Pharmaceutical Industry: Stock Analysis

Forecasting Methods

2021/2022

Group members:
Bruna Matos (52175)
Patrícia Antão (51987)
Pedro Jorge (56801)
Sebastião Cappelle (56284)

Contents

List of Figures

1. Introduction

Given the emergence of COVID-19 and all its impacts in different sectors in the world, we found that the analysis of the pharmaceutical industry would be interesting, allowing us to possibly understand these consequences specifically in this sector.

As for the main goals of the project, we aim to comprehend the variations in pharmaceutical stocks throughout time in general, as well as, particularly during the pandemic time, from 2020 until now. Not only that, but also to gain insights from the tendencies of the financial market.

For that, we decided to study 6 stocks corresponding to: Pfizer Inc. (PFE), Johnson & Johnson (JNJ), AbbVie Inc (ABBV), Moderna Inc (MRNA), BioNTech SE (BNTX) and Novavax Inc (NVAX). For this analysis to be possible we retrieved data from Yahoo Finance, where we found the previously mentioned stocks among others, which are present in following link: https://finance.yahoo.com/u/trea/watchlists/the-fight-against-covid19.

This data will allow us to see the 'open', 'high', 'low' , 'close' and 'adjusted' prices, as well as the amount of 'volume' traded. The datasets obtained have different starting points, as 'PFE', 'JNJ' and 'NVAX' begin in January of 2012, 'ABBV' in January 2013, 'MRNA' in December 2018 and 'BNTX' in October 2019. It is also important to mention that in this time series, we have intra-annual frequency as we are coming across daily data as well as over 11,000 rows.

This way, we will start by joining the data from all these chosen stocks, as well as transform it into a tsibble for the possibility of analysis. In chapter 2, we will have some graphics which include time plots, seasonal plots and seasonal subseries plots with the purpose of seeing the variation and possible trends or seasonality of both close and adjusted stock prices. In chapter 3, we'll have a decomposition using different methods for the possible need of adjustments.

After that, we'll create training and test sets in order to compare the forecasting accuracy of the different forecasting methods used and also try some ETS methods. The last chapter will focus on choosing an ARIMA model more appropriate for our time series, using the Box-Jenkins methodology. In the end, we hope to draw some conclusion related to the changes in the pharmaceutical industry stock market.

2. Time Series Graphs

2.1. Dataset

```
stocks <- tq_get(c('PFE', 'JNJ', 'ABBV', 'MRNA', 'BNTX', 'NVAX'), get = 'stock_price')
stocks
```

```
## # A tibble: 11,759 x 8
##    symbol date        open  high   low close   volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 PFE    2012-01-03  20.7  20.9  20.7  20.8 53124340     14.3
##  2 PFE    2012-01-04  20.8  20.8  20.6  20.7 31912485     14.1
##  3 PFE    2012-01-05  20.6  20.6  20.3  20.5 52764821     14.0
##  4 PFE    2012-01-06  20.6  20.7  20.5  20.5 31613571     14.0
##  5 PFE    2012-01-09  20.5  20.7  20.4  20.7 41786041     14.2
##  6 PFE    2012-01-10  20.8  20.9  20.7  20.8 29227420     14.2
##  7 PFE    2012-01-11  20.8  20.8  20.6  20.8 30317467     14.2
##  8 PFE    2012-01-12  20.8  20.9  20.8  20.9 28835648     14.3
##  9 PFE    2012-01-13  20.8  20.8  20.6  20.7 30636829     14.2
## 10 PFE    2012-01-17  20.9  21.0  20.8  20.8 37463376     14.2
## # ... with 11,749 more rows
```

```
pharma_stocks <- stocks %>%
  mutate(date = as_date(date)) %>%
  as_tsibble(index = date,
             key = symbol)
pharma_stocks
```

```
## # A tsibble: 11,759 x 8 [1D]
## # Key:       symbol [6]
##    symbol date        open  high   low close   volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 ABBV   2013-01-02  34.9  35.4  34.1  35.1 13767900     23.8
##  2 ABBV   2013-01-03  35    35    34.2  34.8 16739300     23.6
##  3 ABBV   2013-01-04  34.6  34.9  34.2  34.4 21372100     23.3
##  4 ABBV   2013-01-07  34.2  35.5  34.2  34.5 17897100     23.3
##  5 ABBV   2013-01-08  34.3  34.6  33.4  33.7 17863300     22.8
##  6 ABBV   2013-01-09  33.6  34.0  33.6  33.9 18800400     23.0
##  7 ABBV   2013-01-10  33.7  34    33.3  34   15658100     23.0
##  8 ABBV   2013-01-11  33.6  33.9  33.3  33.8 11191500     23.2
##  9 ABBV   2013-01-14  34.0  34.2  33.8  34.1 11584900     23.4
## 10 ABBV   2013-01-15  33.7  34.7  33.7  34.6 13040200     23.7
## # ... with 11,749 more rows
```

