Assignment 3

Samia Banafunzi, Mercy Maina, Nachiket Patil, Filipe Soares

07/03/2021

```
setwd('C:/Users/filip/Desktop')
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

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

```
library(lubridate)
```

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

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(tseries)
library(Metrics)
library(ggplot2)
library(timeSeries)
```

```
## Loading required package: timeDate

##
## Attaching package: 'timeSeries'

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

```
library(forecast)
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:Metrics':
##
##     accuracy
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:timeSeries':
##
##     filter

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(prophet)
```

```
## Loading required package: Rcpp

## Loading required package: rlang

##
## Attaching package: 'rlang'

## The following object is masked from 'package:Metrics':
##
##     ll
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------- tidyverse 1.3.0 --

## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------ tidyverse_conflicts() --
## x purrr::%@%()             masks rlang::%@%()
## x lubridate::as.difftime() masks base::as.difftime()
## x purrr::as_function()     masks rlang::as_function()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks plotly::filter(), timeSeries::filter(), stats::filter()
## x dplyr::first()           masks xts::first()
## x purrr::flatten()         masks rlang::flatten()
## x purrr::flatten_chr()     masks rlang::flatten_chr()
## x purrr::flatten_dbl()     masks rlang::flatten_dbl()
## x purrr::flatten_int()     masks rlang::flatten_int()
## x purrr::flatten_lgl()     masks rlang::flatten_lgl()
## x purrr::flatten_raw()     masks rlang::flatten_raw()
## x lubridate::intersect()   masks base::intersect()
```

```
## x purrr::invoke()           masks rlang::invoke()
## x dplyr::lag()              masks timeSeries::lag(), stats::lag()
## x dplyr::last()             masks xts::last()
## x purrr::list_along()       masks rlang::list_along()
## x rlang::ll()               masks Metrics::ll()
## x purrr::modify()           masks rlang::modify()
## x purrr::prepend()          masks rlang::prepend()
## x lubridate::setdiff()      masks base::setdiff()
## x purrr::splice()           masks rlang::splice()
## x lubridate::union()        masks base::union()
```

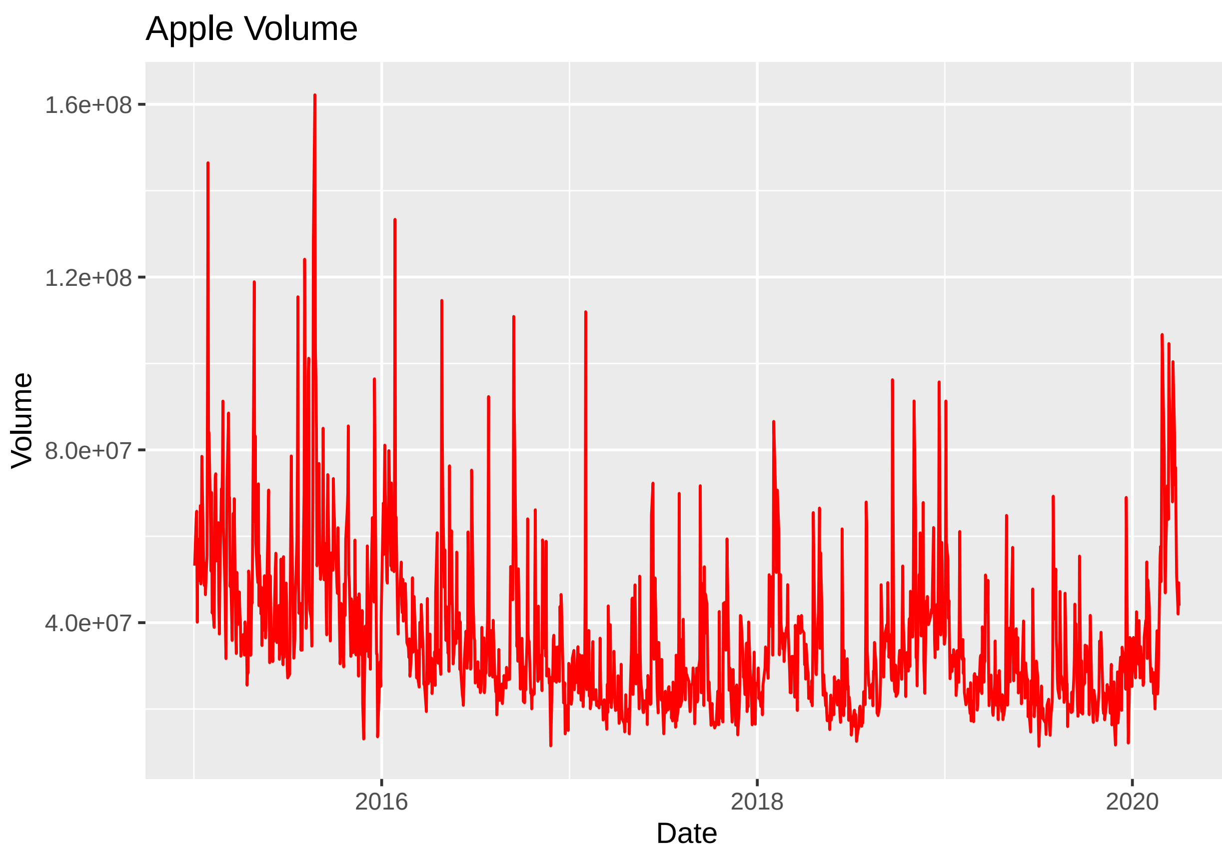
```
library(dplyr)
```

```
# Import Data
apple = read.csv('AAPL.csv')
amgen = read.csv('AMGN.csv')
comcast = read.csv('CMCSA.csv')
gilead = read.csv('GILD.csv')
microsoft = read.csv('MSFT.csv')
netflix = read.csv('NFLX.csv')
```

```
# Convert to Date

apple$Date <- as.Date(apple$Date, format= "%Y-%m-%d")
amgen$Date <- as.Date(amgen$Date, format= "%Y-%m-%d")
comcast$Date <- as.Date(comcast$Date, format= "%Y-%m-%d")
gilead$Date <- as.Date(gilead$Date, format= "%Y-%m-%d")
microsoft$Date <- as.Date(microsoft$Date, format= "%Y-%m-%d")
netflix$Date <- as.Date(netflix$Date, format= "%Y-%m-%d")
```

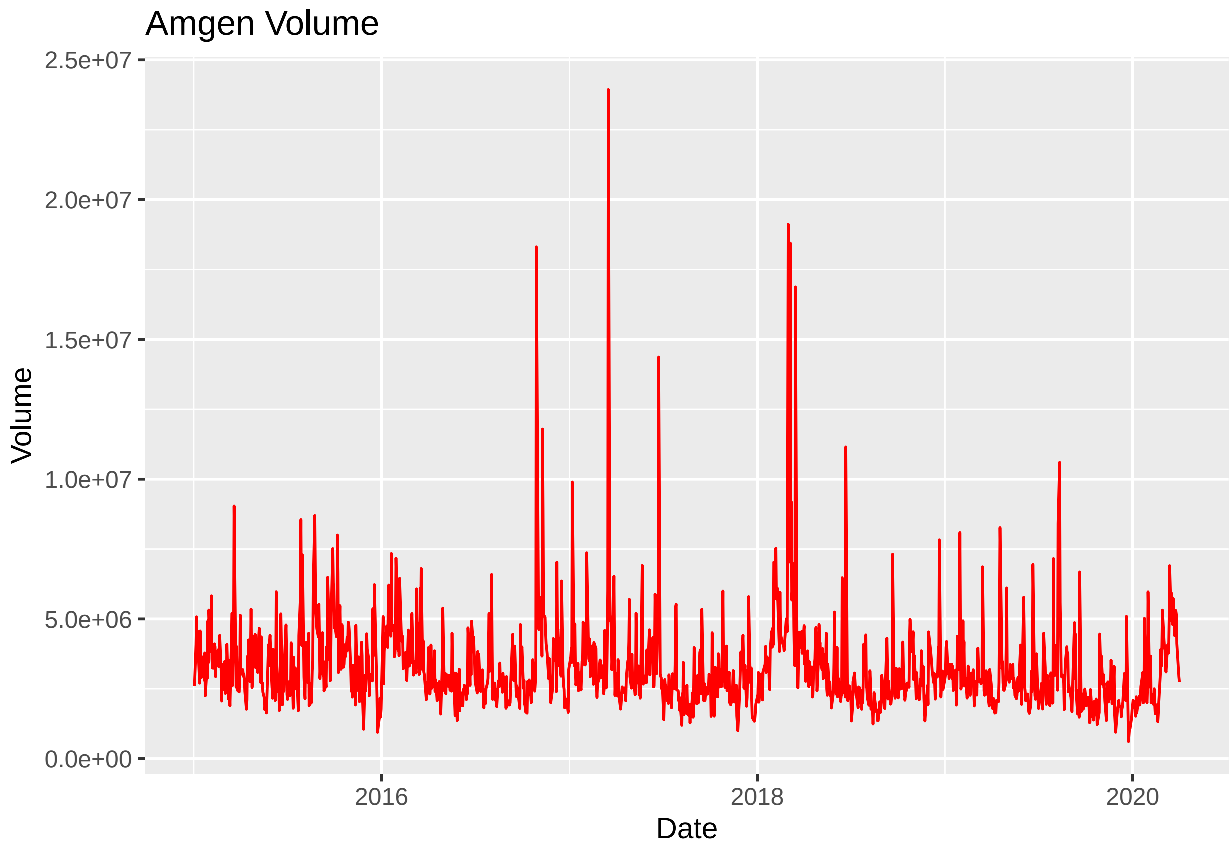
```
# Drop all dates before January 1 2015

app_drop <- subset(apple, Date>= "2015-01-01")
amg_drop <- subset(amgen, Date>= "2015-01-01")
com_drop <- subset(comcast, Date>= "2015-01-01")
gil_drop <- subset(gilead, Date>= "2015-01-01")
mic_drop <- subset(microsoft, Date>= "2015-01-01")
net_drop <- subset(netflix, Date>= "2015-01-01")
```

```
# Plot Volume and Closing Price
ggplot(data=app_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Apple Volume")
```

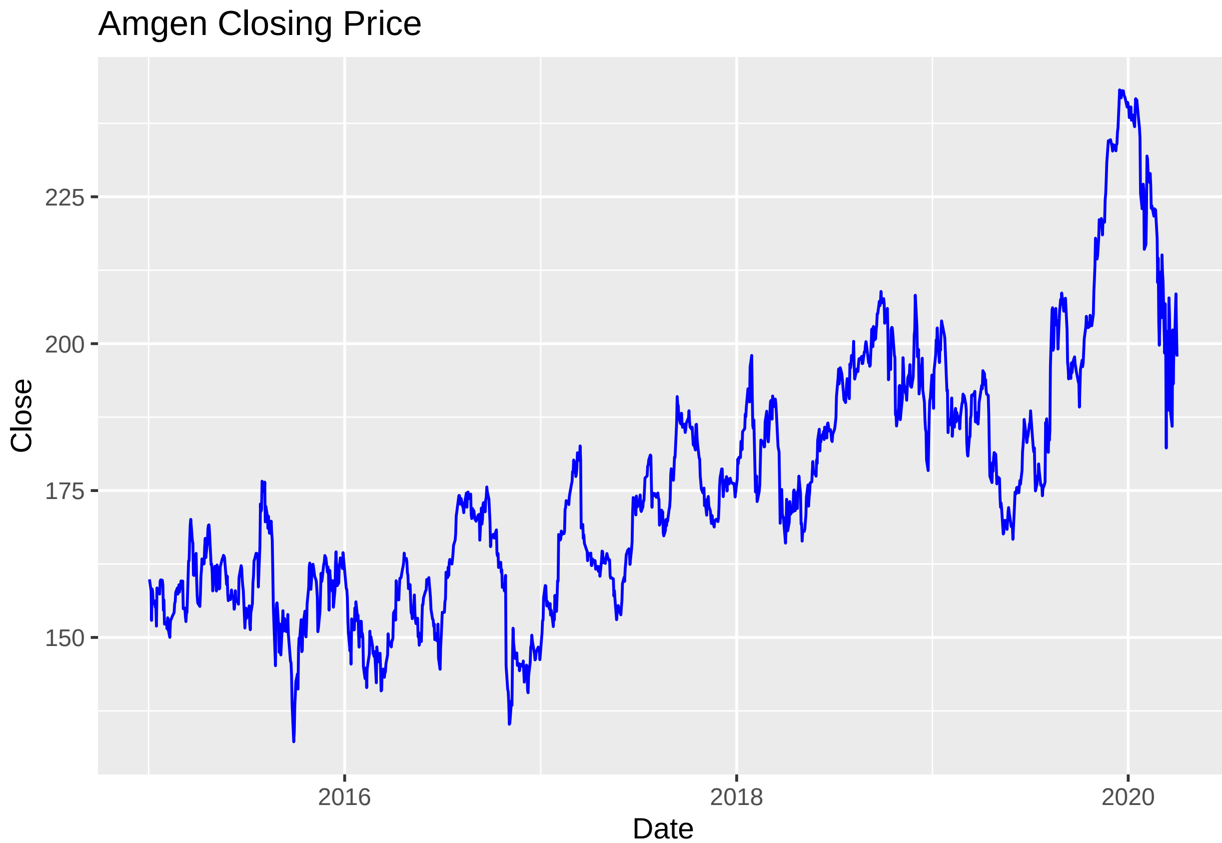



```
ggplot(data=app_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Apple Closing Price")
```

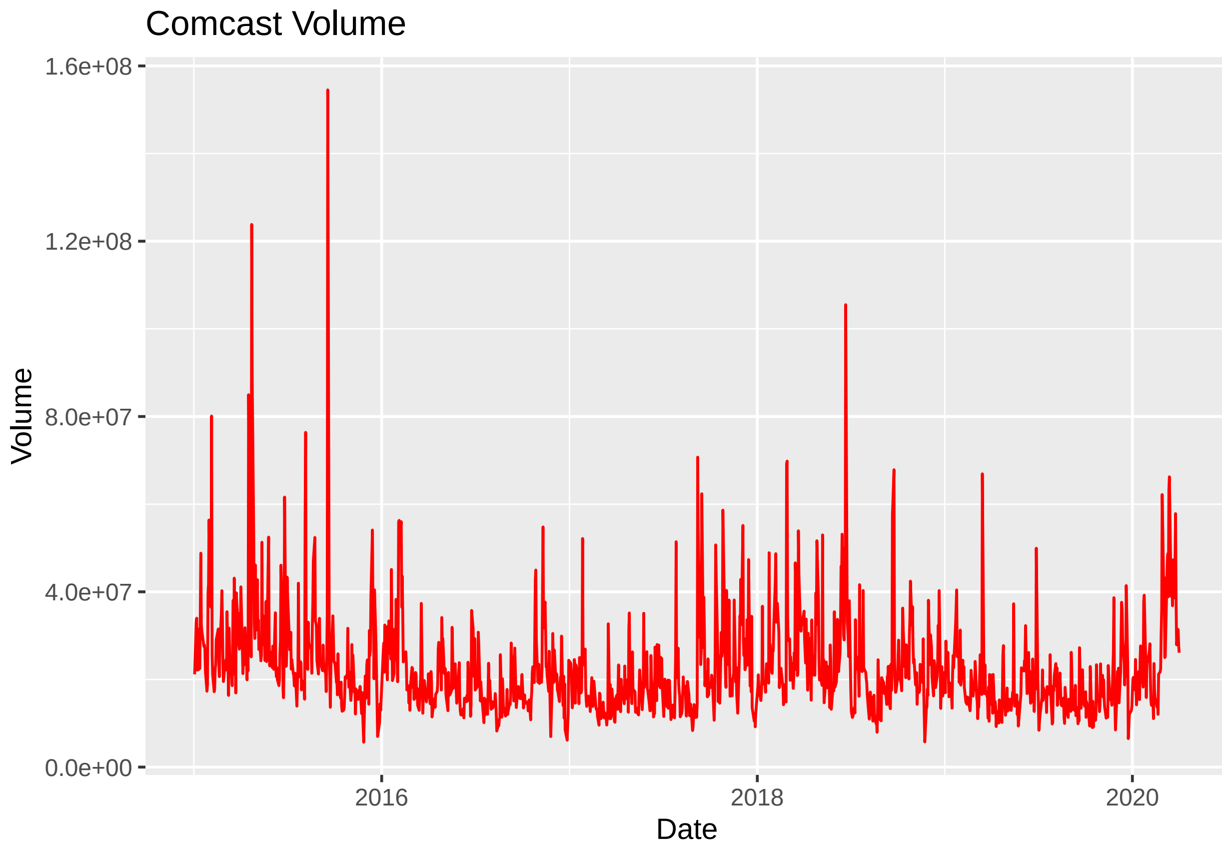

```
ggplot(data=amg_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Amgen Volume")
```




```
ggplot(data=amg_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Amgen Closing Price")
```

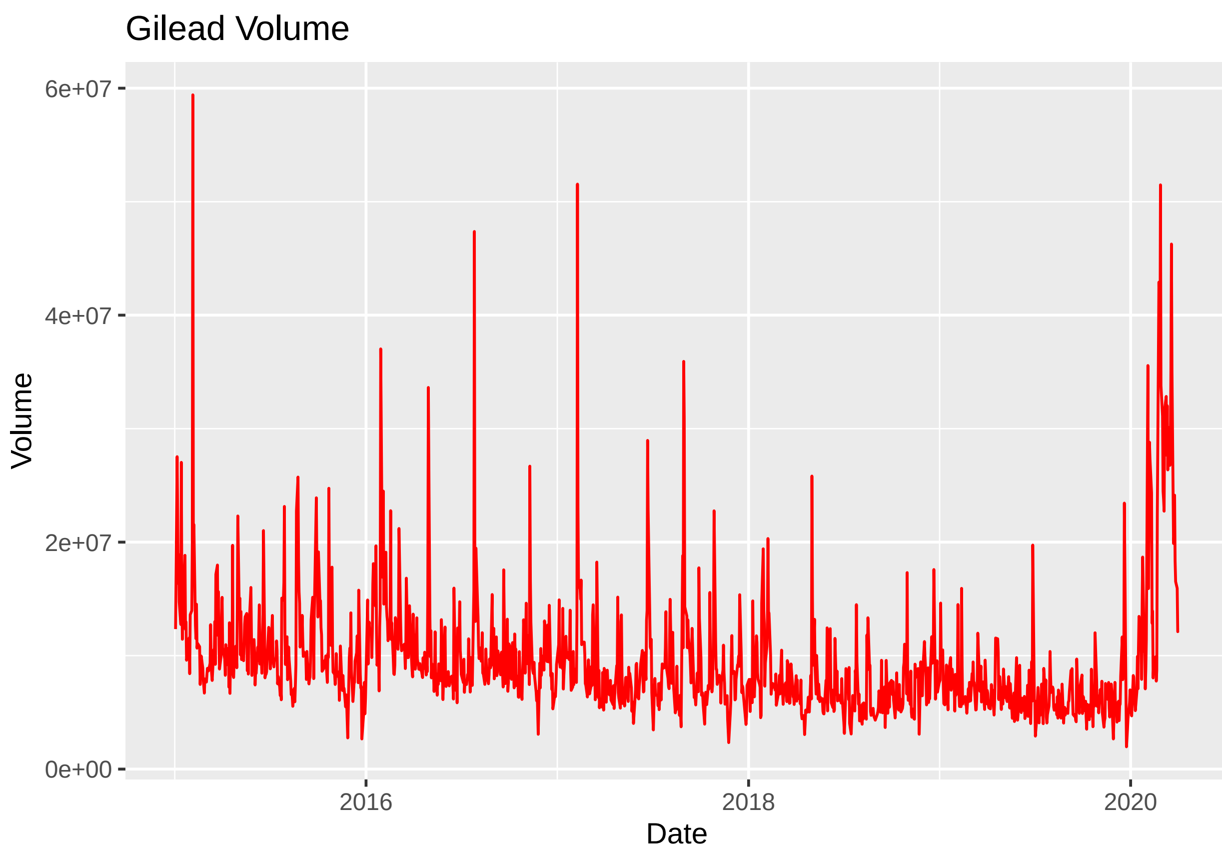



```
ggplot(data=com_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Comcast Volume")
```



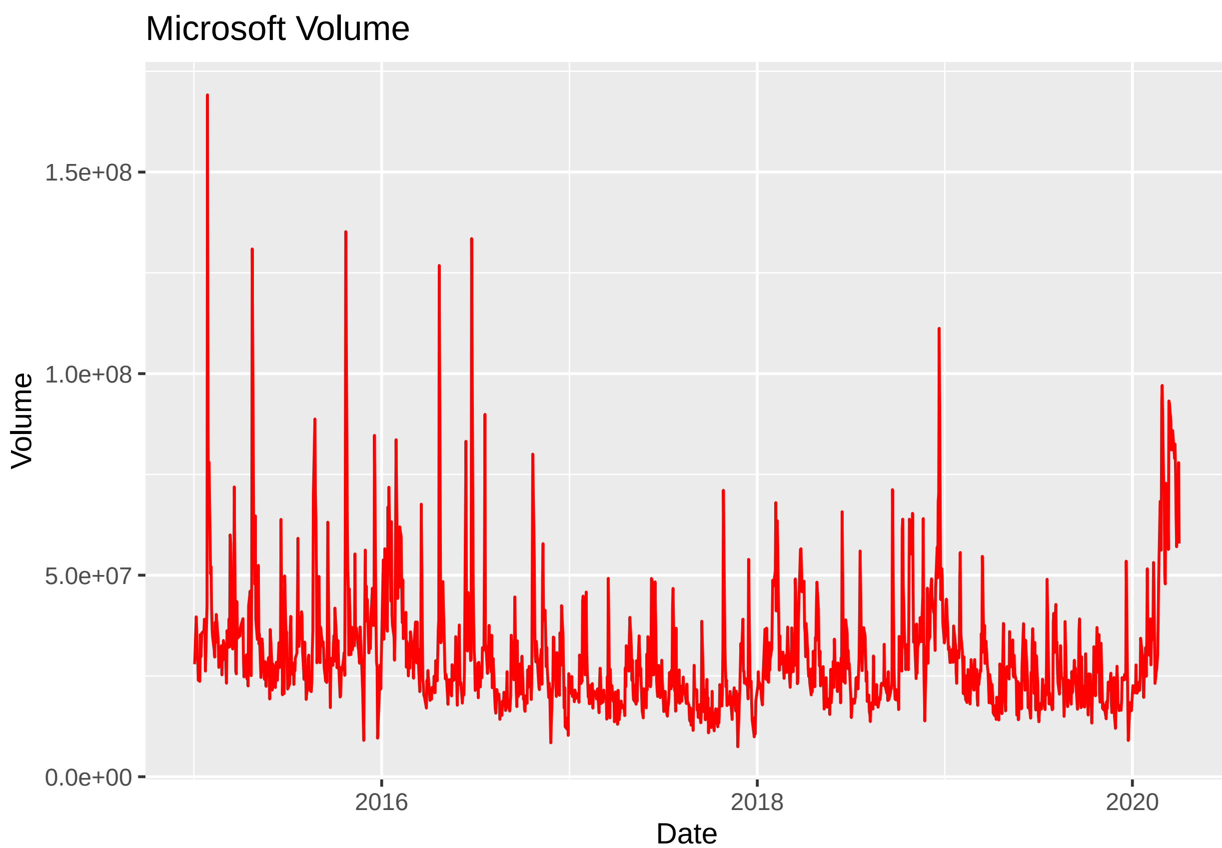

```r
ggplot(data=com_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Comcast Closing Price")
```


```
ggplot(data=gil_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Gilead Volume")
```



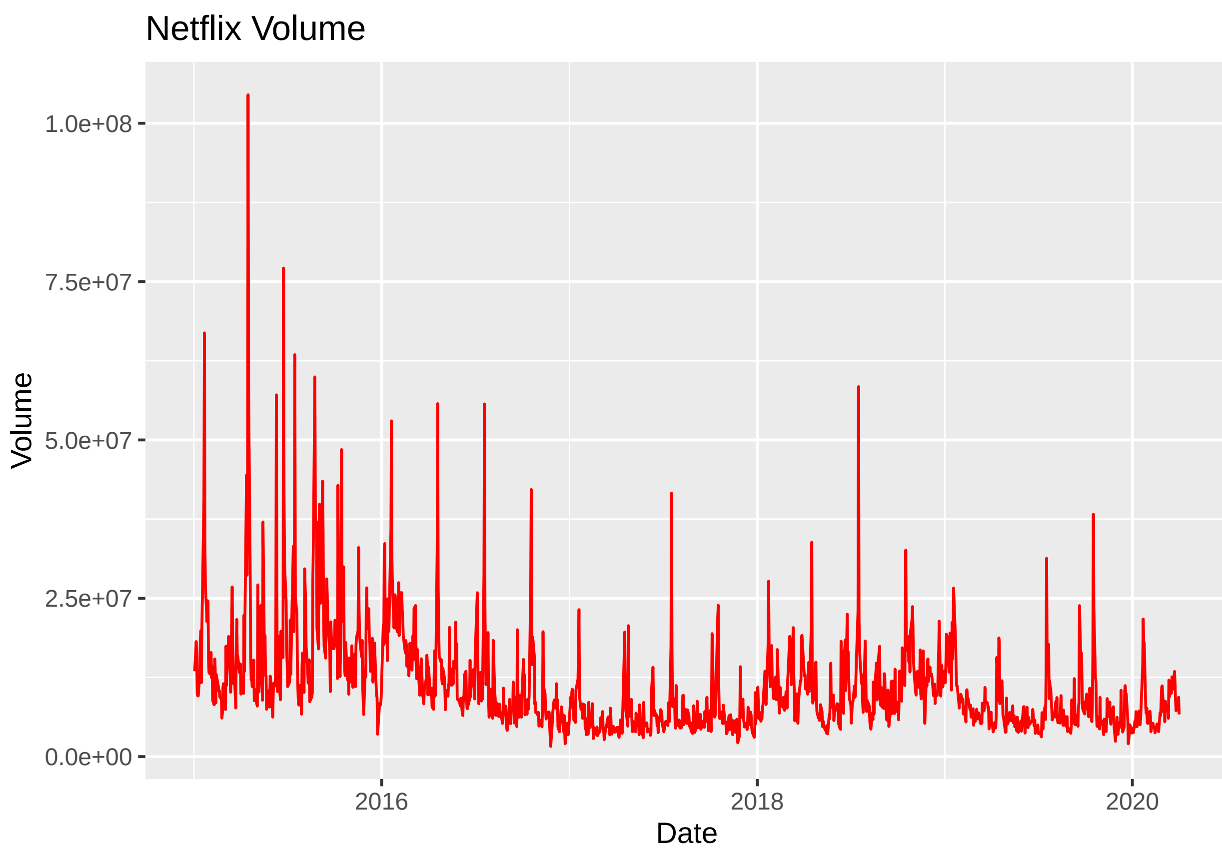

