

Stock prices forecasting project

Erika Harrell

2/21/2021

Disclaimer: This project was conducted for educational purposes only. Do not attempt to use results of this analysis to make financial decisions. The views expressed in this document do not necessarily reflect the views of the U.S. Department of Justice.

Summary

This project compares the high stock prices for 5 stocks: Apple, Netflix, CBS/Viacom, Amazon, and Disney. It took prices from the Yahoo Finance website <https://finance.yahoo.com/> for the dates from September 23, 2020 to November 17, 2020 when the New York Stock Exchange was open. For each stock, it used the high prices for the first 24 dates in the time period to predict the high prices for the next 16 dates. For Amazon, Apple, and Netflix high stock prices, the forecasted model performed better than the average one-step, naïve forecast computed in-sample. However, for CBS/Viacom and Disney high stock prices, the forecasted model performed worse than the average one-step, naïve forecast computed in-sample.

```
#attach libraries
```

```
library(ggplot2)
```

```
library(ggthemes)
```

```
library(tidyr)
```

```
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
```

```
## Registered S3 method overwritten by 'quantmod':
```

```
##      method      from
```

```
##      as.zoo.data.frame zoo
```

```
library(dplyr)
```

```
##  
## Attaching package: 'dplyr'  
  
## The following objects are masked from 'package:xts':  
##  
##     first, last  
  
## The following objects are masked from 'package:stats':  
##  
##     filter, lag  
  
## The following objects are masked from 'package:base':  
##  
##     intersect, setdiff, setequal, union
```

```
library(forecast)
```

Loading Data

Creating data folder, downloading datasets from Yahoo Finance and loading data into R Studio.

```
if(!dir.exists("./data")) {dir.create("./data")}  
download.file("https://query1.finance.yahoo.com/v7/finance/download/AMZN?period1=1600819200&period2=1600819200&events=history&includeAdjustedClose=true")  
download.file("https://query1.finance.yahoo.com/v7/finance/download/AAPL?period1=1600819200&period2=1600819200&events=history&includeAdjustedClose=true")  
download.file("https://query1.finance.yahoo.com/v7/finance/download/VIIAC?period1=1600819200&period2=1600819200&events=history&includeAdjustedClose=true")  
download.file("https://query1.finance.yahoo.com/v7/finance/download/DIS?period1=1600819200&period2=1600819200&events=history&includeAdjustedClose=true")  
download.file("https://query1.finance.yahoo.com/v7/finance/download/NFLX?period1=1600819200&period2=1600819200&events=history&includeAdjustedClose=true")  
amazon<-read.csv("./data/amazon.csv")  
apple<-read.csv("./data/apple.csv")  
disney<-read.csv("./data/disney.csv")  
cbs<-read.csv("./data/cbs.csv")  
netflix<-read.csv("./data/netflix.csv")
```

Data Wrangling

Merge into a single dataset and look at merged dataset.

```
## 'data.frame':   40 obs. of  7 variables:  
##  $ Date      : chr  "2020-09-23" "2020-09-24" "2020-09-25" "2020-09-28" ...  
##  $ Open      : num  3120 2978 3055 3149 3175 ...  
##  $ High      : num  3127 3069 3102 3175 3188 ...  
##  $ Low       : num  2992 2965 2999 3117 3133 ...
```

```

## $ Close      : num  3000 3020 3095 3174 3145 ...
## $ Adj.Close: num  3000 3020 3095 3174 3145 ...
## $ Volume     : int  5652700 5529400 4615200 4224200 3495800 4896100 4971900 5613100 3775300 5086900 .

## 'data.frame':   40 obs. of  7 variables:
## $ Date       : chr  "2020-09-23" "2020-09-24" "2020-09-25" "2020-09-28" ...
## $ Open       : num  112 105 108 115 115 ...
## $ High       : num  112 110 112 115 115 ...
## $ Low        : num  107 105 108 113 114 ...
## $ Close      : num  107 108 112 115 114 ...
## $ Adj.Close: num  106 108 112 114 113 ...
## $ Volume     : int  150718700 167743300 149981400 137672400 99382200 142675200 116120400 144712000 100000000 100000000

## 'data.frame':   40 obs. of  7 variables:
## $ Date       : chr  "2020-09-23" "2020-09-24" "2020-09-25" "2020-09-28" ...
## $ Open       : num  29.7 28.6 28.9 30 29.6 ...
## $ High       : num  30 29.5 29.6 30.2 29.9 ...
## $ Low        : num  28.6 28.1 28.8 29.6 28.9 ...
## $ Close      : num  28.8 29.1 29.5 29.8 29 ...
## $ Adj.Close: num  28.2 28.5 28.9 29.2 28.4 ...
## $ Volume     : int  9744900 13542900 9418000 9323400 11466500 9358800 13944100 9109000 5874900 8724400

## 'data.frame':   40 obs. of  7 variables:
## $ Date       : chr  "2020-09-23" "2020-09-24" "2020-09-25" "2020-09-28" ...
## $ Open       : num  127 122 121 126 126 ...
## $ High       : num  127 124 124 127 126 ...
## $ Low        : num  123 121 121 125 124 ...
## $ Close      : num  123 122 124 126 125 ...
## $ Adj.Close: num  123 122 124 126 125 ...
## $ Volume     : int  8323600 8480000 6851800 6283700 7405800 13642500 8908300 6490200 5919200 9052000

## 'data.frame':   40 obs. of  7 variables:
## $ Date       : chr  "2020-09-23" "2020-09-24" "2020-09-25" "2020-09-28" ...
## $ Open       : num  491 471 474 489 490 ...
## $ High       : num  491 477 485 492 496 ...
## $ Low        : num  469 468 468 478 487 ...
## $ Close      : num  471 473 483 491 493 ...
## $ Adj.Close: num  471 473 483 491 493 ...
## $ Volume     : int  3726400 3727200 3769400 4773500 3541500 4634100 8153700 6071200 4088100 4199000 .

##      Date      amazonOpen    amazonHigh    amazonLow
## Min.   :2020-09-23   Min.   :2978      Min.   :3069      Min.   :2950
## 1st Qu.:2020-10-06   1st Qu.:3140      1st Qu.:3173      1st Qu.:3086
## Median :2020-10-20   Median :3187      Median :3209      Median :3137
## Mean   :2020-10-20   Mean   :3193      Mean   :3235      Mean   :3140
## 3rd Qu.:2020-11-03   3rd Qu.:3226      3rd Qu.:3289      3rd Qu.:3179
## Max.   :2020-11-17   Max.   :3468      Max.   :3496      Max.   :3424
## amazonClose  amazonAdj.Close  amazonVolume    appleOpen
## Min.   :3000      Min.   :3000      Min.   :3174100   Min.   :105.2
## 1st Qu.:3128      1st Qu.:3128      1st Qu.:4304800   1st Qu.:114.0
## Median :3181      Median :3181      Median :4939900   Median :116.0
## Mean   :3185      Mean   :3185      Mean   :5189758   Mean   :115.9
## 3rd Qu.:3226      3rd Qu.:3226      3rd Qu.:5795150   3rd Qu.:118.8