2.2. Time Series Plots

```
pharma_stocks %>%
  filter(year(date) >= '2019') %>%
  autoplot(close) +
  xlab("Day") + ylab("Close Price ($)")
```

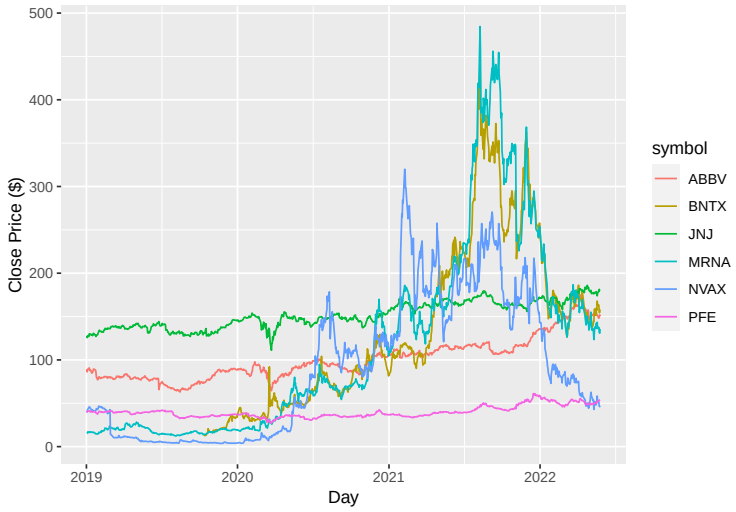

Figure 1: Pharmaceutical Industry Closing Stock Price

The closing stock price allows the investor to see the change of feeling through time and as our goal is to understand that, specifically in the pharmaceutical industry, we decided to start by an analysis of all the companies in question.

Starting with the year of 2019, we don't detect any major differences until the beginning of 2021, as Moderna Inc (MRNA), was the only to show a growth in 2020. Looking at the year of 2021 as well as the beginning of 2022, we can see a huge increase, with the previously mentioned company, as it almost reaches 500$ in August.

As we can see in the table above, this represents a growth of almost 450$.

```
pharma_stocks %>%
  group_by(symbol) %>%
  filter(close == max(close))
```

```
## # A tsibble: 6 x 8 [1D]
## # Key:       symbol [6]
## # Groups:    symbol [6]
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 ABBV   2022-04-08 173   176.  172.  175.   7204200    173.
## 2 BNTX   2021-08-09 408.  460.  404.  447.  12022600    447.
## 3 JNJ    2022-04-25 182.  187.  182.  186.  11176500    185.
## 4 MRNA   2021-08-09 411.  494.  410.  484.  42269900    484.
## 5 NVAX   2021-02-08 295.  330.  293.  320.   6974800    320.
## 6 PFE    2021-12-16  58.4  61.4  57.8  61.2 75183600     60.3
```

```
pharma_stocks %>%
  filter(year(date) >= '2019') %>%
  autoplot(close) +
  facet_grid(symbol ~ ., scales = "free_y") +
  xlab("Day") + ylab("Close Price ($)")
```

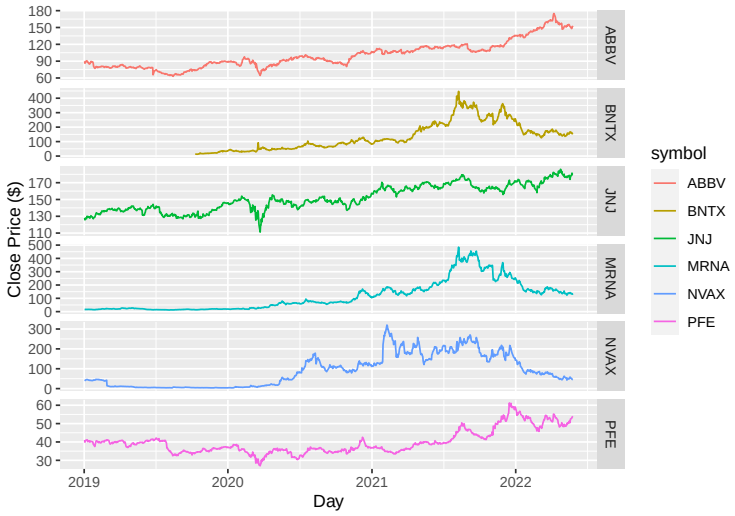

Figure 2: Closing Stock Price per Company

After visualising the pharmaceutical industry more generally, and viewing the disparities among the selected companies, we decided to study in more detail and this way look at the closing price of each of the enterprises' stocks from 2019 to 2022, in order to view more specifically this critical period of pre and during the pandemic.