```
ggplot(data=gil_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Gilead Closing Price")
```


```r
ggplot(data=mic_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Microsoft Volume")
```



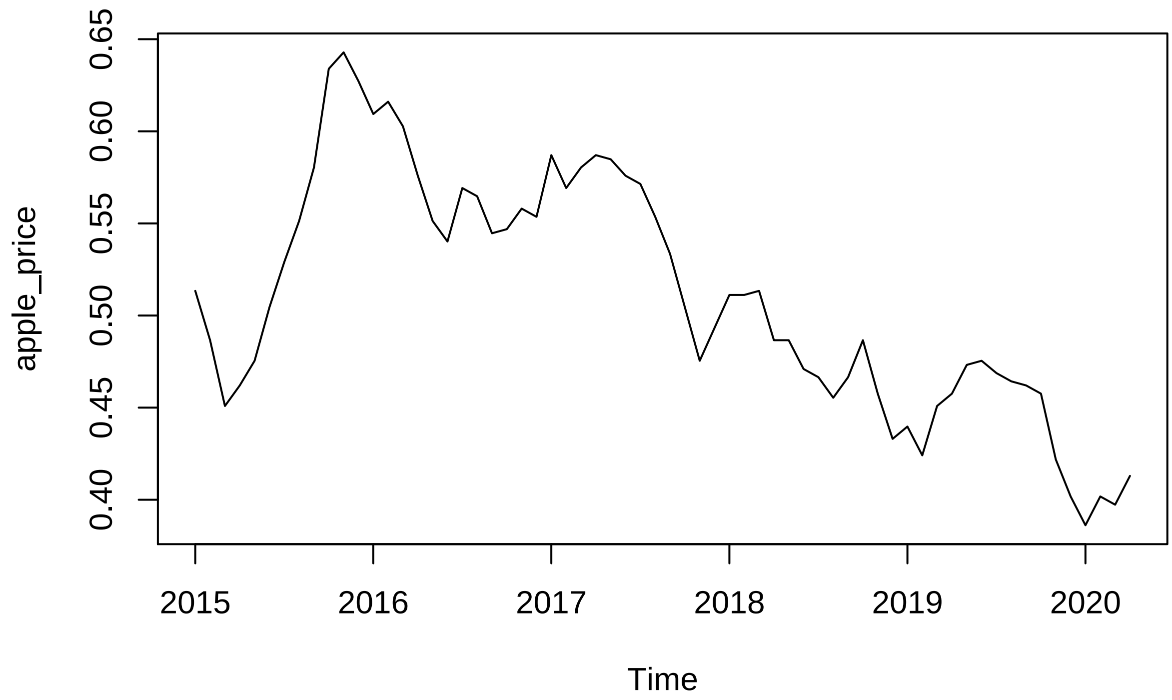

```
ggplot(data=mic_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Microsoft Closing Price")
```


```
ggplot(data=net_drop, aes(x=Date, y=Volume)) +
  geom_line(color="red") + ggtitle("Netflix Volume")
```

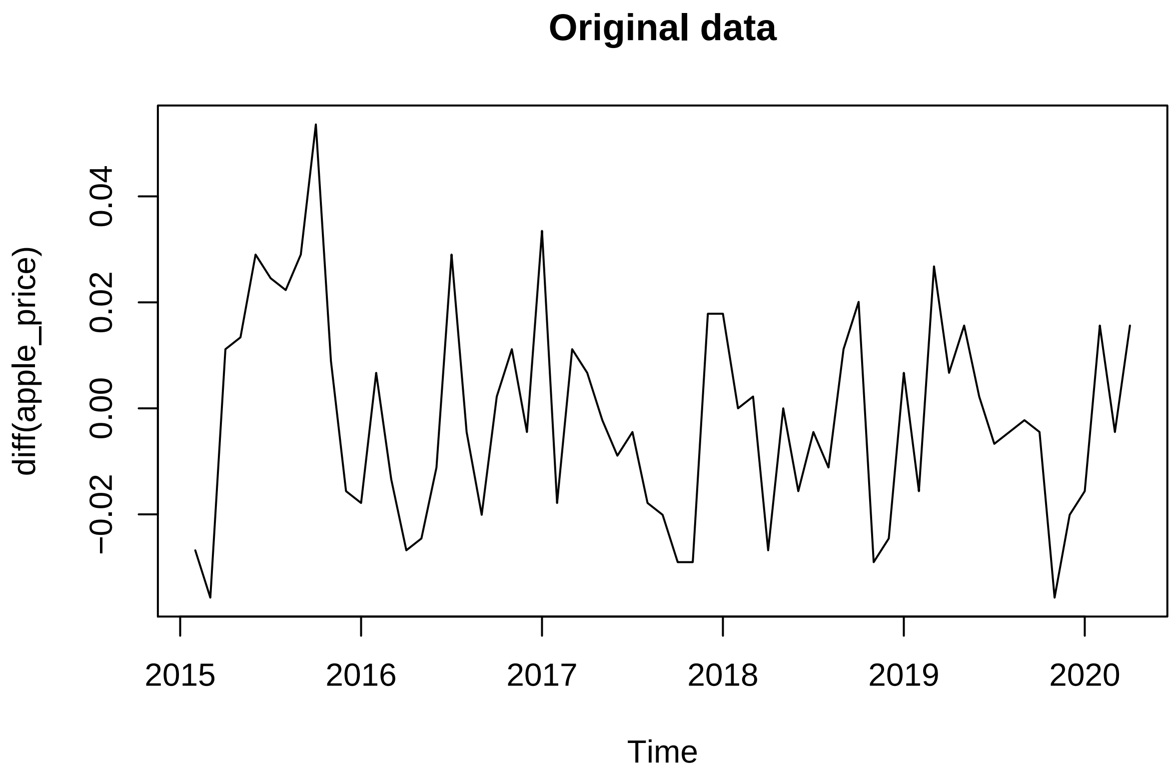



```
ggplot(data=net_drop, aes(x=Date, y=Close)) +
  geom_line(color="blue") + ggtitle("Netflix Closing Price")
```

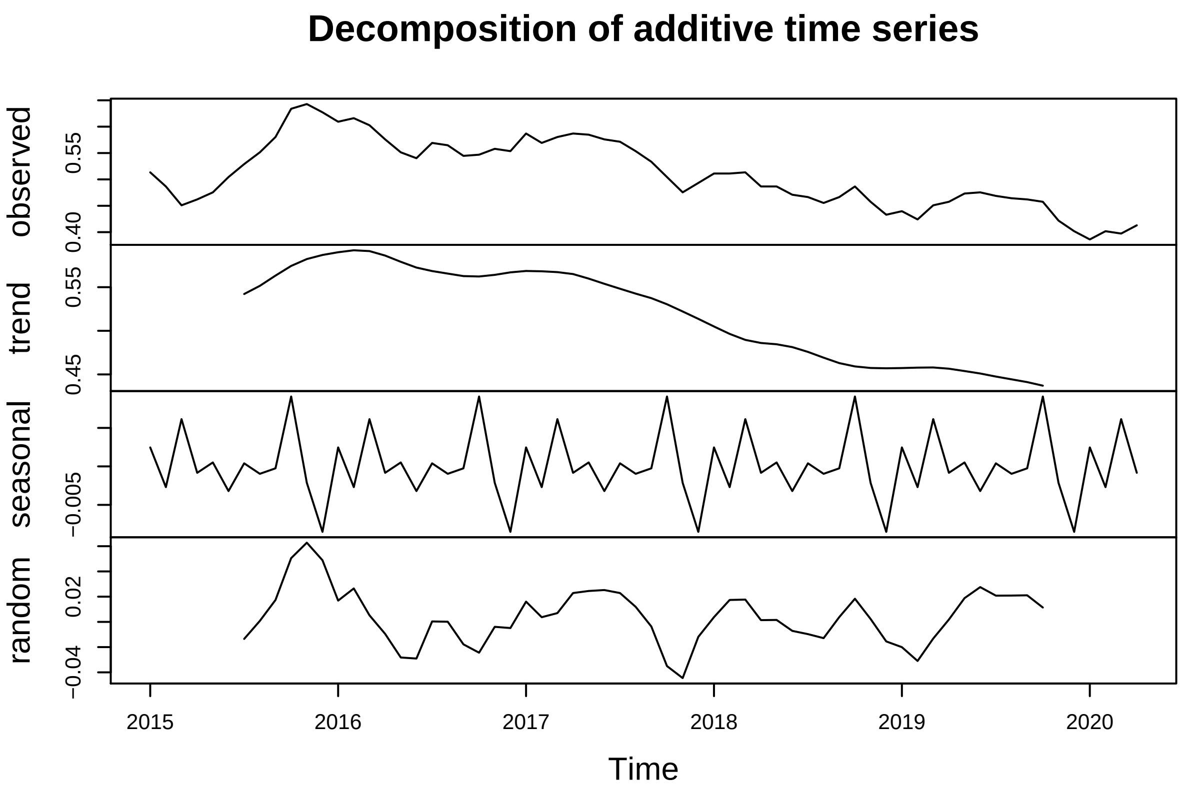

```
# APPLE STOCK
apple_price <- ts(apple$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(apple_price, type = "l")
```



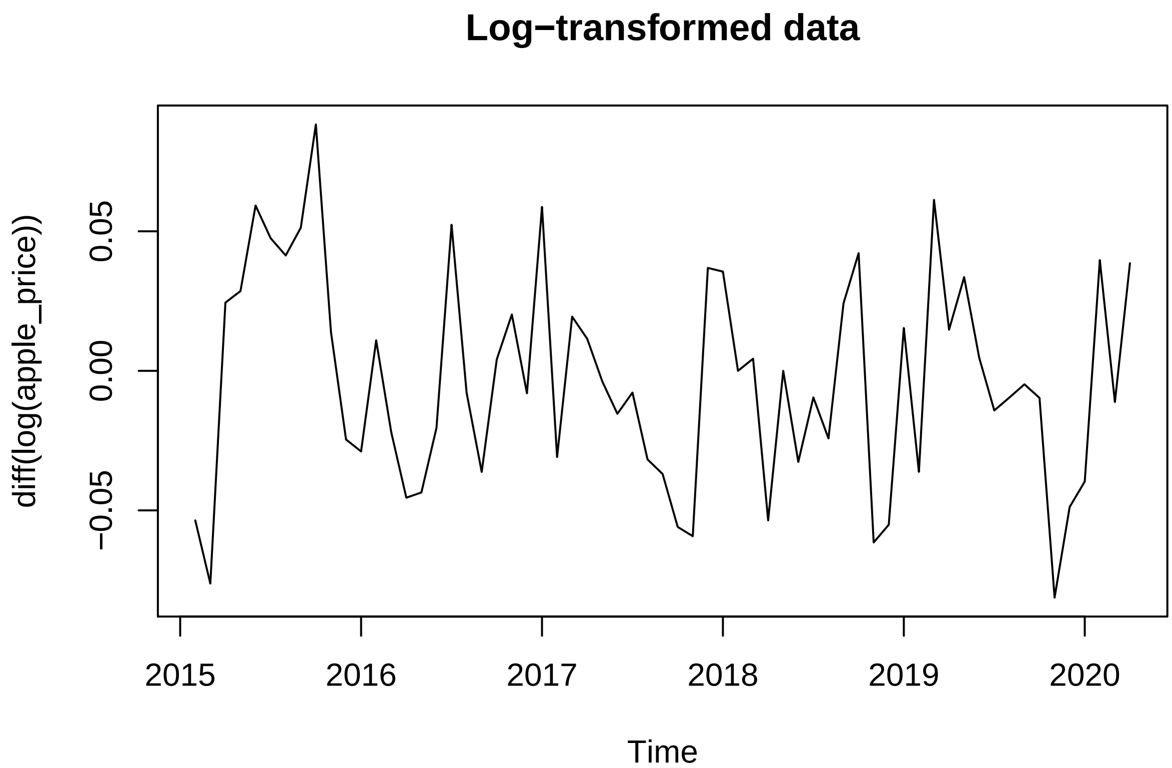

```
plot(diff(apple_price), type = "l", main = "Original data")
```




```r
# Decompose Apple Data
plot(decompose(apple_price))
```




```r
# Convert to ln format
apple_lnprice <- log(apple_price)
plot(diff(log(apple_price)), type = "l", main = "Log-transformed data")
```




```r
# Moving average on ln of stock price
apple_difflnprice <- diff(apple_lnprice,1)
```

```r
#Dickey-Fuller Test
adf.test(apple_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  apple_price
## Dickey-Fuller = -4.0769, Lag order = 3, p-value = 0.01196
## alternative hypothesis: stationary
```

```r
adf.test(apple_lnprice)
```

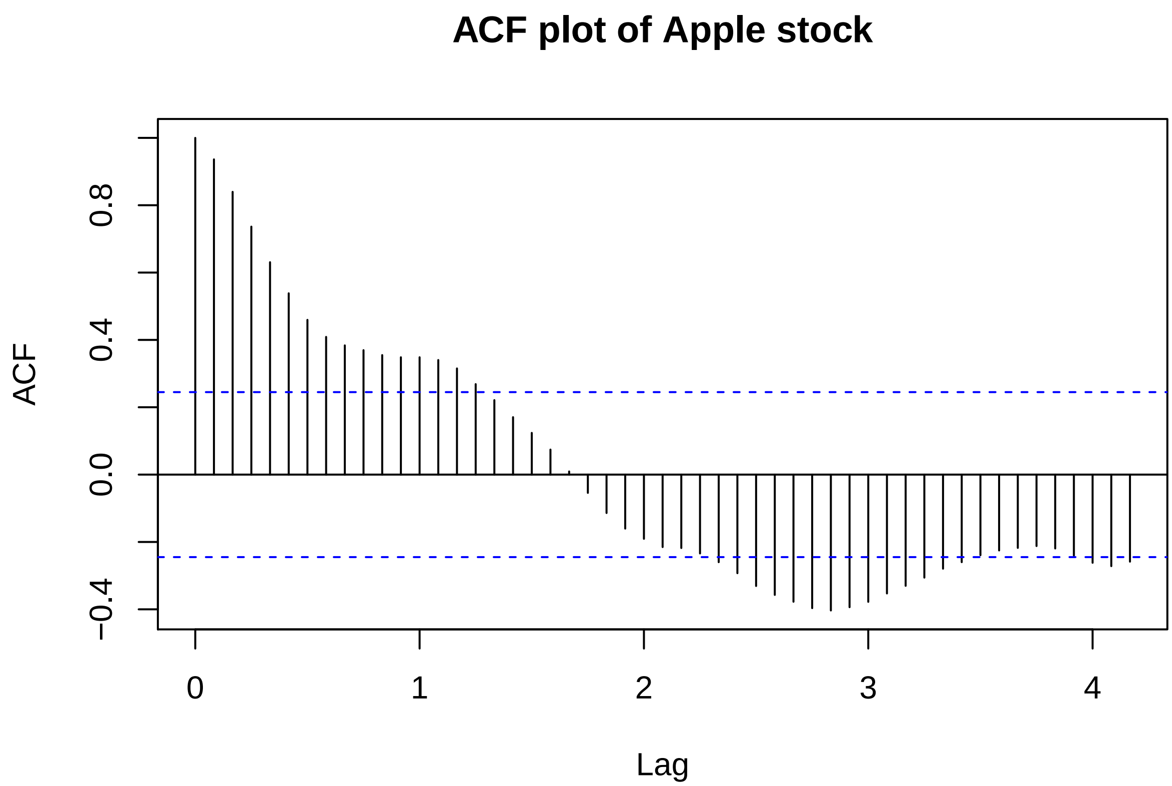
```
##
##  Augmented Dickey-Fuller Test
##
## data:  apple_lnprice
## Dickey-Fuller = -3.9695, Lag order = 3, p-value = 0.01671
## alternative hypothesis: stationary
```

```r
adf.test(apple_difflnprice)
```

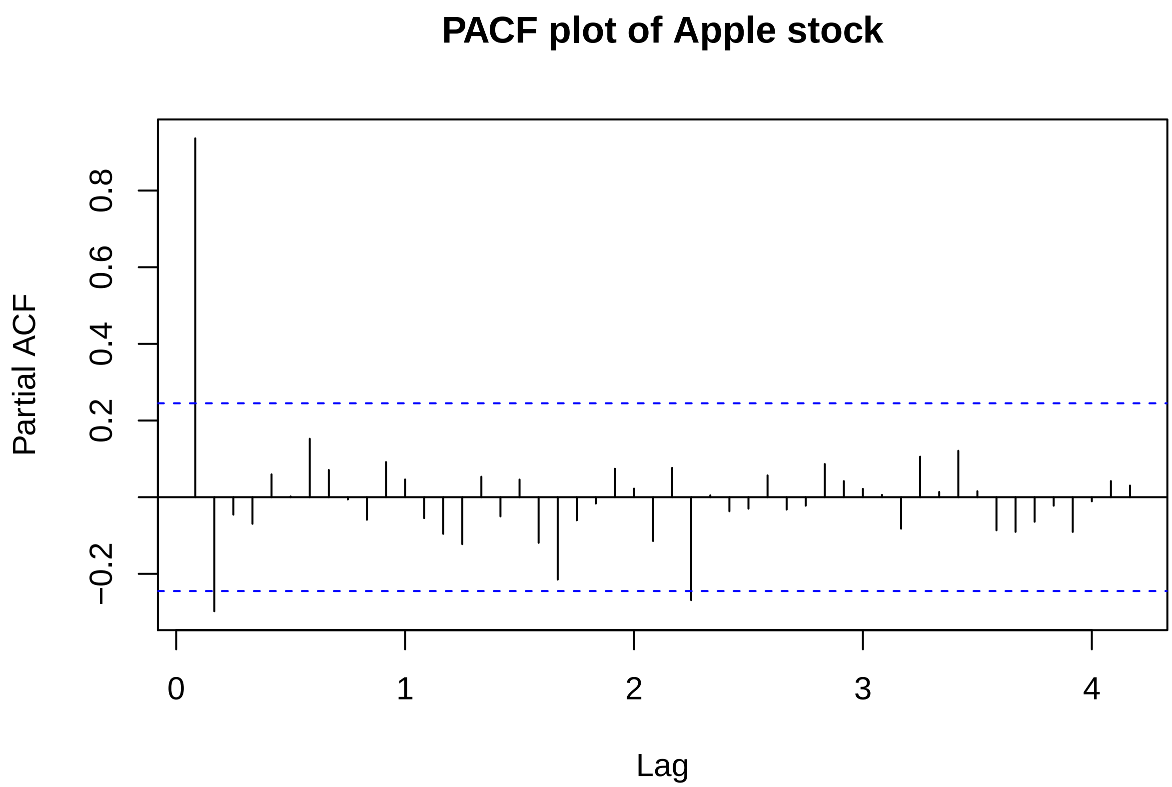
```
##
##  Augmented Dickey-Fuller Test
##
## data:  apple_difflnprice
```

```
## Dickey-Fuller = -3.8655, Lag order = 3, p-value = 0.02137
## alternative hypothesis: stationary
```

```r
#ACF, PACF
acf(apple_lnprice, lag.max=50, main="ACF plot of Apple stock")
```

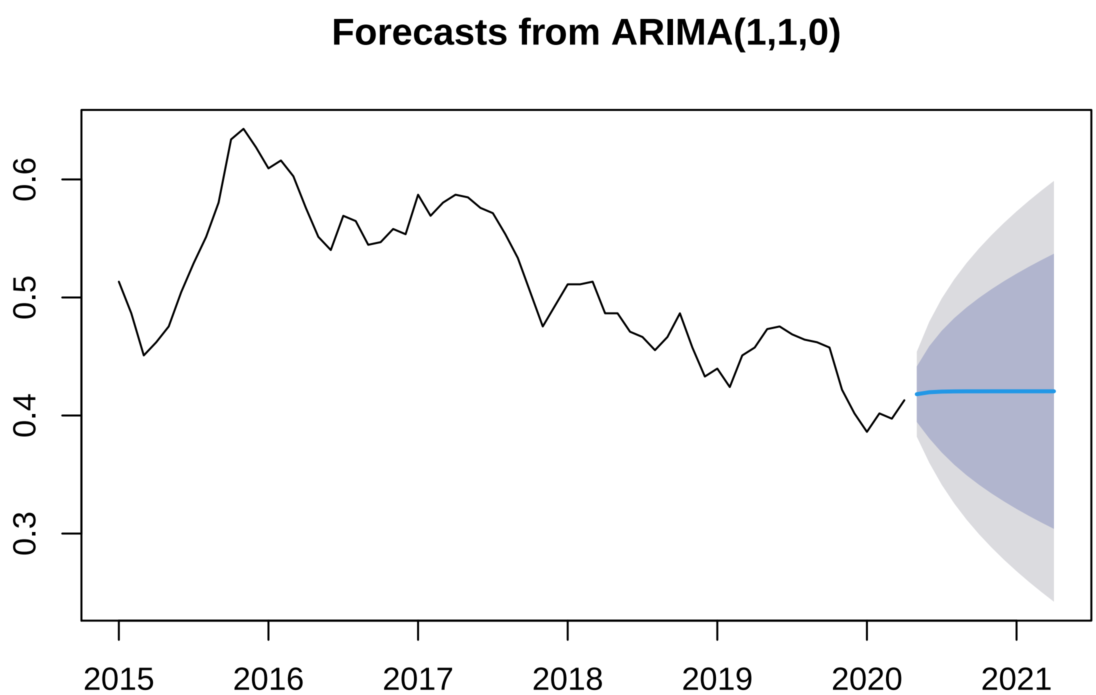



```r
pacf(apple_lnprice, lag.max=50, main="PACF plot of Apple stock")
```




```r
# Run Auto Arima to determine best Arima Model
arima_apple <- auto.arima(apple_price)
```

```r
# Forecast Using Forecast function on Arima Model
forecast_apple <- forecast(arima_apple, h=12)
plot(forecast_apple)
```

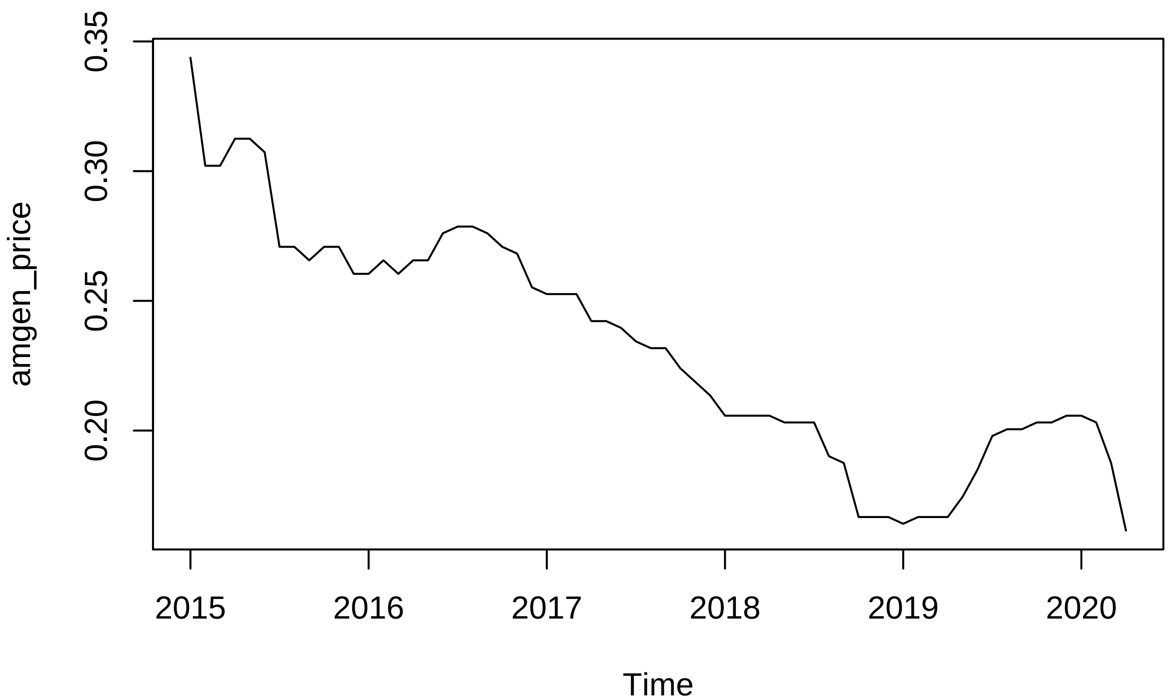



```r
# Summarize Results
summary(forecast_apple)
```

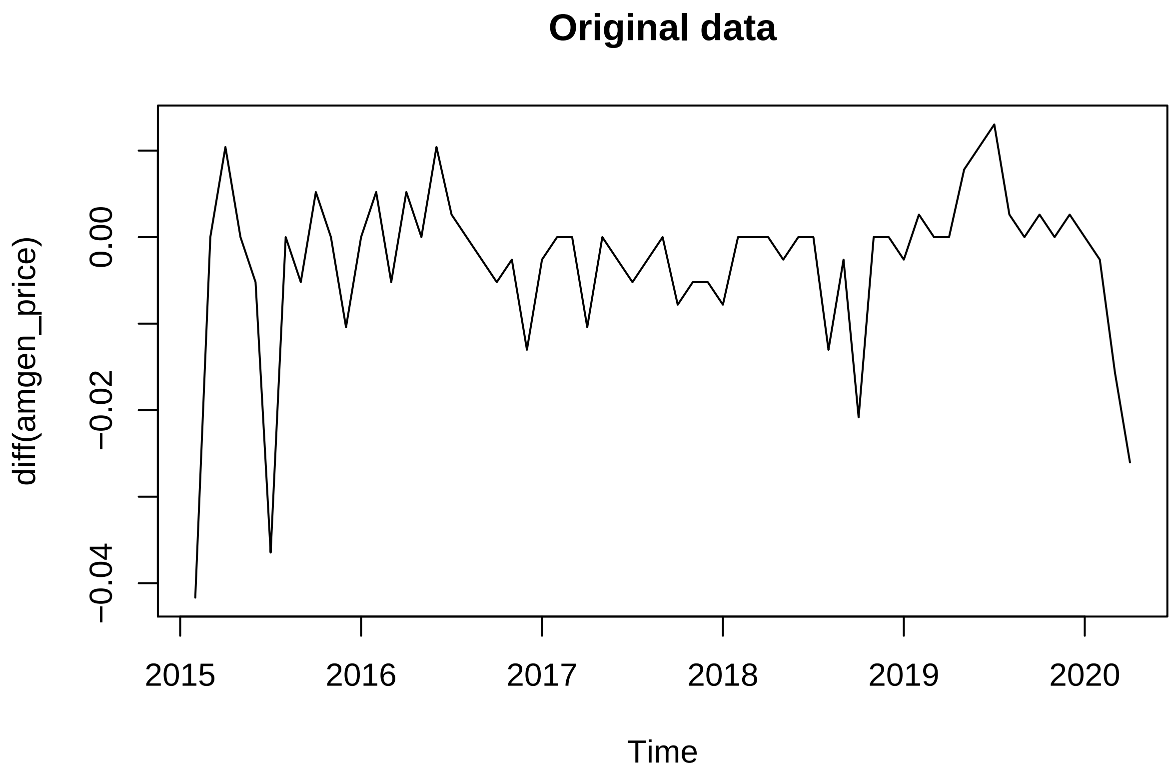
```
##
## Forecast method: ARIMA(1,1,0)
##
## Model Information:
## Series: apple_price
## ARIMA(1,1,0)
##
## Coefficients:
##          ar1
##       0.3263
## s.e.  0.1204
##
## sigma^2 estimated as 0.0003368:  log likelihood=162.93
## AIC=-321.86   AICc=-321.66   BIC=-317.57
##
## Error measures:
##                         ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.0009467805 0.01806294 0.01508402 -0.2375838 2.991729 0.2859213
##                     ACF1
## Training set -0.01430318
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
```

