -0.0812
## s.e.     0.0995
##
## sigma^2 estimated as 8.856:  log likelihood = -1882.21,  aic = 3782.42
```

```r
arima(insmp_diff, order = c(0,0,15))
```

```
##
## Call:
## arima(x = insmp_diff, order = c(0, 0, 15))
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5      ma6     ma7      ma8
##       -0.0532  0.0229  -0.0308  0.0230  0.0262  -0.0510  0.0878  -0.0959
## s.e.   0.0365  0.0364   0.0366  0.0366  0.0371   0.0369  0.0367   0.0374
##          ma9    ma10    ma11    ma12    ma13     ma14     ma15  intercept
##       0.0540  0.0658  0.0073  0.0102  0.0073  -0.0604  -0.0365    -0.0808
## s.e.  0.0366  0.0381  0.0389  0.0397  0.0404   0.0379   0.0390     0.1055
##
## sigma^2 estimated as 8.742:  log likelihood = -1877.42,  aic = 3786.84
```

```r
#Looking strictly at the AIC coefficients, MA(8) would be my preferred choice,
#however I ran a Box-Ljung test to ensure there is no autocorrelation in the MA order.

#Fitted all the data to their respective orders to test
model_ma7 <- arima(insmp_diff, order = c(0,0,7))
model_ma8 <- arima(insmp_diff, order = c(0,0,8))
model_ma15 <- arima(insmp_diff, order = c(0,0,15))

#Ran a Ljung-Box test to check to see if there is any auto-correlation aka
#if the model residuals are normally distributed to ensure there is random noise.
Box.test(residuals(model_ma7), type="Ljung-Box", lag=10)
```

```
##
##  Box-Ljung test
##
## data:  residuals(model_ma7)
## X-squared = 12.203, df = 10, p-value = 0.2717
```

```r
Box.test(residuals(model_ma8), type="Ljung-Box", lag=10)
```

```
##
##  Box-Ljung test
##
## data:  residuals(model_ma8)
## X-squared = 5.142, df = 10, p-value = 0.8815
```

```r
Box.test(residuals(model_ma15), type="Ljung-Box", lag=10)
```

```
##
```

```
##  Box-Ljung test
##
## data:  residuals(model_ma15)
## X-squared = 0.035294, df = 10, p-value = 1
```

```r
#After looking at the p-values, I have two options to choose from, which are either
#MA(8) or MA(15), with MA(8) having the better AIC with a p-value of 0.88, while
#MA(15) has a slightly worse AIC but a p-value of 1. Ultimately, I chose MA(8)
#because it requires less data to have an accurate result.
```

PROBLEM 3 ARMA PART

```r
#I used the AIC criterion for the ARMA just like I did with the MA and AR models
#This specific AIC criterion is referenced from the recitation to find the best ARMA model
aic.matrix <- function(data, ar_order, ma_order)
{
  AIC_matrix <- matrix(NA, nrow = ar_order+1, ncol = ma_order+1)
  for(i in 0 : ar_order)
  {
    for(j in 0 : ma_order)
    {
      tem <- tryCatch(arima(data, order = c(i, 0, j))$aic,
                      error = function(cond)
                      {
                        if(grepl("non-stationary AR part", cond$message)) {
                          message("Non-stationary AR part detected for AR:", i, "; MA:", j)
                          return(NA)
                        } else {
                          stop(cond)
                        }
                      }
      )
      AIC_matrix[i+1, j+1] <- tem
    }
  }
  AIC_matrix
}