```

```

## Max. :3444 Max. :3444 Max. :8386400 Max. :125.3
## appleHigh appleLow appleClose appleAdj.Close
## Min. :110.2 Min. :105.0 Min. :107.1 Min. :106.4
## 1st Qu.:115.5 1st Qu.:112.3 1st Qu.:114.7 1st Qu.:114.0
## Median :117.3 Median :114.6 Median :116.0 Median :115.3
## Mean :117.6 Mean :114.2 Mean :115.9 Mean :115.2
## 3rd Qu.:119.9 3rd Qu.:116.6 3rd Qu.:119.0 3rd Qu.:118.3
## Max. :125.4 Max. :119.7 Max. :124.4 Max. :123.6
## appleVolume cbsOpen cbsHigh cbsLow
## Min. : 74271000 Min. :27.15 Min. :27.77 Min. :26.99
## 1st Qu.:101617725 1st Qu.:28.07 1st Qu.:28.52 1st Qu.:27.50
## Median :118379850 Median :28.84 Median :29.50 Median :28.52
## Mean :127686085 Mean :28.94 Mean :29.49 Mean :28.42
## 3rd Qu.:145066300 3rd Qu.:29.79 3rd Qu.:30.11 3rd Qu.:28.90
## Max. :262330500 Max. :31.60 Max. :32.89 Max. :30.99
## cbsClose cbsAdj.Close cbsVolume disneyOpen
## Min. :27.13 Min. :26.56 Min. : 4870500 Min. :118.2
## 1st Qu.:28.10 1st Qu.:27.51 1st Qu.: 7796575 1st Qu.:123.2
## Median :28.89 Median :28.27 Median : 9282400 Median :125.0
## Mean :28.98 Mean :28.37 Mean :10299698 Mean :127.3
## 3rd Qu.:29.60 3rd Qu.:28.98 3rd Qu.:11809500 3rd Qu.:127.9
## Max. :32.29 Max. :31.61 Max. :26843600 Max. :144.4
## disneyHigh disneyLow disneyClose disneyAdj.Close
## Min. :121.5 Min. :117.2 Min. :118.5 Min. :118.5
## 1st Qu.:124.2 1st Qu.:122.2 1st Qu.:123.3 1st Qu.:123.3
## Median :126.9 Median :123.7 Median :125.0 Median :125.0
## Mean :128.8 Mean :125.6 Mean :127.3 Mean :127.3
## 3rd Qu.:128.5 3rd Qu.:126.1 3rd Qu.:127.5 3rd Qu.:127.5
## Max. :147.7 Max. :142.6 Max. :144.7 Max. :144.7
## disneyVolume netflixOpen netflixHigh netflixLow
## Min. : 5177700 Min. :470.5 Min. :477.0 Min. :463.4
## 1st Qu.: 6758225 1st Qu.:486.7 1st Qu.:491.5 1st Qu.:478.2
## Median : 8401800 Median :493.6 Median :505.3 Median :484.9
## Mean :10167048 Mean :503.8 Mean :511.8 Mean :495.0
## 3rd Qu.:11094200 3rd Qu.:518.2 3rd Qu.:530.6 3rd Qu.:507.1
## Max. :35634700 Max. :562.6 Max. :572.5 Max. :541.0
## netflixClose netflixAdj.Close netflixVolume
## Min. :470.5 Min. :470.5 Min. : 3002700
## 1st Qu.:485.0 1st Qu.:485.0 1st Qu.: 4171275
## Median :492.1 Median :492.1 Median : 5255050
## Mean :502.8 Mean :502.8 Mean : 6030592
## 3rd Qu.:525.9 3rd Qu.:525.9 3rd Qu.: 7140300
## Max. :554.1 Max. :554.1 Max. :17405700

## 'data.frame': 40 obs. of 31 variables:
## $ Date : Date, format: "2020-09-23" "2020-09-24" ...
## $ amazonOpen : num 3120 2978 3055 3149 3175 ...
## $ amazonHigh : num 3127 3069 3102 3175 3188 ...
## $ amazonLow : num 2992 2965 2999 3117 3133 ...
## $ amazonClose : num 3000 3020 3095 3174 3145 ...
## $ amazonAdj.Close : num 3000 3020 3095 3174 3145 ...
## $ amazonVolume : num 5652700 5529400 4615200 4224200 3495800 ...
## $ appleOpen : num 112 105 108 115 115 ...
## $ appleHigh : num 112 110 112 115 115 ...

```

```

## $ appleLow      : num  107 105 108 113 114 ...
## $ appleClose    : num  107 108 112 115 114 ...
## $ appleAdj.Close : num  106 108 112 114 113 ...
## $ appleVolume    : num  1.51e+08 1.68e+08 1.50e+08 1.38e+08 9.94e+07 ...
## $ cbsOpen        : num  29.7 28.6 28.9 30 29.6 ...
## $ cbsHigh        : num  30 29.5 29.6 30.2 29.9 ...
## $ cbsLow         : num  28.6 28.1 28.8 29.6 28.9 ...
## $ cbsClose       : num  28.8 29.1 29.5 29.8 29 ...
## $ cbsAdj.Close   : num  28.2 28.5 28.9 29.2 28.4 ...
## $ cbsVolume      : num  9744900 13542900 9418000 9323400 11466500 ...
## $ disneyOpen     : num  127 122 121 126 126 ...
## $ disneyHigh     : num  127 124 124 127 126 ...
## $ disneyLow      : num  123 121 121 125 124 ...
## $ disneyClose    : num  123 122 124 126 125 ...
## $ disneyAdj.Close : num  123 122 124 126 125 ...
## $ disneyVolume   : num  8323600 8480000 6851800 6283700 7405800 ...
## $ netflixOpen    : num  491 471 474 489 490 ...
## $ netflixHigh    : num  491 477 485 492 496 ...
## $ netflixLow     : num  469 468 468 478 487 ...
## $ netflixClose   : num  471 473 483 491 493 ...
## $ netflixyAdj.Close: num  471 473 483 491 493 ...
## $ netflixVolume  : num  3726400 3727200 3769400 4773500 3541500 ...

