StockMarketSuccess

Dvir Blander and Corey Kozlovski

4/26/2019

R Markdown

Goal and Description of Project

Our goal for this project is to be able to predict stocks by combining four methods of forecasting and weighing each one based on their individual successes in order to create one method that can accurately predict each stock individually. More importantly, we want to learn and understand new methods of analyzing data and being able to apply them in real world situations. Specifically, we will look at the stock of "Disney", "Tesla", "Coca Cola" and based on our analysis, we will determine the future trends of these stocks. We will be using Time Series, Neural Networks, Linear Regression, and lastly our own qualitative analysis of current events in order to come to an aggregate conclusion for each stock.

Importing All the Packages and loading all the Libraries

```r
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

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

```r
library(TimeWarp)
library(xts)
library(forecast)
library(zoo)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library(neuralnet)
library(BBmisc)
```

```
##
## Attaching package: 'BBmisc'
```

```
## The following object is masked from 'package:base':
##
##     isFALSE
```

Getting the Tesla Data from the quantmod library and a plot of the closing values.

```
getSymbols(Symbols = "TSLA", auto.assign = TRUE)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "TSLA"
```

```
plot(TSLA$TSLA.Close)
```

Getting the Coca Cola Data from the quantmod library and a plot of the closing values

```
getSymbols(Symbols = "KO", auto.assign = TRUE)
```

```
## [1] "KO"
```

```r
plot(KO$KO.Close)
```

Getting the Disney Data from the quantmod library and a plot of the closing values.

```r
getSymbols(Symbols = "DIS", auto.assign = TRUE)
```

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

```r
plot(DIS$DIS.Close)
```

Description for Current Event Analysis

Lastly, we will be using a more qualitative form of analysis through the means of current events. This analysis is very different than the other three, but we think it will bring very useful insight and context to our predictions from the other models. As this is a more general analysis, it will be most useful in predicting overall trends in the company's future in regards to real-life situations. Some drawbacks include being too reactionary to current events, as that would impact our predictions greatly, and furthermore this method isn't scientifically based as the other three are, but despite these shortcomings, we believe the context it provides will be essential to our predictions.

Current events for Tesla: When it comes to Tesla and News, these two terms almost come hand in hand. Be it from Elon Musk's Twitter antics, their full line-up of S-3-X-Y cars, and announcements of a new semi-truck and sports car, it can be very hard to use current events in predicting Tesla's future. They have some amazing developments, including battery recycling and releasing new cars recently that are more budget friendly, but also face some sharp downfalls. These include potential SEC violations that Musk faces due to his antics, a slowing down of car manufacturing for Q1 of 2019, and having the electric car as a whole not fully developed and realized hurts them as well. Overall, it is hard to predict the future of Tesla based on current events, but within the context of society and the growth of electric vehicles, it can be safe to assume Tesla will grow.

Current events for Coca Cola: The current events for Coca Cola are as follows. There is going to be advertising for Coca Cola in the newest Star Wars movie, which may bring significant revenue for the company. It is so a recession-proof business that has a healthy dividend, source: https://www.fool.com/investing/2019/04/14/3-reasons-coca-cola-is-a-buy.aspx. Additionally, it is a product that many people want even though people may try for healthier alternatives. Thus, expect continuous growth from Coca Cola.

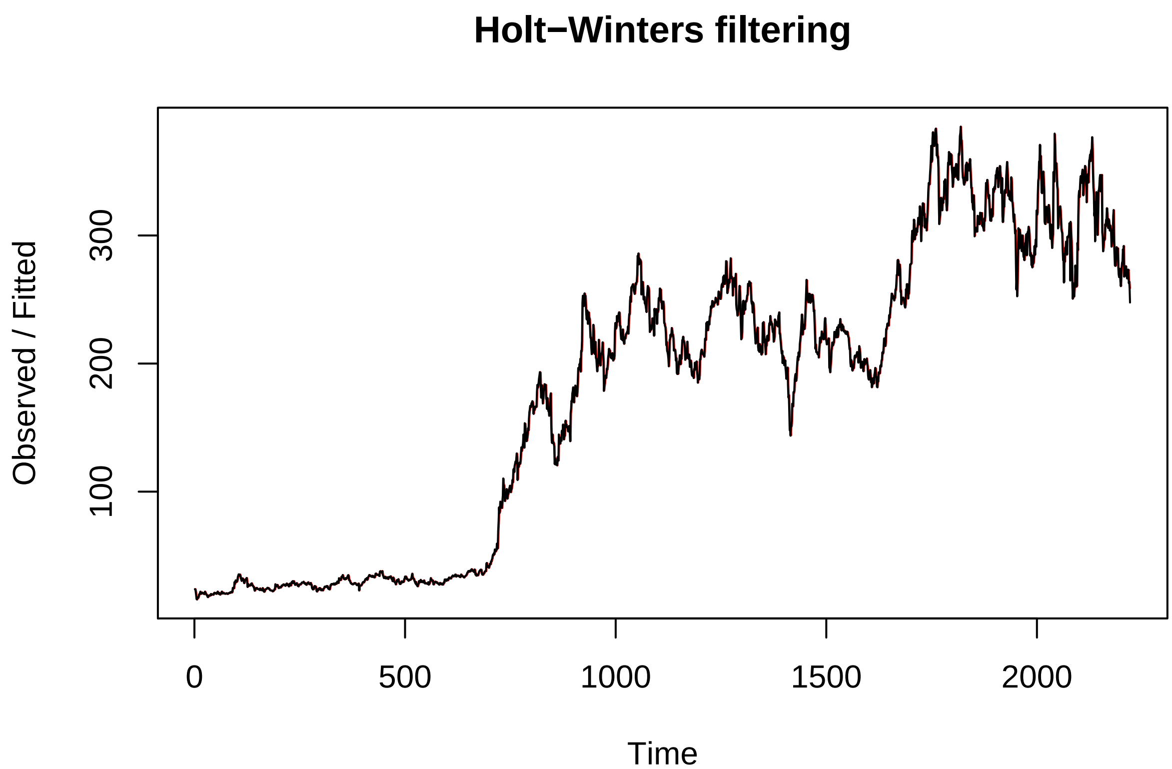
Current ecents for Disney: The current events for Disney are as follows. The company is releasing a streaming service called Disney Plus. It is going to be cheaper than Netflix but it is not expected to create much of a dent into Netflix's market share but, it should create some slow growth for the company. Source:https://www.cnbc.com/2019/04/15/what-a-1000-dollar-investment-in-disney-10-years-ago-would-be-worth-now.html

Description for Time Series

Starting with Time Series, more specifically through the Holt-Winters method, we understand that it is good at seeing long term trends overall and only relies on historical data, which is what we have through the means of the "quantmod" library in R. Some problems in relation to Holt-Winters Time Series is that historical data may not be particularly useful in forecasting future stocks and that it doesn't take current events into account. To do this, we will be using the "forecast" library in conjunction with the native Holt-Winters Time Series method in R and analyzing the trend it predicts from the data.

Here is the Holt-Winters code for the Tesla dataset. First, we create the HoltWinters data set shown below. Here is a plot of the graph predicted by HoltWinters. This is the code that has HoltWinters predict what the future of the TSLA stock could be.

```
TSLA.pred <- HoltWinters(TSLA$TSLA.Close, beta = FALSE, gamma = FALSE)
plot(TSLA.pred)
```

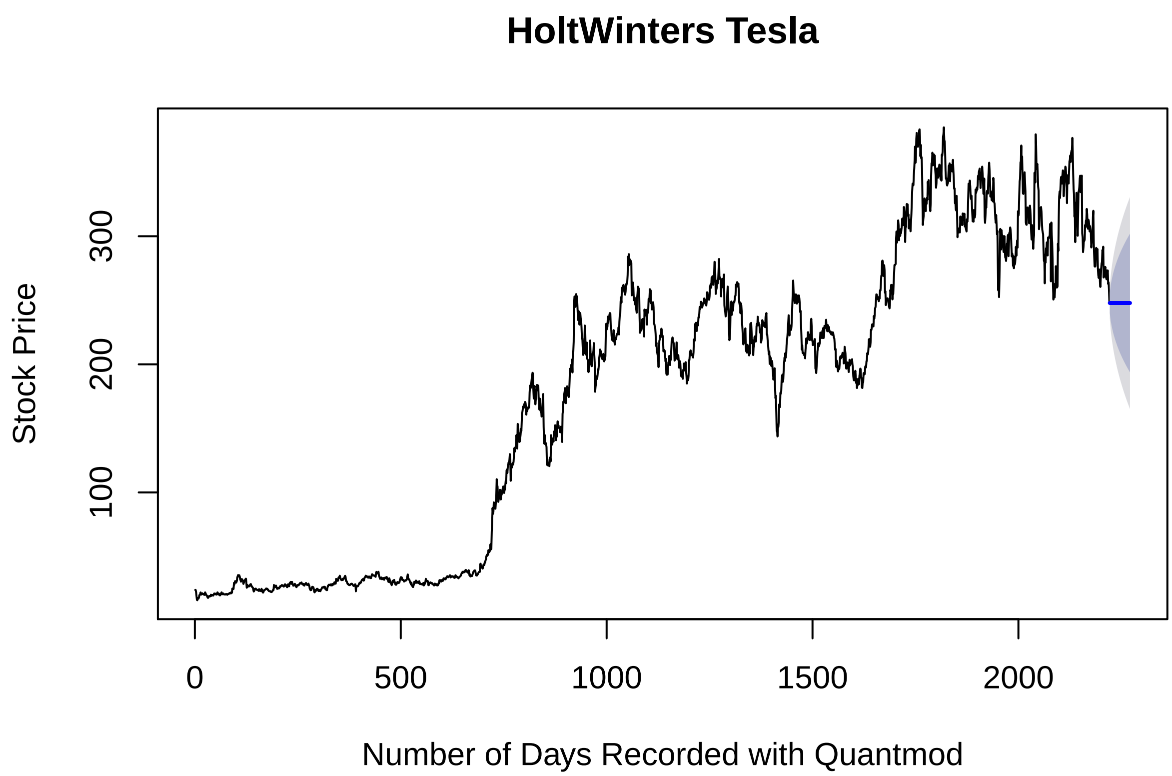