```
## May 2020       0.4180448 0.3945258 0.4415637 0.3820756 0.4540139
## Jun 2020       0.4197083 0.3806423 0.4587743 0.3599621 0.4794545
## Jul 2020       0.4202511 0.3686600 0.4718422 0.3413493 0.4991529
## Aug 2020       0.4204282 0.3583568 0.4824997 0.3254981 0.5153583
## Sep 2020       0.4204860 0.3493343 0.4916377 0.3116689 0.5293031
## Oct 2020       0.4205049 0.3412690 0.4997407 0.2993241 0.5416856
## Nov 2020       0.4205110 0.3339313 0.5070907 0.2880988 0.5529233
## Dec 2020       0.4205130 0.3271619 0.5138641 0.2777448 0.5632812
## Jan 2021       0.4205137 0.3208491 0.5201782 0.2680899 0.5729374
## Feb 2021       0.4205139 0.3149124 0.5261154 0.2590103 0.5820175
## Mar 2021       0.4205140 0.3092918 0.5317361 0.2504144 0.5906136
## Apr 2021       0.4205140 0.3039419 0.5370861 0.2422323 0.5987956
```

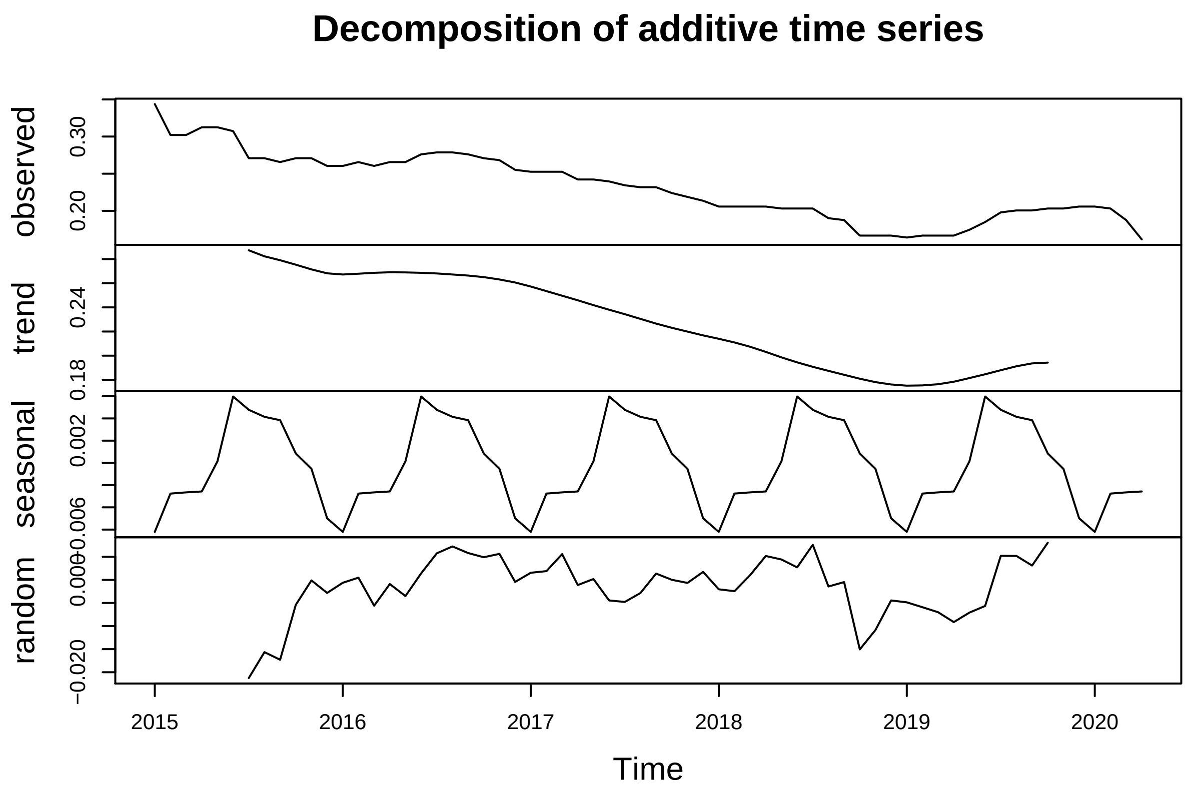
```
# AMGEN STOCK
amgen_price <- ts(amgen$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(amgen_price, type = "l")
```



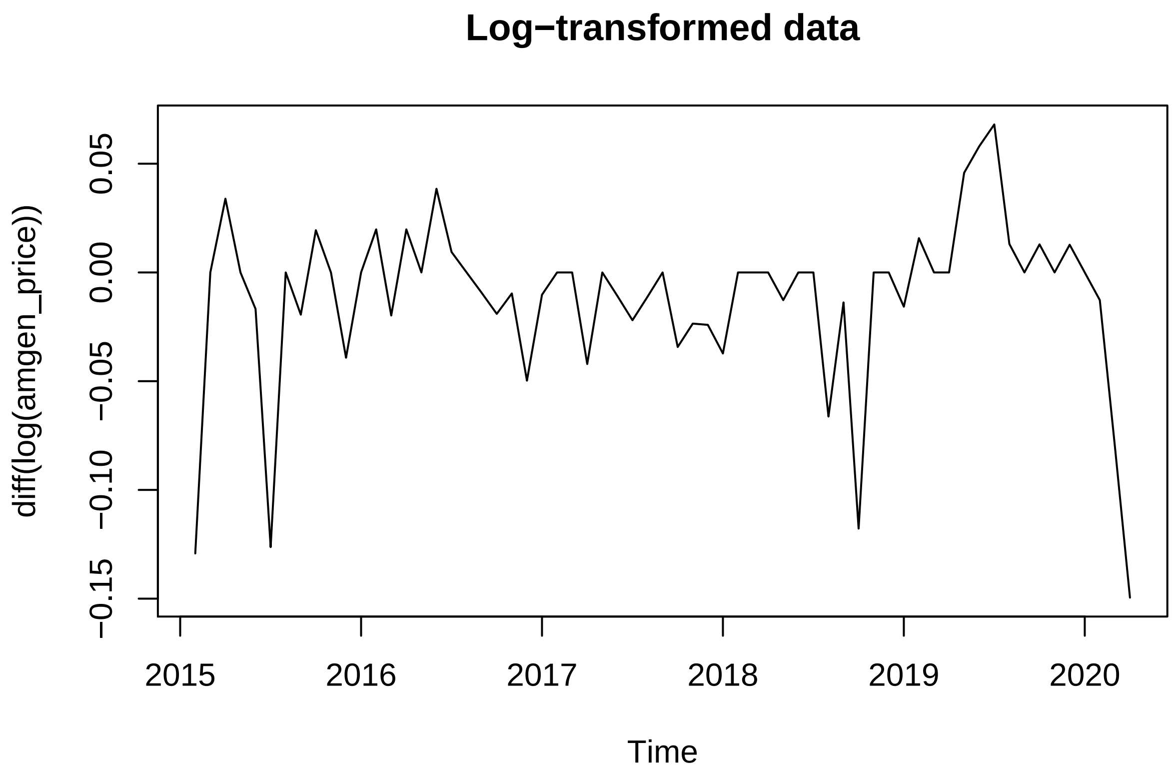

```
plot(diff(amgen_price), type = "l", main = "Original data")
```




```r
# Decompose Amgen Data
plot(decompose(amgen_price))
```




```r
# Convert to ln format
amgen_lnprice <- log(amgen_price)
plot(diff(log(amgen_price)), type = "l", main = "Log-transformed data")
```




```
# Moving average on ln of stock price
amgen_difflnprice <- diff(amgen_lnprice,1)
```

```
#Dickey-Fuller Test
adf.test(amgen_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  amgen_price
## Dickey-Fuller = -2.8335, Lag order = 3, p-value = 0.2372
## alternative hypothesis: stationary
```

```
adf.test(amgen_lnprice)
```

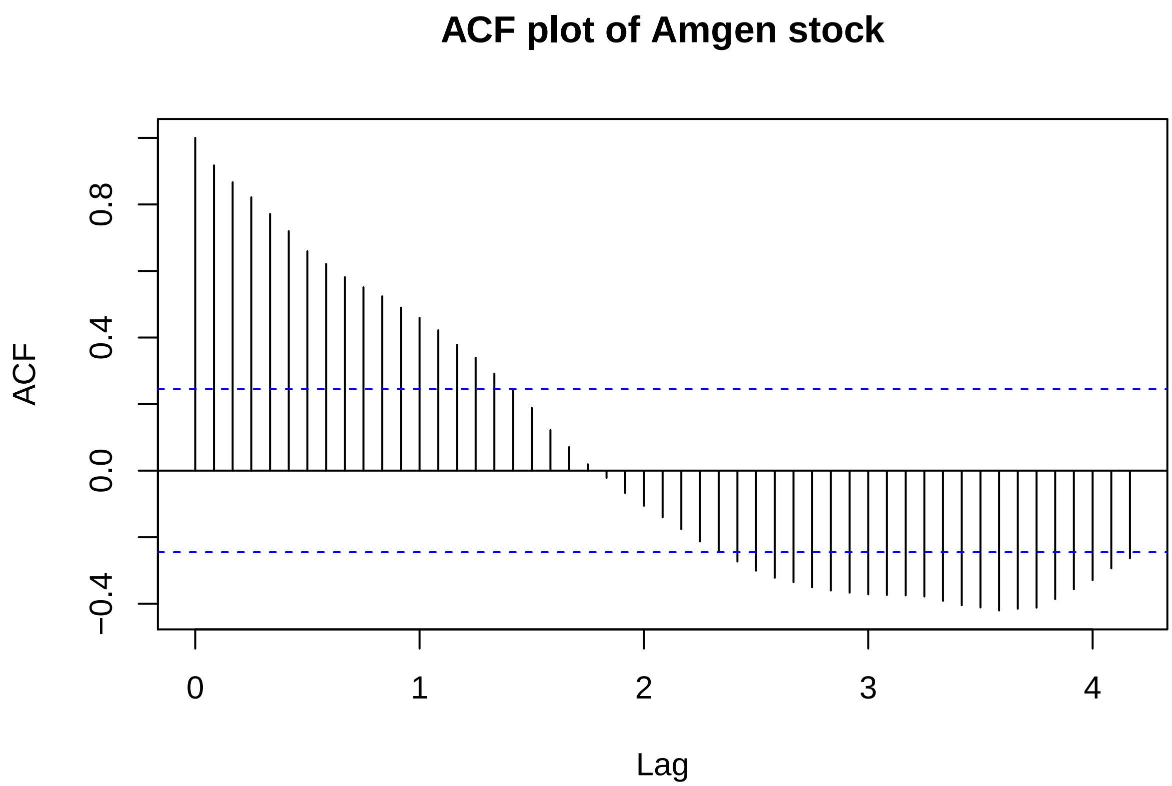
```
##
##  Augmented Dickey-Fuller Test
##
## data:  amgen_lnprice
## Dickey-Fuller = -2.9876, Lag order = 3, p-value = 0.1748
## alternative hypothesis: stationary
```

```
adf.test(amgen_difflnprice)
```

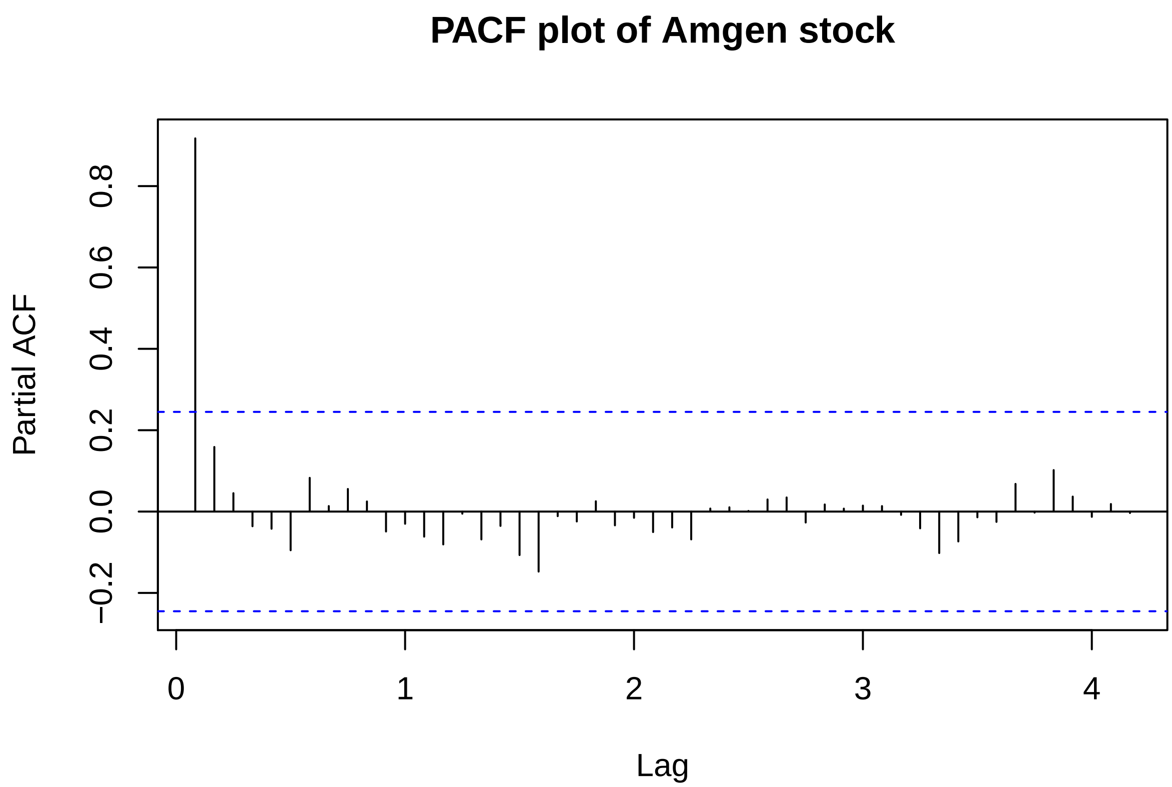
```
##
##  Augmented Dickey-Fuller Test
##
## data:  amgen_difflnprice
```

```
## Dickey-Fuller = -2.4147, Lag order = 3, p-value = 0.407
## alternative hypothesis: stationary
```

```r
#ACF, PACF
acf(amgen_lnprice, lag.max=50, main="ACF plot of Amgen stock")
```

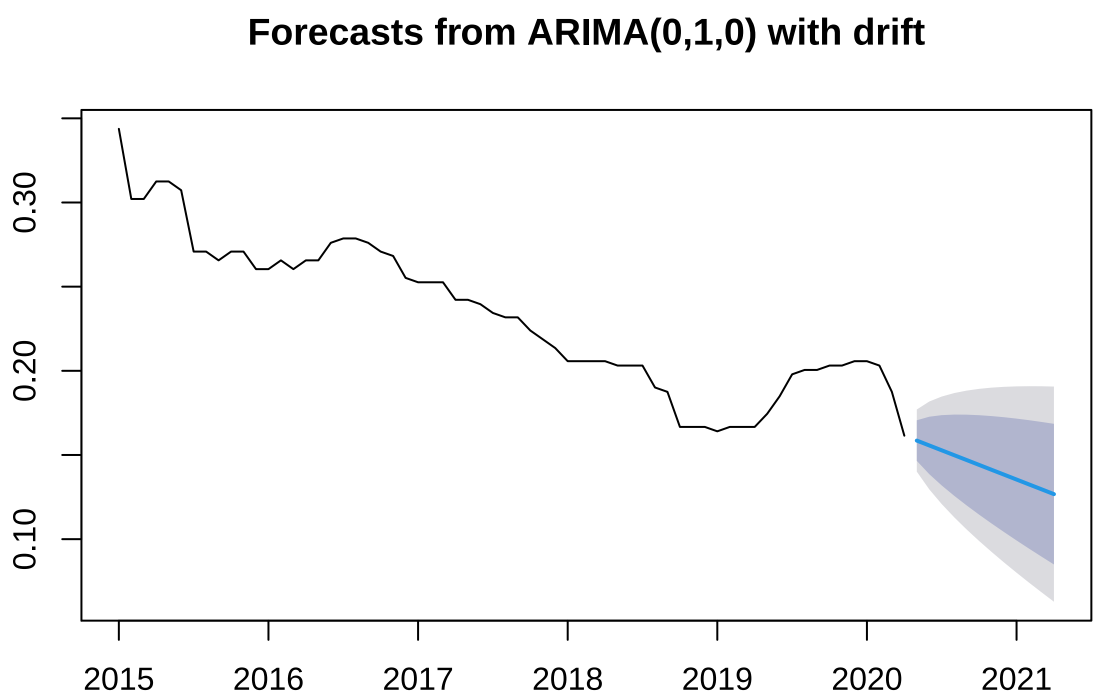



```r
pacf(amgen_lnprice, lag.max=50, main="PACF plot of Amgen stock")
```




```r
# Run Auto Arima to determine best Arima Model
arima_amgen <- auto.arima(amgen_price)
```

```r
# Forecast Using Forecast function on Arima Model
forecast_amgen <- forecast(arima_amgen, h=12)
plot(forecast_amgen)
```

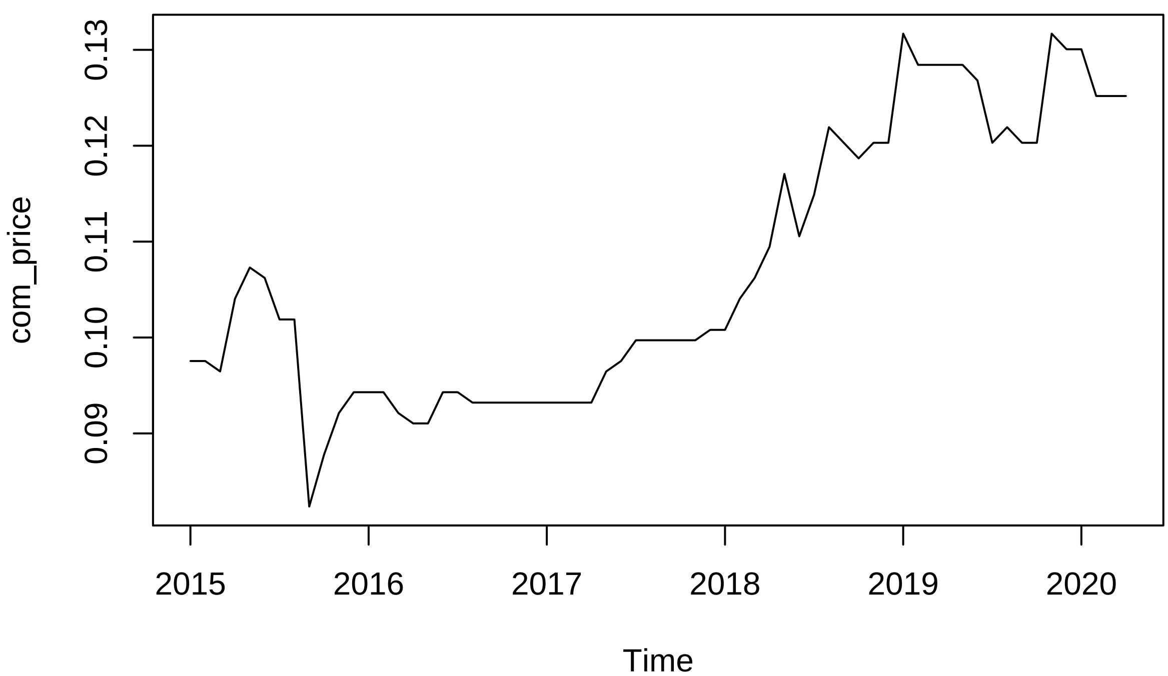



```r
# Summarize Results
summary(forecast_amgen)
```

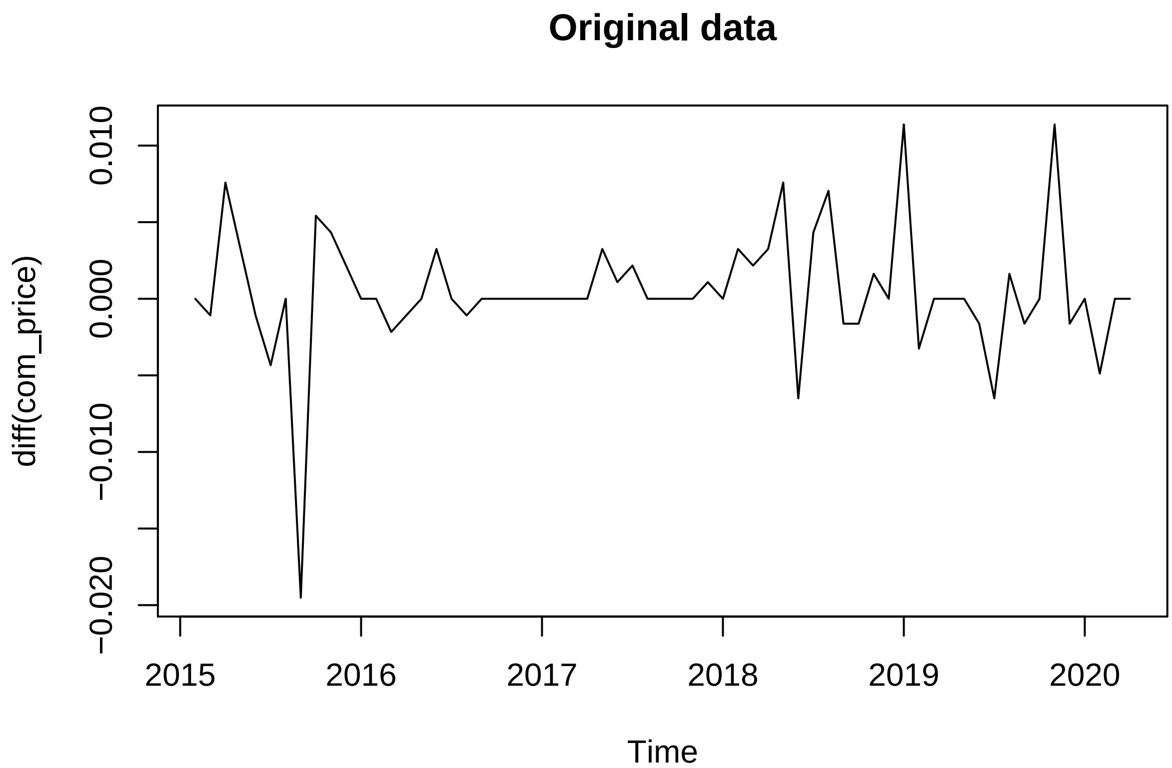
```
##
## Forecast method: ARIMA(0,1,0) with drift
##
## Model Information:
## Series: amgen_price
## ARIMA(0,1,0) with drift
##
## Coefficients:
##         drift
##       -0.0029
## s.e.   0.0012
##
## sigma^2 estimated as 8.863e-05:  log likelihood=205.04
## AIC=-406.08   AICc=-405.88   BIC=-401.8
##
## Error measures:
##                        ME        RMSE        MAE        MPE     MAPE      MASE
## Training set 5.416302e-06 0.009265856 0.00586479 0.01585239 2.625091 0.1812819
##                   ACF1
## Training set 0.1442491
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
```