```

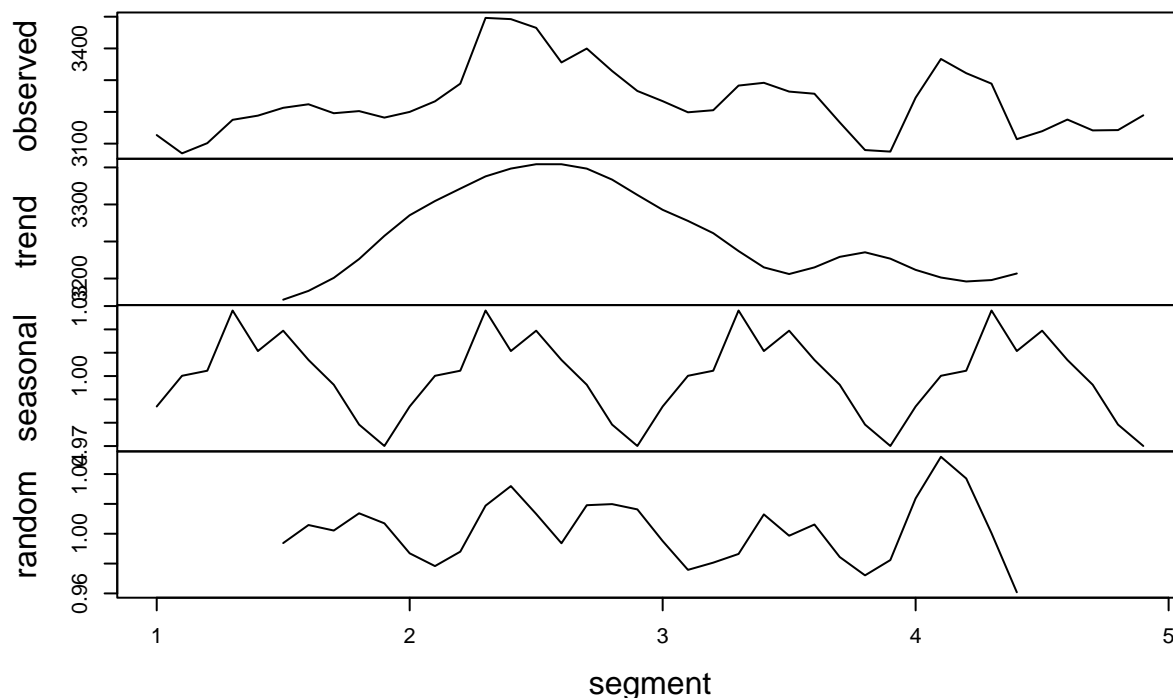
Amazon forecasting analysis

The time chart of the data of the high stock prices for Amazon shows a non-stationary time series. There was an increase in the high stock prices from September 23 to around October 14. From there, the trend generally decreased until around November 2 and increased sharply on November 3. This preceded a decrease in the high stock price until around November 8. After that decrease, the trend generally remained steady.



The data was turned into a time series object in R with 40 observations, one for each day that the stock market was open during the time period. A multiplicative decomposition of the time series was conducted. Plotting the trend-cycle and seasonal indices shows that the data has an upward trend during the 1st 2 segments with a downward trend in the 3rd segment followed by a stable trend in the 4th segment. It also has seasonal fluctuations, with the data increasing at the beginning of each segment, reaching a peak in the middle of the segment and decreasing by the end of the segment. The data also has fairly random residuals.

Decomposition of multiplicative time series



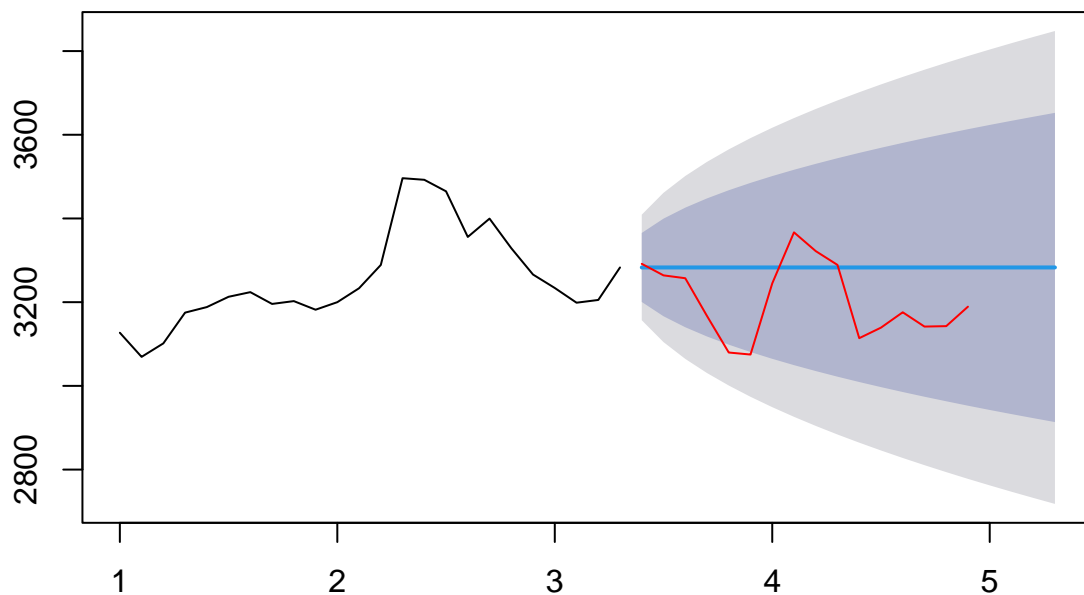
Training and test data sets were created from the time series object with the high stock prices for the first 24 days (60% of the data) being put into the training data while the data for the remaining 16 days (40% of the data) were put into the test data set.

The `ets()` function was applied to the training data to choose the best ets (error, trend, seasonality) model to fit to the data. It returned a model with simple exponential smoothing with multiplicative errors. This model had in a smoothing parameter of 0.9999 which means that in the model, more weight is given to the more recent high stock prices. This ets model was then used to forecast future values. The forecasted data was plotted along with the test data. The plot showed that the test data appears to fall within the 95% prediction intervals from the ets model.

```
## ETS(M,N,N)
##
## Call:
## ets(y = ts1Train)
##
```

```
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 3125.1279
##
## sigma: 0.0196
##
##      AIC      AICc      BIC
## 279.5340 280.7340 283.0681
```

Forecasts from ETS(M,N,N)