```
TSLA.pred2 <- forecast:::forecast.HoltWinters(TSLA.pred, h = 50)
TSLA.pred2
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2222       247.8599 240.0431 255.6767 235.9052 259.8147
## 2223       247.8599 236.9188 258.8010 231.1269 264.5929
## 2224       247.8599 234.5065 261.2133 227.4376 268.2822
## 2225       247.8599 232.4677 263.2521 224.3196 271.4002
## 2226       247.8599 230.6691 265.0508 221.5688 274.1510
## 2227       247.8599 229.0416 266.6783 219.0797 276.6401
## 2228       247.8599 227.5440 268.1758 216.7894 278.9304
## 2229       247.8599 226.1495 269.5703 214.6567 281.0631
## 2230       247.8599 224.8393 270.8805 212.6530 283.0669
## 2231       247.8599 223.5998 272.1200 210.7573 284.9626
## 2232       247.8599 222.4206 273.2992 208.9539 286.7660
```

```
## 2233       247.8599 221.2937 274.4261 207.2304 288.4894
## 2234       247.8599 220.2127 275.5072 205.5771 290.1427
## 2235       247.8599 219.1724 276.5475 203.9861 291.7337
## 2236       247.8599 218.1685 277.5513 202.4508 293.2690
## 2237       247.8599 217.1975 278.5224 200.9657 294.7541
## 2238       247.8599 216.2562 279.4636 199.5263 296.1936
## 2239       247.8599 215.3423 280.3776 198.1285 297.5914
## 2240       247.8599 214.4533 281.2666 196.7689 298.9510
## 2241       247.8599 213.5873 282.1325 195.4445 300.2753
## 2242       247.8599 212.7428 282.9771 194.1529 301.5670
## 2243       247.8599 211.9180 283.8018 192.8915 302.8283
## 2244       247.8599 211.1118 284.6081 191.6585 304.0614
## 2245       247.8599 210.3228 285.3970 190.4519 305.2679
## 2246       247.8599 209.5502 286.1697 189.2702 306.4496
## 2247       247.8599 208.7928 286.9271 188.1119 307.6080
## 2248       247.8599 208.0498 287.6701 186.9755 308.7443
## 2249       247.8599 207.3204 288.3995 185.8600 309.8598
## 2250       247.8599 206.6039 289.1159 184.7643 310.9556
## 2251       247.8599 205.8996 289.8202 183.6872 312.0326
## 2252       247.8599 205.2070 290.5128 182.6279 313.0919
## 2253       247.8599 204.5254 291.1944 181.5856 314.1343
## 2254       247.8599 203.8544 291.8654 180.5593 315.1605
## 2255       247.8599 203.1935 292.5263 179.5485 316.1713
## 2256       247.8599 202.5422 293.1776 178.5525 317.1674
## 2257       247.8599 201.9002 293.8197 177.5705 318.1493
## 2258       247.8599 201.2670 294.4529 176.6021 319.1177
## 2259       247.8599 200.6422 295.0776 175.6467 320.0731
## 2260       247.8599 200.0257 295.6942 174.7037 321.0161
## 2261       247.8599 199.4169 296.3029 173.7728 321.9471
## 2262       247.8599 198.8158 296.9041 172.8534 322.8665
## 2263       247.8599 198.2219 297.4979 171.9451 323.7747
## 2264       247.8599 197.6350 298.0848 171.0476 324.6722
## 2265       247.8599 197.0550 298.6649 170.1604 325.5594
## 2266       247.8599 196.4814 299.2384 169.2833 326.4365
## 2267       247.8599 195.9142 299.8056 168.4158 327.3040
## 2268       247.8599 195.3531 300.3667 167.5577 328.1621
## 2269       247.8599 194.7980 300.9218 166.7087 329.0111
## 2270       247.8599 194.2486 301.4712 165.8685 329.8513
## 2271       247.8599 193.7048 302.0150 165.0368 330.6830
```

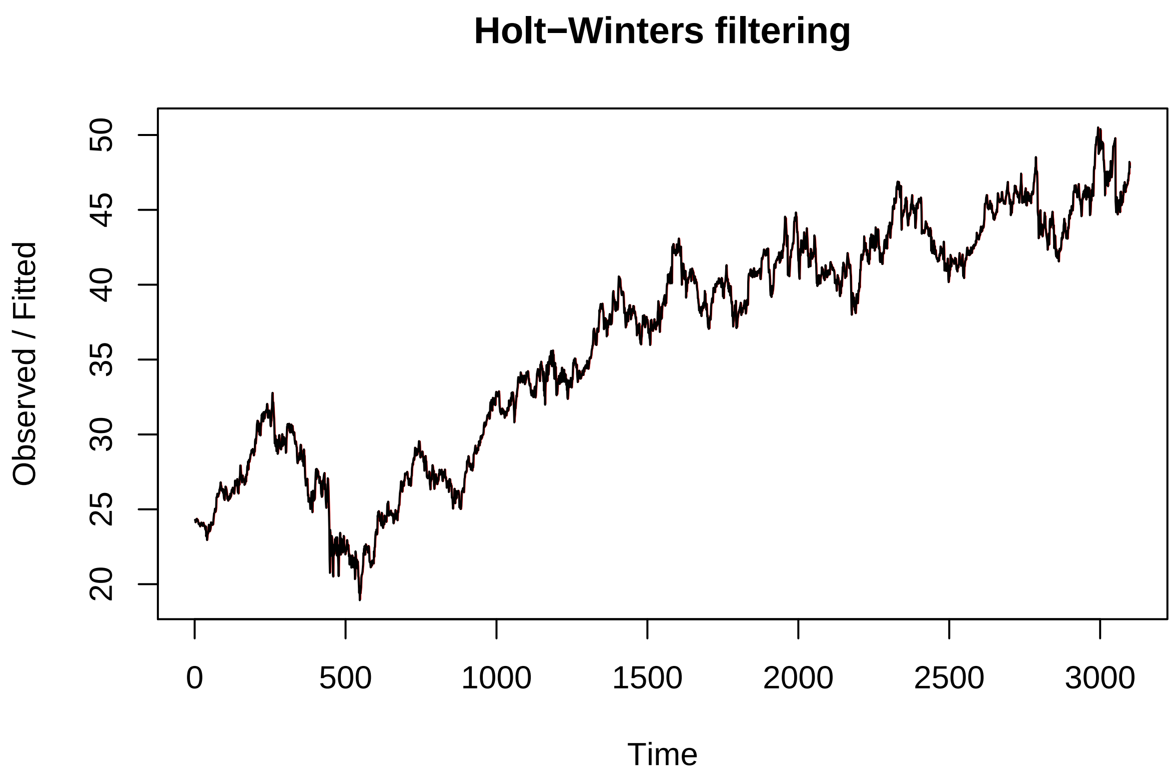
```
forecast:::plot.forecast(TSLA.pred2, xlab = "Number of Days Recorded with Quantmod", ylab = "Stock Pric
```



This shows the expected range in stock for the next 50 days.

Here is the Holt-Winters code for the Coca Cola dataset. First, we create the HoltWinters data set shown below. Here is a plot of the graph predicted by HoltWinters. This is the code that has HoltWinters predict what the future of the TSLA stock could be.

```
KO.pred <- HoltWinters(KO$KO.Close, beta = FALSE, gamma = FALSE)
plot(KO.pred)
```

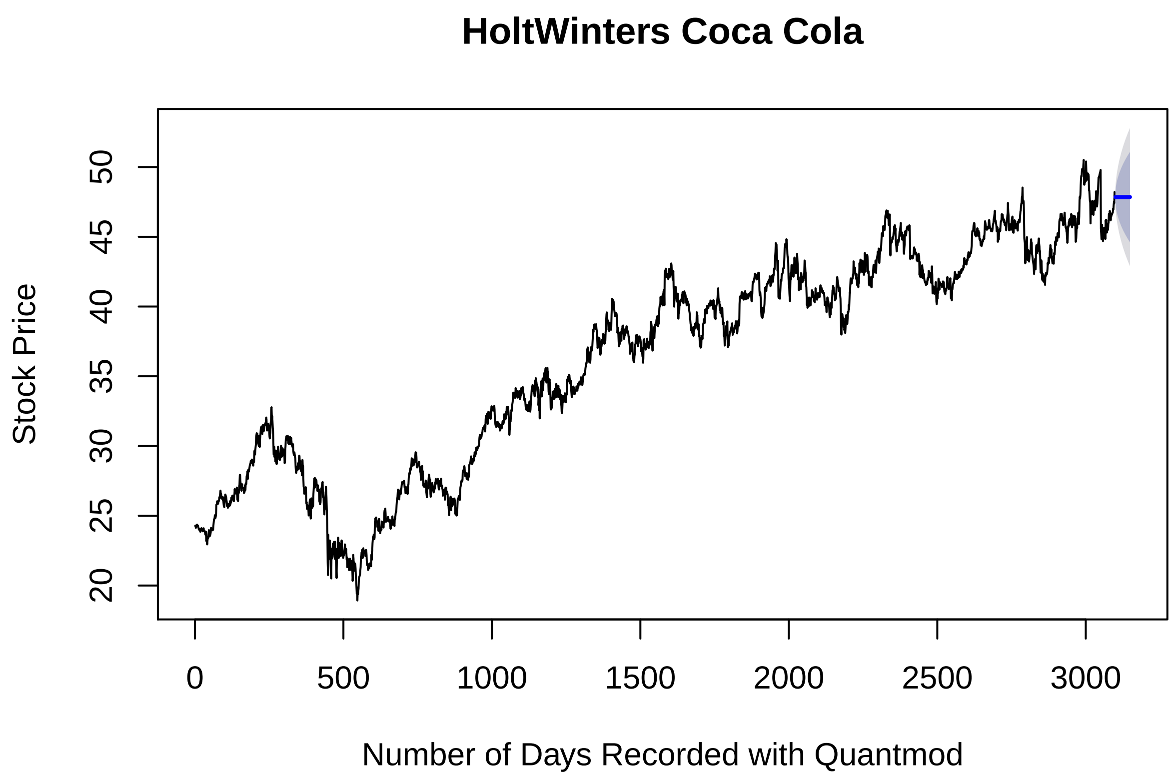