#The range for the max AR and MA order is 10 and 10 in this instance
matrix <- aic.matrix(insmp_diff, 10, 10)
which(matrix == min(na.omit(matrix)), arr.ind = TRUE) - 1
```

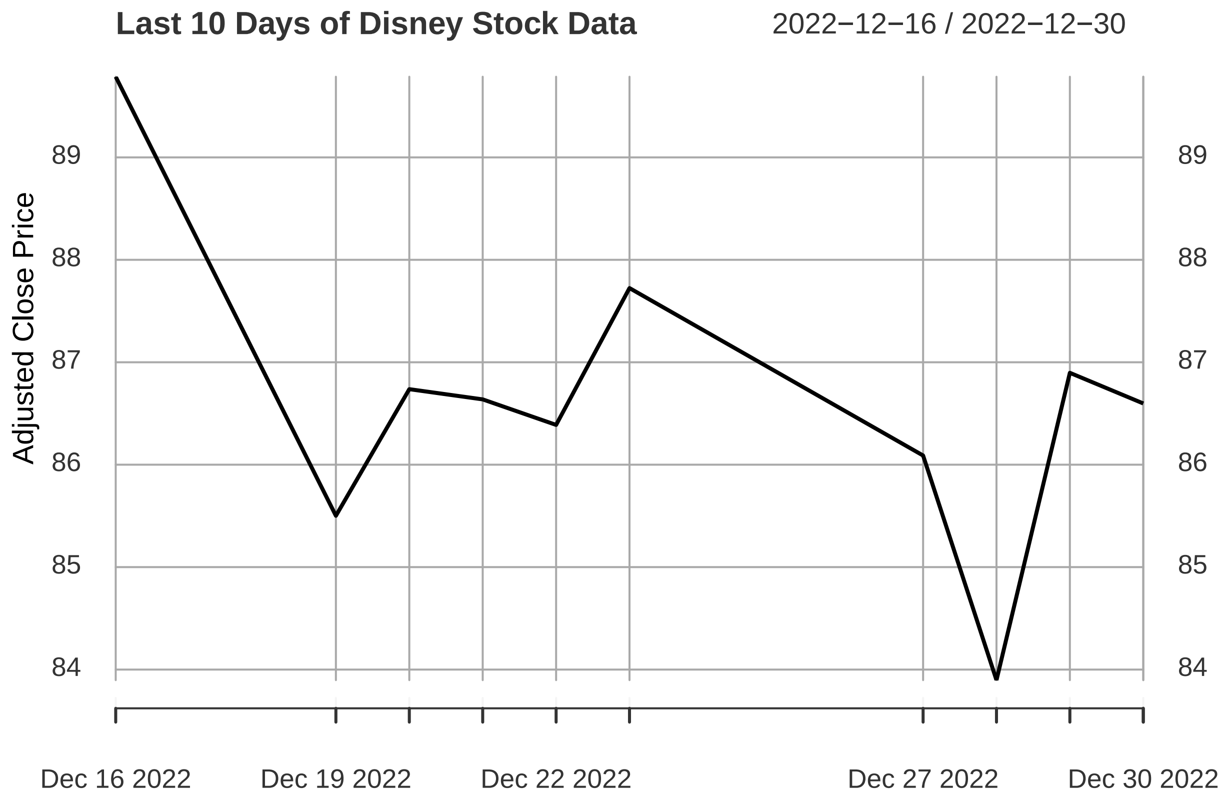
```
##      row col
## [1,]   5   7
```

Using the AIC method, the recommended ARMA model to use would be ARMA(5, 7).

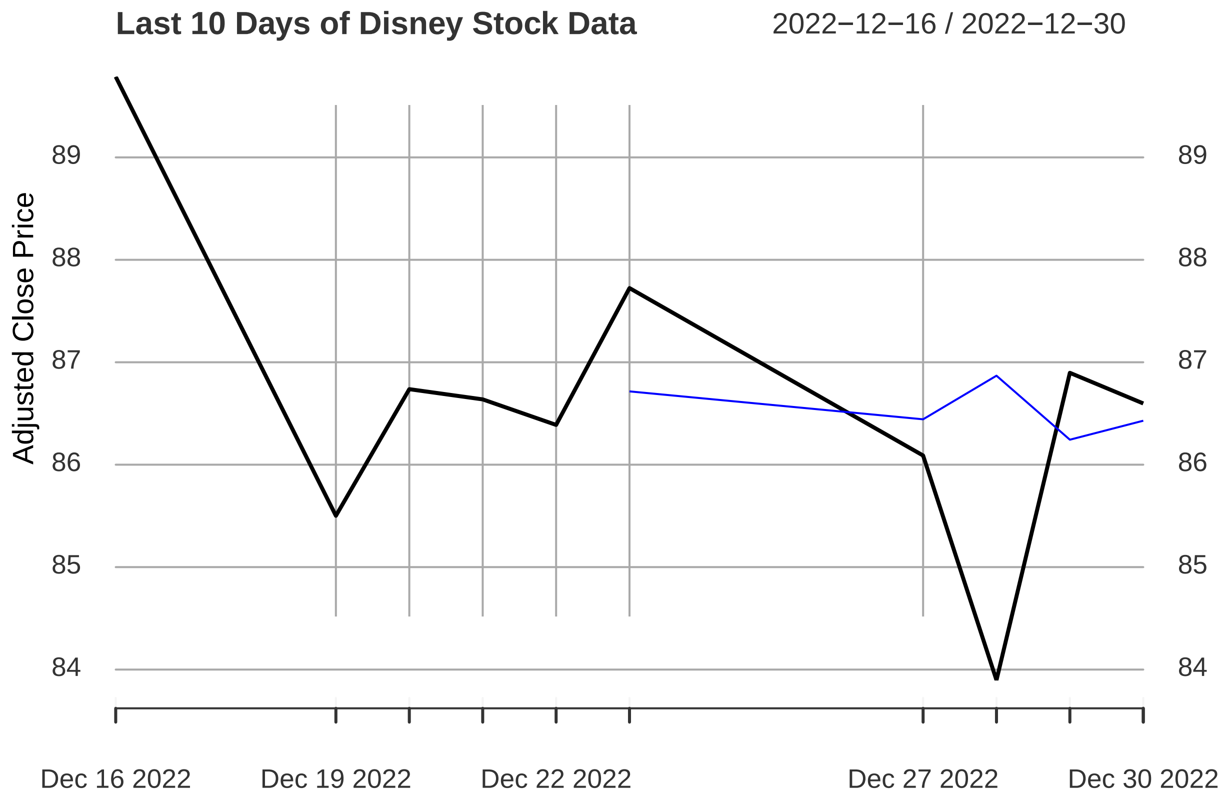
PROBLEM 4