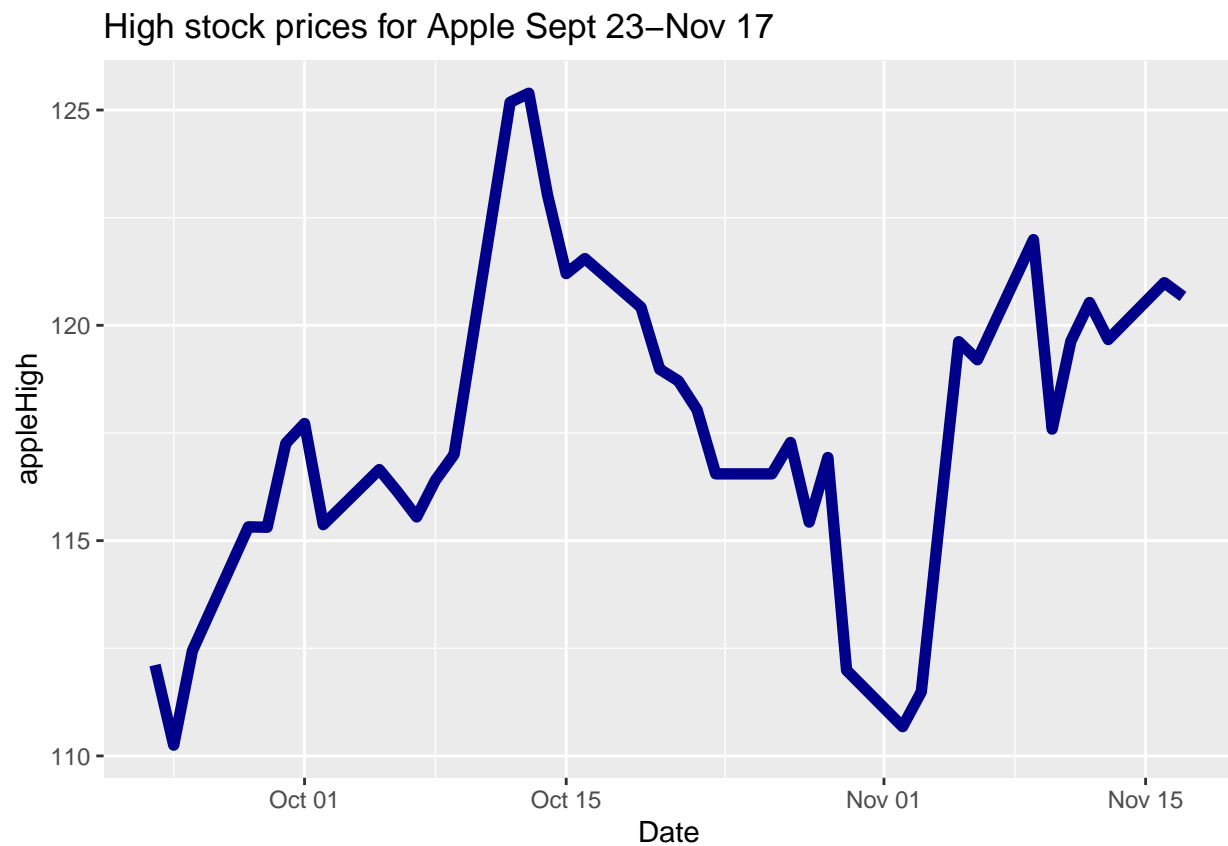


In terms of accuracy, the mean absolute scaled error (MASE) of the forecast was about 0.6 for the test data. With $MASE < 1$, the forecast did better in predicting the later high stock prices than the average one-step, naïve forecast computed in-sample.

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set  6.577504 61.67829 43.88522 0.1883105 1.330191 0.2775902
## Test set     -79.229107 116.80991 96.42549 -2.5462514 3.060235 0.6099267
##           ACF1 Theil's U
## Training set 0.1853532    NA
## Test set     0.5447402 1.498007
```

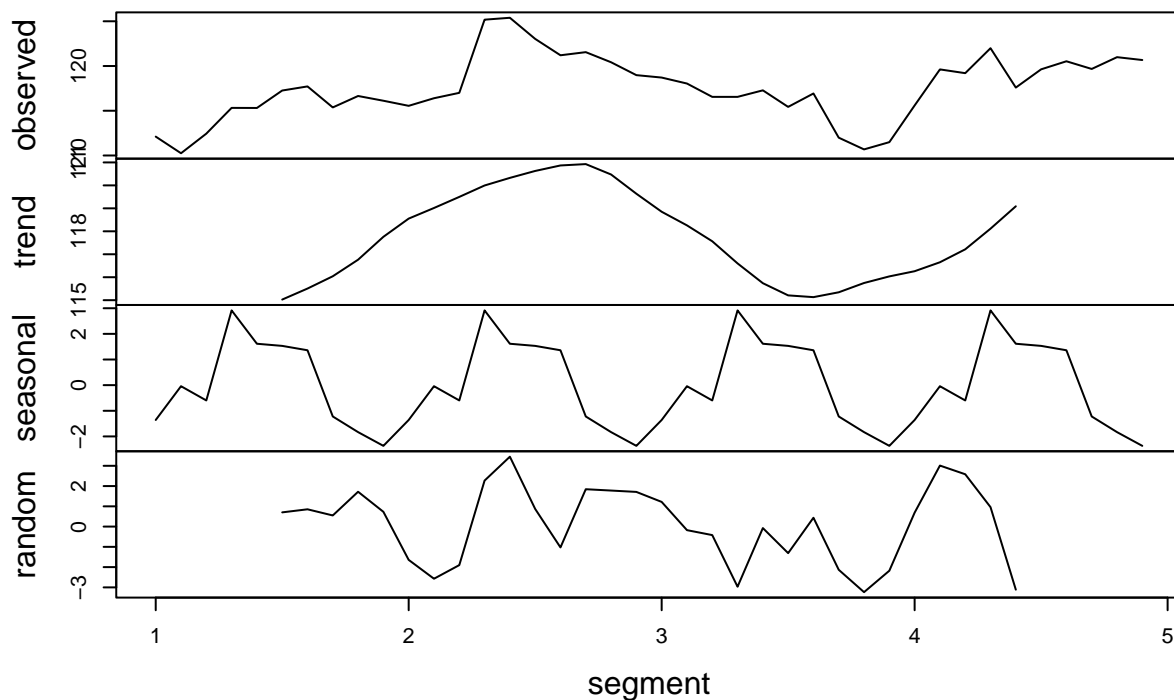

Apple forecasting analysis

The time chart of the data of the high stock prices for Apple shows a non-stationary time series. There was an increase in the high stock prices from September 23 to around October 14. From there, the trend generally decreased until around November 2 and increased sharply on November 3 until around November 7. This preceded a slight decrease in the high stock price until around November 8. After that decrease, the trend generally increased until the end of the time period.



The data was turned into a time series object in R with 40 observations, one for each day that the stock market was open during the time period. An additive decomposition of the time series was conducted. Plotting the trend-cycle and seasonal indices shows that the data have an upward trend during the 1st segment lasting through the 1st half of the 2nd segment. The trend declined from the 2nd half of the 2nd segment until the middle of the 3rd segment. From there, it increased through the 4th segment. It also has seasonal fluctuations, with the data increasing at the beginning of each segment, reaching a peak in the middle of the segment and decreasing by the end of the segment. The data also has fairly random residuals.

Decomposition of additive time series



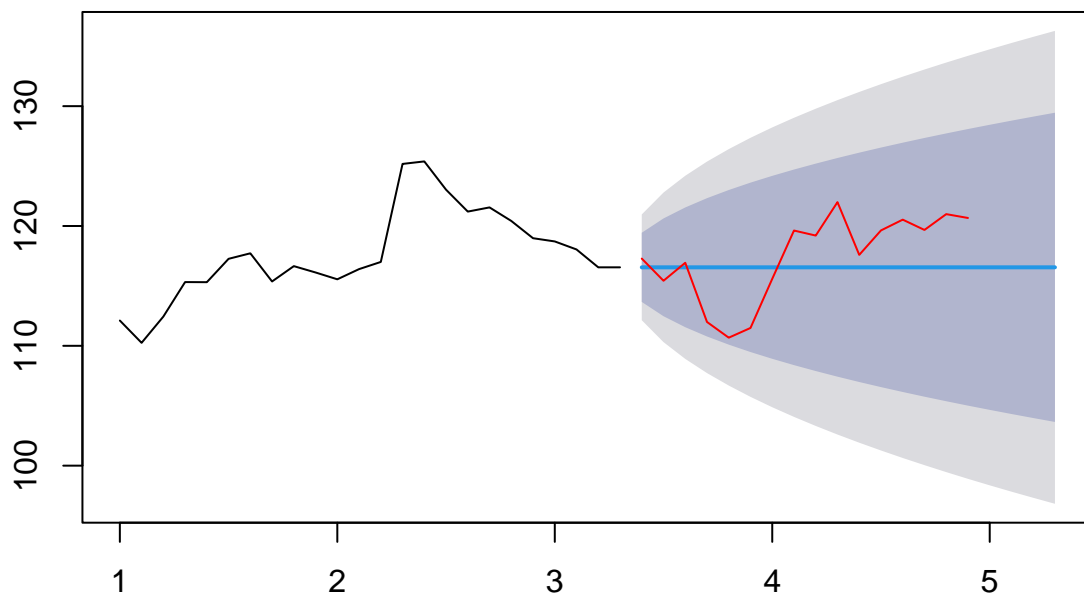
Training and test data sets were created from the time series object with the high stock prices for the first 24 days (60% of the data) being put into the training data while the data for the remaining 16 days (40% of the data) were put into the test data set.

The `ets()` function was applied to the training data to choose the best ets (error, trend, seasonality) model to fit to the data. It returned a model with simple exponential smoothing with multiplicative errors. This model had in a smoothing parameter of 0.9999 which means that in the model, more weight is given to the more recent stock prices. This ets model was then used to forecast future values. The forecasted data was plotted along with the test data. The plot showed that the test data appears to fall within the 80% prediction intervals from the ets model.