```
## May 2020       0.1585648 0.14650013 0.1706295 0.14011348 0.1770161
## Jun 2020       0.1556713 0.13860926 0.1727333 0.12957717 0.1817654
## Jul 2020       0.1527778 0.13188114 0.1736744 0.12081913 0.1847364
## Aug 2020       0.1498843 0.12575490 0.1740136 0.11298159 0.1867869
## Sep 2020       0.1469907 0.12001329 0.1739682 0.10573230 0.1882492
## Oct 2020       0.1440972 0.11454491 0.1736495 0.09890086 0.1892936
## Nov 2020       0.1412037 0.10928356 0.1731238 0.09238606 0.1900213
## Dec 2020       0.1383102 0.10418612 0.1724342 0.08612193 0.1904984
## Jan 2021       0.1354167 0.09922263 0.1716107 0.08006266 0.1907707
## Feb 2021       0.1325231 0.09437128 0.1706750 0.07417490 0.1908714
## Mar 2021       0.1296296 0.08961561 0.1696436 0.06843347 0.1908258
## Apr 2021       0.1267361 0.08494283 0.1685294 0.06281881 0.1906534
```

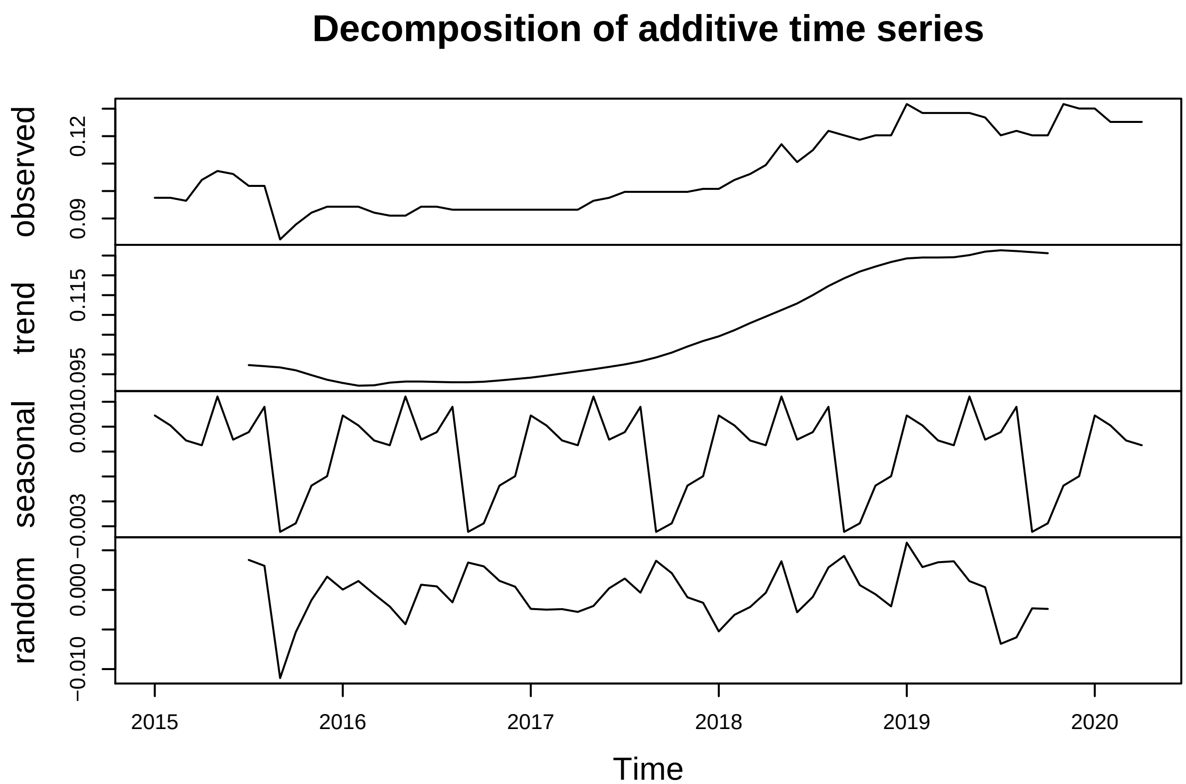
```r
# COMCAST STOCK
com_price <- ts(comcast$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(com_price, type = "l")
```



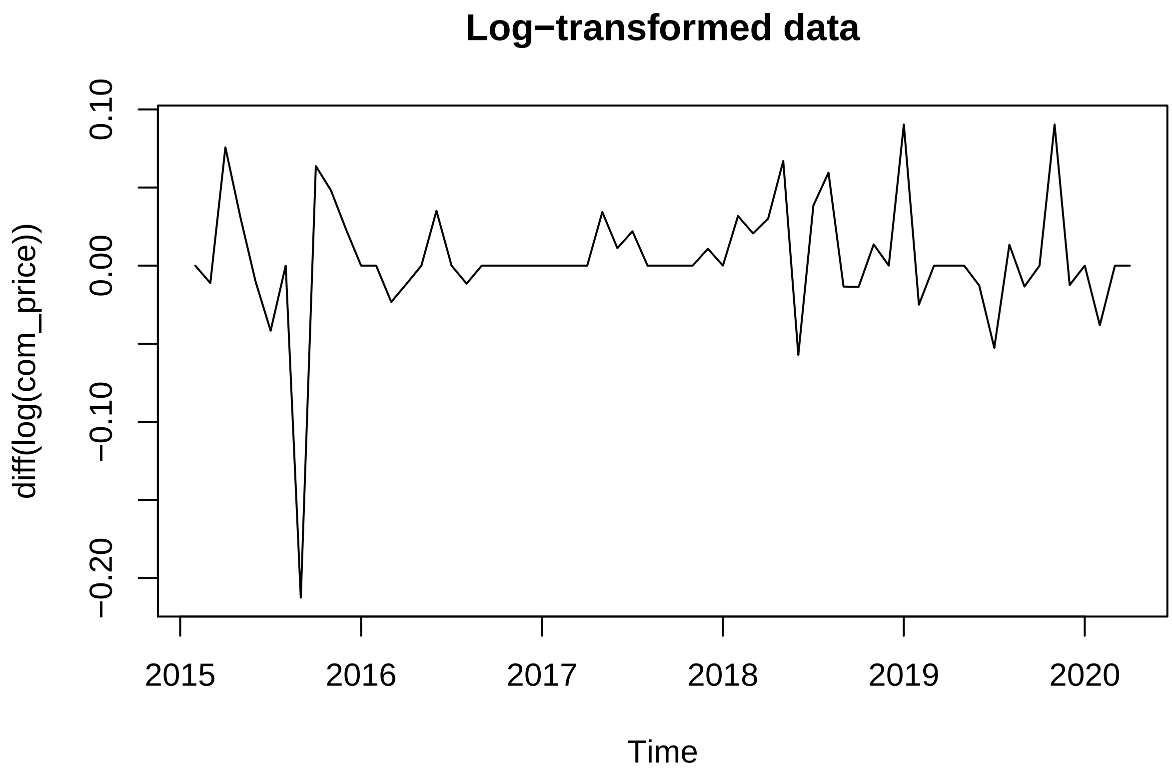

```r
plot(diff(com_price), type = "l", main = "Original data")
```




```
# Decompose Comcast Data
plot(decompose(com_price))
```




```r
# Convert to ln format
com_lnprice <- log(com_price)
plot(diff(log(com_price)), type = "l", main = "Log-transformed data")
```




```r
# Moving average on ln of stock price
com_difflnprice <- diff(com_lnprice,1)
```

```r
#Dickey-Fuller Test
adf.test(com_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  com_price
## Dickey-Fuller = -2.5727, Lag order = 3, p-value = 0.3429
## alternative hypothesis: stationary
```

```r
adf.test(com_lnprice)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  com_lnprice
## Dickey-Fuller = -2.658, Lag order = 3, p-value = 0.3083
## alternative hypothesis: stationary
```

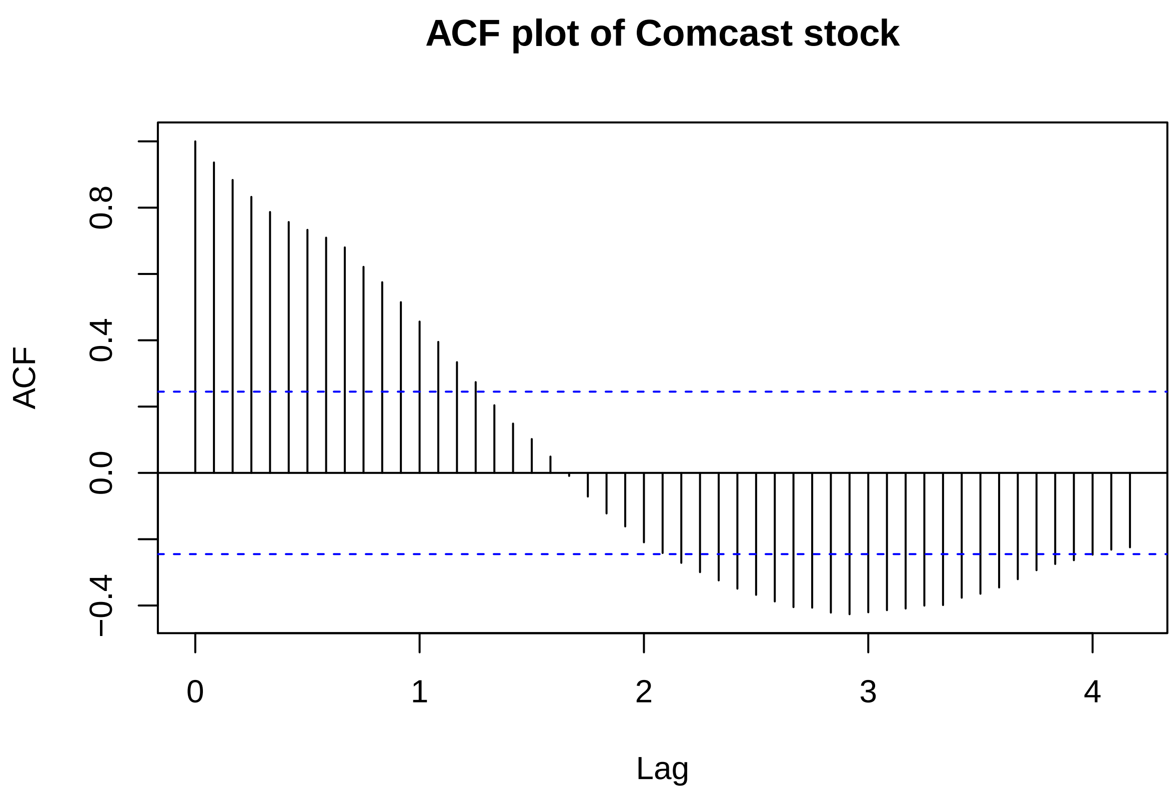
```r
adf.test(com_difflnprice)
```

```
## Warning in adf.test(com_difflnprice): p-value smaller than printed p-value
```

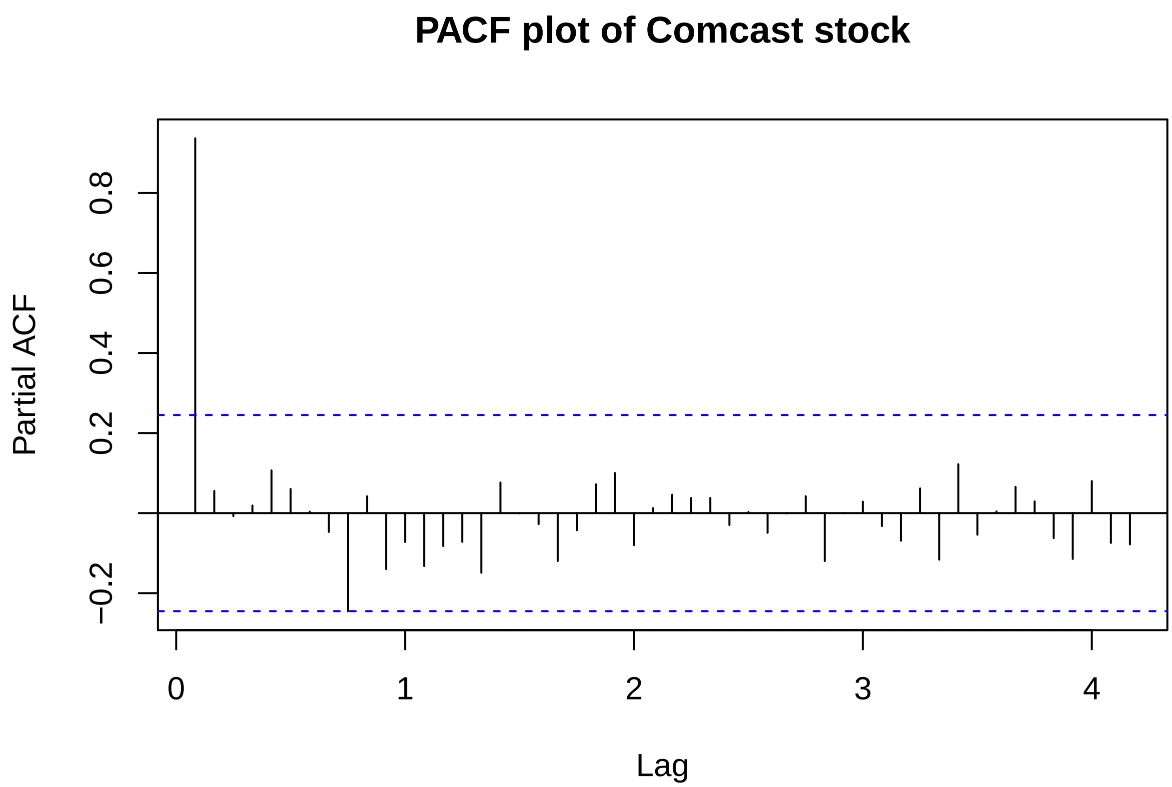
```
##
##  Augmented Dickey-Fuller Test
##
```

```
## data:  com_difflnprice
## Dickey-Fuller = -5.2695, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```r
#ACF, PACF
acf(com_lnprice, lag.max=50, main="ACF plot of Comcast stock")
```

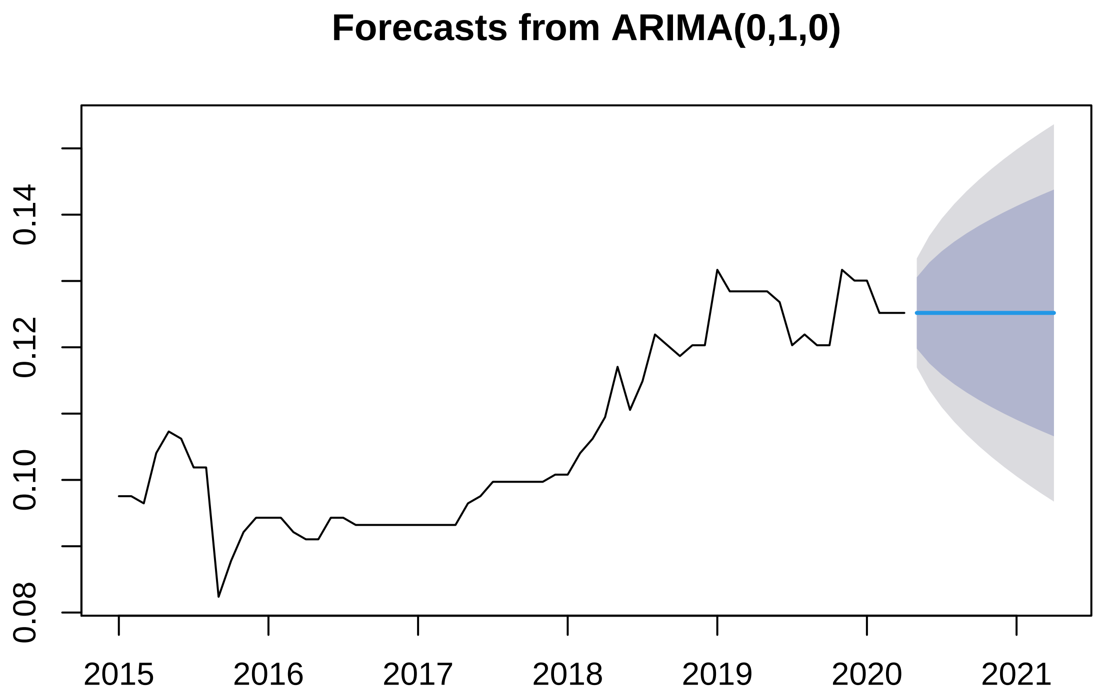



```r
pacf(com_lnprice, lag.max=50, main="PACF plot of Comcast stock")
```




```
# Run Auto Arima to determine best Arima Model
arima_comcast <- auto.arima(com_price)
```

```
# Forecast Using Forecast function on Arima Model
forecast_comcast <- forecast(arima_comcast, h=12)
plot(forecast_comcast)
```

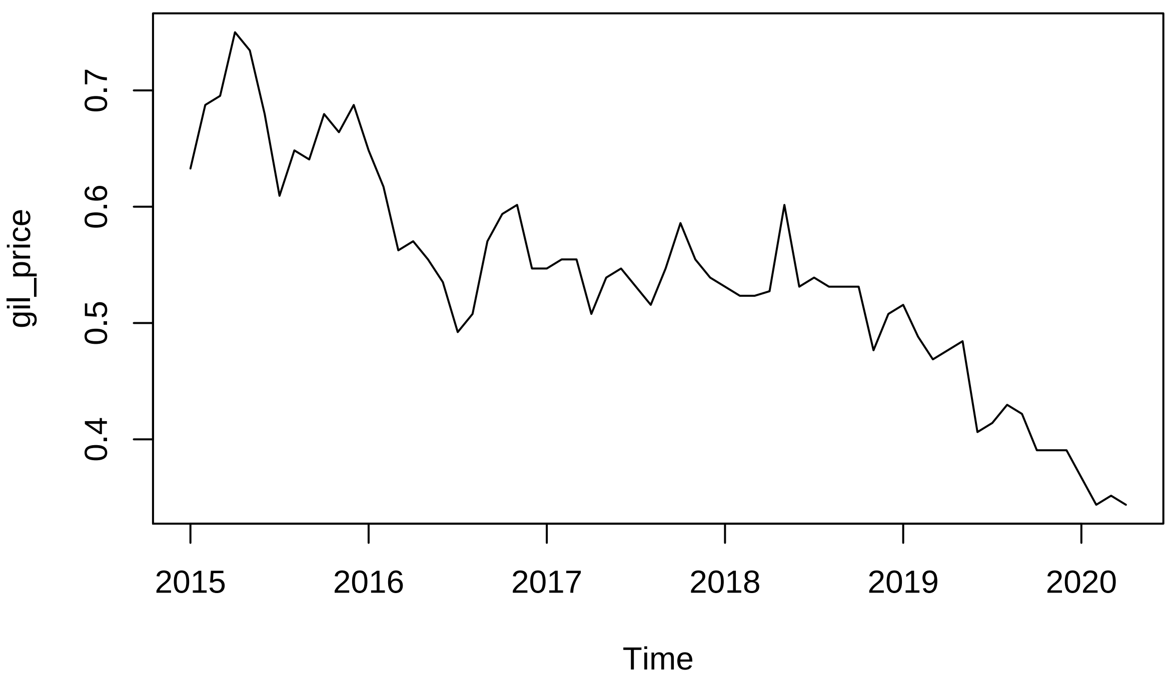



```
# Summarize Results
summary(forecast_comcast)
```

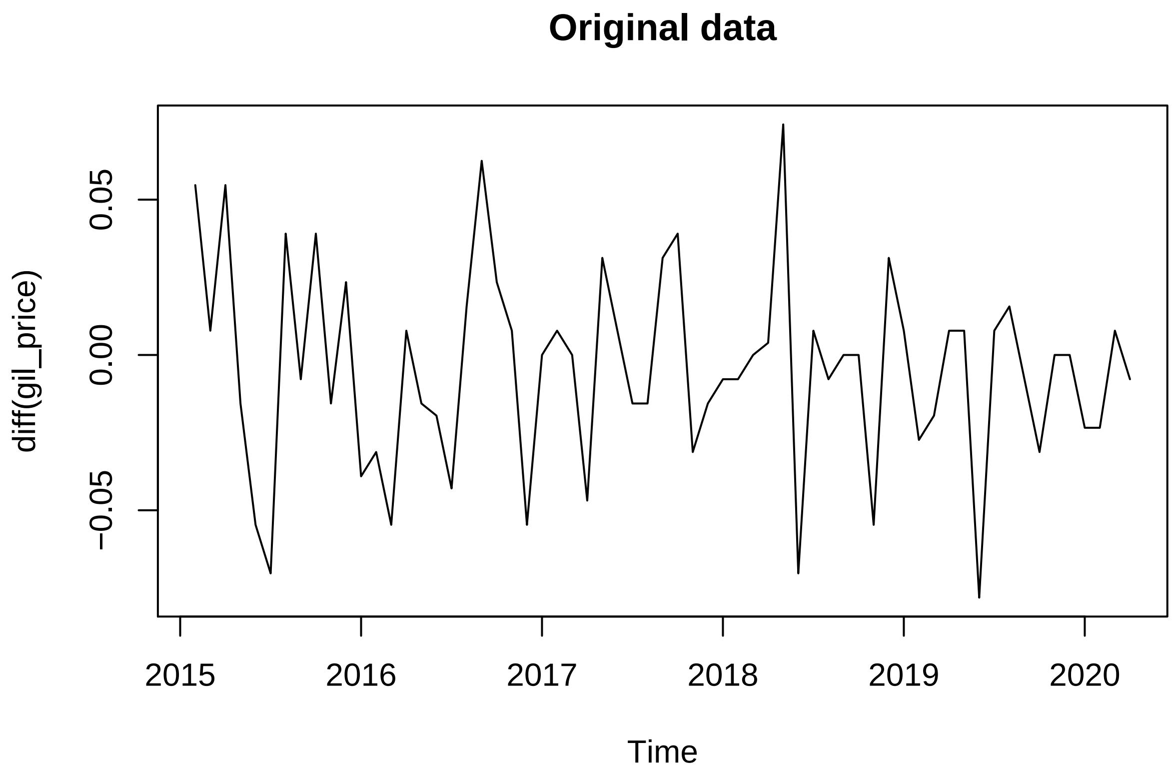
```
##
## Forecast method: ARIMA(0,1,0)
##
## Model Information:
## Series: com_price
## ARIMA(0,1,0)
##
## sigma^2 estimated as 1.756e-05:  log likelihood=255.53
## AIC=-509.06   AICc=-508.99   BIC=-506.91
##
## Error measures:
##                        ME        RMSE         MAE       MPE     MAPE      MASE
## Training set 0.0004333688 0.004157642 0.002296228 0.3094784 2.156662 0.2379414
##                    ACF1
## Training set -0.1251772
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80      Lo 95     Hi 95
## May 2020      0.1251842 0.1198138 0.1305545 0.11697092 0.1333974
## Jun 2020      0.1251842 0.1175893 0.1327790 0.11356888 0.1367994
## Jul 2020      0.1251842 0.1158824 0.1344859 0.11095840 0.1394099
## Aug 2020      0.1251842 0.1144435 0.1359249 0.10875767 0.1416107
## Sep 2020      0.1251842 0.1131757 0.1371926 0.10681879 0.1435495
```

```
## Oct 2020       0.1251842 0.1120295 0.1383388 0.10506590 0.1453024
## Nov 2020       0.1251842 0.1109755 0.1393928 0.10345396 0.1469144
## Dec 2020       0.1251842 0.1099945 0.1403738 0.10195359 0.1484147
## Jan 2021       0.1251842 0.1090731 0.1412952 0.10054442 0.1498239
## Feb 2021       0.1251842 0.1082016 0.1421667 0.09921160 0.1511567
## Mar 2021       0.1251842 0.1073727 0.1429956 0.09794391 0.1524244
## Apr 2021       0.1251842 0.1065807 0.1437876 0.09673264 0.1536357
```

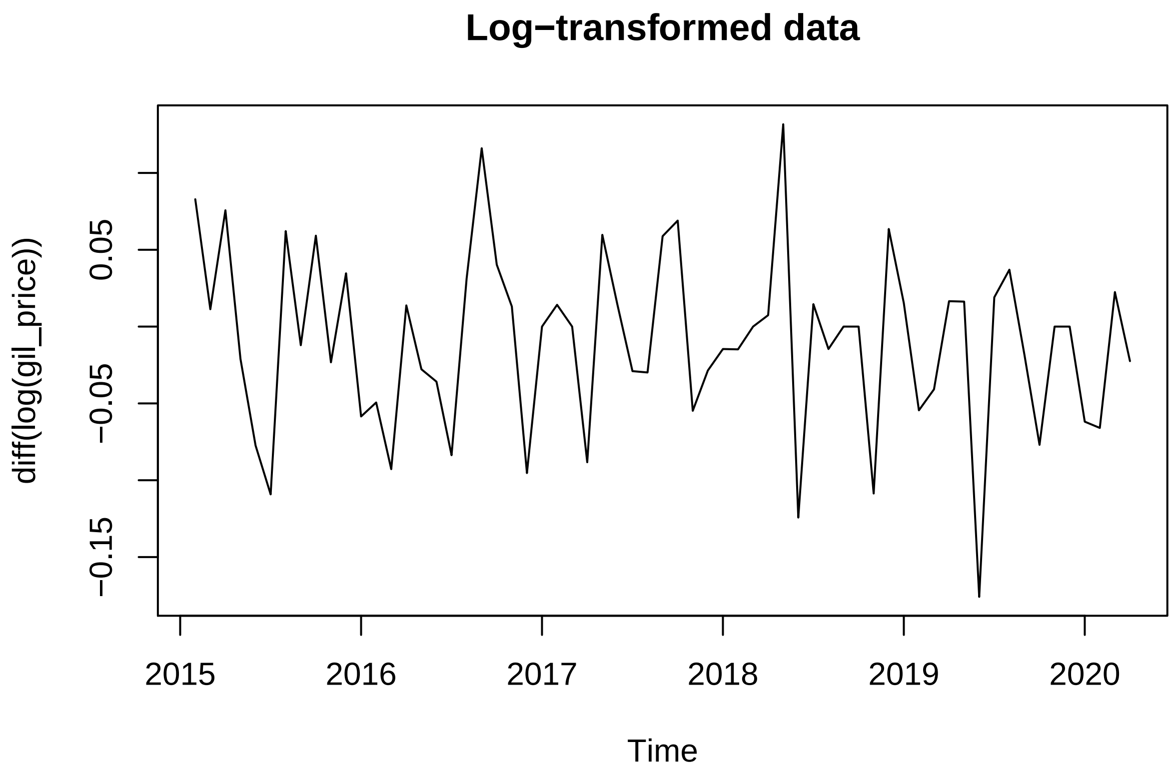
```r
# GILEAD SCIENCES STOCK
gil_price <- ts(gilead$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(gil_price, type = "l")
```




```r
plot(diff(gil_price), type = "l", main = "Original data")
```

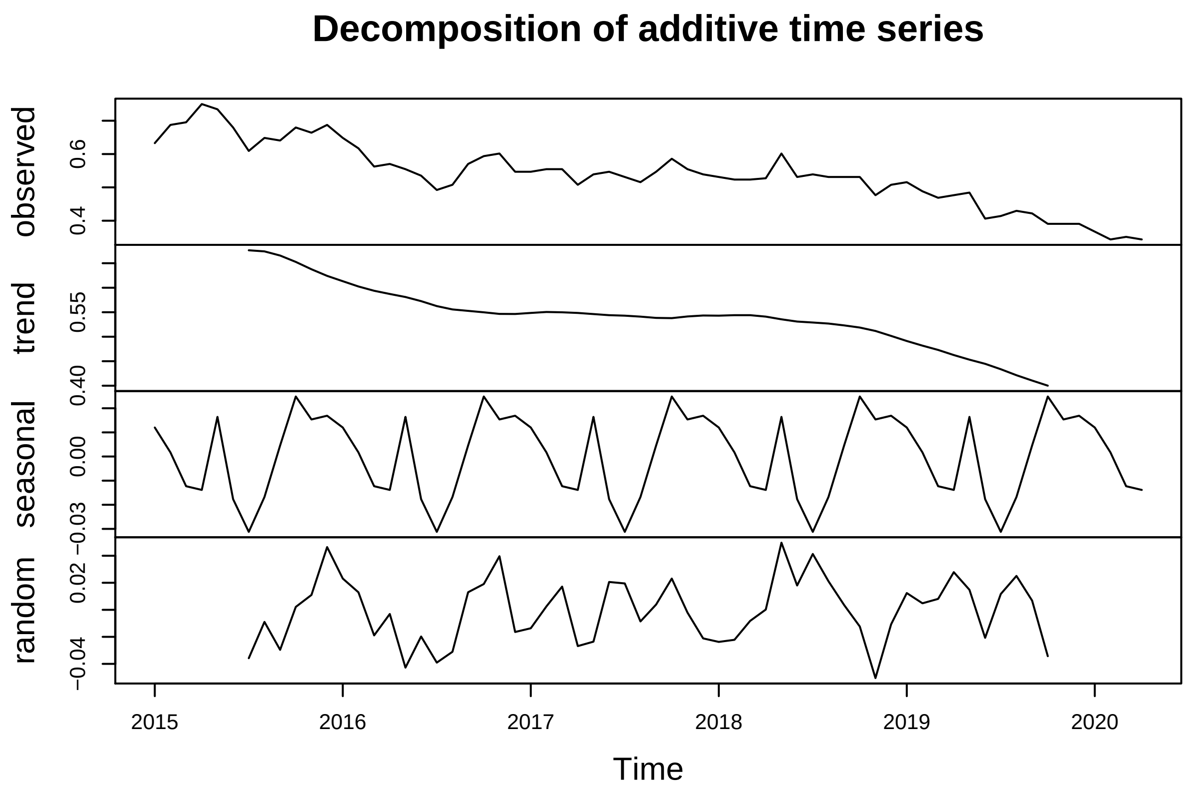