As we can see in the previous graph, there's an increase in the first half of 2020, explained by the beginning of the impact of the emergence of COVID-19, that will be analysed in the next sections. This growth continues until 2022, where a decrease starts, except for Johnson & Johnson (JNJ) and AbbVie Inc (ABBV) that appear to have a continuous increase with their highest values in this year.

Starting with AbbVie Inc (ABBV), there is a slight growth starting in 2020, but not very sharp. The value is, at its peak, $175 in 2022, as previously shown in the values of maximum closing price of the stocks.

When analyzing BioNTech SE (BNTX), we can immediately see a much steeper growth than in the first one, showing its peak 447$, followed by a decrease in the more recent months.

With Johnson & Johnson (JNJ), we see exactly the same situation as with AbbVie Inc (ABBV), with a continuous increase, reaching its maximum at 186$.

Looking at Moderna Inc (MRNA), we again see a very sharp rise, peaking at $484, but on the other hand a big decrease starting in the second half of 2021.

With Novavax Inc (NVAX), we observe as before a growth throughout 2021, that begins in the middle of 2020, although with some oscillations, and reaching its highest value of 320$.

With Pfizer Inc. (PFE), even though we can see that there was, in 2021, a rise in the closing price of the stock, this rise is not very significant when compared to all the others, showing only a rise up to $61.

```
pharma_stocks %>%
  group_by(symbol) %>%
  filter(volume == max(volume))
```

```
## # A tsibble: 6 x 8 [1D]
## # Key:       symbol [6]
## # Groups:    symbol [6]
##   symbol date        open  high   low close    volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 ABBV   2014-10-15  52.3  55.7  52.1  54.6 122740200     39.7
## 2 BNTX   2020-07-23  97.5 101.   87    88.5  15899700     88.5
## 3 JNJ    2012-06-13  64.2  64.7  63.2  64.4  98440200     48.9
## 4 MRNA   2020-12-01 178.  178.  130.  141.  125130400    141.
## 5 NVAX   2020-05-12  39.0  44.9  36.1  39.8  74649600     39.8
## 6 PFE    2016-04-05  29.6  30.1  29.0  29.8 299829377     23.6
```

```
pharma_stocks %>%
  filter(year(date) >= '2019') %>%
  autoplot(volume) +
  ylab("Volume") +
  xlab("Day")
```

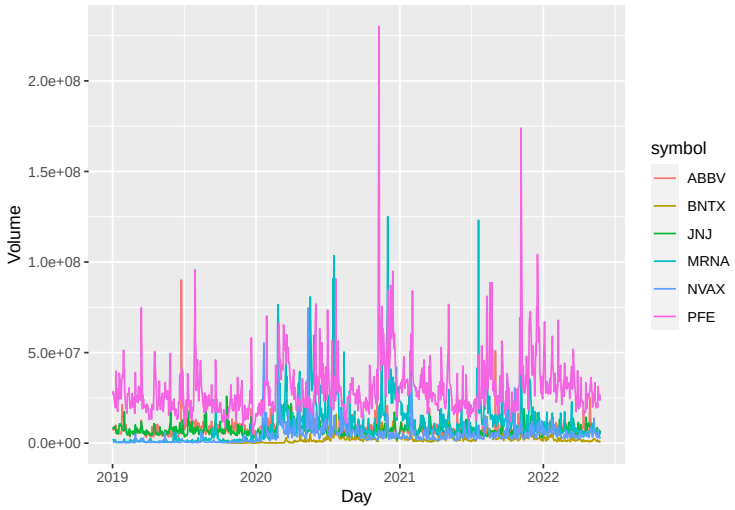

Figure 3: Stock Volume

As far as volume is concerned, half of the companies have reached their highest amounts in 2020. Even though all companies reached their maximum closing stock price in 2021 and 2022, three of them reached their pick of volume a few years earlier. As we can see, for example, Johnson & Johnson peaked in June 2012.

Taking a deeper look at the volume available in each of the companies, we can conclude that Pfizer Inc. (PFE) reaches its peak - much higher than the others - even before 2021, reaching another peak at the end of this same year.

All the others, except Moderna Inc (MRNA), with the same behavior of Pfizer Inc. (PFE), always vary between the same values throughout the three years studied, not providing a very interesting analysis.