```
## ETS(A,N,N)
##
## Call:
## ets(y = ts1Train)
```

```
##
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 112.1082
##
## sigma: 2.2513
##
##      AIC      AICc      BIC
## 119.1378 120.3378 122.6720
```

Forecasts from ETS(A,N,N)

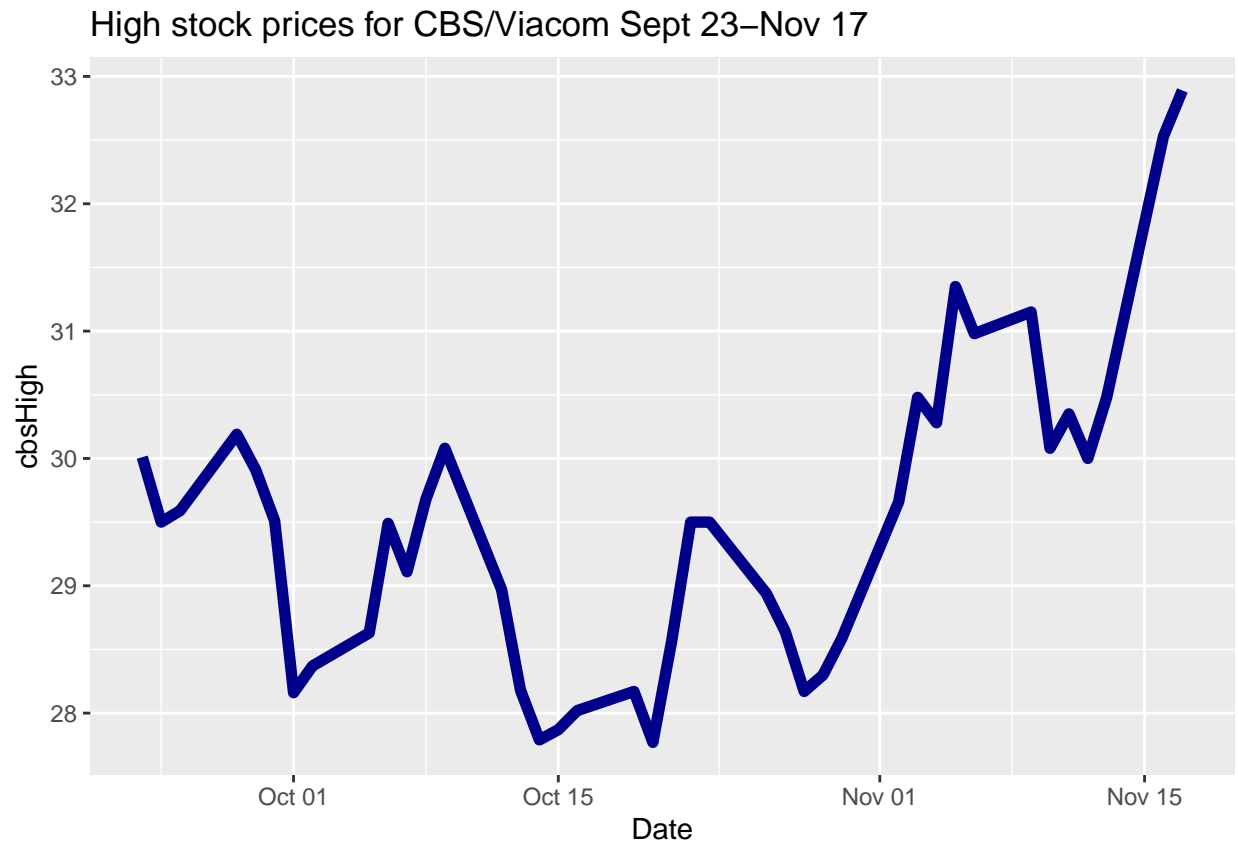


In terms of accuracy, the mean absolute scaled error (MASE) of the forecast was about 0.6 for the test data. With $MASE < 1$, the forecast did better in predicting the later high stock prices than the average one-step, naïve forecast computed in-sample.

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.1850931 2.155474 1.394282 0.1457702 1.173510 0.2787370
## Test set    0.9050000 3.558258 3.101250 0.6833750 2.647713 0.6199843
##           ACF1 Theil's U
## Training set 0.09397187    NA
## Test set    0.70776444 1.422214
```

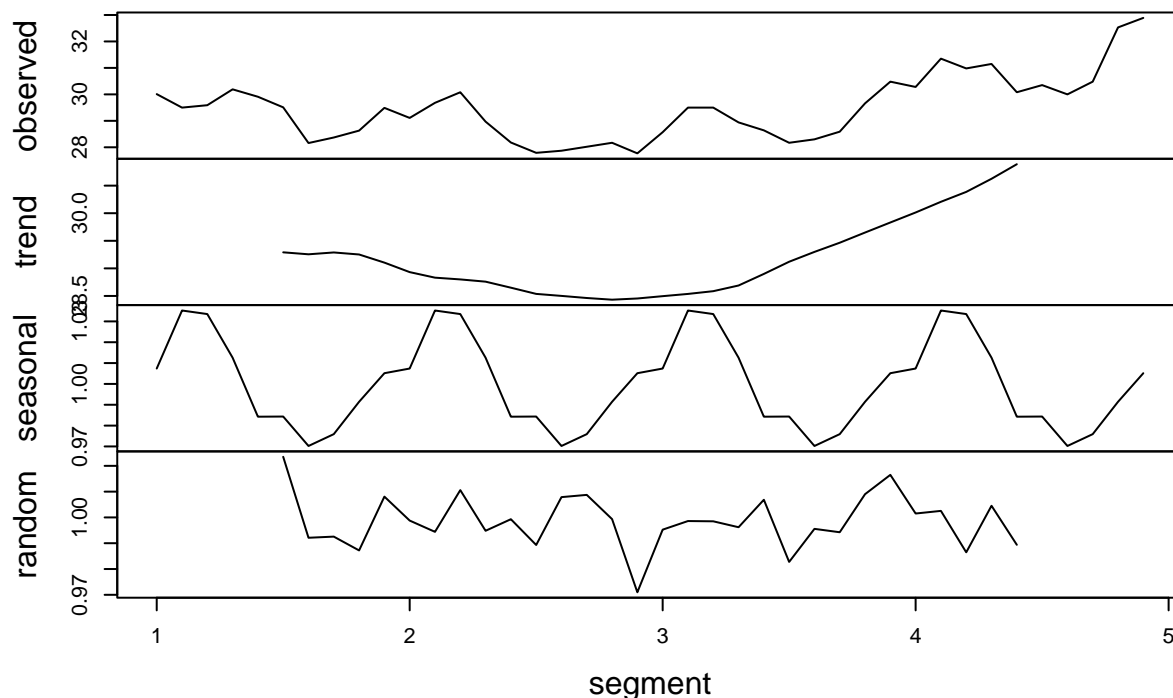
CBS/Viacom forecasting analysis

The plot of time series of the high stock prices for CBS/Viacom shows a more stationary time series than the plots of the Apple and Amazon time series. However, the CBS/Viacom time series is still nonstationary, with the high stock prices increasing from around October 28 until around November 4 and another sharp increase from November 9 to the end of the time series.