```
# Convert to ln format
gil_lnprice <- log(gil_price)
plot(diff(log(gil_price)), type = "l", main = "Log-transformed data")
```




```
# Moving average on ln of stock price
gil_difflnprice <- diff(gil_lnprice,1)
```

```
# Decompose Gilead Data
plot(decompose(gil_price))
```




```r
# Dickey-Fuller Test
adf.test(gil_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gil_price
## Dickey-Fuller = -1.9531, Lag order = 3, p-value = 0.594
## alternative hypothesis: stationary
```

```r
adf.test(gil_lnprice)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gil_lnprice
## Dickey-Fuller = -1.0438, Lag order = 3, p-value = 0.9241
## alternative hypothesis: stationary
```

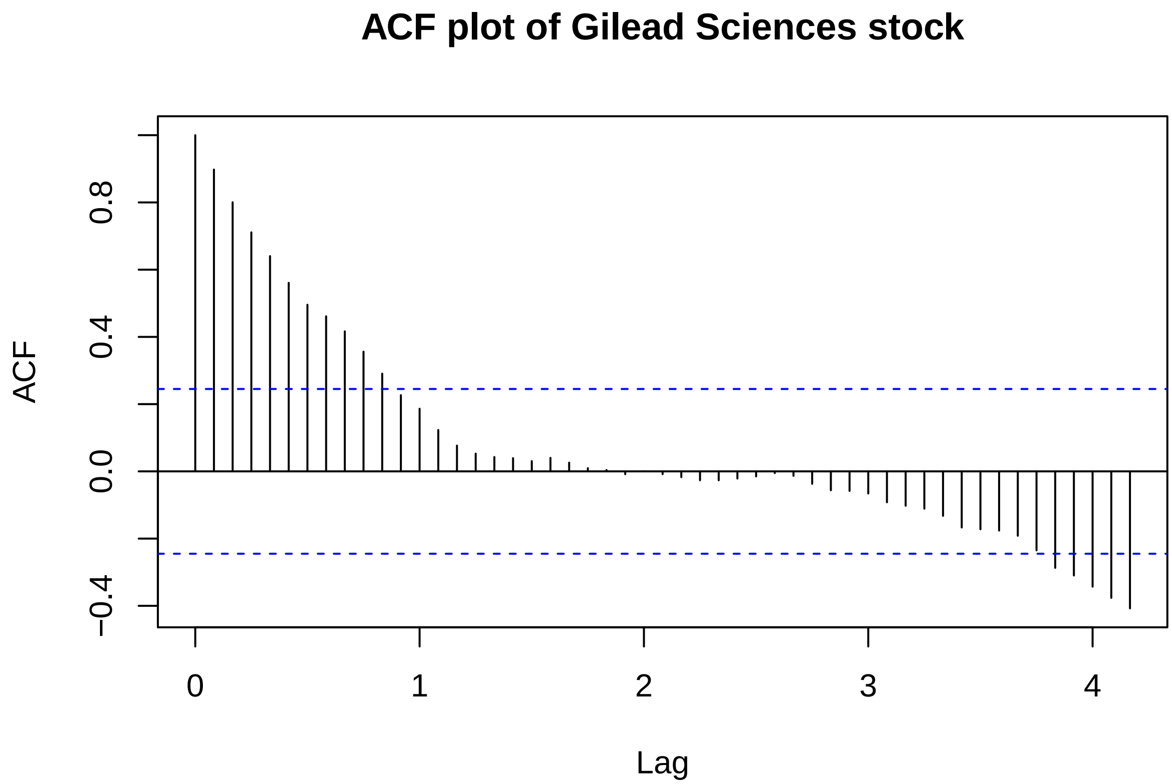
```r
adf.test(gil_difflnprice)
```

```
## Warning in adf.test(gil_difflnprice): p-value smaller than printed p-value
```

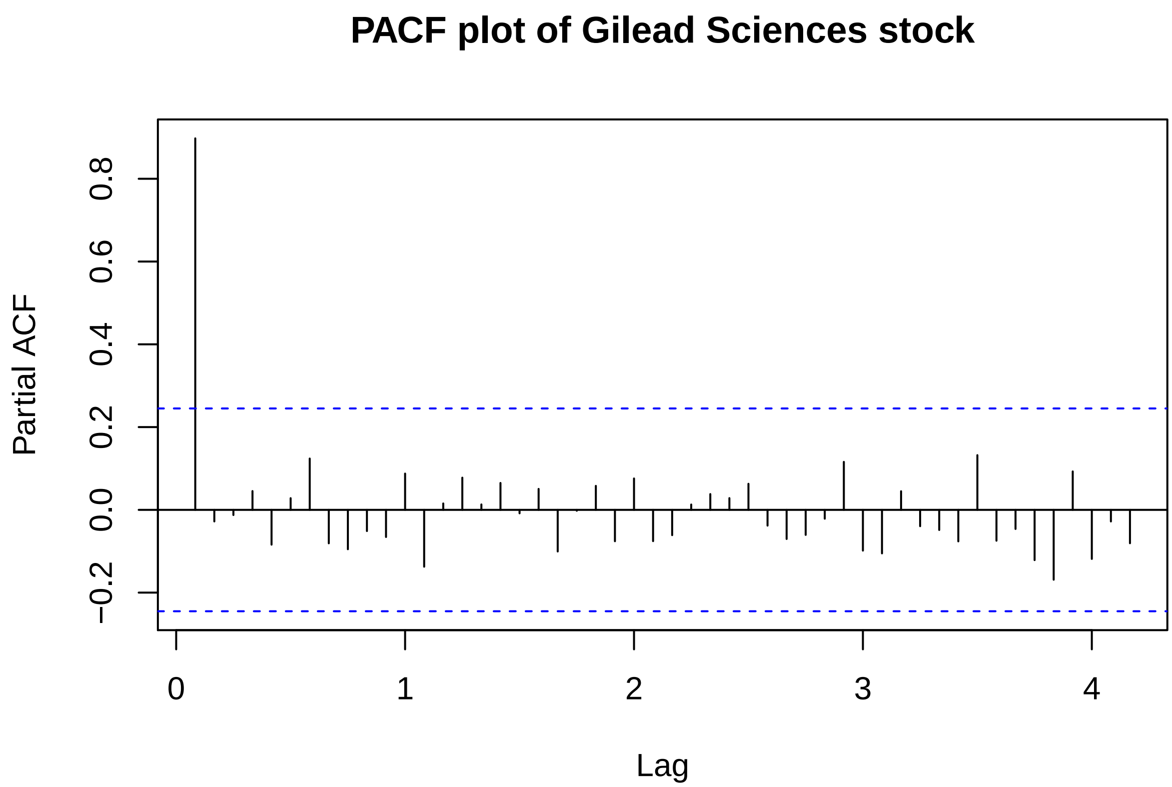
```
##
##  Augmented Dickey-Fuller Test
##
## data:  gil_difflnprice
## Dickey-Fuller = -4.5417, Lag order = 3, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```r
# ACF, PACF
acf(gil_lnprice, lag.max=50, main="ACF plot of Gilead Sciences stock")
```

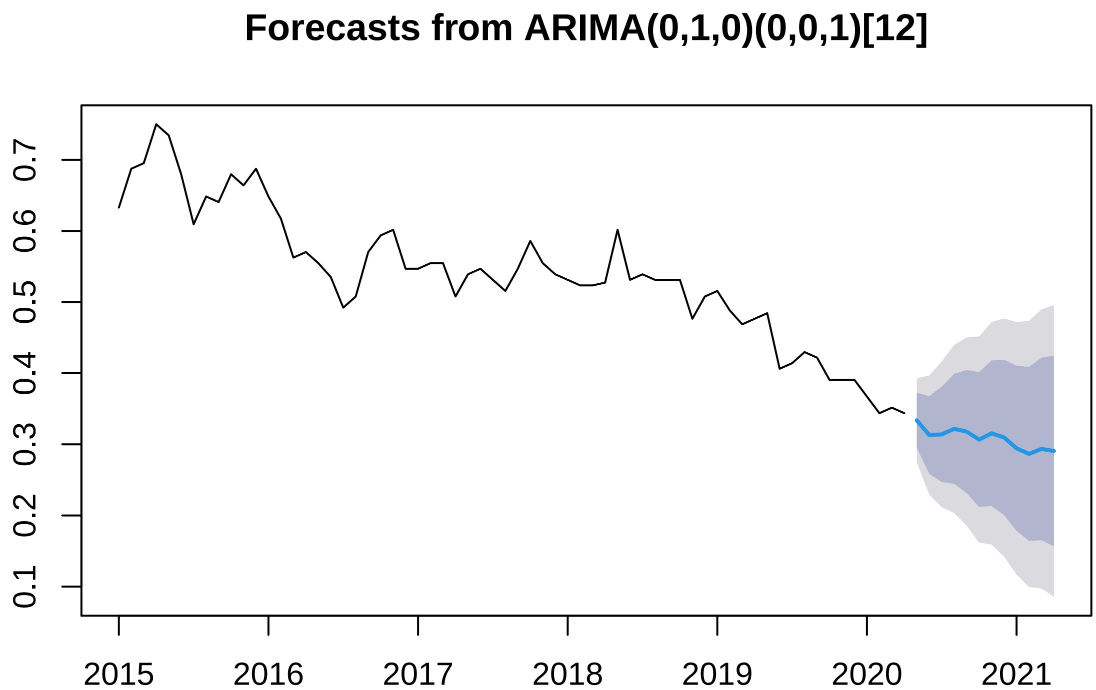



```r
pacf(gil_lnprice, lag.max=50, main="PACF plot of Gilead Sciences stock")
```




```
# Run Auto Arima to determine best Arima Model
arima_gil <- auto.arima(gil_price)
```

```
# Forecast Using Forecast function on Arima Model
forecast_gil <- forecast(arima_gil, h=12)
plot(forecast_gil)
```

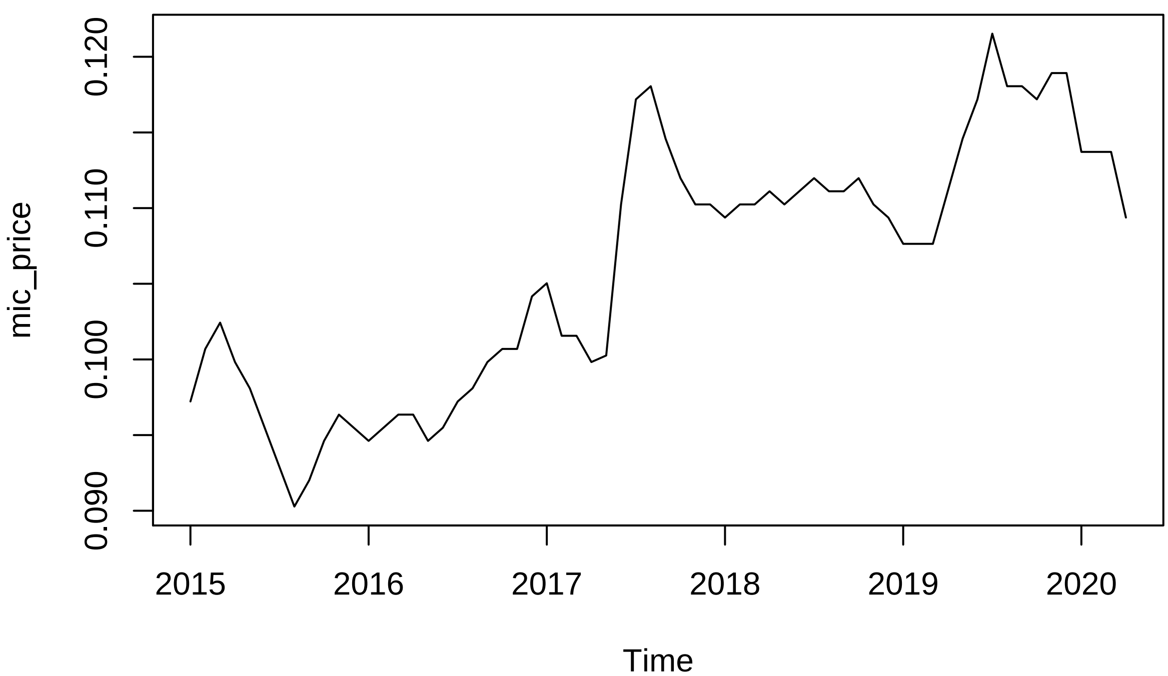



```r
# Summarize Results
summary(forecast_gil)
```

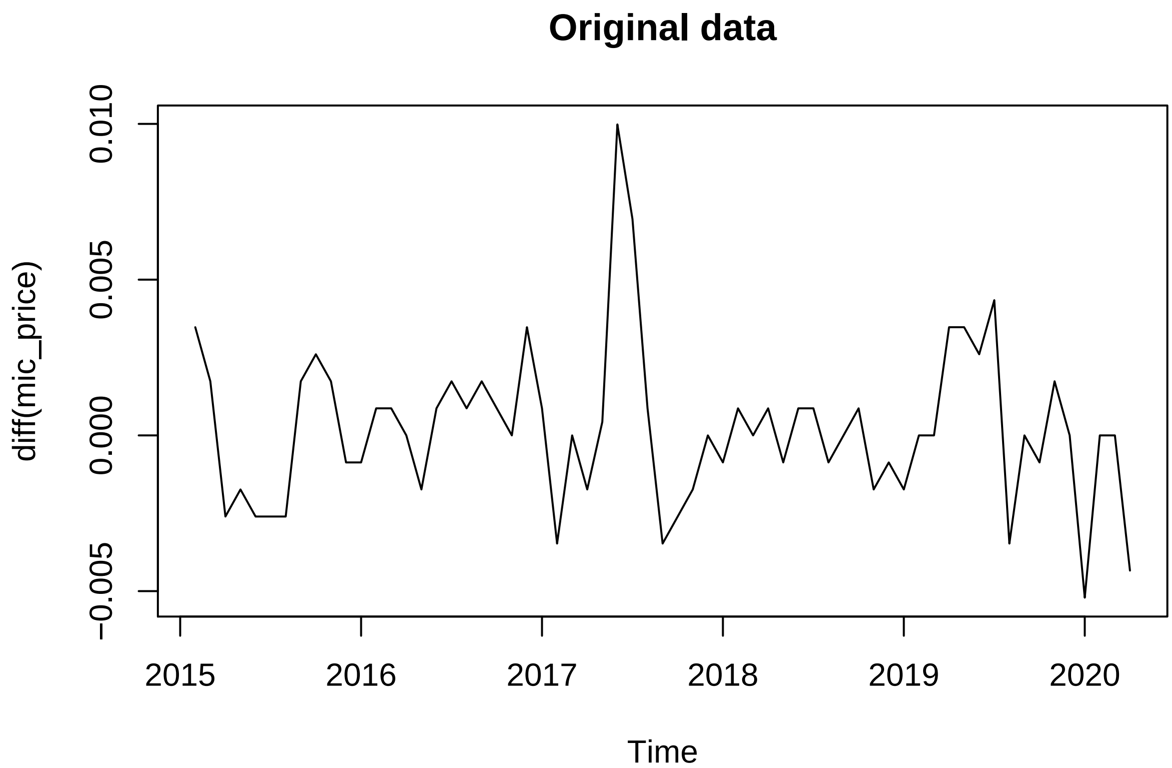
```
##
## Forecast method: ARIMA(0,1,0)(0,0,1)[12]
##
## Model Information:
## Series: gil_price
## ARIMA(0,1,0)(0,0,1)[12]
##
## Coefficients:
##         sma1
##       0.4954
## s.e.  0.1927
##
## sigma^2 estimated as 0.0009117:  log likelihood=129.93
## AIC=-255.86   AICc=-255.66   BIC=-251.57
##
## Error measures:
##                        ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.003561889 0.02971888 0.02333314 -0.8634698 4.308489 0.3249067
##                     ACF1
## Training set -0.03751999
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80      Lo 95     Hi 95
```

```
## May 2020       0.3337415 0.2950426 0.3724404 0.27455669 0.3929264
## Jun 2020       0.3130719 0.2583434 0.3678004 0.22937190 0.3967719
## Jul 2020       0.3140060 0.2469775 0.3810344 0.21149481 0.4165171
## Aug 2020       0.3217753 0.2443776 0.3991731 0.20340565 0.4401450
## Sep 2020       0.3179372 0.2314038 0.4044705 0.18559583 0.4502785
## Oct 2020       0.3066992 0.2119066 0.4014917 0.16172648 0.4516718
## Nov 2020       0.3155367 0.2131491 0.4179244 0.15894836 0.4721251
## Dec 2020       0.3096454 0.2001884 0.4191024 0.14224535 0.4770454
## Jan 2021       0.2943154 0.1782187 0.4104121 0.11676085 0.4718699
## Feb 2021       0.2865149 0.1641390 0.4088908 0.09935712 0.4736726
## Mar 2021       0.2935533 0.1652050 0.4219015 0.09726154 0.4898450
## Apr 2021       0.2906319 0.1565771 0.4246867 0.08561278 0.4956511
```

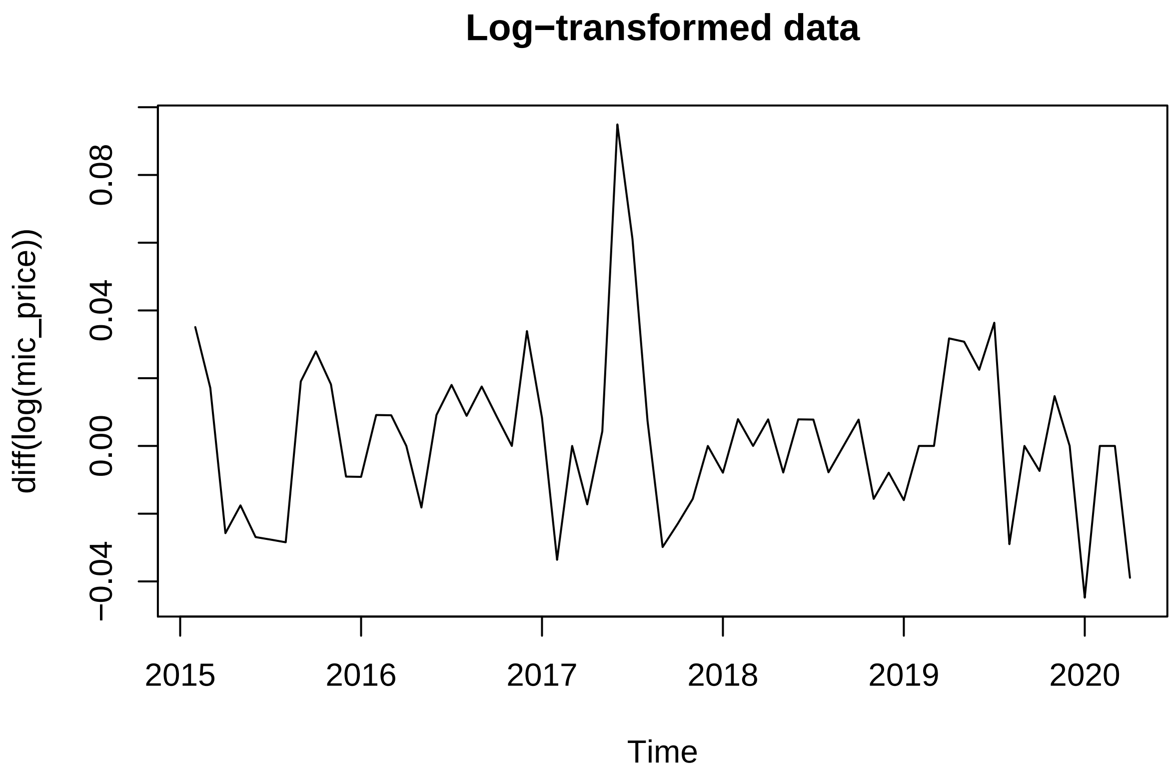
```r
# MICROSOFT STOCK
mic_price <- ts(microsoft$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(mic_price, type = "l")
```




```r
plot(diff(mic_price), type = "l", main = "Original data")
```

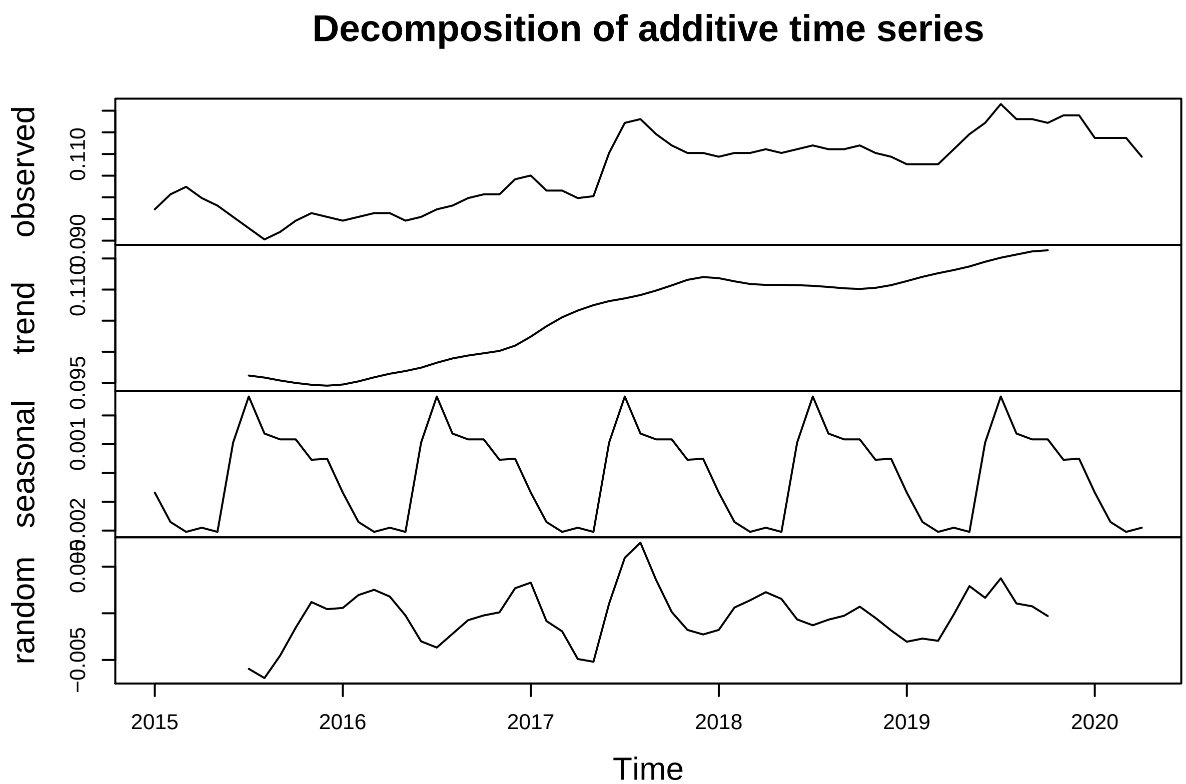



```r
# Convert to ln format
mic_lnprice <- log(mic_price)
plot(diff(log(mic_price)), type = "l", main = "Log-transformed data")
```




```
# Moving average on ln of stock price
mic_difflnprice <- diff(mic_lnprice,1)
```

```
# Decompose Microsoft Data
plot(decompose(mic_price))
```




```r
#Dickey-Fuller Test
adf.test(mic_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mic_price
## Dickey-Fuller = -2.7838, Lag order = 3, p-value = 0.2574
## alternative hypothesis: stationary
```

```r
adf.test(mic_lnprice)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mic_lnprice
## Dickey-Fuller = -2.7299, Lag order = 3, p-value = 0.2792
## alternative hypothesis: stationary
```

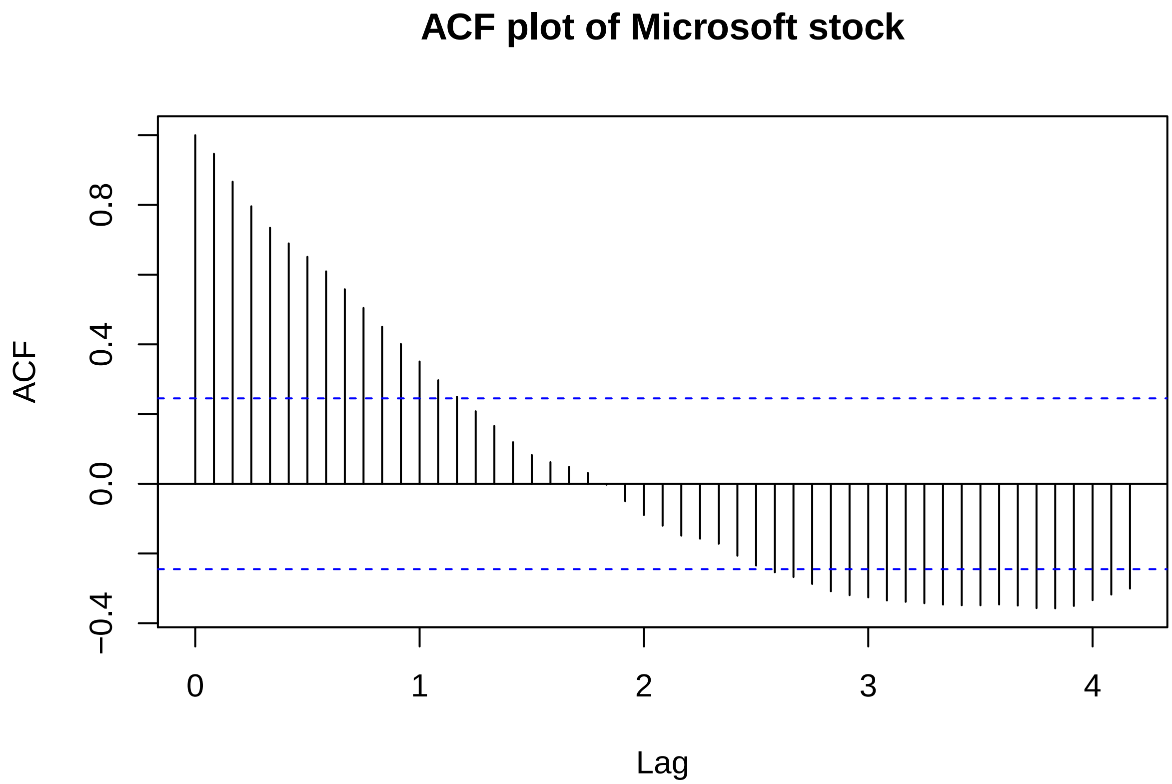
```r
adf.test(mic_difflnprice)
```