2.3. Time Series Patterns

```
monthly_stocks_adj <- pharma_stocks %>%
  index_by(month = yearmonth(date)) %>%
  summarise(average_adjusted = mean(adjusted))

monthly_stocks_adj %>%
  gg_season(average_adjusted, labels = 'right') +
  xlab("Month") +
  ylab('Adjusted Price ($)')
```

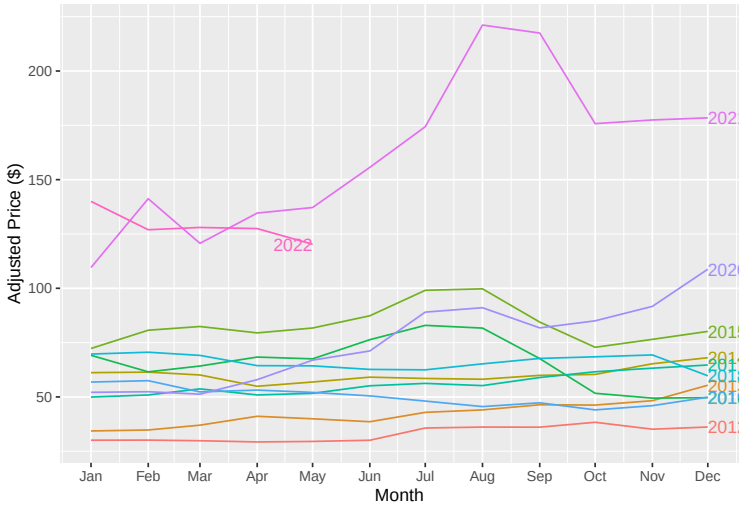

Figure 4: Seasonal Plot - Adjusted Closing Stock Price

As to analyze the pharmaceutical stock market as a whole, based on the stocks we have obtained, we decided to transform the daily data into monthly data, to have a clearer and more overall idea of the changes. In this case, our analysis is going to be based on the adjusted closing stock price, in order to have into consideration other factors such as dividends, stock splits, and new stock offerings.

From 2012 to 2019, the changes aren't very significant, except for 2015, that sees a sharp growth in the beginning of the year, as there seems to be an increase of the attention in healthcare companies, although by the end of the year it has gone down a bit.

Focusing on the on-going pandemic, and this way, from 2020 on, we can also see drastic changes, as there's a big shift specially in March 2020 up until September 2021, with a big growth of stock prices getting to over 100$ as seen in the previous graph. This can be explained by the opportunity seen by investors during these tough times in this sector given its crucial role in the containment of the situation, including the quick development of effective vaccines.

From September 2021 until now, we see a radical alteration with a big decrease of stock price, although still a lot higher that pre-pandemic. As COVID-19 is more under control and investors don't see the pharmaceutical and biotech companies as much as an opportunity anymore, leaning more towards other types of companies that will benefit from the aim to go back to normality and the reopening of the economy in the upcoming months.

```
monthly_stocks_adj %>%
  gg_subseries(average_adjusted) +
```

```
  xlab("Month") +
  ylab('Adjusted Price ($)')
```

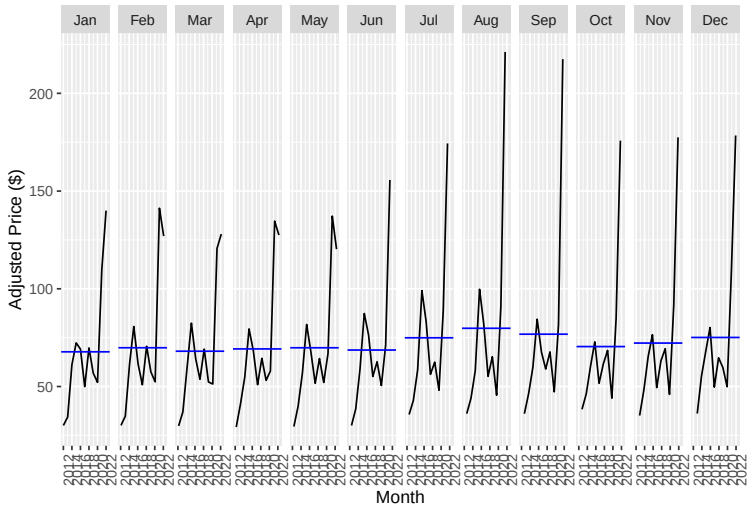

Figure 5: Seasonal Subseries Plot - Adjusted Closing Stock Price

As there are significant variations throughout the years regarding the adjusted closing stock price, the means for each month are very similar.