```r
KO.pred2 <- forecast:::forecast.HoltWinters(KO.pred, h = 50)
KO.pred2
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 3100       47.84671 47.36704 48.32638 47.11312 48.58030
## 3101       47.84671 47.18348 48.50994 46.83238 48.86103
## 3102       47.84671 47.04069 48.65273 46.61400 49.07941
## 3103       47.84671 46.91963 48.77378 46.42887 49.26454
## 3104       47.84671 46.81266 48.88076 46.26526 49.42815
## 3105       47.84671 46.71576 48.97766 46.11706 49.57635
## 3106       47.84671 46.62652 49.06689 45.98060 49.71282
## 3107       47.84671 46.54339 49.15003 45.85345 49.83996
## 3108       47.84671 46.46525 49.22817 45.73394 49.95947
## 3109       47.84671 46.39129 49.30212 45.62084 50.07257
## 3110       47.84671 46.32092 49.37250 45.51322 50.18020
## 3111       47.84671 46.25365 49.43976 45.41034 50.28307
## 3112       47.84671 46.18911 49.50430 45.31164 50.38178
## 3113       47.84671 46.12700 49.56642 45.21663 50.47678
## 3114       47.84671 46.06704 49.62637 45.12495 50.56847
## 3115       47.84671 46.00905 49.68437 45.03625 50.65717
## 3116       47.84671 45.95282 49.74059 44.95026 50.74315
## 3117       47.84671 45.89822 49.79519 44.86676 50.82666
## 3118       47.84671 45.84511 49.84830 44.78553 50.90788
## 3119       47.84671 45.79337 49.90004 44.70640 50.98701
## 3120       47.84671 45.74291 49.95051 44.62922 51.06419
## 3121       47.84671 45.69362 49.99979 44.55385 51.13957
## 3122       47.84671 45.64544 50.04797 44.48016 51.21325
## 3123       47.84671 45.59830 50.09512 44.40806 51.28536
## 3124       47.84671 45.55212 50.14130 44.33743 51.35598
## 3125       47.84671 45.50685 50.18657 44.26820 51.42522
```

```
## 3126       47.84671 45.46244 50.23098 44.20028 51.49313
## 3127       47.84671 45.41884 50.27458 44.13361 51.55981
## 3128       47.84671 45.37601 50.31740 44.06811 51.62531
## 3129       47.84671 45.33392 50.35950 44.00372 51.68969
## 3130       47.84671 45.29251 50.40091 43.94040 51.75302
## 3131       47.84671 45.25177 50.44165 43.87809 51.81533
## 3132       47.84671 45.21165 50.48176 43.81674 51.87668
## 3133       47.84671 45.17214 50.52127 43.75631 51.93710
## 3134       47.84671 45.13320 50.56021 43.69676 51.99665
## 3135       47.84671 45.09482 50.59860 43.63806 52.05536
## 3136       47.84671 45.05696 50.63646 43.58016 52.11326
## 3137       47.84671 45.01961 50.67381 43.52303 52.17038
## 3138       47.84671 44.98275 50.71067 43.46666 52.22676
## 3139       47.84671 44.94635 50.74707 43.41099 52.28242
## 3140       47.84671 44.91041 50.78301 43.35602 52.33739
## 3141       47.84671 44.87490 50.81852 43.30171 52.39170
## 3142       47.84671 44.83981 50.85361 43.24805 52.44537
## 3143       47.84671 44.80512 50.88830 43.19500 52.49841
## 3144       47.84671 44.77083 50.92259 43.14255 52.55086
## 3145       47.84671 44.73691 50.95651 43.09068 52.60273
## 3146       47.84671 44.70336 50.99006 43.03937 52.65404
## 3147       47.84671 44.67016 51.02325 42.98860 52.70481
## 3148       47.84671 44.63731 51.05610 42.93836 52.75506
## 3149       47.84671 44.60479 51.08862 42.88862 52.80479
```

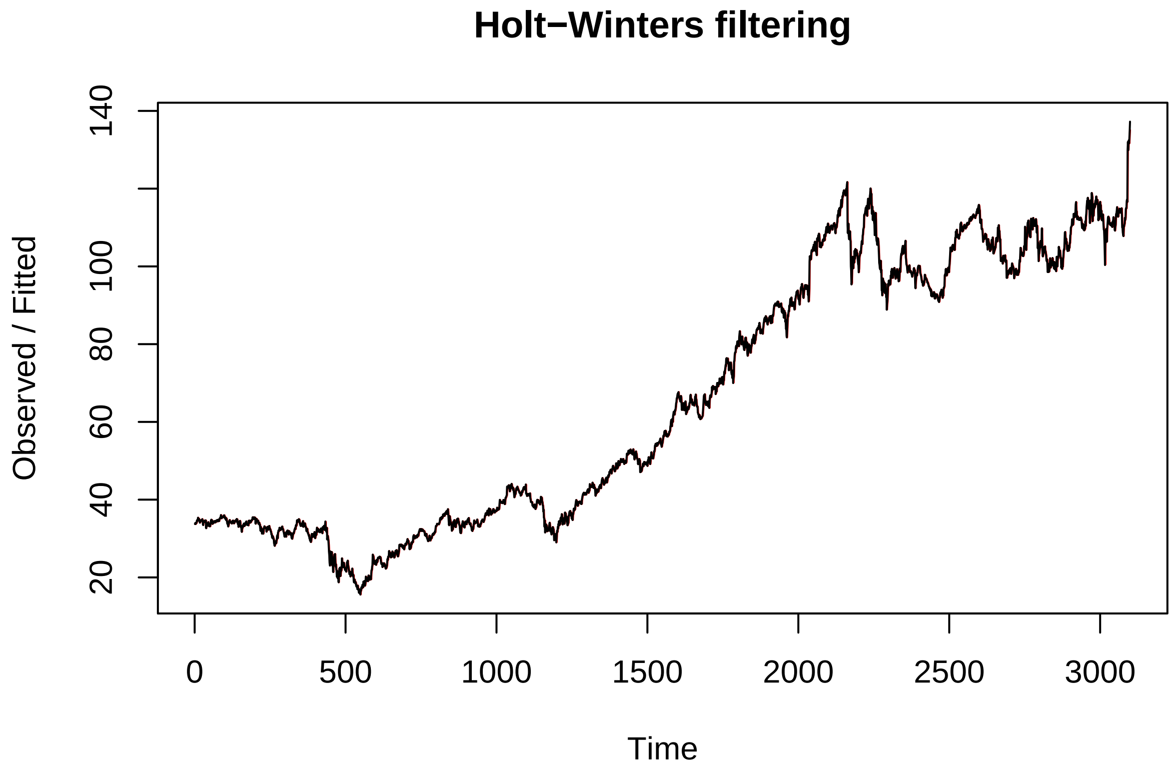
```
forecast:::plot.forecast(KO.pred2, xlab = "Number of Days Recorded with Quantmod", ylab = "Stock Price"
```



This shows the expected range in stock for the next 50 days.

Here is the Holt-Winters code for the Disney dataset. First, we create the HoltWinters data set shown below. Here is a plot of the graph predicted by HoltWinters. This is the code that has HoltWinters predict what the future of the TSLA stock could be.

```r
DIS.pred <- HoltWinters(DIS$DIS.Close, beta = FALSE, gamma = FALSE)
plot(DIS.pred)
```

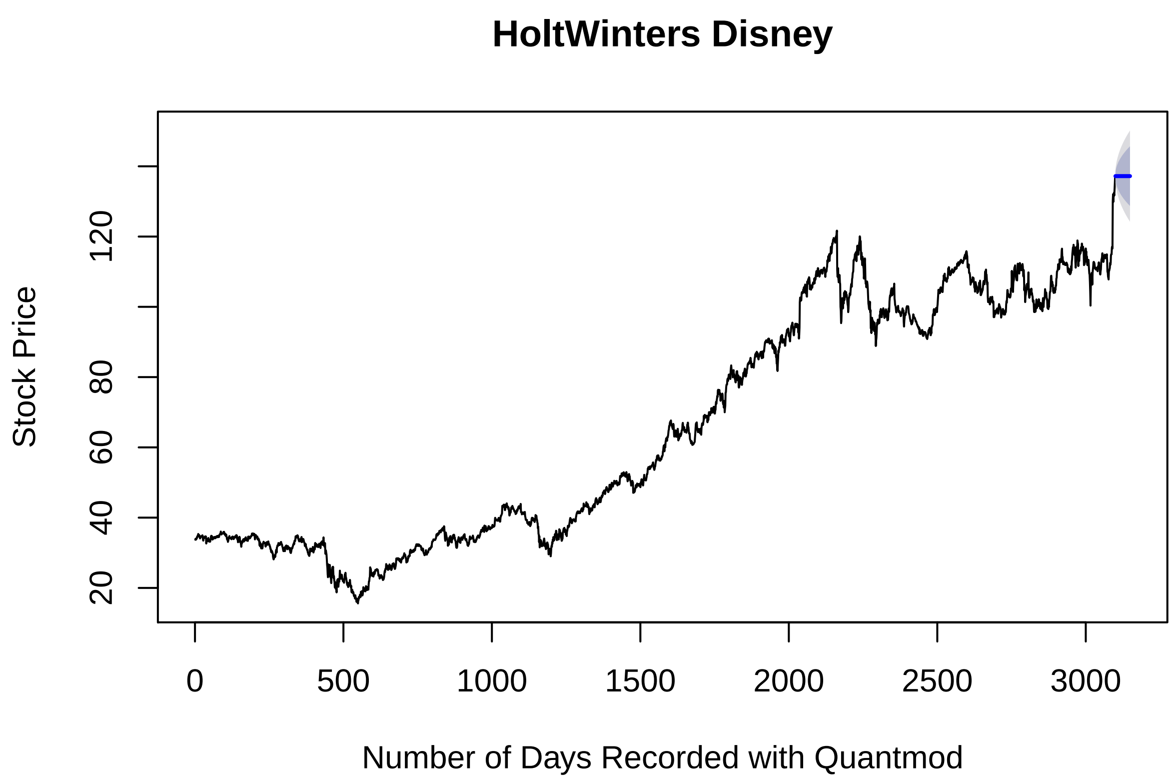