```
## Warning in adf.test(mic_difflnprice): p-value smaller than printed p-value
```

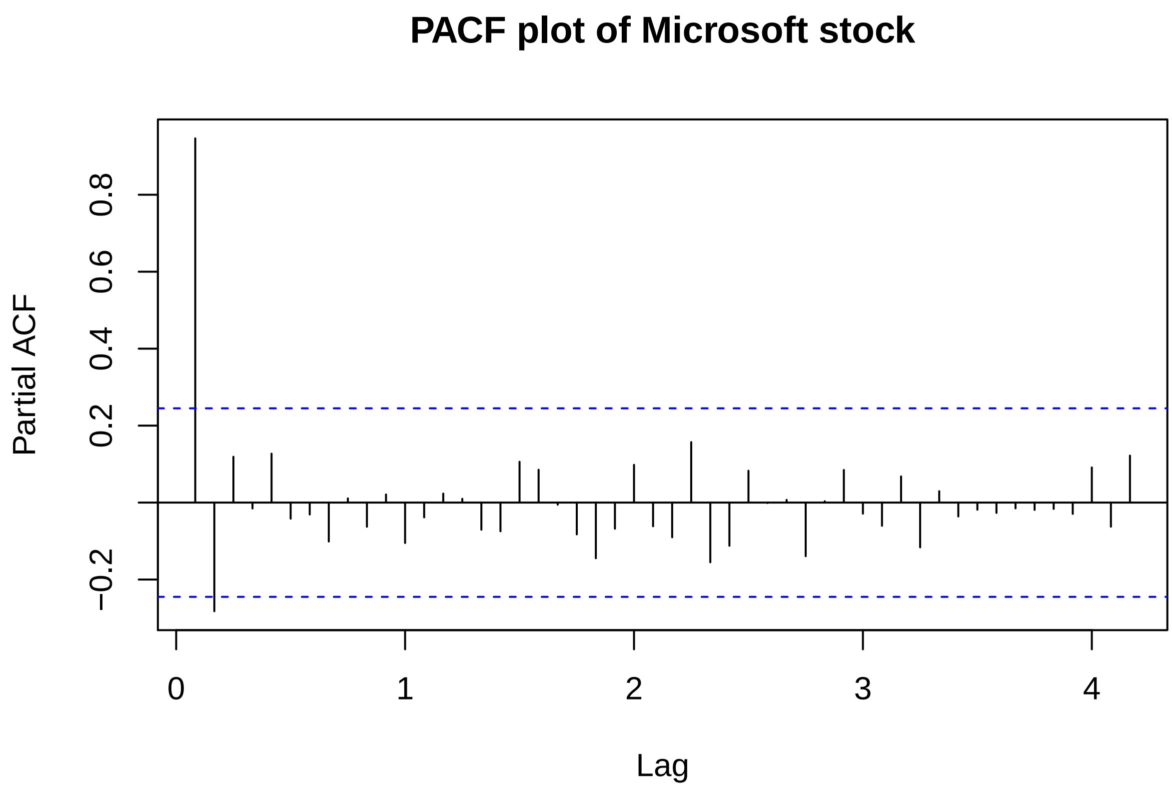
```
##
##  Augmented Dickey-Fuller Test
##
## data:  mic_difflnprice
## Dickey-Fuller = -4.7799, Lag order = 3, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```r
#ACF, PACF
acf(mic_lnprice, lag.max=50, main="ACF plot of Microsoft stock")
```

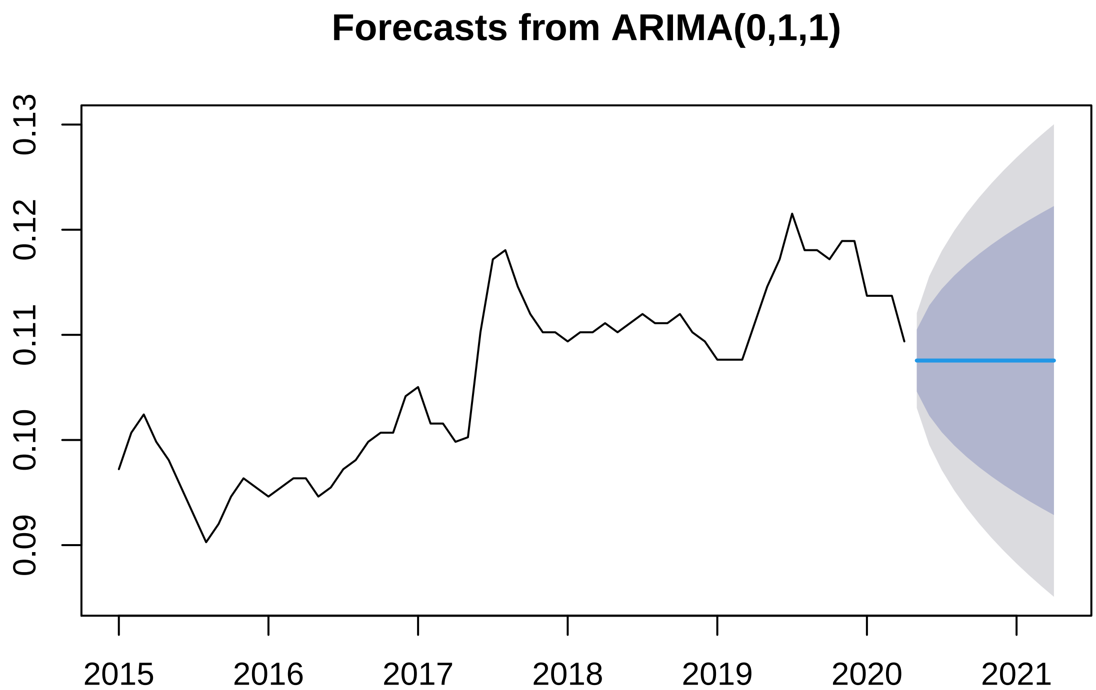



```r
pacf(mic_lnprice, lag.max=50, main="PACF plot of Microsoft stock")
```




```r
# Run Auto Arima to determine best Arima Model
arima_mic <- auto.arima(mic_price)
```

```r
# Forecast Using Forecast function on Arima Model
forecast_mic <- forecast(arima_mic, h=12)
plot(forecast_mic)
```

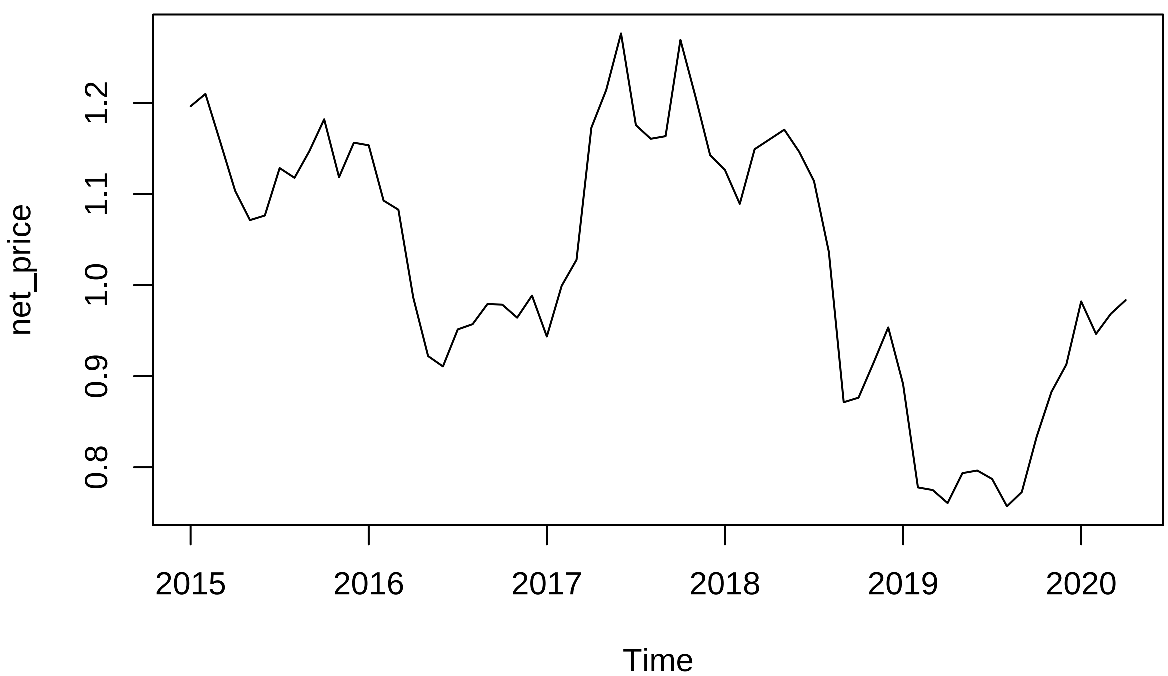



```r
# Summarize Results
summary(forecast_mic)
```

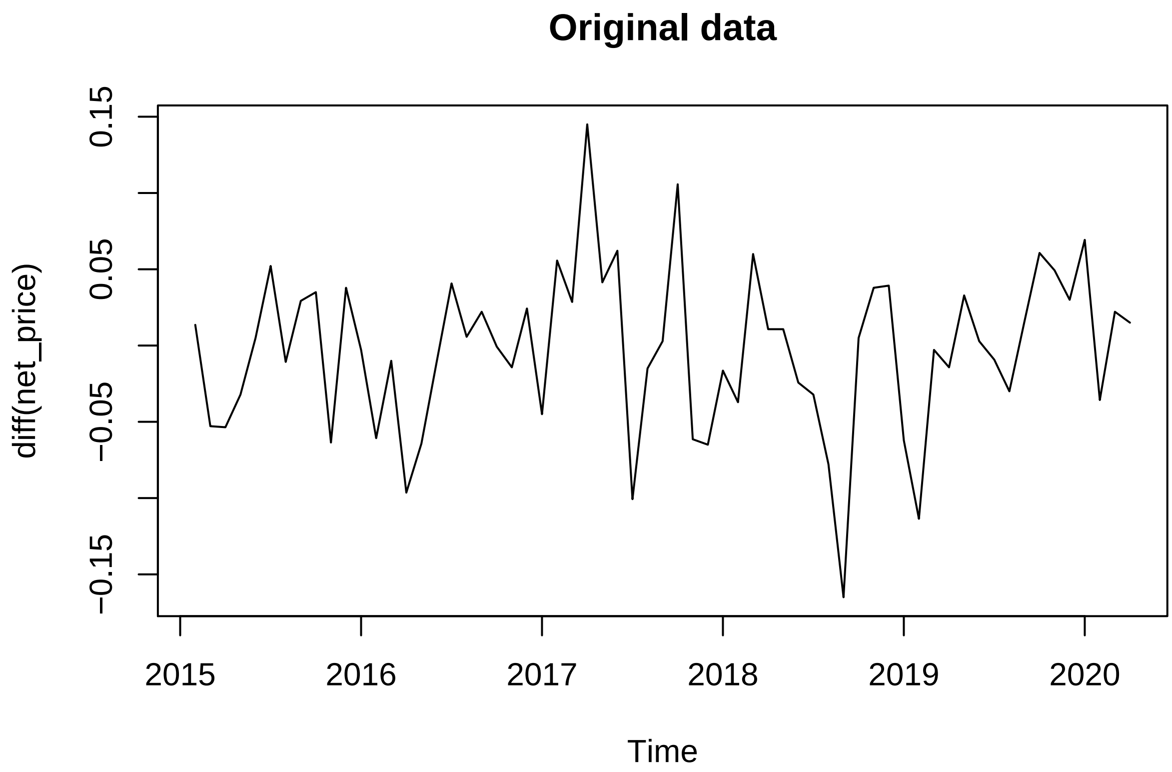
```
##
## Forecast method: ARIMA(0,1,1)
##
## Model Information:
## Series: mic_price
## ARIMA(0,1,1)
##
## Coefficients:
##          ma1
##       0.4711
## s.e.  0.1197
##
## sigma^2 estimated as 5.3e-06:  log likelihood=293.64
## AIC=-583.29   AICc=-583.09   BIC=-579
##
## Error measures:
##                        ME        RMSE         MAE        MPE     MAPE      MASE
## Training set 0.0001092999 0.002265869 0.001704006 0.09503491 1.590398 0.2672166
##                     ACF1
## Training set -0.01809195
##
## Forecasts:
##          Point Forecast      Lo 80      Hi 80      Lo 95     Hi 95
```

```
## May 2020       0.1075574 0.10460716 0.1105077 0.10304537 0.1120695
## Jun 2020       0.1075574 0.10230938 0.1128055 0.09953123 0.1155837
## Jul 2020       0.1075574 0.10074715 0.1143677 0.09714200 0.1179729
## Aug 2020       0.1075574 0.09948168 0.1156332 0.09520662 0.1199083
## Sep 2020       0.1075574 0.09838924 0.1167257 0.09353589 0.1215790
## Oct 2020       0.1075574 0.09741378 0.1177011 0.09204405 0.1230708
## Nov 2020       0.1075574 0.09652423 0.1185907 0.09068360 0.1244313
## Dec 2020       0.1075574 0.09570123 0.1194137 0.08942493 0.1256900
## Jan 2021       0.1075574 0.09493177 0.1201831 0.08824814 0.1268668
## Feb 2021       0.1075574 0.09420658 0.1209083 0.08713905 0.1279758
## Mar 2021       0.1075574 0.09351880 0.1215961 0.08608719 0.1290277
## Apr 2021       0.1075574 0.09286318 0.1222517 0.08508450 0.1300304
```

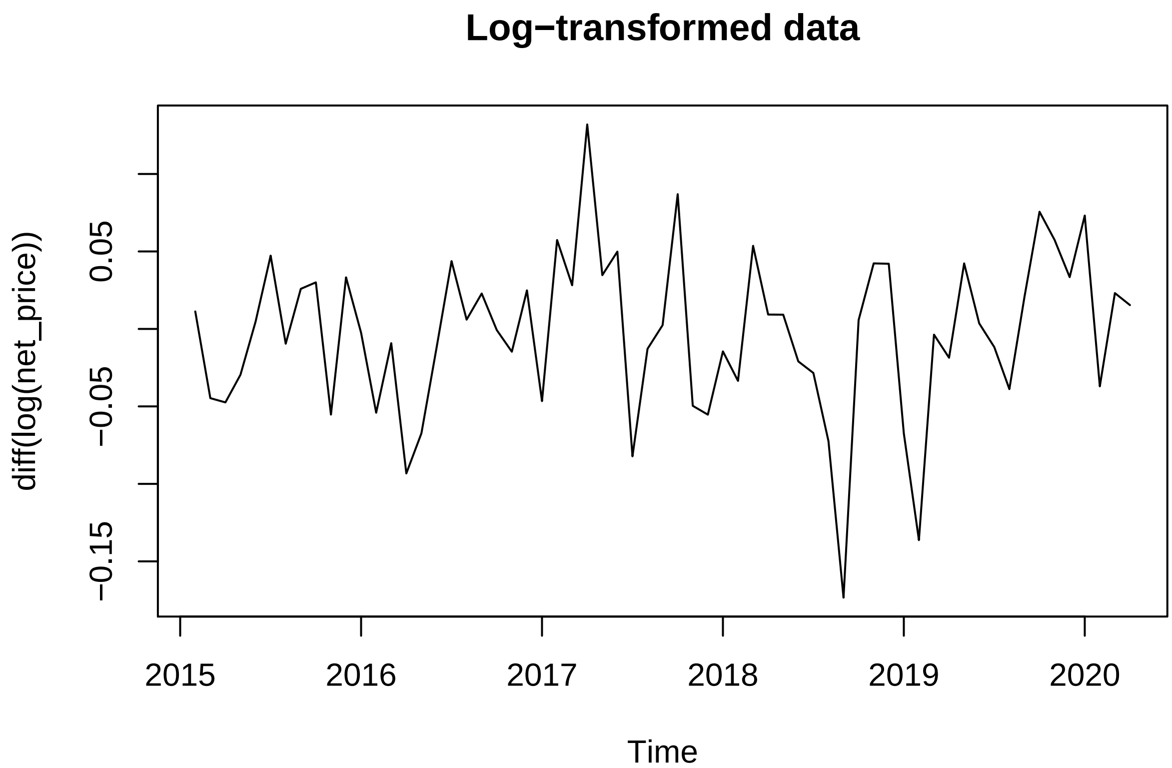
```
# NETFLIX STOCK
net_price <- ts(netflix$Close, start = c(2015,1), end = c(2020,4), frequency = 12)
plot(net_price, type = "l")
```




```
plot(diff(net_price), type = "l", main = "Original data")
```

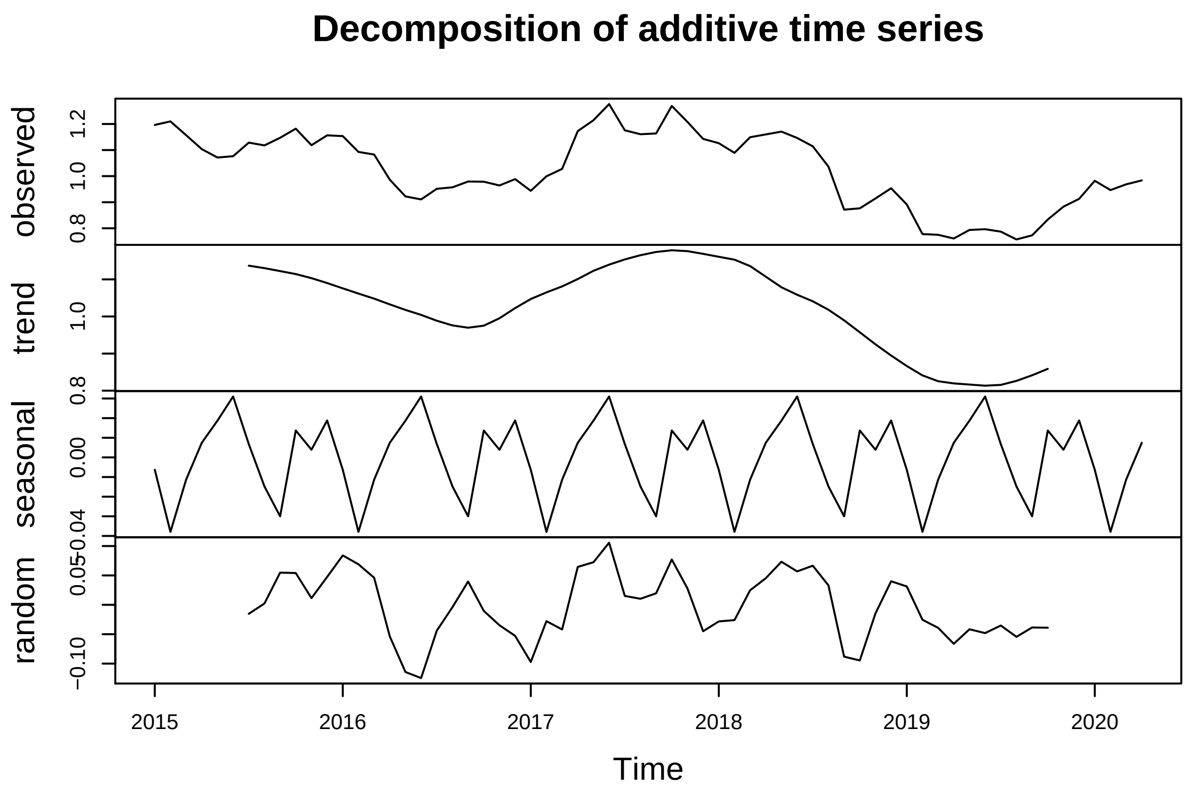



```r
# Convert to ln format
net_lnprice <- log(net_price)
plot(diff(log(net_price)), type = "l", main = "Log-transformed data")
```




```r
# Moving average on ln of stock price
net_difflnprice <- diff(net_lnprice,1)
```

```r
# Decompose Netflix Data
plot(decompose(net_price))
```




```
#Dickey-Fuller Test
adf.test(net_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  net_price
## Dickey-Fuller = -1.685, Lag order = 3, p-value = 0.7026
## alternative hypothesis: stationary
```

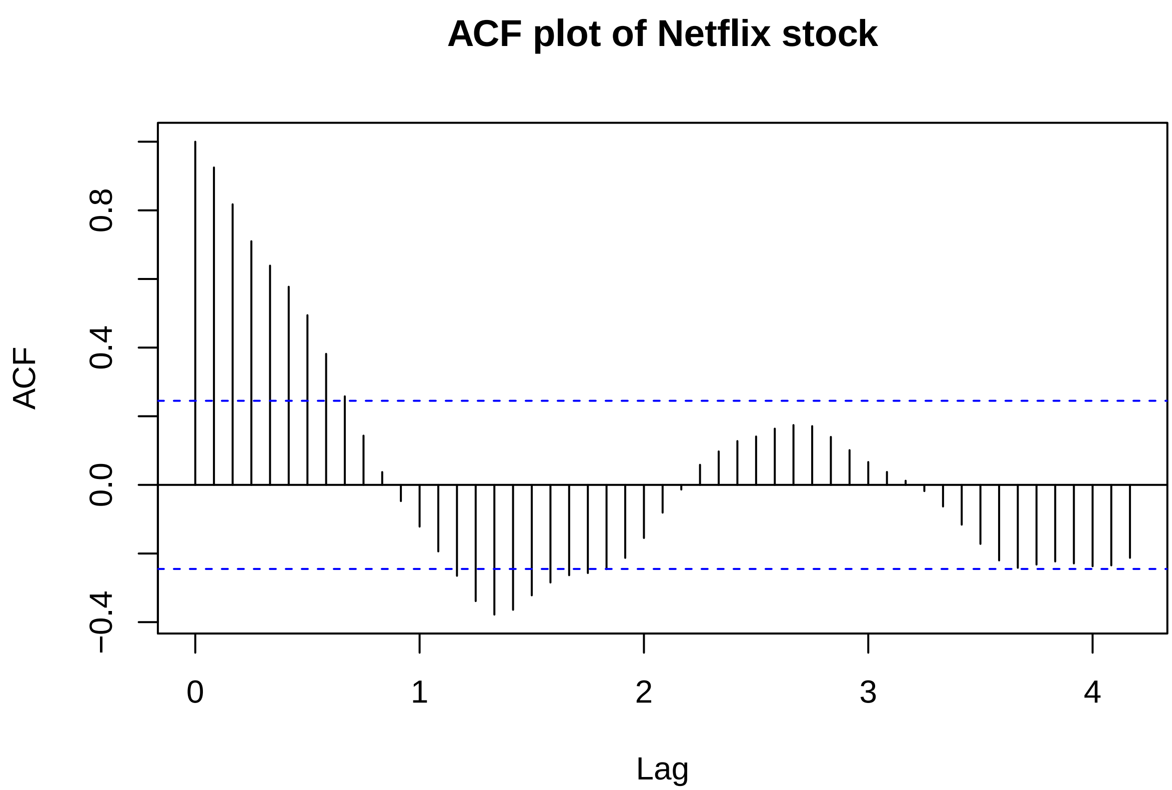
```
adf.test(net_lnprice)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  net_lnprice
## Dickey-Fuller = -1.6354, Lag order = 3, p-value = 0.7227
## alternative hypothesis: stationary
```

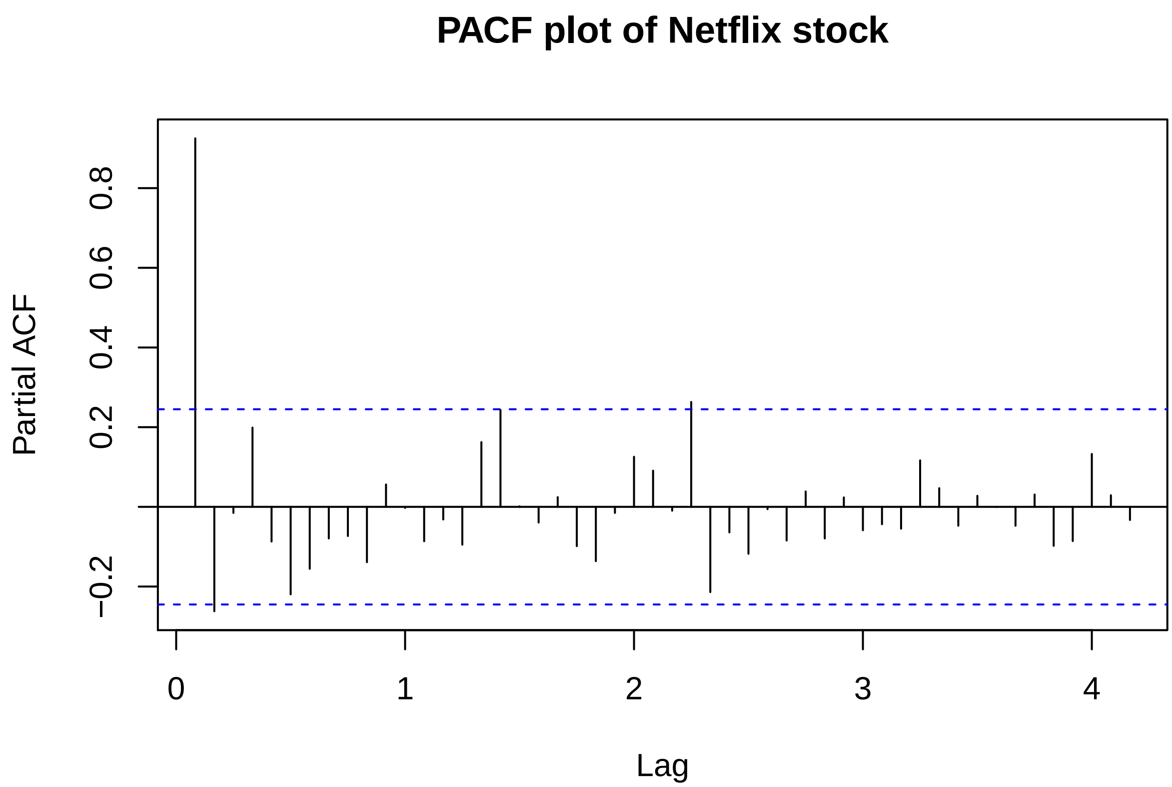
```
adf.test(net_difflnprice)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  net_difflnprice
## Dickey-Fuller = -3.8888, Lag order = 3, p-value = 0.02035
## alternative hypothesis: stationary
```