```r
#These are the fitted AR, MA, and ARMA models respectively.
ar_model <- Arima(insmp_diff, order=c(10,0,0))
ma_model <- Arima(insmp_diff, order=c(0,0,8))
arma_model <- Arima(insmp_diff, order=c(5,0,7))
```

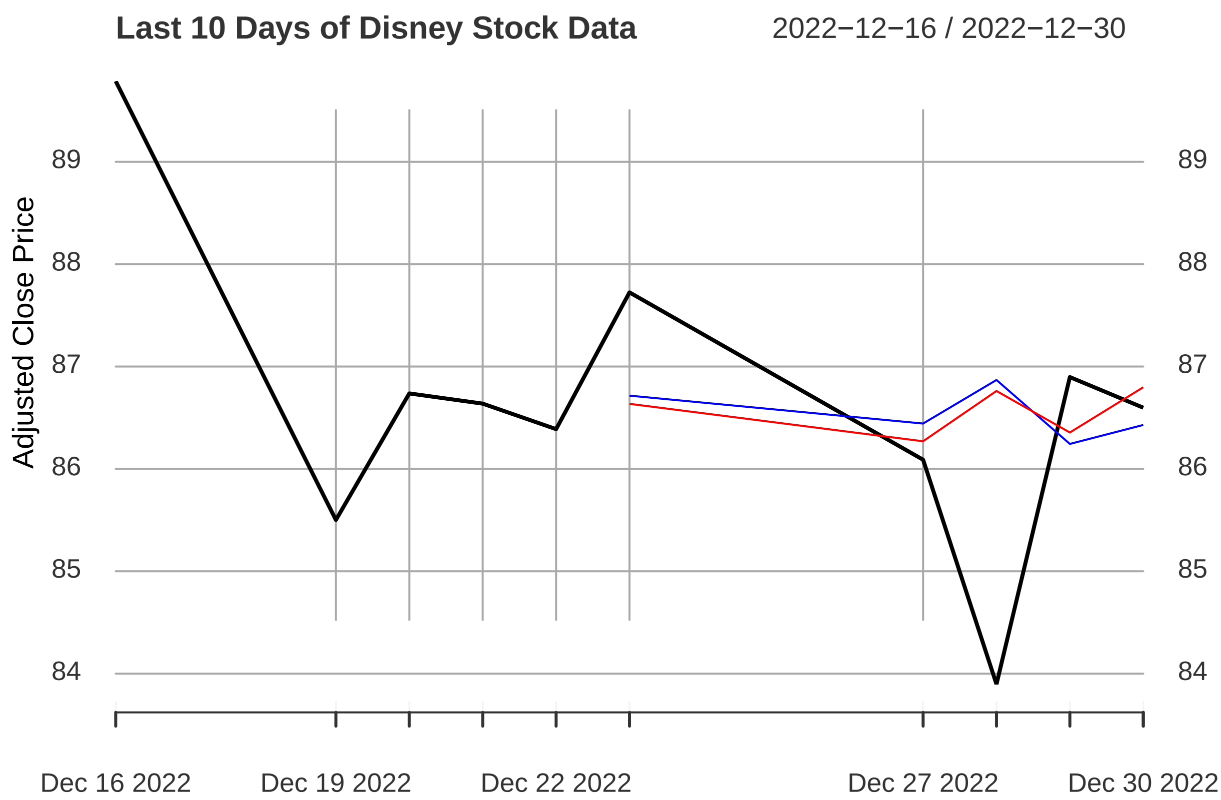
```r
#This is their predicted values 5 days into the future.
ar_pred <- predict(ar_model, n.ahead = 5)
ma_pred <- predict(ma_model, n.ahead = 5)
arma_pred <- predict(arma_model, n.ahead = 5)