```r
DIS.pred2 <- forecast:::forecast.HoltWinters(DIS.pred, h = 50)
DIS.pred2
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 3100       137.2035 135.9835 138.4235 135.3377 139.0694
## 3101       137.2035 135.4926 138.9144 134.5869 139.8201
## 3102       137.2035 135.1140 139.2930 134.0079 140.3991
## 3103       137.2035 134.7942 139.6128 133.5188 140.8882
## 3104       137.2035 134.5121 139.8949 133.0874 141.3196
## 3105       137.2035 134.2569 140.1501 132.6971 141.7099
## 3106       137.2035 134.0222 140.3849 132.3380 142.0690
## 3107       137.2035 133.8035 140.6035 132.0037 142.4033
## 3108       137.2035 133.5982 140.8089 131.6896 142.7174
## 3109       137.2035 133.4039 141.0032 131.3924 143.0146
## 3110       137.2035 133.2190 141.1880 131.1098 143.2973
## 3111       137.2035 133.0424 141.3646 130.8396 143.5674
## 3112       137.2035 132.8730 141.5341 130.5805 143.8265
## 3113       137.2035 132.7099 141.6971 130.3311 144.0759
## 3114       137.2035 132.5526 141.8544 130.0905 144.3165
## 3115       137.2035 132.4004 142.0066 129.8578 144.5493
## 3116       137.2035 132.2529 142.1541 129.6322 144.7748
## 3117       137.2035 132.1096 142.2974 129.4131 144.9939
## 3118       137.2035 131.9703 142.4367 129.2000 145.2070
## 3119       137.2035 131.8346 142.5724 128.9925 145.4145
## 3120       137.2035 131.7023 142.7048 128.7901 145.6170
## 3121       137.2035 131.5730 142.8340 128.5924 145.8146
```

```
## 3122       137.2035 131.4467 142.9604 128.3992 146.0079
## 3123       137.2035 131.3230 143.0840 128.2101 146.1969
## 3124       137.2035 131.2019 143.2051 128.0249 146.3821
## 3125       137.2035 131.0832 143.3238 127.8434 146.5637
## 3126       137.2035 130.9668 143.4402 127.6653 146.7417
## 3127       137.2035 130.8525 143.5545 127.4905 146.9165
## 3128       137.2035 130.7402 143.6668 127.3188 147.0882
## 3129       137.2035 130.6299 143.7772 127.1500 147.2570
## 3130       137.2035 130.5213 143.8857 126.9840 147.4230
## 3131       137.2035 130.4145 143.9925 126.8207 147.5864
## 3132       137.2035 130.3094 144.0976 126.6598 147.7472
## 3133       137.2035 130.2058 144.2012 126.5014 147.9056
## 3134       137.2035 130.1037 144.3033 126.3454 148.0617
## 3135       137.2035 130.0031 144.4039 126.1915 148.2155
## 3136       137.2035 129.9039 144.5031 126.0397 148.3673
## 3137       137.2035 129.8060 144.6010 125.8900 148.5170
## 3138       137.2035 129.7094 144.6976 125.7423 148.6648
## 3139       137.2035 129.6140 144.7930 125.5964 148.8106
## 3140       137.2035 129.5198 144.8872 125.4523 148.9547
## 3141       137.2035 129.4267 144.9803 125.3100 149.0970
## 3142       137.2035 129.3348 145.0722 125.1693 149.2377
## 3143       137.2035 129.2439 145.1631 125.0303 149.3767
## 3144       137.2035 129.1540 145.2530 124.8929 149.5141
## 3145       137.2035 129.0651 145.3419 124.7569 149.6501
## 3146       137.2035 128.9772 145.4298 124.6225 149.7845
## 3147       137.2035 128.8902 145.5168 124.4894 149.9176
## 3148       137.2035 128.8041 145.6029 124.3578 150.0492
## 3149       137.2035 128.7189 145.6881 124.2275 150.1796
```

```r
forecast:::plot.forecast(DIS.pred2, xlab = "Number of Days Recorded with Quantmod", ylab = "Stock Price
```



This

shows the expected range in stock for the next 50 days.

Description for Neural Networks

Moving forward with Neural Networks, we will be using the "neuralnet", "xts", "zoo", "lubridate", "BBmisc" libraries in r. We understand that it is beneficial for larger datasets and that it can be more efficient in terms of time to be run. However, our datasets are not in the size of millions and it is difficult to build and understand a neural network algorithm. Thus, it is to be expected that this method will be weighed less than other methods.

Here is the code for Tesla. We first start by creating a new dataframe, as the values we are using to learn are different than the other methods. We also have to filter the dataset, as seen in the for loop.

```r
TSLA.df <- data.frame("Open" = TSLA$TSLA.Open, "Close" = TSLA$TSLA.Close, "Year"=as.numeric(format(inde
                      "Month"=as.numeric(format(index(TSLA), "%m")), "Day"=as.numeric(format(index(TSLA
                      "Is_Month_Start" = FALSE, "Is_Month_End" = FALSE, "Is_Year_End" = FALSE,
                      "Is_Year_Start" = FALSE, "Is_Quarter_Start" = FALSE, "Is_Quarter_End" = FALSE,
                      "Quarters"=quarters(index(TSLA)))

for (i in 1:nrow(TSLA.df)){

  date <- as.POSIXlt(paste(TSLA.df[i,]$Year, "-", TSLA.df[i,]$Month, "-", TSLA.df[i,]$Day, sep=""))

  #Determine if it is beginning/end of the month and change dataset accordingly
  if (TSLA.df[i,]$Day <= 7){ #7 is the first ~25% of the month
    TSLA.df[i,]$Is_Month_Start = TRUE
  } else if (TSLA.df[i,]$Day > 24){ #24 is the last ~25% of the month
    TSLA.df[i,]$Is_Month_End = TRUE
  }

  #Determine if it is the beginning/end of the year and change dataset accordingly
  if (TSLA.df[i,]$Quarters == 'Q1'){ #Q1 is the first ~25% of the year
    TSLA.df[i,]$Is_Year_Start = TRUE
  } else if (TSLA.df[i,]$Quarters == 'Q4'){ #Q4 is the last ~25% of the year
    TSLA.df[i,]$Is_Year_End = TRUE
  }

  #Determine if it is the beginning of the quarter and change dataset accordingly
  if (yday(date) <= 22){ #days 1-22 is the first ~25% of Q1
    TSLA.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 91 &  yday(date) <= 113){ #days 91-113 is the first ~25% of Q2
    TSLA.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 182 & yday(date) <=204){ #days 182-204 is the first ~25% of Q3
    TSLA.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 273 & yday(date) <=296){ #days 273-296 is the first ~25% of Q4
    TSLA.df[i,]$Is_Quarter_Start = TRUE
  }

  #Determine if it is the end of the quarter and change dataset accordingly
  if (yday(date) <= 91 & yday(date) >= 69){ #days 69-91 is the last ~25% of Q1
    TSLA.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 182 &  yday(date) >= 160){ #days 160-182 is the last ~25% of Q2
    TSLA.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 273 & yday(date) >= 251){ #days 251-273 is the last ~25% of Q3
```

```
    TSLA.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 366 & yday(date) >= 343){ #days 343-365 is the last ~25% of Q4
    TSLA.df[i,]$Is_Quarter_End = TRUE
  }
}

TSLA.df <- TSLA.df[,1:11]#Remove the last column, as it was only used to clean data

#change true/false values into 1/0s to prepare for normalization.
cols <- sapply(TSLA.df, is.logical)
TSLA.df[,cols] <- lapply(TSLA.df[,cols], as.numeric)
```

To use neural networks, we have to normalize the data set. We used the BBmsic library to help us with this.