```r
#ACF, PACF
acf(net_lnprice, lag.max=50, main="ACF plot of Netflix stock")
```

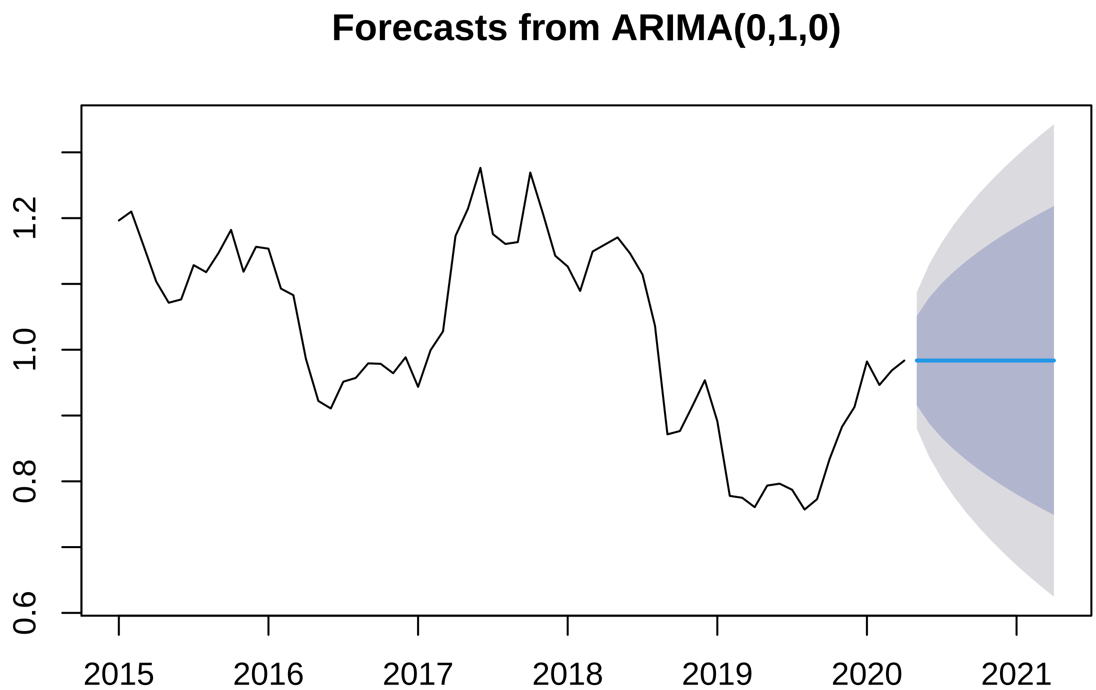



```r
pacf(net_lnprice, lag.max=50, main="PACF plot of Netflix stock")
```




```
# Run Auto Arima to determine best Arima Model
arima_net <- auto.arima(net_price)
```

```
# Forecast Using Forecast function on Arima Model
forecast_net <- forecast(arima_net, h=12)
plot(forecast_net)
```




```r
# Summarize Results
summary(forecast_net)
```

```
##
## Forecast method: ARIMA(0,1,0)
##
## Model Information:
## Series: net_price
## ARIMA(0,1,0)
##
## sigma^2 estimated as 0.002797:  log likelihood=95.8
## AIC=-189.6   AICc=-189.54   BIC=-187.46
##
## Error measures:
##                        ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.003307198 0.05247303 0.03986245 -0.4390323 3.929701 0.2141687
##                   ACF1
## Training set 0.2075696
##
## Forecasts:
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## May 2020      0.9835714 0.9157929 1.051350 0.8799131 1.087230
## Jun 2020      0.9835714 0.8877181 1.079425 0.8369765 1.130166
## Jul 2020      0.9835714 0.8661756 1.100967 0.8040300 1.163113
## Aug 2020      0.9835714 0.8480144 1.119128 0.7762549 1.190888
## Sep 2020      0.9835714 0.8320141 1.135129 0.7517845 1.215358
```

```
## Oct 2020      0.9835714 0.8175487 1.149594 0.7296616 1.237481
## Nov 2020      0.9835714 0.8042464 1.162896 0.7093174 1.257825
## Dec 2020      0.9835714 0.7918649 1.175278 0.6903816 1.276761
## Jan 2021      0.9835714 0.7802359 1.186907 0.6725966 1.294546
## Feb 2021      0.9835714 0.7692370 1.197906 0.6557752 1.311368
## Mar 2021      0.9835714 0.7587756 1.208367 0.6397759 1.327367
## Apr 2021      0.9835714 0.7487798 1.218363 0.6244887 1.342654
```

```r
# Drop Columns for FB Prophet & Rename them

app_drop = subset(app_drop, select= -c(Open, High, Low, Adj.Close, Volume))
amg_drop = subset(amg_drop, select= -c(Open, High, Low, Adj.Close, Volume))
com_drop = subset(com_drop, select= -c(Open, High, Low, Adj.Close, Volume))
gil_drop = subset(gil_drop, select= -c(Open, High, Low, Adj.Close, Volume))
mic_drop = subset(mic_drop, select= -c(Open, High, Low, Adj.Close, Volume))
net_drop = subset(net_drop, select= -c(Open, High, Low, Adj.Close, Volume))
```

```r
# Change Column Names for Prophet to run

names(app_drop)[names(app_drop) == "Date"] <- "ds"
names(app_drop)[names(app_drop) == "Close"] <- "y"

names(amg_drop)[names(amg_drop) == "Date"] <- "ds"
names(amg_drop)[names(amg_drop) == "Close"] <- "y"

names(com_drop)[names(com_drop) == "Date"] <- "ds"
names(com_drop)[names(com_drop) == "Close"] <- "y"

names(gil_drop)[names(gil_drop) == "Date"] <- "ds"
names(gil_drop)[names(gil_drop) == "Close"] <- "y"

names(mic_drop)[names(mic_drop) == "Date"] <- "ds"
names(mic_drop)[names(mic_drop) == "Close"] <- "y"

names(net_drop)[names(net_drop) == "Date"] <- "ds"
names(net_drop)[names(net_drop) == "Close"] <- "y"

head(app_drop,5)
```

```
##              ds      y
## 8589 2015-01-02 109.33
## 8590 2015-01-05 106.25
## 8591 2015-01-06 106.26
## 8592 2015-01-07 107.75
## 8593 2015-01-08 111.89
```

```r
head(amg_drop,5)
```

```
##              ds      y
## 7955 2015-01-02 159.89
## 7956 2015-01-05 157.99
## 7957 2015-01-06 152.90
## 7958 2015-01-07 158.24
## 7959 2015-01-08 157.67
```

```r
head(com_drop,5)
```

```
##              ds      y
## 8778 2015-01-02 28.675
## 8779 2015-01-05 27.980
## 8780 2015-01-06 27.615
## 8781 2015-01-07 27.590
## 8782 2015-01-08 28.190
```

```r
head(gil_drop,5)
```

```
##              ds      y
## 5782 2015-01-02  94.91
## 5783 2015-01-05  96.79
## 5784 2015-01-06  97.65
## 5785 2015-01-07  99.48
## 5786 2015-01-08 102.30
```

```r
head(mic_drop,5)
```

```
##              ds     y
## 7264 2015-01-02 46.76
## 7265 2015-01-05 46.33
## 7266 2015-01-06 45.65
## 7267 2015-01-07 46.23
## 7268 2015-01-08 47.59
```

```r
head(net_drop,5)
```

```
##              ds        y
## 3176 2015-01-02 49.84857
## 3177 2015-01-05 47.31143
## 3178 2015-01-06 46.50143
## 3179 2015-01-07 46.74286
## 3180 2015-01-08 47.78000
```

```r
# Apple Forecast

m1 <- prophet(app_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```r
future1 <- make_future_dataframe(m1, periods = 365)
tail(future1)
```

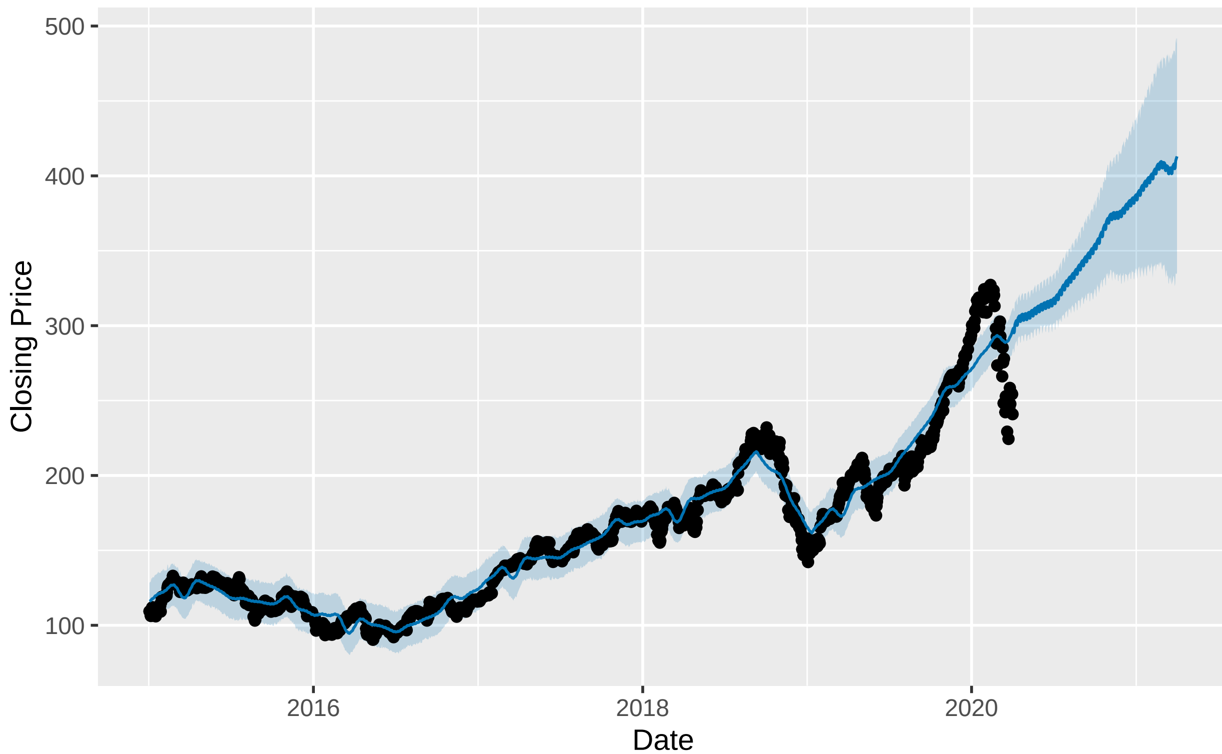
```
##              ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
## 1686 2021-04-01
```

```r
forecast1 <- predict(m1, future1)
tail(forecast1[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##              ds     yhat yhat_lower yhat_upper
## 1681 2021-03-27 404.7167   326.2127   483.6015
```

```
## 1682 2021-03-28 405.4338   327.9639   480.1474
## 1683 2021-03-29 410.0154   335.0882   491.8736
## 1684 2021-03-30 411.0892   334.4756   488.3412
## 1685 2021-03-31 412.0427   334.4276   491.8973
## 1686 2021-04-01 412.9749   335.3948   491.4239
```

```
forecast_plot1 <- plot(m1, forecast1, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot1
```




```
# Amgen Forecast
m2 <- prophet(amg_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
future2 <- make_future_dataframe(m2, periods = 365)
tail(future2)
```

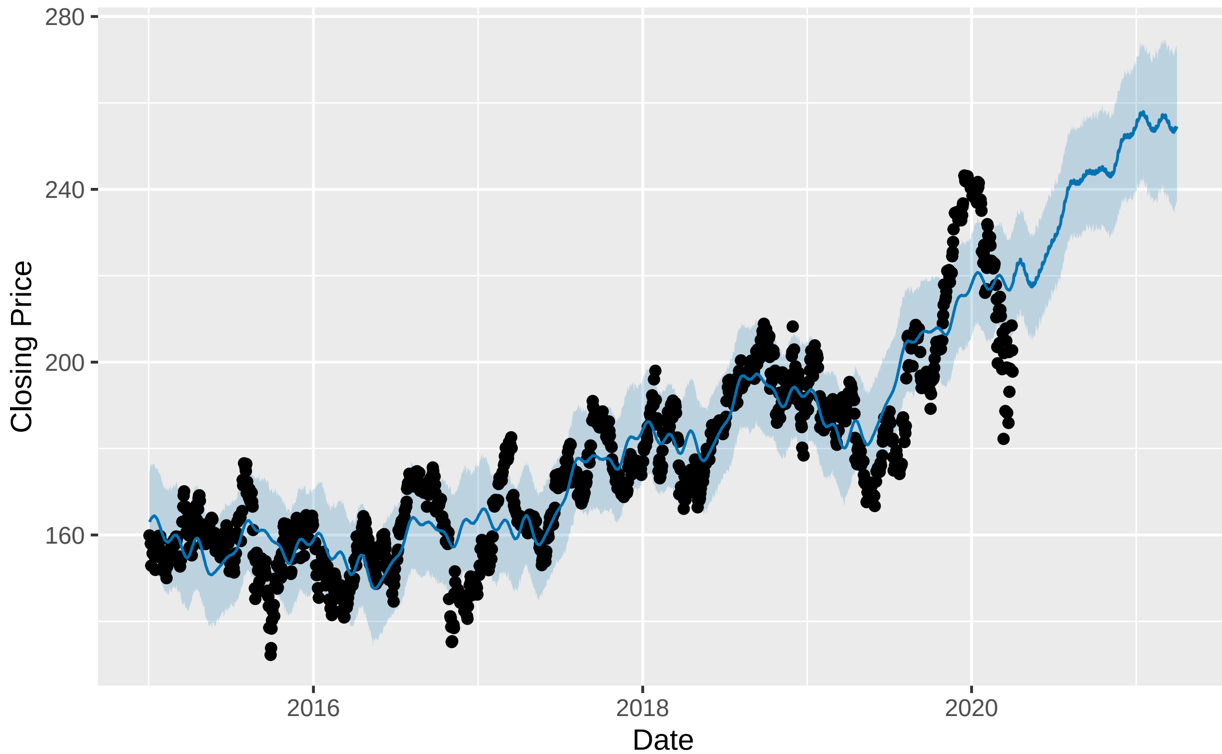
```
##              ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
## 1686 2021-04-01
```

```
forecast2 <- predict(m2, future2)
tail(forecast2[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##              ds     yhat yhat_lower yhat_upper
```

```
## 1681 2021-03-27 254.1915   236.0628   272.0221
## 1682 2021-03-28 254.3491   235.8432   272.9411
## 1683 2021-03-29 253.7787   236.3034   272.4784
## 1684 2021-03-30 254.0939   236.5655   271.7975
## 1685 2021-03-31 254.2732   237.5749   273.3858
## 1686 2021-04-01 254.5397   236.6798   272.9314
```

```r
forecast_plot2 <- plot(m2, forecast2, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot2
```




```r
# Comcast Forecast
m3 <- prophet(com_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

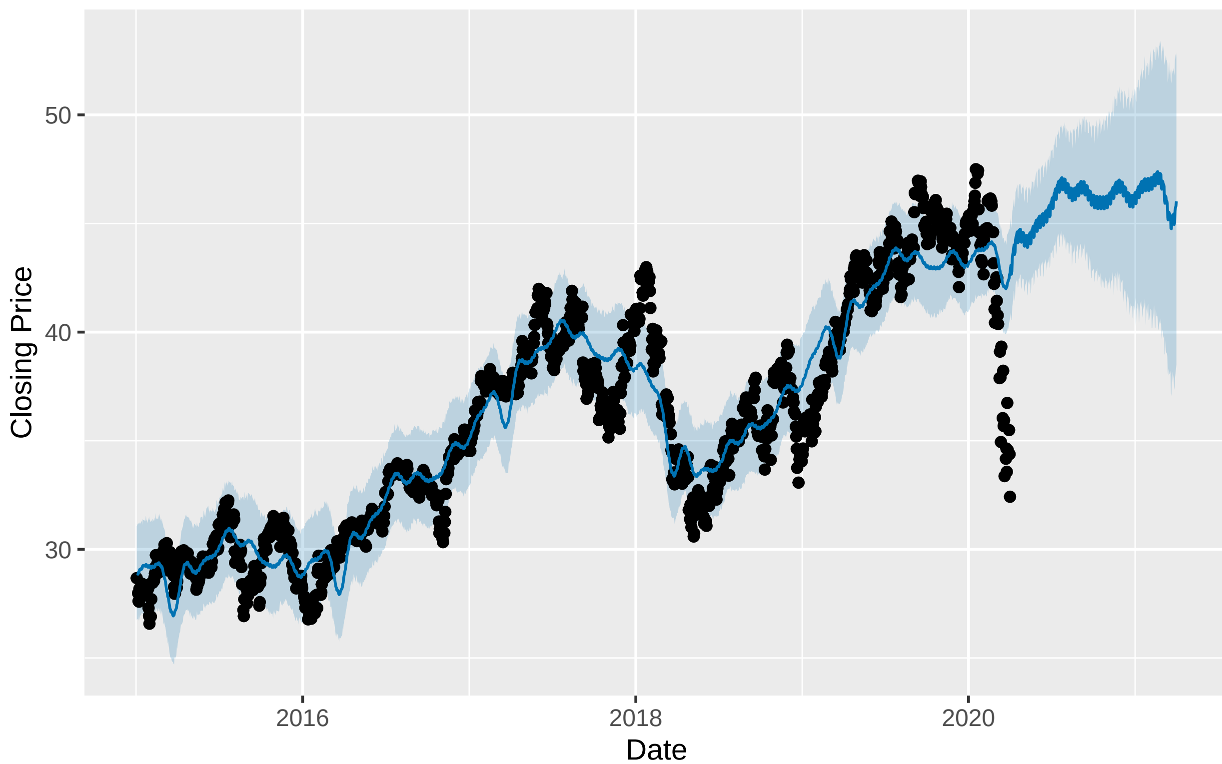
```r
future3 <- make_future_dataframe(m3, periods = 365)
tail(future3)
```

```
##              ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
## 1686 2021-04-01
```

```r
forecast3 <- predict(m3, future3)
tail(forecast3[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##              ds     yhat yhat_lower yhat_upper
## 1681 2021-03-27 44.94827   37.45362   51.95707
## 1682 2021-03-28 45.02905   37.34235   51.81408
## 1683 2021-03-29 45.57790   37.92933   52.80763
## 1684 2021-03-30 45.74186   38.08575   52.72167
## 1685 2021-03-31 45.80648   38.30389   52.56314
## 1686 2021-04-01 46.01798   38.50828   52.87689
```

```r
forecast_plot3 <- plot(m3, forecast3, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot3
```




```r
# Gilead Forecast
m4 <- prophet(gil_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

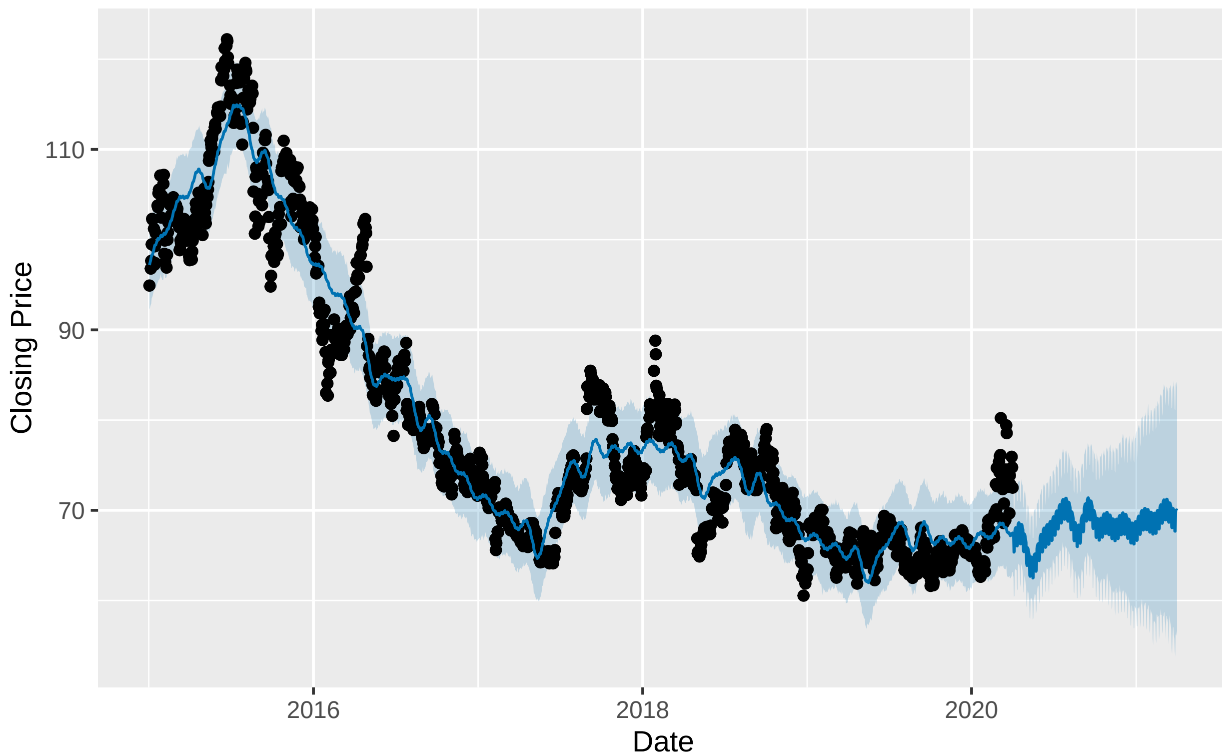
```r
future4 <- make_future_dataframe(m4, periods = 365)
tail(future4)
```

```
##              ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
## 1686 2021-04-01
```

```r
forecast4 <- predict(m4, future4)
tail(forecast4[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##                ds     yhat yhat_lower yhat_upper
## 1681 2021-03-27 67.68985   54.43690   81.24792
## 1682 2021-03-28 67.67304   53.79762   81.07737
## 1683 2021-03-29 69.87807   56.25703   83.88188
## 1684 2021-03-30 70.01194   56.13002   84.24513
## 1685 2021-03-31 70.05502   56.62474   84.01749
## 1686 2021-04-01 69.95568   56.00136   83.62395
```

```
forecast_plot4 <- plot(m4, forecast4, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot4
```




```
# Microsoft Forecast
m5 <- prophet(mic_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

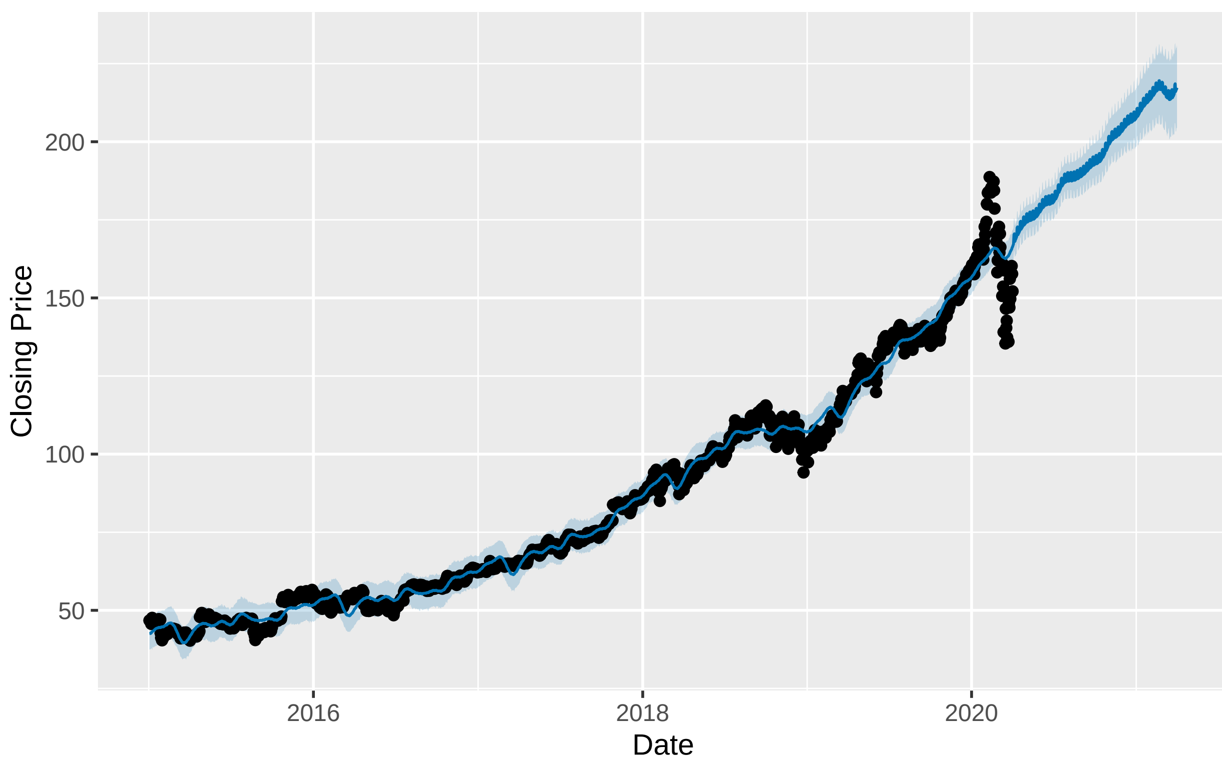
```
future5 <- make_future_dataframe(m5, periods = 365)
tail(future5)
```

```
##                ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
## 1686 2021-04-01
```

```r
forecast5 <- predict(m5, future5)
tail(forecast5[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##              ds     yhat yhat_lower yhat_upper
## 1681 2021-03-27 218.2346   205.9046   231.1006
## 1682 2021-03-28 218.5573   205.9576   231.6122
## 1683 2021-03-29 216.0782   203.3043   229.0161
## 1684 2021-03-30 216.5677   203.2874   230.0511
## 1685 2021-03-31 216.8445   204.5079   230.1040
## 1686 2021-04-01 217.2859   204.3561   230.9279
```

```r
forecast_plot5 <- plot(m5, forecast5, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot5
```




```r
# Netflix Forecast
m6 <- prophet(net_drop)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```r
future6 <- make_future_dataframe(m6, periods = 365)
tail(future6)
```