#Using the timeDate library, I used this to first look 30 days ahead in order to
#safely determine the next 5 business days
#The NYSE holidays functions finds the holidays that the NYSE has off to
#accurately reflect the dates of the stock data being predicted.
#The time sequence function acts as a filter to only keep the NYSE business
#Lastly, only picks the 1-5 business day dates that will be used in the next function cluster.
last_date <- index(insmp_diff)[length(insmp_diff)]
end_date <- as.Date(last_date) + 30
nyse_holidays <- holidayNYSE(2022)
all_days <- timeSequence(from = as.Date(last_date) + 1, to = end_date, by = "day")
business_days <- all_days[isBizday(all_days, holidays = nyse_holidays)]
next_business_days <- business_days[1:5]

#This appends the data from the prediction set to their corresponding date,
#allowing them to be graphed with the traditional closing prices.
ar_updated <- xts(ar_pred$pred, order.by = next_business_days, dimnames=list(NULL, "pred"))
ma_updated <- xts(ma_pred$pred, order.by = next_business_days, dimnames=list(NULL, "pred"))
arma_updated <- xts(arma_pred$pred, order.by = next_business_days, dimnames=list(NULL, "pred"))

#This function converts the daily percent changes of the predictions to the
#predicted adjusted closing price of Disney's stock in the next 5 days (ex: $88.25)
last_price <- last_10$DIS.Adjusted[4]
ar_prices <- cumsum(c(last_price, ar_updated$pred))[-1]
ma_prices <- cumsum(c(last_price, ma_updated$pred))[-1]
arma_prices <- cumsum(c(last_price, arma_updated$pred))[-1]