```
TSLA.df <- BBmisc::normalize(TSLA.df)
```

Now, we make a testing and training set to evaluate the performance of neural networks. We decided to use 75% of the original data set.

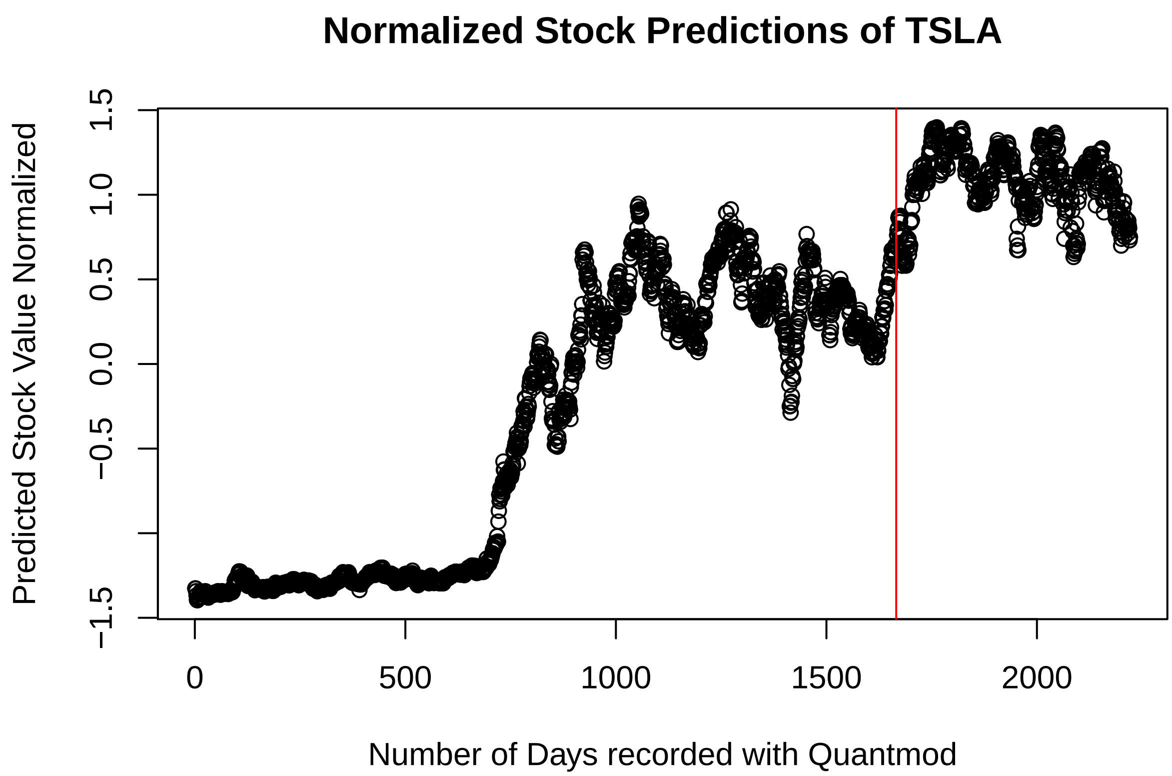
```
TSLA.train <- TSLA.df[1:(.75 * nrow(TSLA.df)),]
TSLA.test <- TSLA.df[(.75 * nrow(TSLA.df)):nrow(TSLA.df),]
```

We now create the neural network and display it.

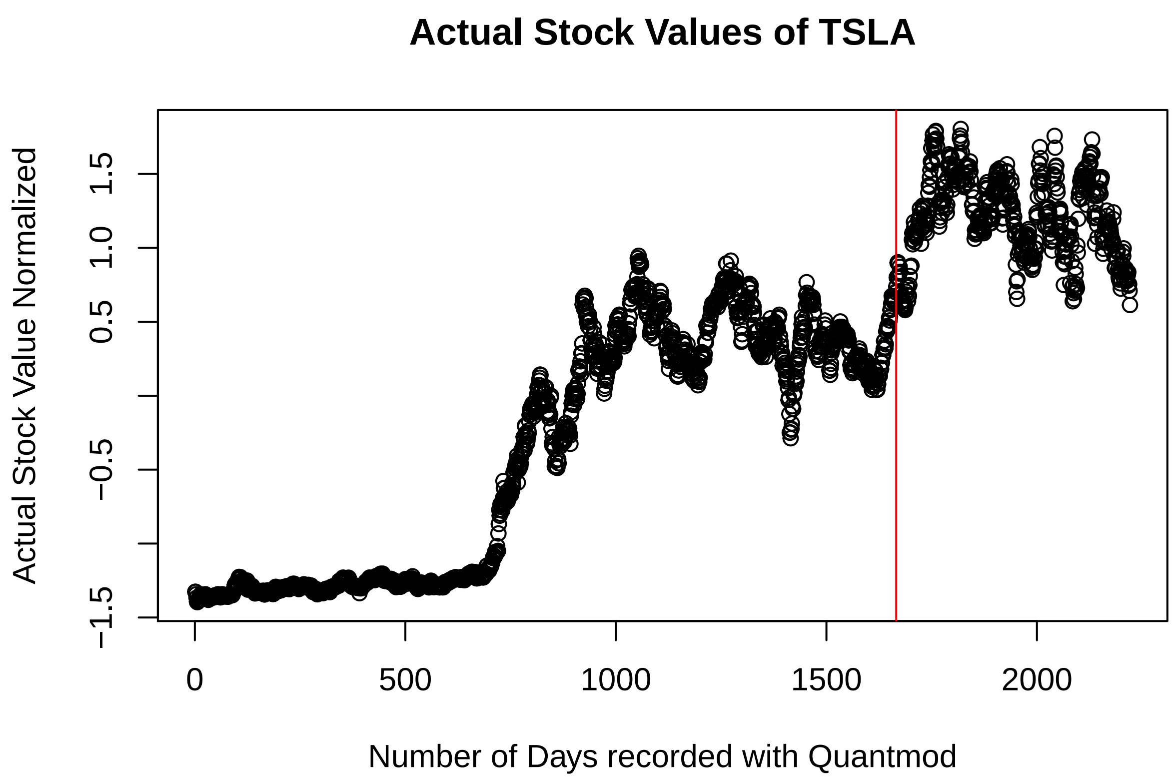
```
TSLA.NN <- neuralnet(TSLA.Close ~ TSLA.Open + Year + Month + Day + Is_Month_Start + Is_Month_End + Is_Y
                       Is_Year_Start + Is_Quarter_Start + Is_Quarter_End
                     , data = TSLA.train, linear.output=TRUE, hidden=c(5,3,3))
plot(TSLA.NN)
```

Using the created neural network, we predict the values and compare them to the actual values. The red line indicates where the training data ends and where the prediction and actual values, respectively, begin.

```
TSLA.pred <- predict(TSLA.NN, TSLA.test)
TSLA.pred.total <- c(TSLA.train$TSLA.Close, TSLA.pred)
plot(TSLA.pred.total, xlab="Number of Days recorded with Quantmod", ylab="Predicted Stock Value Normali
abline(v = (.75 * nrow(TSLA.df)), col='red')
```




```
plot(TSLA.df$TSLA.Close, xlab="Number of Days recorded with Quantmod", ylab="Actual Stock Value Normali
abline(v = (.75 * nrow(TSLA.df)), col='red')
```



Comparing both graphs, we can see that our predictor did a pretty poor job of predicting the values for TSLA. It predicted much less volatility than what actually happened, and did not predict it to go nearly as high as it did based on the normalized scale. The prediction maxed out at just above 1, but the actual values went more towards 1.5. Other than that, it did manage to predict that there would be 3 large "dips", as both the predicted graph and the actual graph show it. I would not recommend using this method, with the parameters we chose, to predict, but it did do well in seeing dips.

Here is the code for Coca Cola. We first start by creating a new dataframe, as the values we are using to learn are different than the other methods. We also have to filter the dataset, as seen in the for loop.

```
KO.df <- data.frame("Open" = KO$KO.Open, "Close" = KO$KO.Close, "Year"=as.numeric(format(index(KO), "%Y
                      "Month"=as.numeric(format(index(KO), "%m")), "Day"=as.numeric(format(index(KO), "
                      "Is_Month_Start" = FALSE, "Is_Month_End" = FALSE, "Is_Year_End" = FALSE,
                      "Is_Year_Start" = FALSE, "Is_Quarter_Start" = FALSE, "Is_Quarter_End" = FALSE,
                      "Quarters"=quarters(index(KO)))

for (i in 1:nrow(KO.df)){

  date <- as.POSIXlt(paste(KO.df[i,]$Year, "-", KO.df[i,]$Month, "-", KO.df[i,]$Day, sep=""))

  #Determine if it is beginning/end of the month and change dataset accordingly
  if (KO.df[i,]$Day <= 7){ #7 is the first ~25% of the month
    KO.df[i,]$Is_Month_Start = TRUE
  } else if (KO.df[i,]$Day > 24){ #24 is the last ~25% of the month
    KO.df[i,]$Is_Month_End = TRUE
  }

  #Determine if it is the beginning/end of the year and change dataset accordingly
  if (KO.df[i,]$Quarters == 'Q1'){ #Q1 is the first ~25% of the year
    KO.df[i,]$Is_Year_Start = TRUE
  } else if (KO.df[i,]$Quarters == 'Q4'){ #Q4 is the last ~25% of the year
    KO.df[i,]$Is_Year_End = TRUE
  }

  #Determine if it is the beginning of the quarter and change dataset accordingly
  if (yday(date) <= 22){ #days 1-22 is the first ~25% of Q1
    KO.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 91 &  yday(date) <= 113){ #days 91-113 is the first ~25% of Q2
    KO.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 182 & yday(date) <=204){ #days 182-204 is the first ~25% of Q3
    KO.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 273 & yday(date) <=296){ #days 273-296 is the first ~25% of Q4
    KO.df[i,]$Is_Quarter_Start = TRUE
  }

  #Determine if it is the end of the quarter and change dataset accordingly
  if (yday(date) <= 91 & yday(date) >= 69){ #days 69-91 is the last ~25% of Q1
    KO.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 182 &  yday(date) >= 160){ #days 160-182 is the last ~25% of Q2
    KO.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 273 & yday(date) >= 251){ #days 251-273 is the last ~25% of Q3
    KO.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 366 & yday(date) >= 343){ #days 343-365 is the last ~25% of Q4
    KO.df[i,]$Is_Quarter_End = TRUE
  }
}

KO.df <- KO.df[,1:11]#Remove the last column, it was only used to clean data
```

```
#change true/false values into 1/0s
cols <- sapply(KO.df, is.logical)
KO.df[,cols] <- lapply(KO.df[,cols], as.numeric)
```

To use neural networks, we have to normalize the data set. We used the BBmsic library to help us with this.