The data was turned into a time series object in R with 40 observations, one for each day that the stock market was open during the time period. A multiplicative decomposition of the time series was conducted. Plotting the trend-cycle and seasonal indices shows that the data have a downward trend during the 1st 2 segments with an upward trend in the 3rd and 4th segments. It also has seasonal fluctuations, with the data increasing in the middle of each segment, reaching a peak in the middle of the segment and a peak at the beginning of each segment. The time series also has fairly random residuals.

Decomposition of multiplicative time series



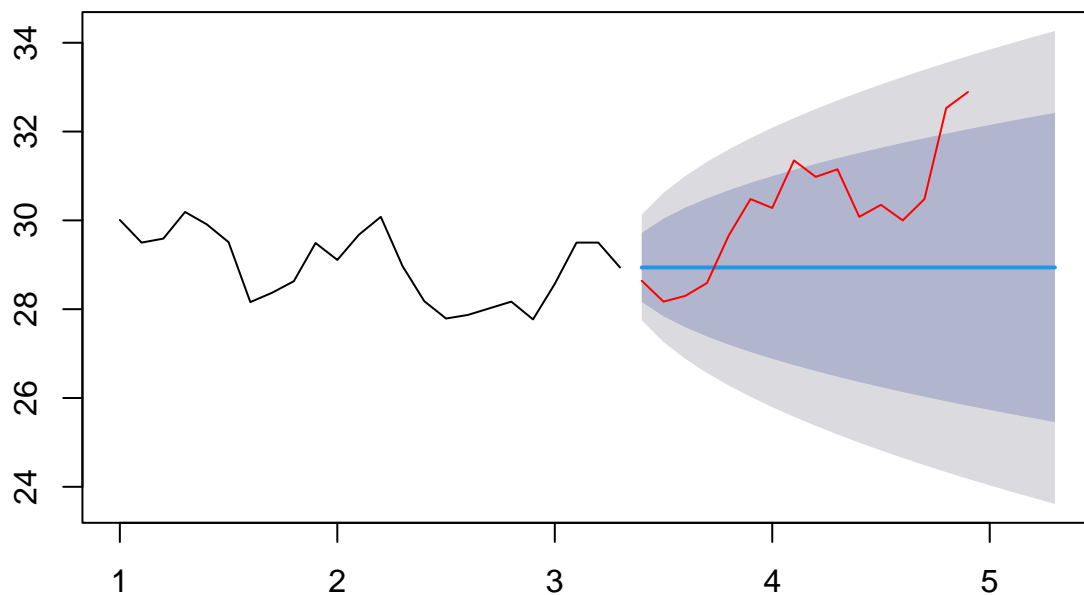
Training and test data sets were created from the time series object with the high stock prices for the first 24 days (60% of the data) being put into the training data while the data for the remaining 16 days (40% of the data) were put into the test data set.

The `ets()` function was applied to the training data to choose the best ets (error, trend, seasonality) model to fit to the data. It returned a model with simple exponential smoothing with multiplicative errors. This model had in a smoothing parameter of 0.9999 which means that in the model, more weight is given to the more recent stock prices. This ets model was then used to forecast future values. The forecasted data was plotted along with the test data. The plot showed that the test data appears to fall within the 95% prediction intervals from the ets model.

```
## ETS(M,N,N)
##
## Call:
## ets(y = ts1Train)
##
```

```
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 29.9983
##
## sigma: 0.0209
##
##      AIC      AICc      BIC
## 56.28348 57.48348 59.81764
```

Forecasts from ETS(M,N,N)



In terms of accuracy, the mean absolute scaled error (MASE) of the forecast was about 2.1 for the test data. With $MASE > 1$, the forecast did worse in predicting the later high stock prices than the average one-step, naïve forecast computed in-sample.