par(mar=c(5, 4, 4, 8) + 0.1)
plot(last_10$DIS.Adjusted, type = "l", col = "black", main = "Last 10 Days of Disney Stock Data", xlab =
```



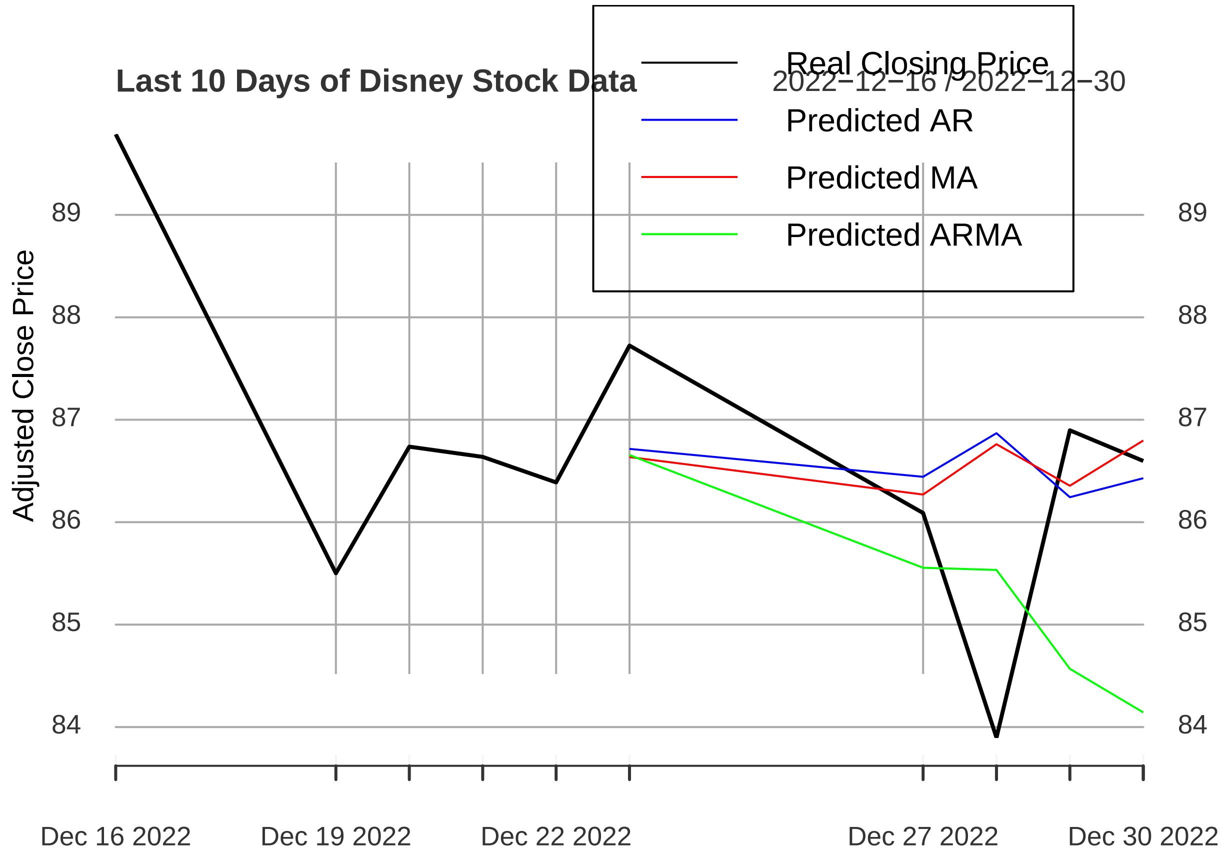

```r
lines(ar_prices, col = "blue")
```




```r
lines(ma_prices, col = "red")
```




```r
lines(arma_prices, col = "green")

# Adjusting margins for the legend
par(xpd=TRUE, mar = par()$mar + c(0, 0, 0, 3))

legend("topright", inset=c(0.1, 0),
       legend = c("Real Closing Price", "Predicted AR", "Predicted MA", "Predicted ARMA"),
       col = c("black", "blue", "red", "green"),
       lty = 1)
```




PROBLEM 5

```
ar_sse <- sum((last_10$DIS.Adjusted[6:10] - ar_prices)^2)
ma_sse <- sum((last_10$DIS.Adjusted[6:10] - ma_prices)^2)
arma_sse <- sum((last_10$DIS.Adjusted[6:10] - arma_prices)^2)

ar_sse
```

```
## [1] 10.43016
```

```
ma_sse
```

```
## [1] 9.756237
```

```
arma_sse
```

```
## [1] 15.55213
```

Looking at the sum of squared errors, I would conclude that the MA model is the most accurate at 9.756, followed closely by the AR model at 10.430, and lastly the ARMA model at 15.552. Hence I chose the MA model. PROBLEM 6

```
#In order to calculate the percentage of time the MA model is correct, I need all my data to be in perc
#This function converts the adjusted close price to daily returns, putting them in the same form as my f
#I then removed the 22nd from the array to ensure that the length of both the MA and actual returns wil
```

```r
DIS_price <- last_10$DIS.Adjusted[5:10]
DIS_perchange <- dailyReturn(DIS_price['2022-12-22/2022-12-30'])
DIS_perchangefin <- DIS_perchange['2022-12-23/']

#This checks the signs of both the predicted and actual returns
#It then compares the two returns for each date in the arrays
#It returns 1 for a positive sign and -1 for a negative sign
#per_right calculates the % of correct predictions
pred_dtma <- sign(ma_updated)
act_dt <- sign(DIS_perchangefin)
correct_predma <- sum(pred_dtma == act_dt)
per_rightma <- (correct_predma / length(pred_dtma)) * 100

pred_dtar <- sign(ar_updated)
correct_predar <- sum(pred_dtar == act_dt)
per_rightar <- (correct_predar / length(pred_dtar)) * 100

pred_dtarma <- sign(arma_updated)
correct_predarma <- sum(pred_dtarma == act_dt)
per_rightarma <- (correct_predarma / length(pred_dtarma)) * 100

print(paste("MA Directional Prediction Success Rate:", per_rightma, "%"))
```

```
## [1] "MA Directional Prediction Success Rate: 20 %"
```

```r
print(paste("AR Directional Prediction Success Rate:", per_rightar, "%"))
```