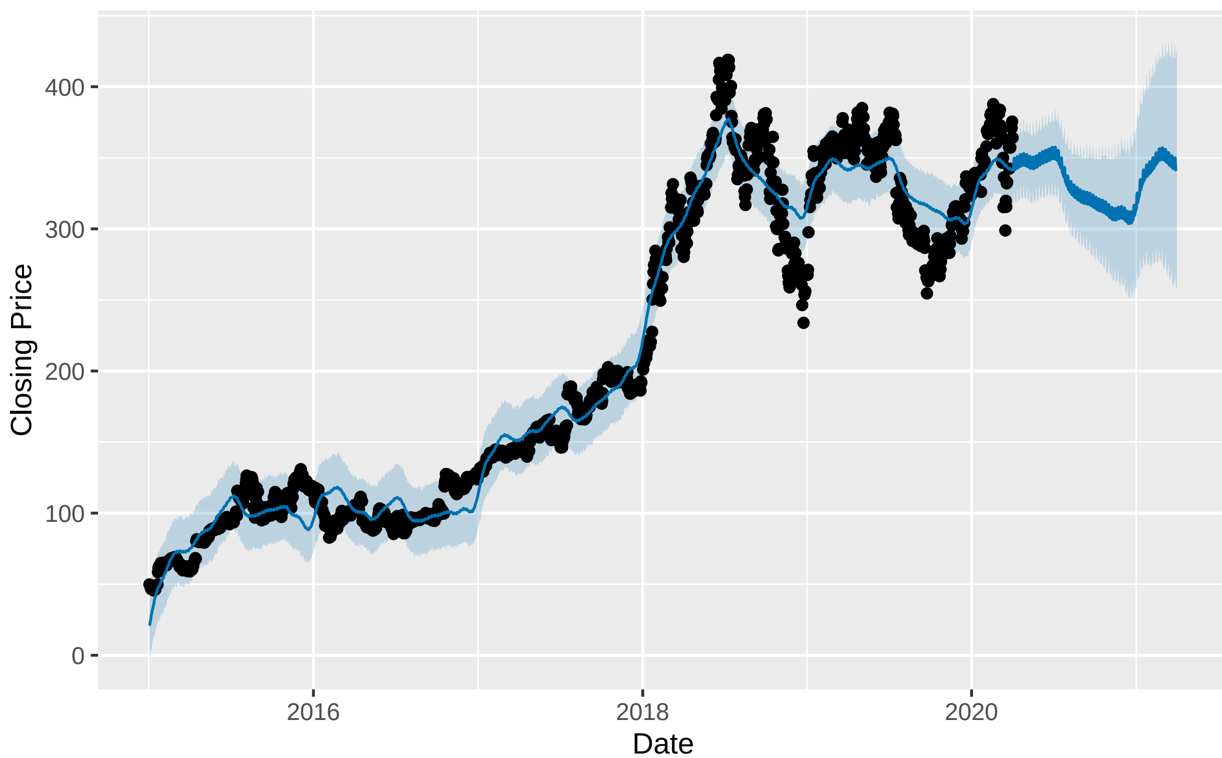
```
##              ds
## 1681 2021-03-27
## 1682 2021-03-28
## 1683 2021-03-29
## 1684 2021-03-30
## 1685 2021-03-31
```

```
## 1686 2021-04-01
```

```r
forecast6 <- predict(m6, future6)
tail(forecast6[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##              ds     yhat yhat_lower yhat_upper
## 1681 2021-03-27 350.0129   267.8535   431.8509
## 1682 2021-03-28 349.8855   265.7516   427.2333
## 1683 2021-03-29 341.4205   260.3253   420.0349
## 1684 2021-03-30 342.2534   258.6489   421.6609
## 1685 2021-03-31 341.9360   257.9795   425.1060
## 1686 2021-04-01 341.9094   257.3491   426.5237
```

```r
forecast_plot6 <- plot(m6, forecast6, xlabel = 'Date', ylabel = "Closing Price")
forecast_plot6
```




```r
# Apple Evaluation

app_cv <- cross_validation(m1, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

```r
head(app_cv)
```

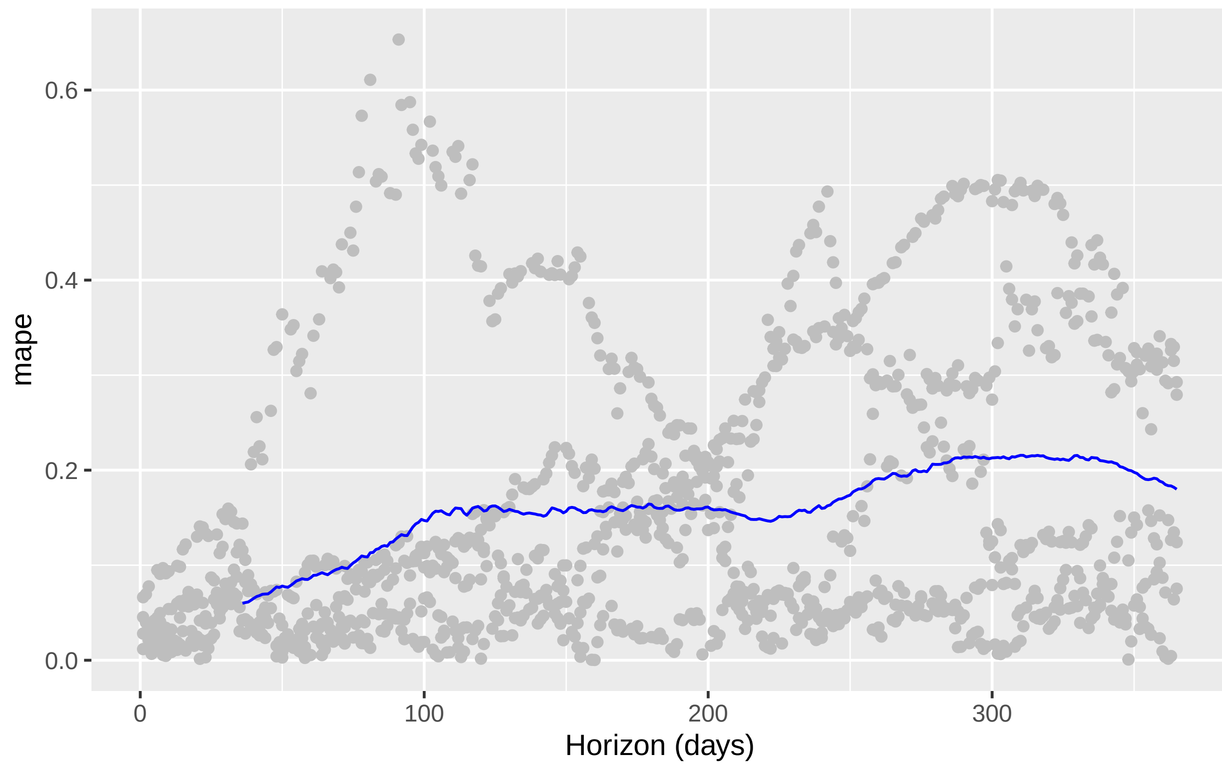
```
##        y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 141.05 2017-04-13 147.4463   143.2733   152.1332 2017-04-12
## 2 141.83 2017-04-17 147.7661   143.1296   152.3697 2017-04-12
## 3 141.20 2017-04-18 147.5084   142.5252   151.8505 2017-04-12
## 4 140.68 2017-04-19 147.6685   143.1506   152.1183 2017-04-12
```

```
## 5 142.44 2017-04-20 147.4569   142.6550   151.9279 2017-04-12
## 6 142.27 2017-04-21 147.1283   142.8957   151.5146 2017-04-12
```

```r
app_perf <- performance_metrics(app_cv)
head(app_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 204.5786 14.30310 11.32127 0.05994330 0.3120000
## 2 37 days 208.3522 14.43441 11.46861 0.06068139 0.3066667
## 3 38 days 210.5856 14.51157 11.58032 0.06124137 0.2933333
## 4 39 days 224.5532 14.98510 11.94960 0.06313905 0.2720000
## 5 40 days 240.1119 15.49554 12.34673 0.06521411 0.2533333
## 6 41 days 257.3208 16.04122 12.67973 0.06702853 0.2500000
```

```r
plot_cross_validation_metric(app_cv, metric = 'mape')
```




```r
# Amgen Evaluation
amg_cv <- cross_validation(m2, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

```r
head(amg_cv)
```

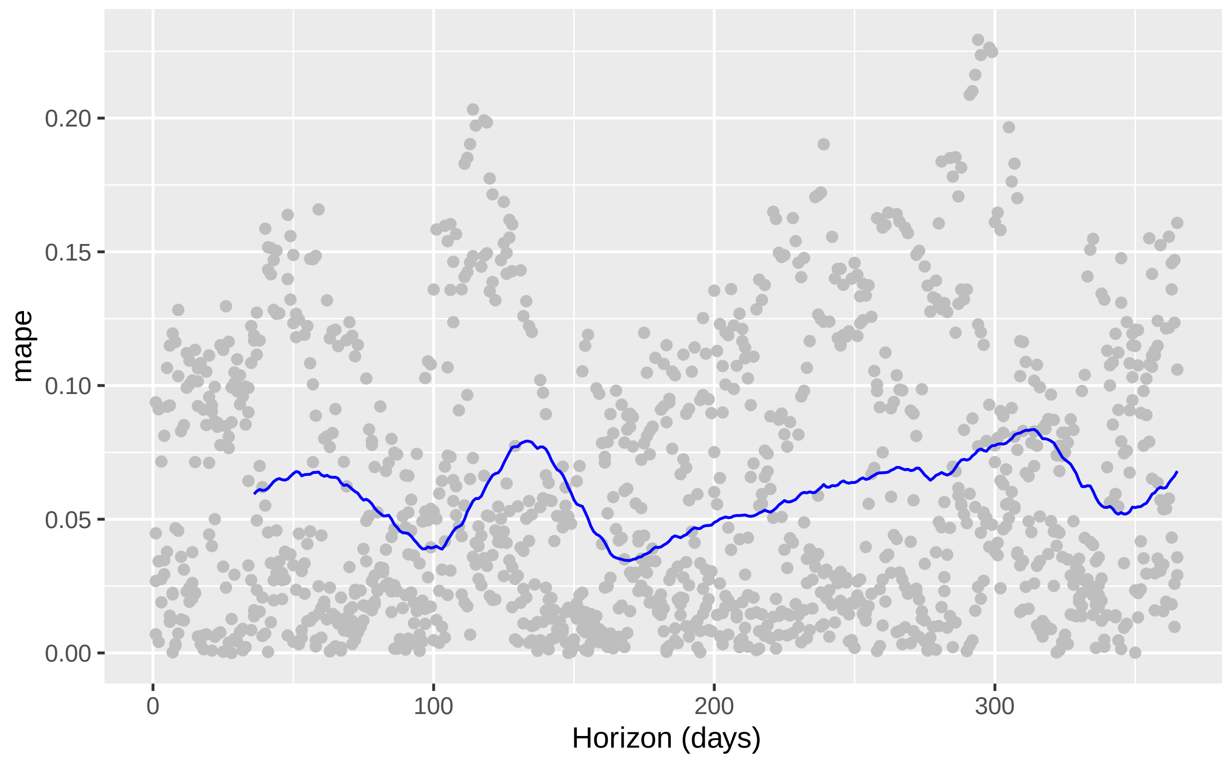
```
##        y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 161.61 2017-04-13 176.7549   169.9416   183.2389 2017-04-12
## 2 162.11 2017-04-17 179.3887   173.1073   186.0171 2017-04-12
## 3 161.25 2017-04-18 179.7860   173.4058   186.4556 2017-04-12
## 4 161.26 2017-04-19 180.5249   173.7842   187.3669 2017-04-12
## 5 162.04 2017-04-20 180.8859   174.4772   187.2028 2017-04-12
```

```
## 6 160.41 2017-04-21 180.9877   174.0186   187.4597 2017-04-12
```

```r
amg_perf <- performance_metrics(amg_cv)
head(amg_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 153.3108 12.38187 10.21699 0.05939368 0.4760000
## 2 37 days 157.3851 12.54532 10.39079 0.06050901 0.4693333
## 3 38 days 160.0541 12.65125 10.49600 0.06115114 0.4613333
## 4 39 days 159.1089 12.61384 10.42230 0.06077358 0.4613333
## 5 40 days 160.6066 12.67306 10.41622 0.06089290 0.4586667
## 6 41 days 164.8291 12.83858 10.51382 0.06158776 0.4600000
```

```r
plot_cross_validation_metric(amg_cv, metric = 'mape')
```




```r
# Comcast Evaluation
com_cv <- cross_validation(m3, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

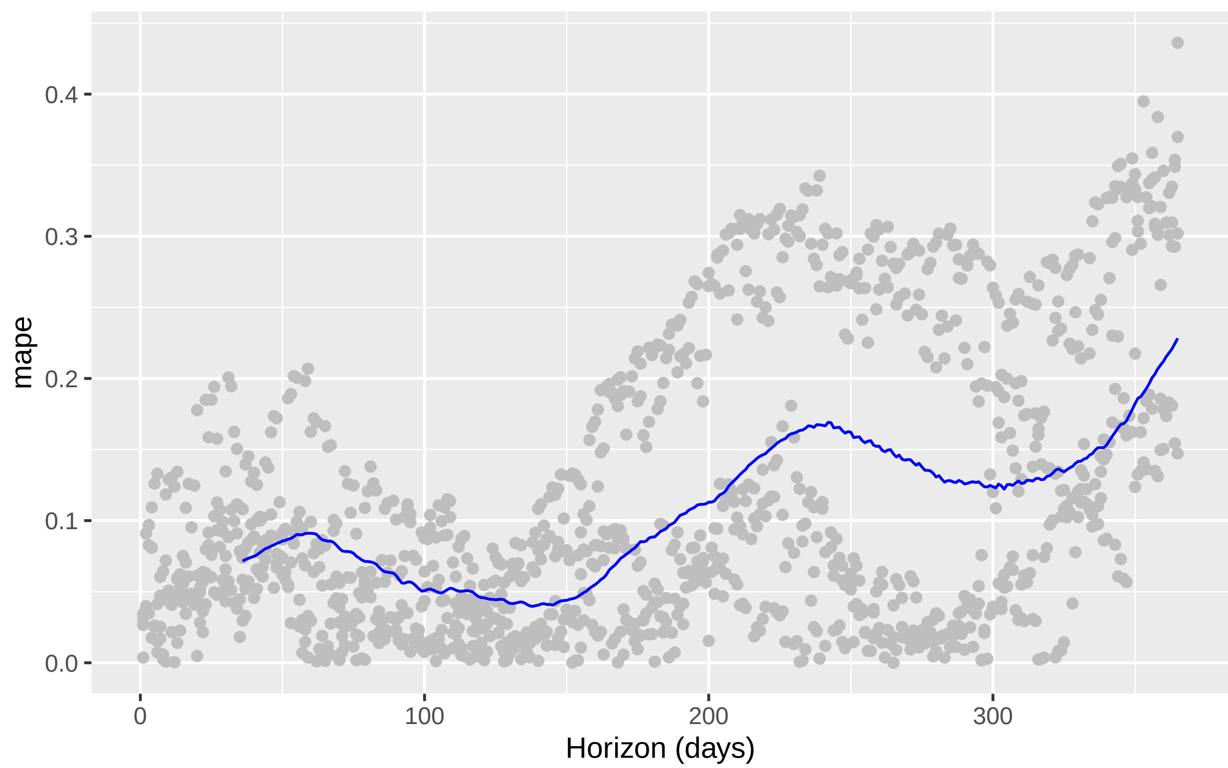
```r
head(com_cv)
```

```
##       y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 37.14 2017-04-13 38.12789   37.46081   38.91476 2017-04-12
## 2 37.20 2017-04-17 38.17455   37.46489   38.91116 2017-04-12
## 3 37.59 2017-04-18 38.15367   37.39992   38.94068 2017-04-12
## 4 37.53 2017-04-19 38.16711   37.48949   38.90728 2017-04-12
## 5 38.00 2017-04-20 38.22098   37.50204   38.96151 2017-04-12
## 6 38.16 2017-04-21 38.19214   37.42973   38.91748 2017-04-12
```

```r
com_perf <- performance_metrics(com_cv)
head(com_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 8.854283 2.975615 2.589959 0.07152249 0.1280000
## 2 37 days 9.062441 3.010389 2.632239 0.07268745 0.1200000
## 3 38 days 9.225972 3.037429 2.658689 0.07339677 0.1200000
## 4 39 days 9.374374 3.061760 2.687592 0.07420527 0.1146667
## 5 40 days 9.478174 3.078664 2.711980 0.07484813 0.1066667
## 6 41 days 9.639781 3.104800 2.756706 0.07593819 0.0920000
```

```r
plot_cross_validation_metric(com_cv, metric = 'mape')
```




```r
# Gilead Evaluation
gil_cv <- cross_validation(m4, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

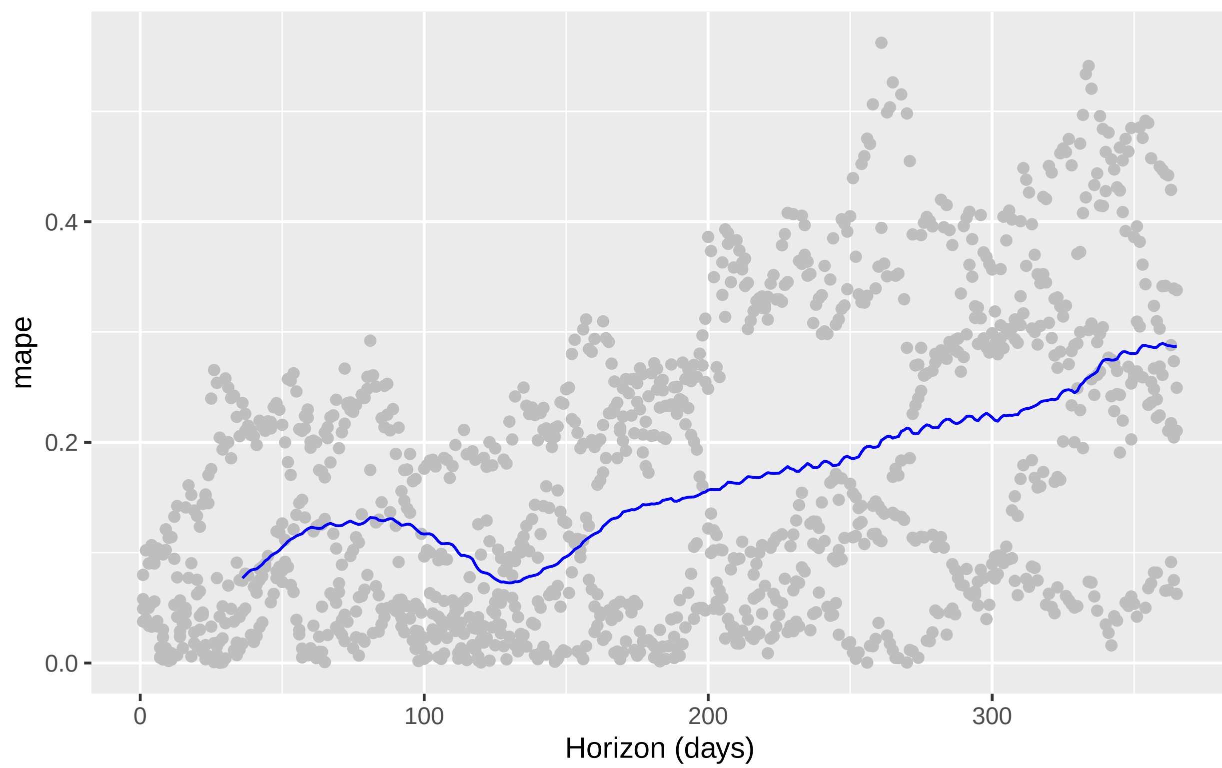
```r
head(gil_cv)
```

```
##       y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 66.51 2017-04-13 71.81972   67.60573   76.23379 2017-04-12
## 2 66.70 2017-04-17 72.71560   68.00446   77.07724 2017-04-12
## 3 66.06 2017-04-18 72.92179   68.46257   77.28572 2017-04-12
## 4 66.28 2017-04-19 73.05839   68.55009   77.41297 2017-04-12
## 5 66.50 2017-04-20 73.04441   68.74708   77.42582 2017-04-12
## 6 65.93 2017-04-21 72.67805   68.52131   77.17845 2017-04-12
```

```r
gil_perf <- performance_metrics(gil_cv)
head(gil_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 53.91191 7.342473 5.428956 0.07693670 0.5920000
## 2 37 days 57.20250 7.563233 5.598017 0.07954984 0.5813333
## 3 38 days 60.43637 7.774083 5.771662 0.08217291 0.5653333
## 4 39 days 63.25519 7.953313 5.907075 0.08422168 0.5573333
## 5 40 days 64.17451 8.010900 5.932070 0.08476614 0.5546667
## 6 41 days 64.97589 8.060762 5.960503 0.08533709 0.5560000
```

```r
plot_cross_validation_metric(gil_cv, metric = 'mape')
```




```r
# Microsoft Evaluation
mic_cv <- cross_validation(m5, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

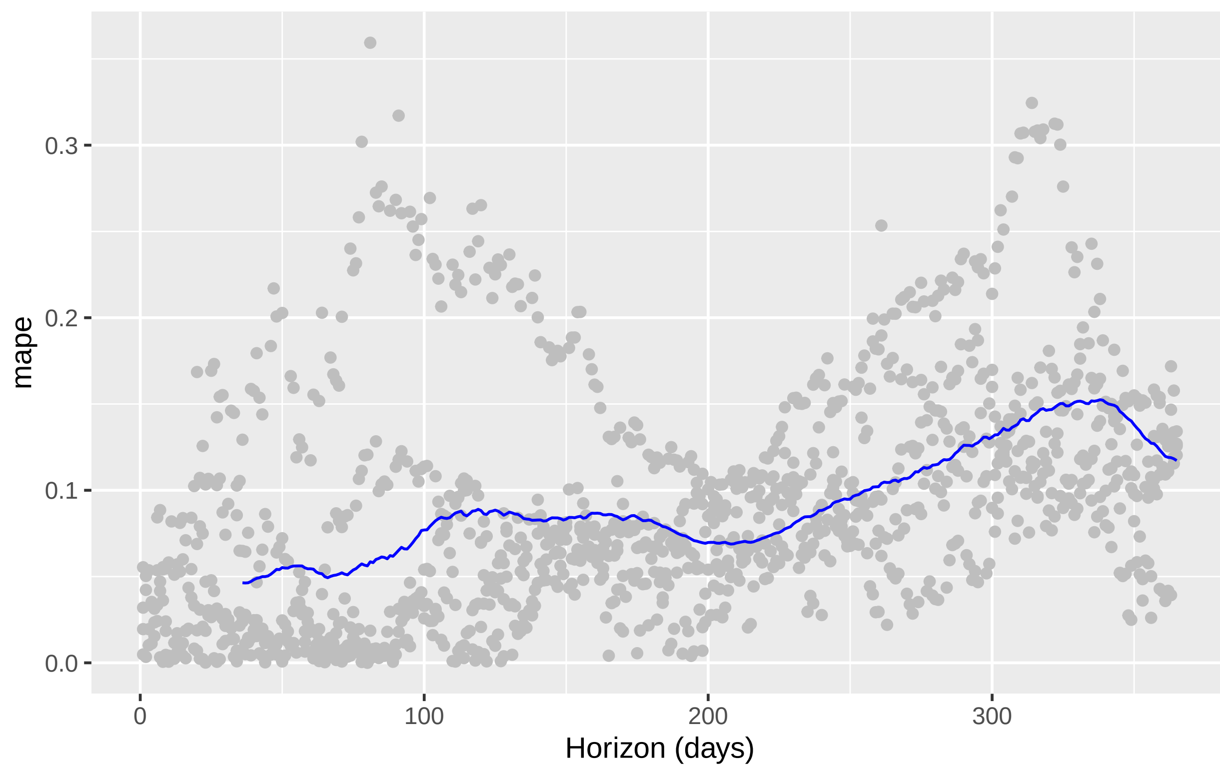
```r
head(mic_cv)
```

```
##       y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 64.95 2017-04-13 67.02616   65.26458   68.68112 2017-04-12
## 2 65.48 2017-04-17 67.53296   65.77186   69.31523 2017-04-12
## 3 65.39 2017-04-18 67.57971   65.83370   69.30114 2017-04-12
## 4 65.04 2017-04-19 67.75139   66.15494   69.52869 2017-04-12
## 5 65.50 2017-04-20 67.86833   66.23606   69.70391 2017-04-12
## 6 66.40 2017-04-21 67.99190   66.22739   69.60377 2017-04-12
```

```r
mic_perf <- performance_metrics(mic_cv)
head(mic_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 49.17180 7.012261 4.964589 0.04636474 0.3640000
## 2 37 days 49.24616 7.017561 4.956593 0.04627249 0.3786667
## 3 38 days 49.62615 7.044583 4.977020 0.04638558 0.3813333
## 4 39 days 51.68970 7.189555 5.061034 0.04715903 0.3866667
## 5 40 days 53.88260 7.340477 5.171786 0.04818351 0.3866667
## 6 41 days 56.21789 7.497859 5.260845 0.04895345 0.3920000
```

```r
plot_cross_validation_metric(mic_cv, metric = 'mape')
```




```r
# Netflix Evaluation
net_cv <- cross_validation(m6, initial = 730, period = 180, horizon = 365, units = 'days')
```

```
## Making 5 forecasts with cutoffs between 2017-04-12 and 2019-04-02
```

```r
head(net_cv)
```

```
##        y         ds     yhat yhat_lower yhat_upper     cutoff
## 1 142.92 2017-04-13 153.0848   145.6823   160.7698 2017-04-12
## 2 147.25 2017-04-17 154.4662   146.2827   162.3005 2017-04-12
## 3 143.36 2017-04-18 154.8627   147.7040   162.6146 2017-04-12
## 4 139.76 2017-04-19 155.4890   147.3016   163.2870 2017-04-12
## 5 141.18 2017-04-20 155.7976   148.1458   163.6770 2017-04-12
## 6 142.87 2017-04-21 155.8497   148.5247   163.3863 2017-04-12
```

```r
net_perf <- performance_metrics(net_cv)
head(net_perf)
```

```
##   horizon      mse     rmse      mae       mape  coverage
## 1 36 days 1903.121 43.62478 27.54966 0.08995221 0.4760000
## 2 37 days 1891.745 43.49420 27.29025 0.08930804 0.4826667
## 3 38 days 1889.356 43.46673 27.26292 0.08931354 0.4773333
## 4 39 days 2002.231 44.74630 27.99623 0.09207850 0.4746667
## 5 40 days 2114.319 45.98172 28.69623 0.09463269 0.4773333
## 6 41 days 2218.122 47.09694 29.15128 0.09638135 0.4900000
```

```r
plot_cross_validation_metric(net_cv, metric = 'mape')
```