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.04409949 0.584818 0.4700718 -0.1700425 1.624446 0.633398
## Test set      1.30556897 1.871256 1.5630970  4.1293770 5.037798 2.106194
##           ACF1 Theil's U
## Training set 0.1531483    NA
## Test set     0.6702398 2.377506
```

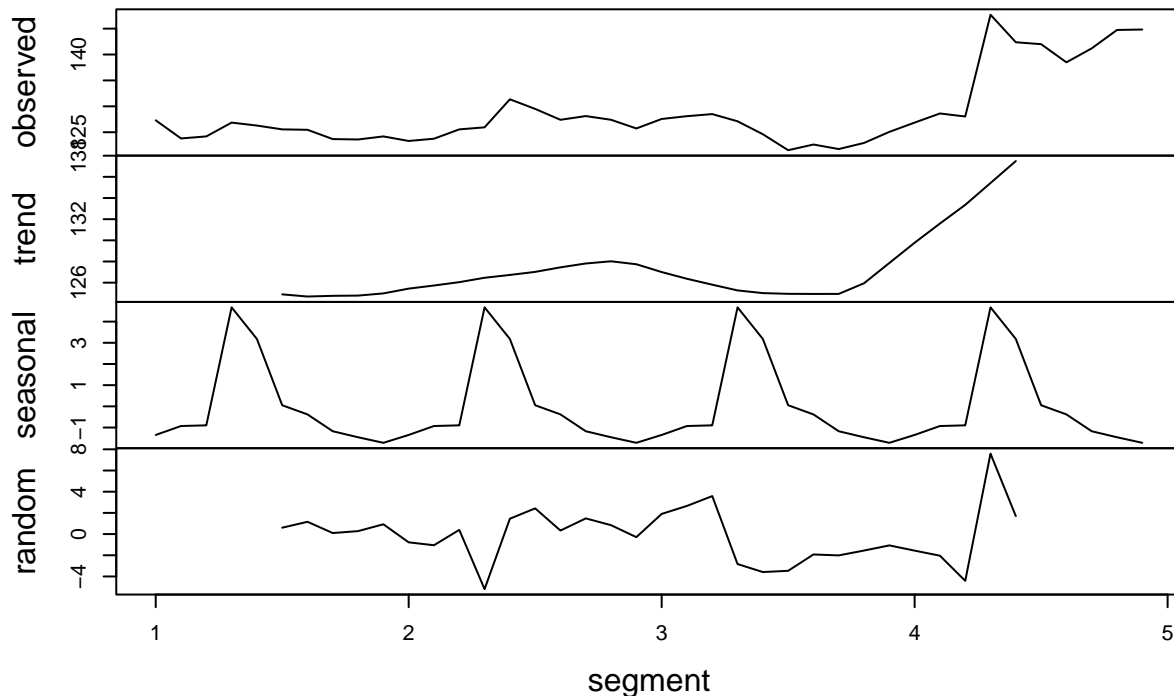
Disney forecasting analysis

The time chart of the data of the high stock prices for Disney shows a non-stationary time series. This is primarily due to the trend at the end of the time period where the high stock prices increase sharply from around November 6 to November 8.



The data was turned into a time series object in R with 40 observations, one for each day that the stock market was open during the time period. An additive decomposition of the time series was conducted. Plotting the trend-cycle and seasonal indices shows that the trend was basically stable until a sharp increase starting at the end of the 3rd segment. It also has seasonal fluctuations, with the data increasing at the beginning of each segment, reaching a peak in the middle of the segment and decreasing by the end of the segment. The data also had random residuals.

Decomposition of additive time series



Training and test data sets were created from the time series object with the high stock prices for the first 24 days (60% of the data) being put into the training data while the data for the remaining 16 days (40% of the data) were put into the test data set.

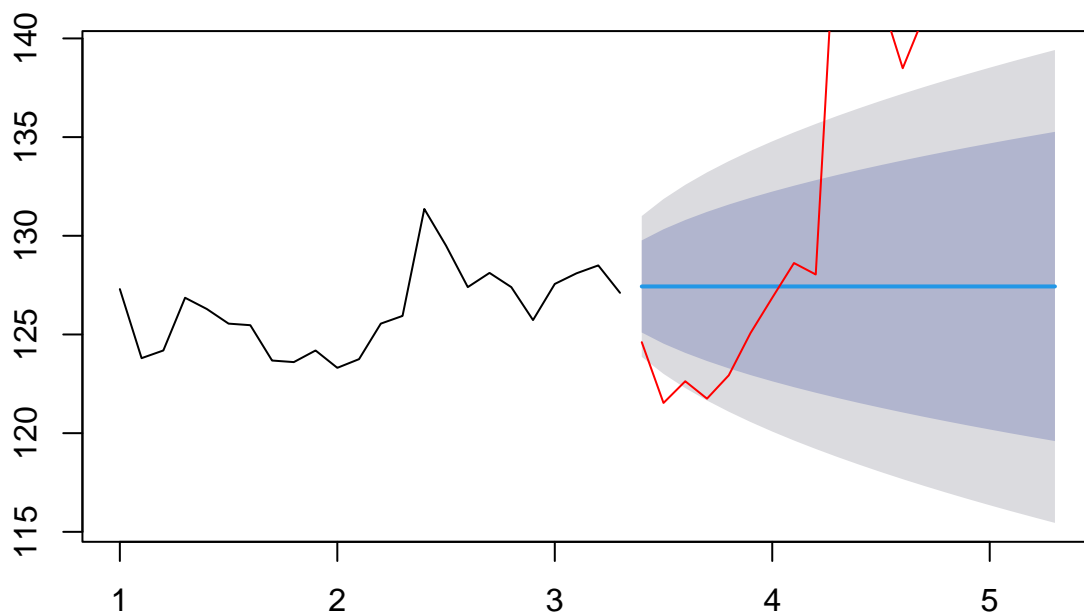
The `ets()` function was applied to the training data to choose the best ets (error, trend, seasonality) model to fit to the data. It returned a model with simple exponential smoothing with additive errors. This model had in a smoothing parameter of 0.7352 which means that in the model, more weight is given to the more recent stock prices. This ets model was then used to forecast future values. The forecasted data was plotted along with the test data. The plot showed that a good portion of the test data fell outside the 80% and 95% prediction intervals from the ets model.

```
## ETS(A,N,N)
##
## Call:
## ets(y = ts1Train)
##
```



```
## Smoothing parameters:
##   alpha = 0.7352
##
## Initial states:
##   l = 126.447
##
## sigma: 1.8208
##
##      AIC      AICc      BIC
## 108.9505 110.1505 112.4846
```

Forecasts from ETS(A,N,N)

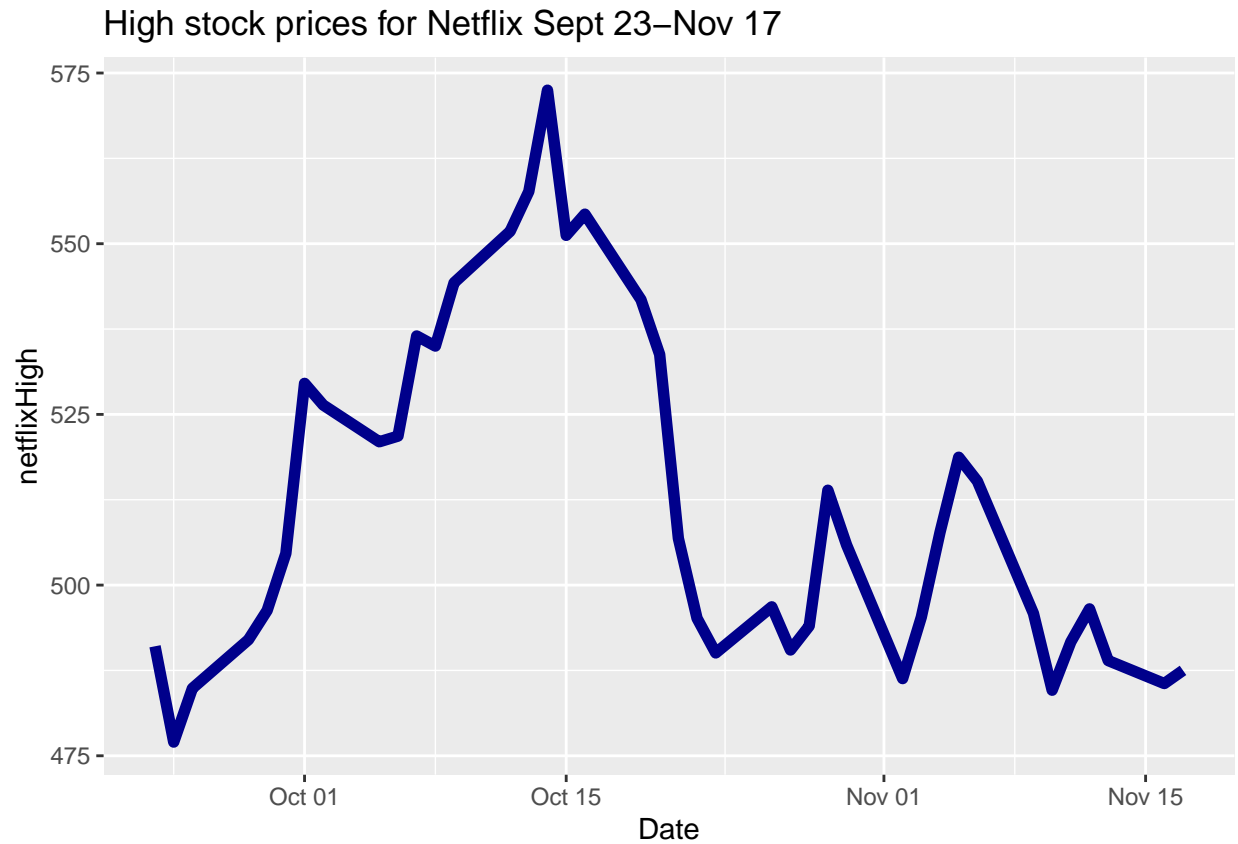


In terms of accuracy, the mean absolute scaled error (MASE) of the forecast was about 3.0 for the test data. With $MASE > 1$, the forecast did worse in predicting the later high stock prices than the average one-step, naïve forecast computed in-sample.

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.05591236  1.743291  1.242970  0.03219821  0.9798233  0.4375556
## Test set    5.27529531 10.854211  8.607068  3.48983108  6.2040540  3.0298958
##           ACF1 Theil's U
## Training set 0.05275571      NA
## Test set    0.76285026  1.879288
```