```
## [1] "AR Directional Prediction Success Rate: 40 %"
```

```r
print(paste("ARMA Directional Prediction Success Rate:", per_rightarma, "%"))
```

```
## [1] "ARMA Directional Prediction Success Rate: 80 %"
```

Looking at MA's Directional Prediction success rate of 20%, it didn't do the best. However,it is important to note that when testing the other two models' directional success rate, the AR's success rate was 40%, while the ARMA's success rate was 80%. However, when looking at the graph and calculating the SSE for the models, the MA model was closest to the actual model's closing price values. While ARMA was the least accurate in predicting the adjusted close price it was successful in determining the general direction of the stock on a daily basis, lastly the AR model was the middle-ground between the two.

INTERVIEW PROBLEMS

PROBLEM 1 Data1 PART

```r
data1 <- read.csv("C:/Users/Laurent/Downloads/data1.csv")
data2 <- read.csv("C:/Users/Laurent/Downloads/data2.csv")
data3 <- read.csv("C:/Users/Laurent/Downloads/data3.csv")

vec1 <- as.numeric(data1$data)
vec2 <- as.numeric(data2$data)
vec3 <- as.numeric(data3$data)
```

```r
ts1 <- ts(vec1)
ts2 <- ts(vec2)
ts3 <- ts(vec3)

adf.test(ts1, alternative = "stationary")
```

```
## Warning in adf.test(ts1, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts1
## Dickey-Fuller = -8.4506, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(ts2, alternative = "stationary")
```

```
## Warning in adf.test(ts2, alternative = "stationary"): p-value smaller than
## printed p-value
```

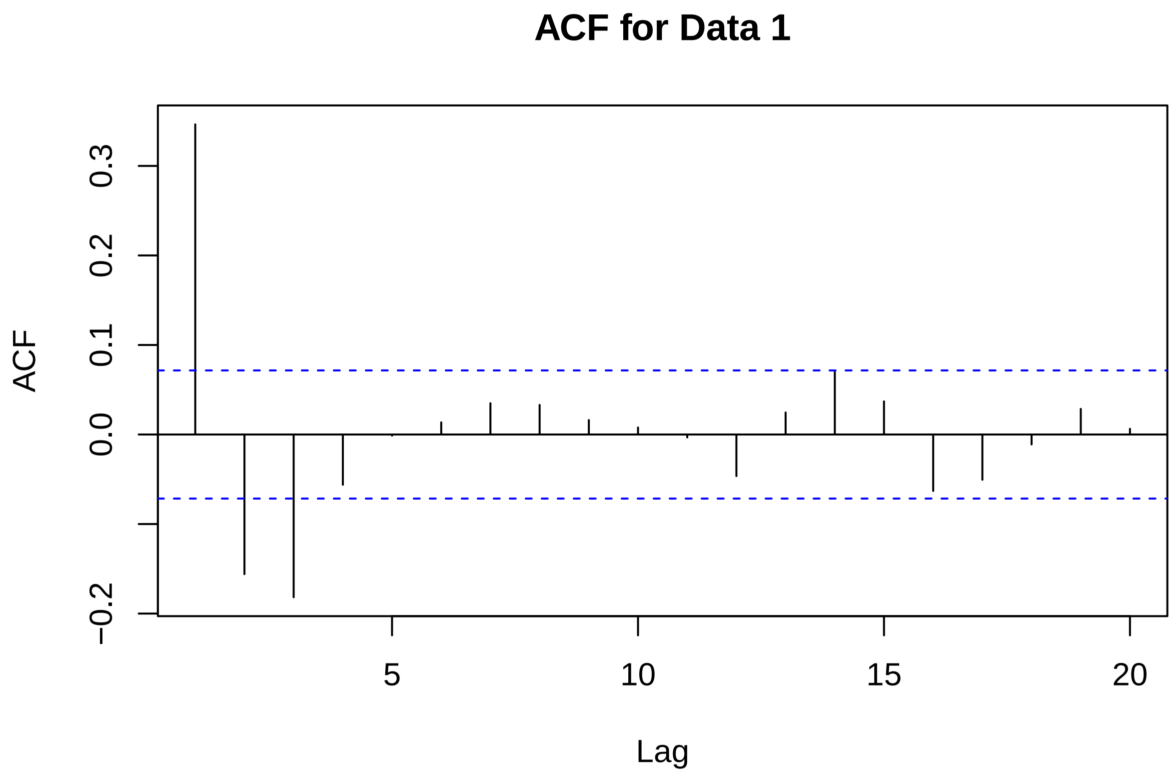
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts2
## Dickey-Fuller = -8.7559, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(ts3, alternative = "stationary")
```