```
KO.df <- BBmisc::normalize(KO.df)
```

Now, we make a testing and training set to evaluate the performance of neural networks. We decided to use 75% of the original data set.

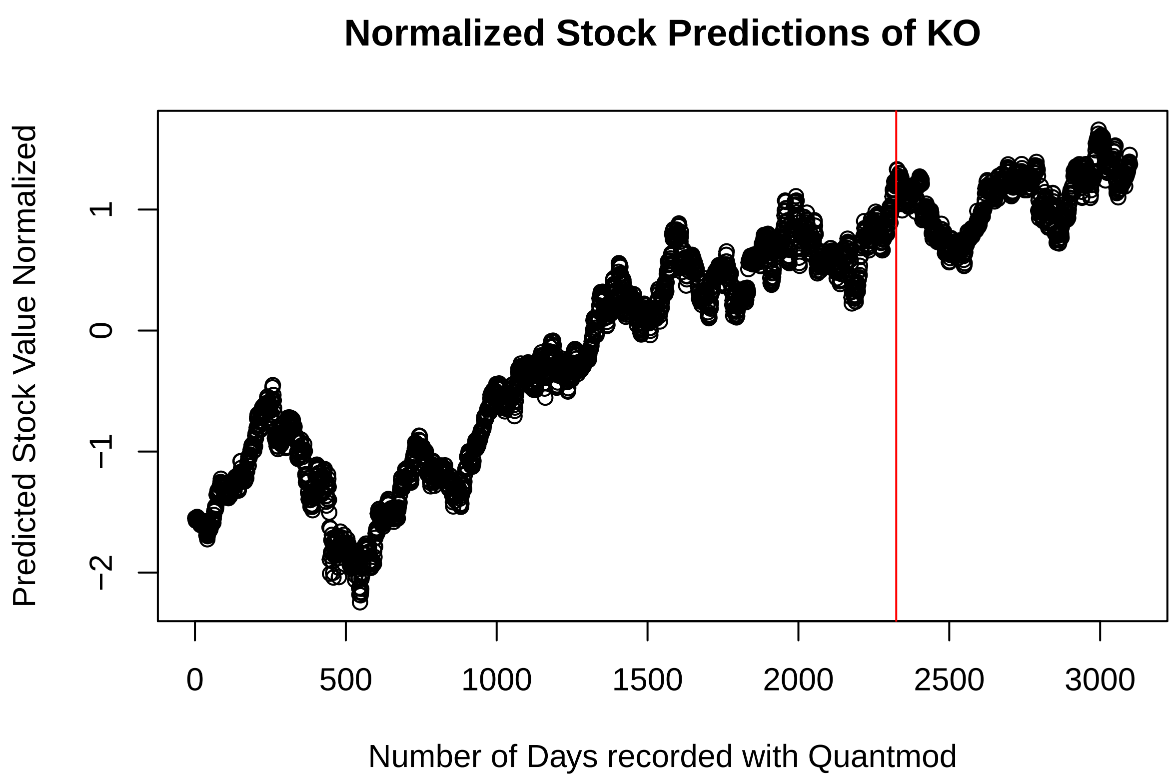
```
KO.train <- KO.df[1:(.75 * nrow(KO.df)),] #75% of data set
KO.test <- KO.df[(.75 * nrow(KO.df)):nrow(KO.df),]
```

We now create the neural network and display it.

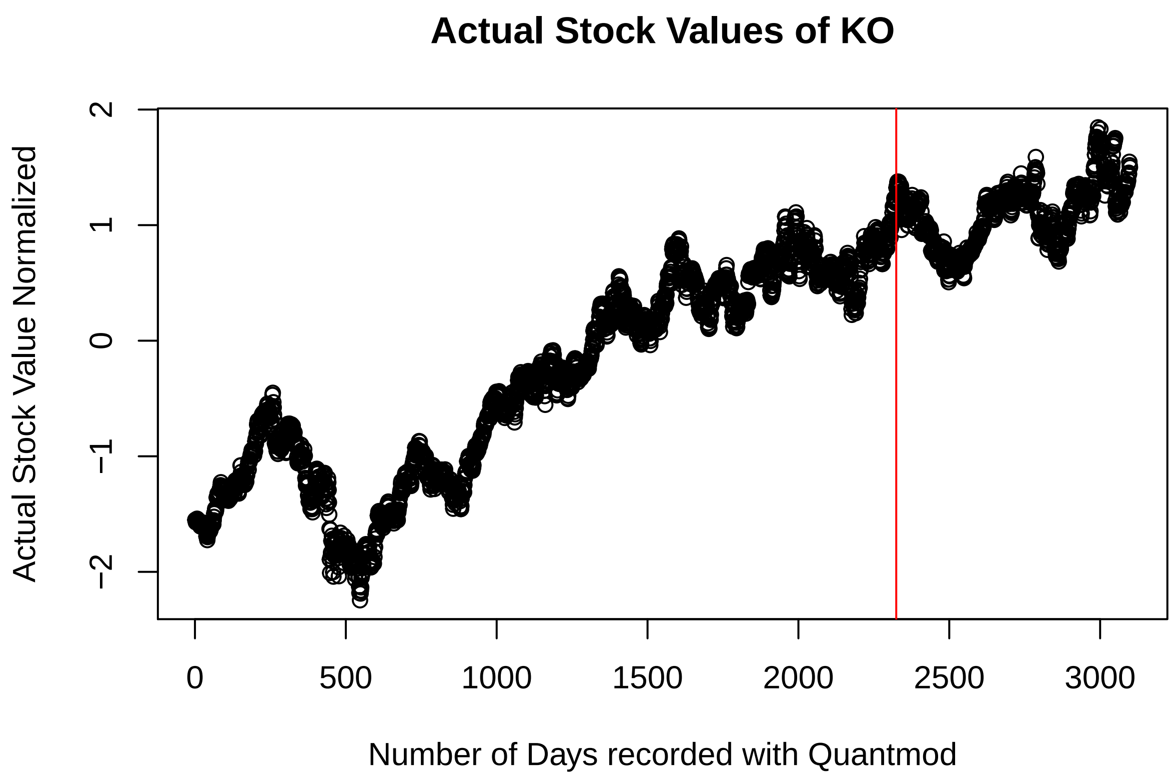
```
KO.NN <- neuralnet(KO.Close ~ KO.Open + Year + Month + Day + Is_Month_Start + Is_Month_End + Is_Year_End
                       Is_Year_Start + Is_Quarter_Start + Is_Quarter_End
                     , data = KO.train, linear.output=TRUE, hidden=c(5,3,3))
plot(KO.NN)
```

Using the created neural network, we predict the values and compare them to the actual values. The red line indicates where the training data ends and where the prediction and actual values, respectively, begin.

```
KO.pred <- predict(KO.NN, KO.test)
KO.pred.total <- c(KO.train$KO.Close, KO.pred)
plot(KO.pred.total, xlab="Number of Days recorded with Quantmod", ylab="Predicted Stock Value Normalized
abline(v = (.75 * nrow(KO.df)), col='red')
```




```
plot(KO.df$KO.Close, xlab="Number of Days recorded with Quantmod", ylab="Actual Stock Value Normalized"
abline(v = (.75 * nrow(KO.df)), col='red')
```



Surprisingly, this model worked very well for Coca Cola. It managed to predict the 3 peaks fairly accurately and both dips as well. It is nearly exact overall but the only sigificant difference would be that the neural network was significantly less aggressive when predicting peaks. The actual values ended up going a bit higher than predicted, but they were very close. For this data set, I would recommend using this method.

Here is the code for Disney. We first start by creating a new dataframe, as the values we are using to learn are different than the other methods. We also have to filter the dataset, as seen in the for loop.

```
DIS.df <- data.frame("Open" = DIS$DIS.Open, "Close" = DIS$DIS.Close, "Year"=as.numeric(format(index(DIS
                     "Month"=as.numeric(format(index(DIS), "%m")), "Day"=as.numeric(format(index(DIS),
                     "Is_Month_Start" = FALSE, "Is_Month_End" = FALSE, "Is_Year_End" = FALSE,
                     "Is_Year_Start" = FALSE, "Is_Quarter_Start" = FALSE, "Is_Quarter_End" = FALSE,
                     "Quarters"=quarters(index(DIS)))

for (i in 1:nrow(DIS.df)){

  date <- as.POSIXlt(paste(DIS.df[i,]$Year, "-", DIS.df[i,]$Month, "-", DIS.df[i,]$Day, sep=""))

  #Determine if it is beginning/end of the month and change dataset accordingly
  if (DIS.df[i,]$Day <= 7){ #7 is the first ~25% of the month
    DIS.df[i,]$Is_Month_Start = TRUE
  } else if (DIS.df[i,]$Day > 24){ #24 is the last ~25% of the month
    DIS.df[i,]$Is_Month_End = TRUE
  }

  #Determine if it is the beginning/end of the year and change dataset accordingly
  if (DIS.df[i,]$Quarters == 'Q1'){ #Q1 is the first ~25% of the year
    DIS.df[i,]$Is_Year_Start = TRUE
  } else if (DIS.df[i,]$Quarters == 'Q4'){ #Q4 is the last ~25% of the year
    DIS.df[i,]$Is_Year_End = TRUE
  }
```

```
  #Determine if it is the beginning of the quarter and change dataset accordingly
  if (yday(date) <= 22){ #days 1-22 is the first ~25% of Q1
    DIS.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 91 &  yday(date) <= 113){ #days 91-113 is the first ~25% of Q2
    DIS.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 182 & yday(date) <=204){ #days 182-204 is the first ~25% of Q3
    DIS.df[i,]$Is_Quarter_Start = TRUE
  } else if (yday(date) > 273 & yday(date) <=296){ #days 273-296 is the first ~25% of Q4
    DIS.df[i,]$Is_Quarter_Start = TRUE
  }

  #Determine if it is the end of the quarter and change dataset accordingly
  if (yday(date) <= 91 & yday(date) >= 69){ #days 69-91 is the last ~25% of Q1
    DIS.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 182 &  yday(date) >= 160){ #days 160-182 is the last ~25% of Q2
    DIS.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 273 & yday(date) >= 251){ #days 251-273 is the last ~25% of Q3
    DIS.df[i,]$Is_Quarter_End = TRUE
  } else if (yday(date) <= 366 & yday(date) >= 343){ #days 343-365 is the last ~25% of Q4
    DIS.df[i,]$Is_Quarter_End = TRUE
  }
}

DIS.df <- DIS.df[,1:11]#Remove the last column, it was only used to clean data

#change true/false values into 1/0s
cols <- sapply(DIS.df, is.logical)
DIS.df[,cols] <- lapply(DIS.df[,cols], as.numeric)
```

To use neural networks, we have to normalize the data set. We used the BBmsic library to help us with this.