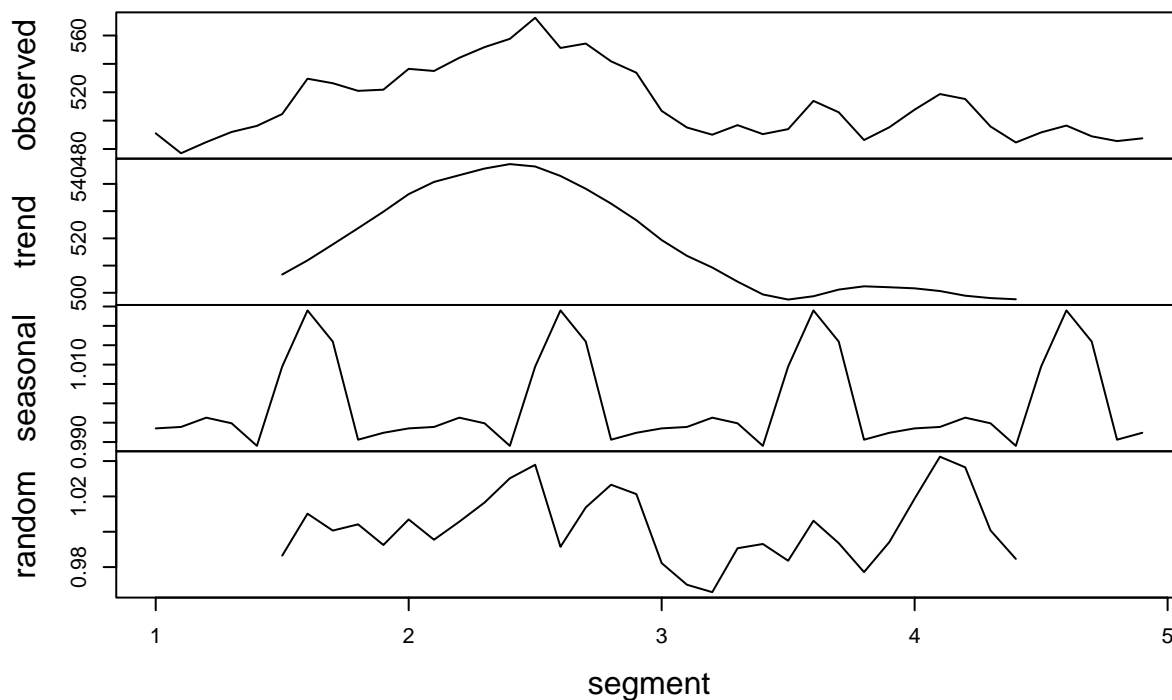
Netflix forecasting analysis

The time chart of the data of the high stock prices for Netflix shows a non-stationary time series. This is primarily due to the activity of the time series at the beginning of the time period. The data increases from the beginning of the time period to October 14. It then decreases until around October 22. Afterwards, it begins to become more stationary with a few fluctuations until around November 9.



The data was turned into a time series object in R with 40 observations, one for each day that the stock market was open during the time period. An multiplicative decomposition of the time series was conducted. Plotting the trend-cycle and seasonal indices shows that the data increased from the middle of the 1st segment until the middle of the 2nd segment. Afterwards, it decreased until the middle of the 3rd segment and remained flat until the end of the time period. It also has seasonal fluctuations, with the data increasing at the beginning of each segment, reaching a peak in the middle of the segment and decreasing by the end of the segment. The data also has fairly random residuals.

Decomposition of multiplicative time series



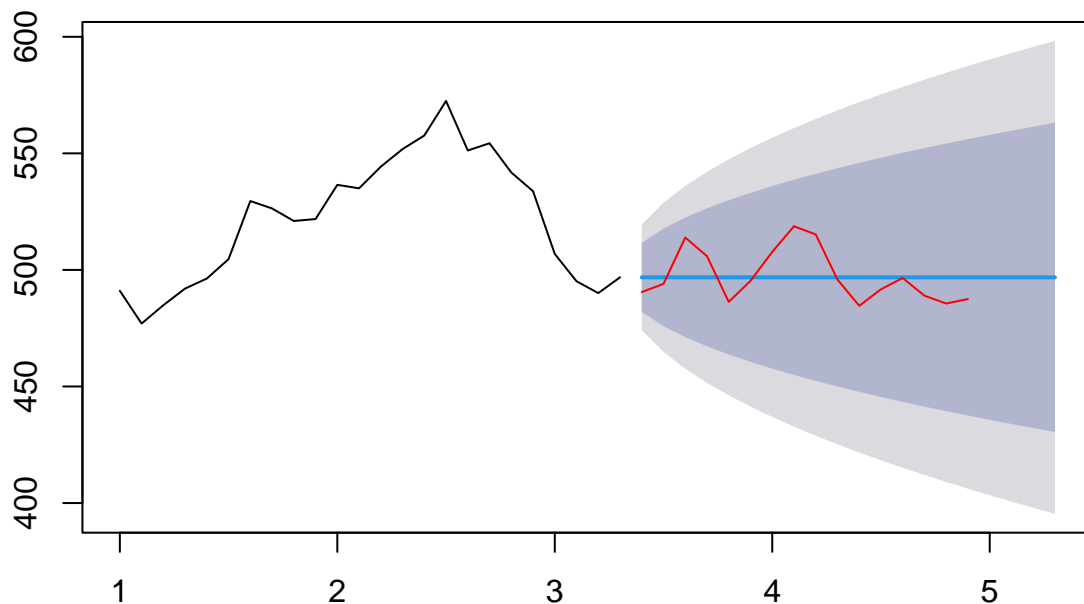
Training and test data sets were created from the time series object with the high stock prices for the first 24 days (60% of the data) being put into the training data while the data for the remaining 16 days (40% of the data) were put into the test data set.

The `ets()` function was applied to the training data to choose the best ets (error, trend, seasonality) model to fit to the data. It returned a model with simple exponential smoothing with multiplicative errors. This model had in a smoothing parameter of 0.9999 which means that in the model, more weight is given to the more recent stock prices. This ets model was then used to forecast future values. The forecasted data was plotted along with the test data. The plot showed that the test data appears to fall within the 80% prediction intervals from the ets model.

```
## ETS(M,N,N)
##
## Call:
## ets(y = ts1Train)
```

```
##
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 490.7364
##
## sigma: 0.0232
##
##      AIC      AICc      BIC
## 199.8201 201.0201 203.3542
```

Forecasts from ETS(M,N,N)



In terms of accuracy, the mean absolute scaled error (MASE) of the forecast was about 0.2 for the test data. With $MASE < 1$, the forecast did better in predicting the later high stock prices than the average one-step, naïve forecast computed in-sample.

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.2534795 11.70942 9.389351 0.02641231 1.797039 0.2144387
## Test set    0.5700515 10.96571 9.099121 0.06687166 1.815276 0.2078103
##           ACF1 Theil's U
## Training set 0.1710957      NA
## Test set    0.4742796 1.001417
```