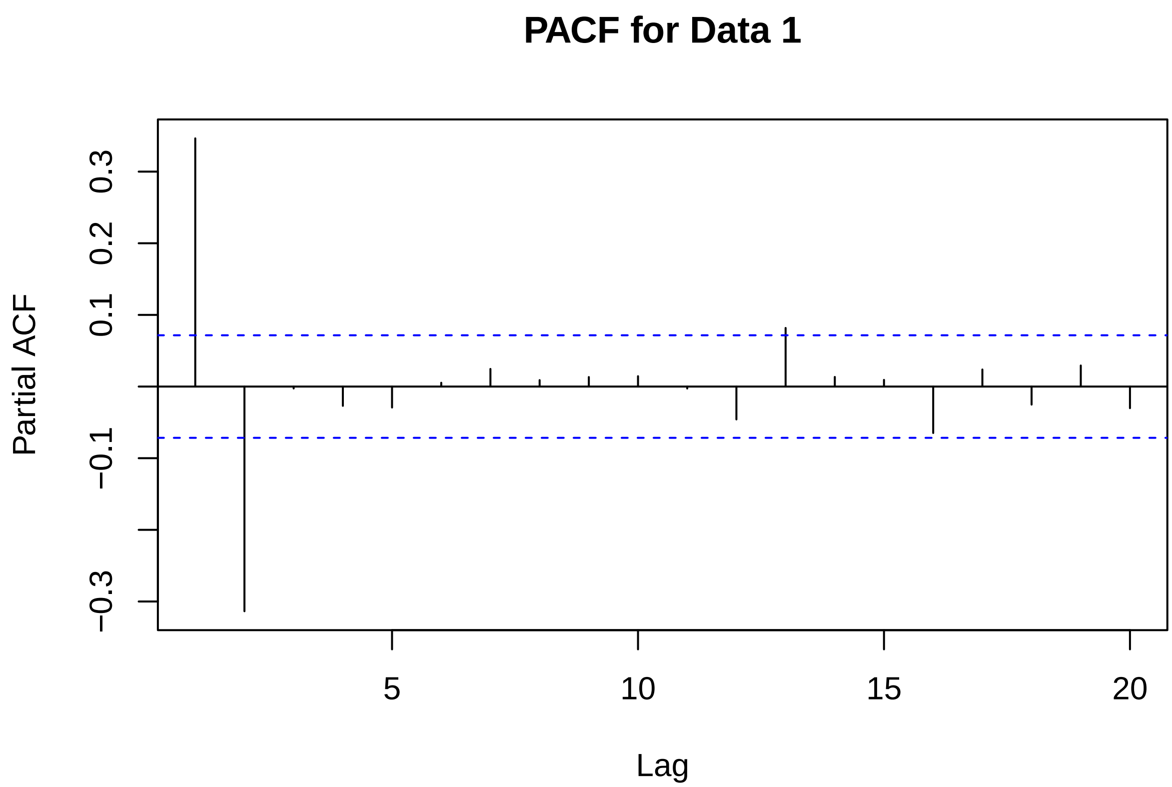
```
## Warning in adf.test(ts3, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts3
## Dickey-Fuller = -8.286, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf(ts1, lag.max = 20, main = "ACF for Data 1")
```




```r
pacf(ts1, lag.max = 20, main = "PACF for Data 1")
```




```r
#Looking at the PACF, recommended AR order is 1 or 2
#Looking at the ACF, recommended MA order is 1 or 3

arma1 <- Arima(ts1, order=c(1,0,0))
rec_arma <- Arima(ts1, order = c(1, 0, 1))
rec_arma2 <- Arima(ts1, order = c(1, 0, 3))
rec_arma3 <- Arima(ts1, order = c(2, 0, 1))
rec_arma4 <- Arima(ts1, order = c(2, 0, 3))

print(AIC(arma1))
```

```
## [1] -4723.637
```

```r
print(AIC(rec_arma))
```

```
## [1] -4776.206
```

```r
print(AIC(rec_arma2))
```

```
## [1] -4795.348
```

```r
print(AIC(rec_arma3))
```

```
## [1] -4797.335
```

```r
print(AIC(rec_arma4))
```

```
## [1] -4794.161
```