```
DIS.df <- BBmisc::normalize(DIS.df)
```

Now, we make a testing and training set to evaluate the performance of neural networks. We decided to use 75% of the original data set.

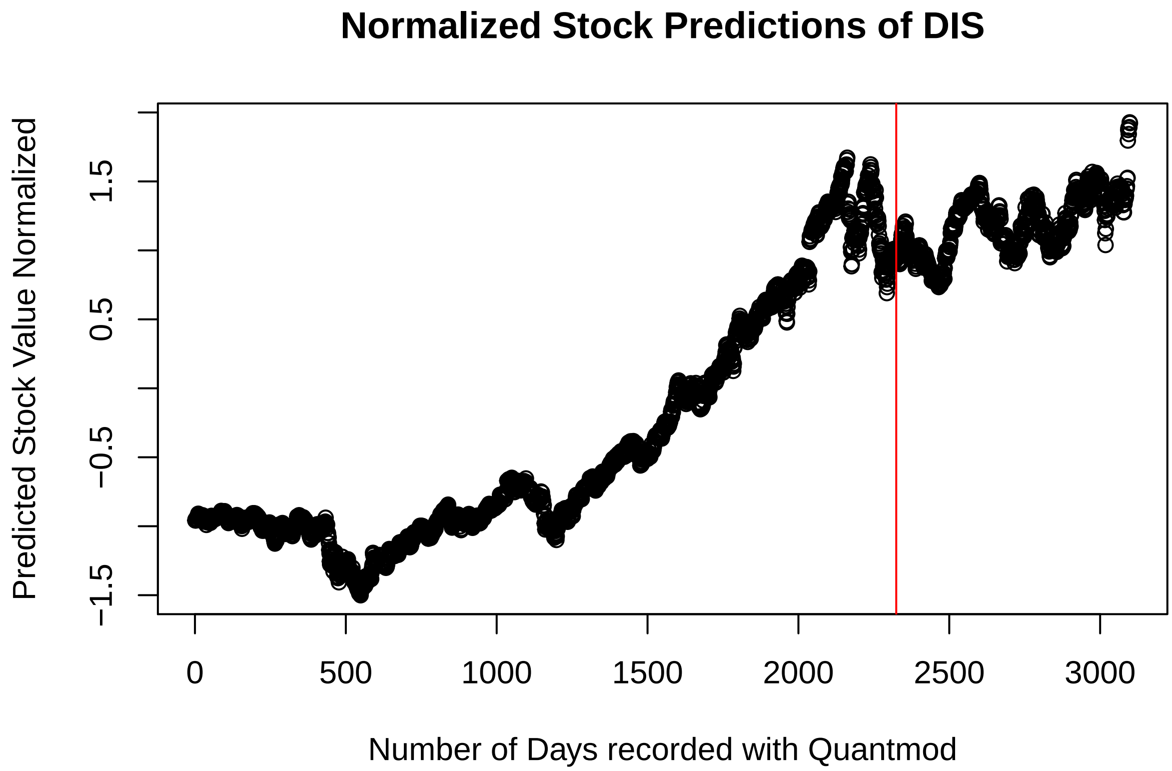
```
DIS.train <- DIS.df[1:(.75 * nrow(DIS.df)),] #75% of data set
DIS.test <- DIS.df[(.75 * nrow(DIS.df)):nrow(DIS.df),]
```

We now create the neural network and display it.

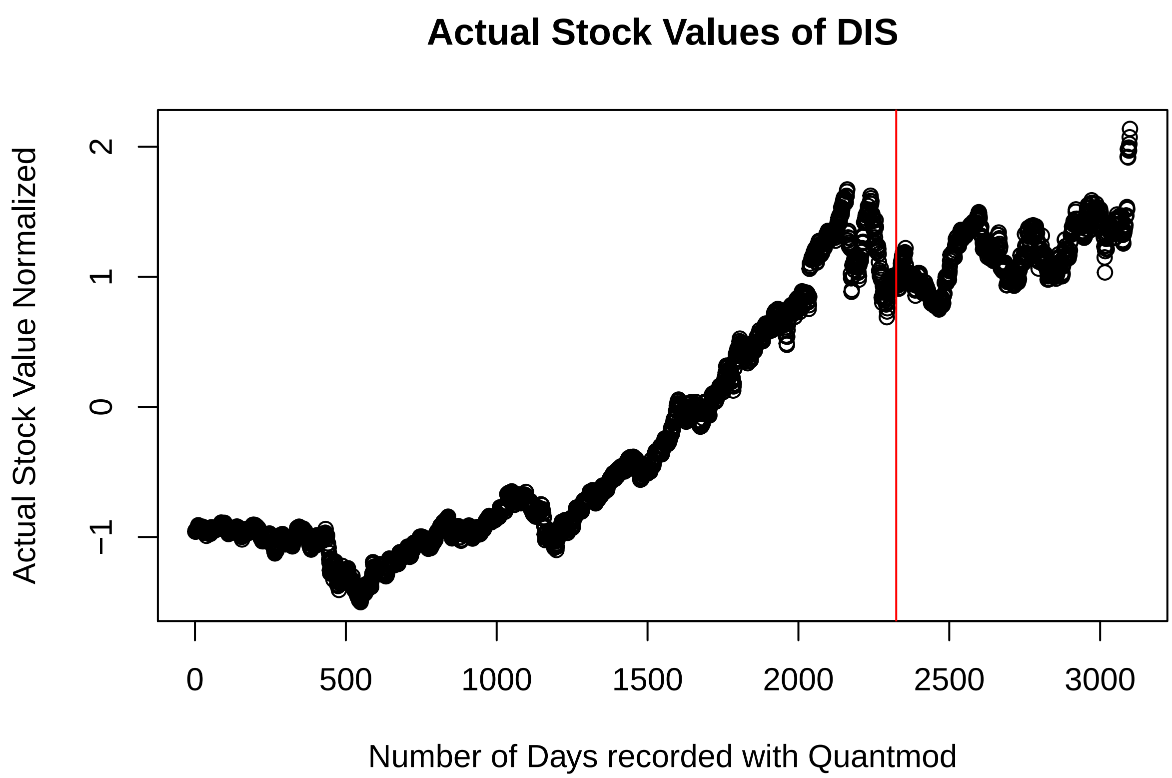
```
DIS.NN <- neuralnet(DIS.Close ~ DIS.Open + Year + Month + Day + Is_Month_Start + Is_Month_End + Is_Year_
                        Is_Year_Start + Is_Quarter_Start + Is_Quarter_End
                      , data = DIS.train, linear.output=TRUE, hidden=c(5,3,3))
plot(DIS.NN)
```

Using the created neural network, we predict the values and compare them to the actual values. The red line indicates where the training data ends and where the prediction and actual values, respectively, begin.

```
DIS.pred <- predict(DIS.NN, DIS.test)
DIS.pred.total <- c(DIS.train$DIS.Close, DIS.pred)
plot(DIS.pred.total, xlab="Number of Days recorded with Quantmod", ylab="Predicted Stock Value Normaliz
abline(v = (.75 * nrow(DIS.df)), col='red')
```




```
plot(DIS.df$DIS.Close, xlab="Number of Days recorded with Quantmod", ylab="Actual Stock Value Normalized
abline(v = (.75 * nrow(DIS.df)), col='red')
```



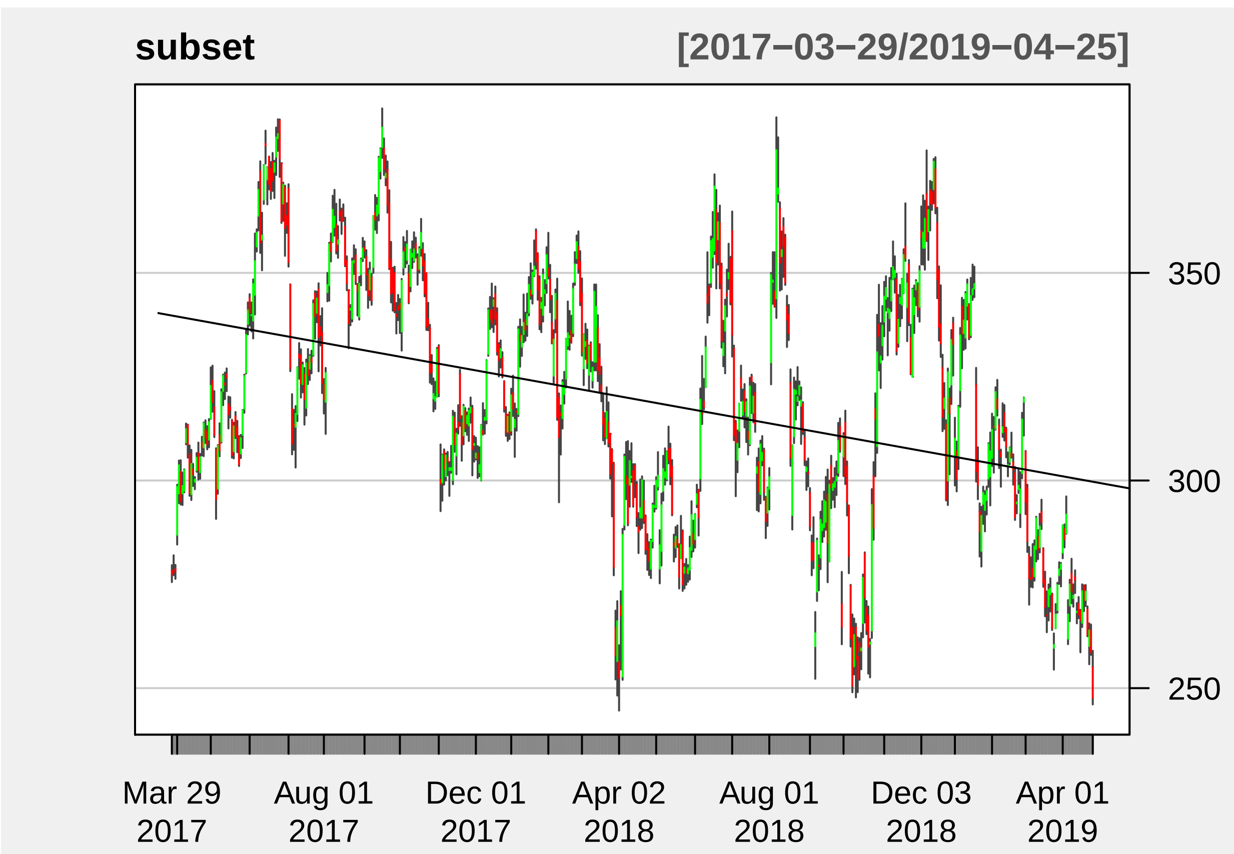
Again, this model performed exceedingly well! It was able to predict 3 peaks, 3 dips, and even manage to predict, towards the very end, the start of a peak. As with the last one, Neural Networks seem to be less aggressive with the predictions overall, as the actual data set reached and exceeded 2, but the predictions didn't quite get there. I would again recommend using this model for our overall stock market prediciton.

Description for Linear Regression

Onward, we will be looking into the Linear Regression method of predicting the future values of our chosen stocks. The method in which this will be executed is by choosing the most recent trend in the data by looking at the plot for each graph and finding a linear regression of the graphs to make predictions. Some benefits for this method is that it will use all of the data and fit the data in the best possible way and the model will also favor newer data rather than older data. Moreover, a property in Calculus is that if one restricts the domain in any graph enough, the graph will show up as linear. This is why Linear Regression is a valid form of analyzinng the data. However, there is a risk in excluding necessary data to create the trend of the graph. Additionally, this is a pretty simplistic way of predicting future stocks. Thus, this may be weighed less than other options. The Timewarp library was used in addition to the quantmod library.

Here are the steps for Linear Regression in the Tesla dataset. First we restrict the domain of the dataset to only view the most recent trend of the data. We next use Chartseries with subset as the main argument. We then use a linear model on the same range of the chart in addition to plotting the chart.

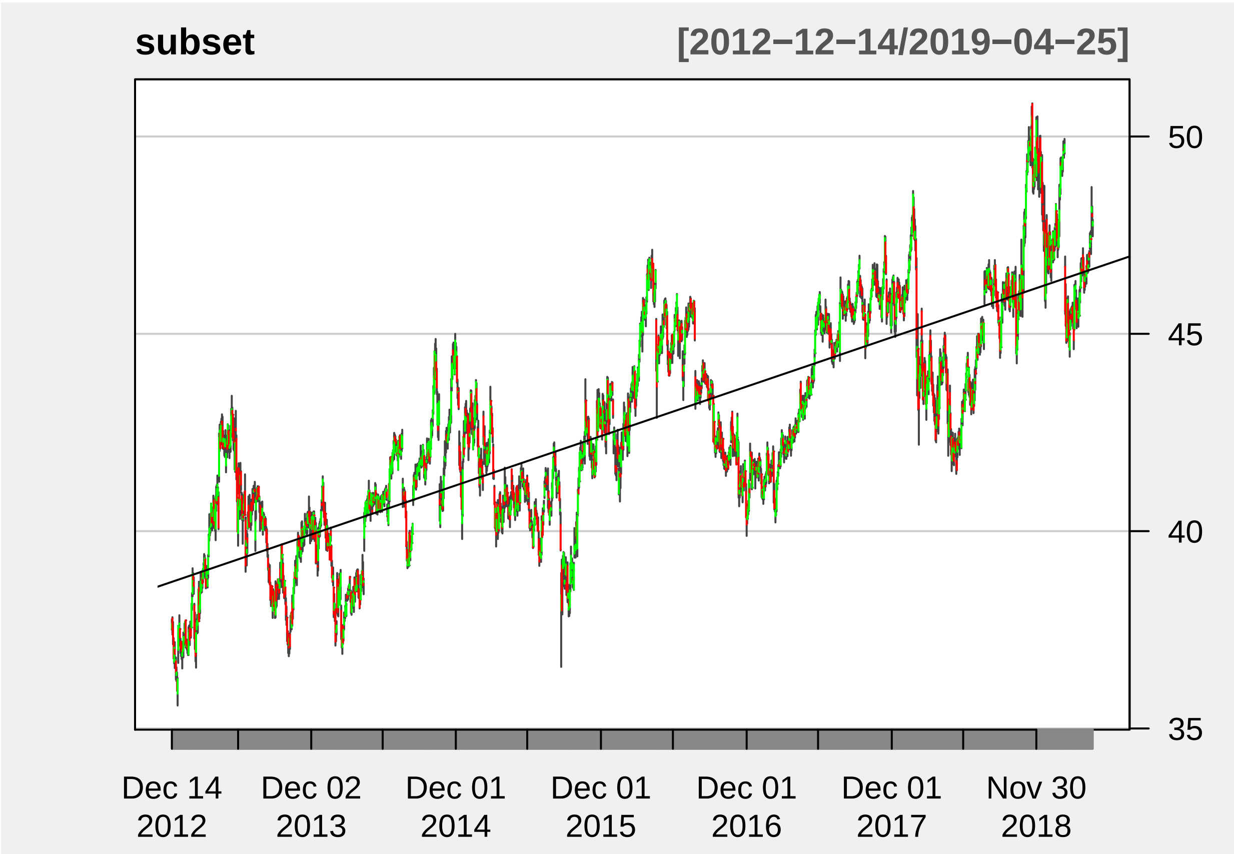
```
subset <- TSLA[1700:nrow(TSLA),]
chartSeries(subset, TA = NULL, theme = "white", up.col = "green", dn.col = "red")
indices = 1:nrow(subset)
model=lm(TSLA.Close~indices,data=subset)
abline(model$coefficients[1],model$coefficients[2])
```



Based on the graph above, Tesla seems to be a stock that is decreasing in value.

Here are the steps for Linear Regression in the Coca Cola dataset. First we restrict the domain of the dataset to only view the most recent trend of the data. We next use Chartseries with subset as the main argument. We then use a linear model on the same range of the chart in addition to plotting the chart.

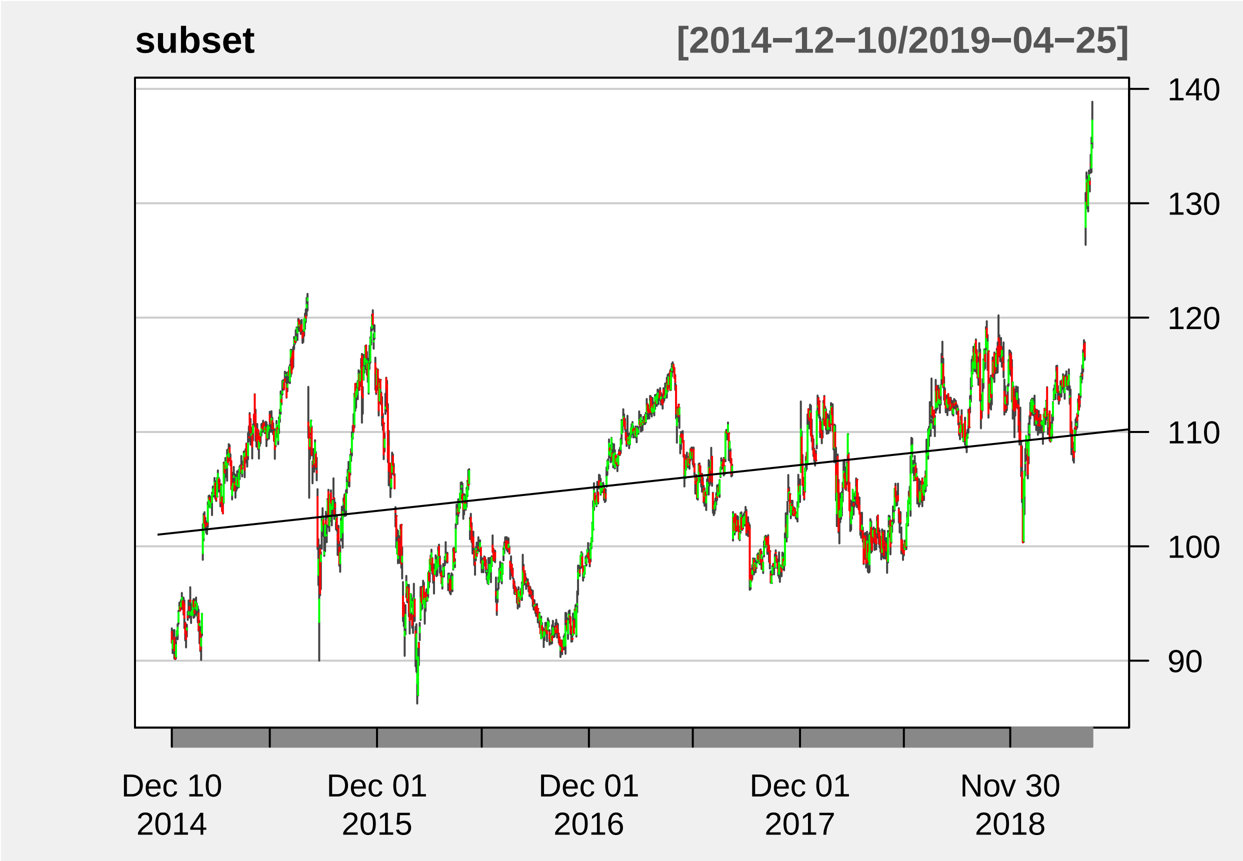
```
subset = KO[1500:nrow(KO),]
chartSeries(subset, TA = NULL, theme = "white", up.col = "green", dn.col = "red")
indices = 1:nrow(subset)
model=lm(KO.Close~indices,data=subset)
abline(model$coefficients[1],model$coefficients[2])
```



Based on the graph above, Coca Cola seems to be a stock that is increasing in value.

Here are the steps for Linear Regression in the Disney dataset. First we restrict the domain of the dataset to only view the most recent trend of the data. We next use Chartseries with subset as the main argument. We then use a linear model on the same range of the chart in addition to plotting the chart.

```
subset <- DIS[2000:nrow(DIS),]
chartSeries(subset, TA = NULL, theme = "white", up.col = "green", dn.col = "red")
indices = 1:nrow(subset)
model=lm(DIS.Close~indices,data=subset)
abline(model$coefficients[1],model$coefficients[2])
```



Based on the graph above, Disney seems to be a stock that is increasing in value.

Conclusion

Based on the pros and cons of each method of forecasting, we are going to weigh each model differently, placing higher weight on Time-Series and Current Events, and lower weights on Neural Networks and Linear Regression. The different weights for each model will be based on how we expect each model to contribute to our conclusion. Since our pros and cons for the Time Series and the Current Events has led us to believe more in those methods, they will have higher contribution than Neural Networks and Linear Regression. If we learn that our method is successful, we think it could be very useful in predicting the outcome of future stocks for a variety of other companies as well.

In terms of Tesla, we predicted that the current events would have an increase for the stock value. The Holt-winters analysis predicted a large range of values for the future, which creates a lot of uncertainty. The Linear regression analysis showed that the Tesla stock tended to decrease. The Neural Network analysis did not really fit the Tesla data and we wouldn't feel safe using this, in a different context, to predict Tesla's future. Thus, we are not sure what will happen with the Tesla stock in the future due to the conflicting analysis' for Linear Regression and Current Events.

For Coca Cola, we found that in almost every test that it would end up growing. For Current Events, we found that everything that is happening for that company overall benefits them and Holt Winters Time Series we found that the range of values isn't nearly as volatile as Tesla's. These two tests we valued higher than the other two, but despite that, we found that Linear Regression also increased alongside having Neural Networks be very accurate in predictions. With that knowledge, if we could apply Neural Networks in a different context and predict it, we believe it would also see an increase. Overall, all of our tests tend to lead towards Coca Cola only growing from here.

Finally, the Disney stock should grow slowly based on the following results. The Current Event analysis had the stock increasing, the Linear Regression indicated in an increasing stock value, and the Holt-Winters had a smaller range of possible values for Disney. The Neural Networks was also very accurate in its analysis. With that knowledge, if we could apply Neural Networks in a different context and predict it, we believe it would also see an increase.

From this project, we noticed a few trends that were apparent. First of all, if Holt Winters performed well and had not that much uncertainty, we found that Neural Networks would perform very well. If it did, Neural Networks struggled to perform, as seen in the tests ran on Tesla. In addition, we found that for all 3 companies, current events would always lead to us believing in an increase to the stock value. This could be very dependent on our biases, in the sense of how we value each event, in addition to news source articles that we received the information from. Overall, it was very interesting applying and learning new ways to predict real-life situations.