Given that ARMA(2, 0, 1)'s AIC is the lowest, (2, 0, 1) is data1's recommended order to me. And since it is given the rest of the data has the same order and parameters, data2's and data3's recommended orders are also ARMA(2, 0, 1).

PROBLEM 2

```r
fit1 <- Arima(ts1, order=c(2,0,1))
fit2 <- Arima(ts2, order=c(2,0,1))
fit3 <- Arima(ts3, order=c(2,0,1))

summary(fit1)
```

```
## Series: ts1
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1   mean
##       0.4688  -0.3183  -0.0147  1e-04
## s.e.  0.1205   0.0525   0.1282  4e-04
##
## sigma^2 = 9.681e-05:  log likelihood = 2403.67
## AIC=-4797.33   AICc=-4797.25   BIC=-4774.23
##
## Training set error measures:
##                        ME        RMSE        MAE       MPE     MAPE      MASE
## Training set 2.693995e-06 0.009812955 0.00780437 -56.75598 350.3323 0.7866795
##                      ACF1
## Training set 0.0006309388
```

```r
summary(fit2)
```

```
## Series: ts2
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1    mean
##       0.5367  -0.3399  -0.0265  0.0010
## s.e.  0.0974   0.0487   0.1028  0.0022
##
## sigma^2 = 0.002569:  log likelihood = 1174.14
## AIC=-2338.29   AICc=-2338.21   BIC=-2315.19
##
## Training set error measures:
##                        ME       RMSE       MAE       MPE     MAPE      MASE
## Training set 2.129978e-06 0.05055341 0.0403589 -16.63057 332.7382 0.7825428
##                      ACF1
## Training set 0.0002109283
```

```r
summary(fit3)
```

```
## Series: ts3
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1     mean
##       0.4957  -0.3107  -0.0160  -0.0040
## s.e.  0.1066   0.0506   0.1115   0.0043
##
## sigma^2 = 0.009537:  log likelihood = 682.34
## AIC=-1354.68   AICc=-1354.6   BIC=-1331.58
##
## Training set error measures:
##                         ME       RMSE        MAE      MPE     MAPE      MASE
## Training set -5.353682e-05 0.09739776 0.07742859 75.31768 152.7177 0.7938319
##                       ACF1
## Training set 0.0004116471
```

PROBLEM 3

```r
ar1_mean <- (0.4688 + 0.5367 + 0.4957) / 3
ar2_mean <- (-0.3183 + -0.3399 + -0.3107) / 3
ma_mean <- (-0.0147 + -0.0265 + -0.0160) / 3
intercept_mean <- (1e-04 + 0.0010 + -0.0042) / 3
sig_mean <- (0.0001071 + 0.002879 + 0.01049) / 3

cat("Averaged Results for the Model Coefficients:\n")
```

```
## Averaged Results for the Model Coefficients:
```

```r
cat(sprintf("Average AR1 Coefficient: %f\n", ar1_mean))
```

```
## Average AR1 Coefficient: 0.500400
```

```r
cat(sprintf("Average AR2 Coefficient: %f\n", ar2_mean))
```

```
## Average AR2 Coefficient: -0.322967
```

```r
cat(sprintf("Average MA1 Coefficient: %f\n", ma_mean))
```

```
## Average MA1 Coefficient: -0.019067
```

```r
cat(sprintf("Average Intercept: %f\n", intercept_mean))
```

```
## Average Intercept: -0.001033
```

```r
cat(sprintf("Average Noise Variance: %f\n", sig_mean))
```

```
## Average Noise Variance: 0.004492
```

My best guess for the model coefficients are above, where I took the average coefficients for each of the models.