In the plot above, it is clearly possible to see the changes in seasonality over time, given that it has had dramatic changes with a special increase from the year of 2020. It's also important to examine that a decrease is starting in 2022, as seen in the months that have already passed of February, March and April, which confirms the analysis done in the previous seasonal plot.

After visualizing these seasonal and seasonal subseries plots, we can conclude that there is a trend starting in the beginning of 2020, as previously mentioned, since there seems to be a high increase of stock prices up until the following year. On the other hand, there also seems to be a trend in the opposite direction starting in the end of 2021, for the reasons also touched beforehand. As for cyclic or seasonal behavior those don't seem to be present in this time series.

2.4. ACF and White Noise

```
monthly_stocks_adj %>%
  ACF(average_adjusted) %>%
  autoplot() +
  ylab("ACF")
```

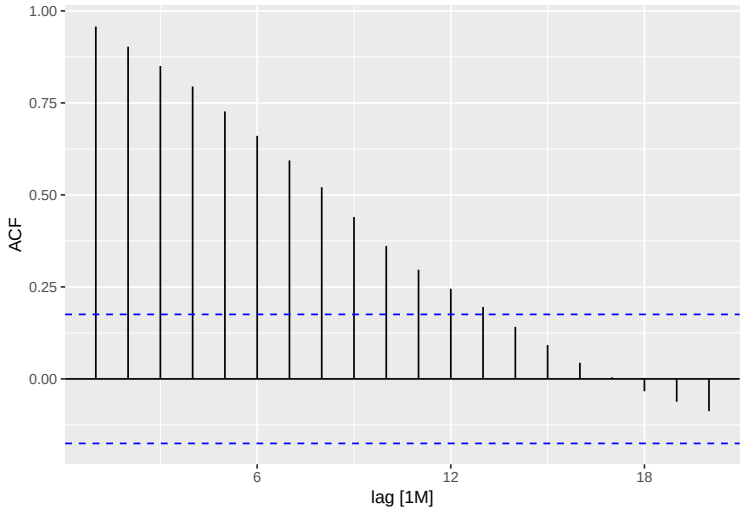

Figure 6: ACF - Monthly Adjusted Closing Stock Prices

In the study of the ACF plot above, it is possible to recognize that there is, for the most part, a high and positive autocorrelation, although with a prominent decreasing as the lags increase, which is explained by the trend previously found. We can also visually conclude that there is no seasonal pattern.

As most of the lags are considerably higher than zero, showing a significant autocorrelation, we can conclude that there is no white noise present in the studied time series.

3. Time Series Decomposition

3.1. Adjustments

When dealing with historical data there are several adjustments and transformations that can be done. Either to remove sources of variation or to increase the pattern consistency.

Given all the pandemic context and the unstable global economy, we found critical to have in mind the inflation when analyzing data such as stock related.

Inflation adjustments

```
monthly_inflation <- tibble(month=c("2019 Jan","2019 Feb","2019 Mar","2019 Apr","2019 May",
                                    "2019 Jun","2019 Jul","2019 Aug","2019 Sep","2019 Oct",
                                    "2019 Nov","2019 Dec","2020 Jan","2020 Feb","2020 Mar",
                                    "2020 Apr","2020 May","2020 Jun","2020 Jul","2020 Aug",
                                    "2020 Sep","2020 Oct", "2020 Nov","2020 Dec","2021 Jan",
                                    "2021 Feb","2021 Mar","2021 Apr","2021 May","2021 Jun",
                                    "2021 Jul","2021 Aug","2021 Sep","2021 Oct", "2021 Nov",
                                    "2021 Dec","2021 Jan","2021 Feb","2021 Mar"),
                            rate = c(0.0155,0.0152,0.0186,0.02,0.0179,0.0165,0.0181,0.0175,
                                     0.0171,0.0176,0.0205,0.0229,0.0249,0.0233,0.0154,0.0033,
                                     0.0012,0.0065,0.0099,0.0131,0.0137,0.0118,0.0117,0.0136,
                                     0.0140,0.0168,0.0262,0.0416,0.0499,0.0539,0.0537,0.0525,
                                     0.0539,0.0622,0.0681,0.0704,0.0748,0.0787,0.0854))

monthly_inflation
```

```
## # A tibble: 39 x 2
##    month      rate
##    <chr>     <dbl>
##  1 2019 Jan 0.0155
##  2 2019 Feb 0.0152
##  3 2019 Mar 0.0186
##  4 2019 Apr 0.02
##  5 2019 May 0.0179
##  6 2019 Jun 0.0165
##  7 2019 Jul 0.0181
##  8 2019 Aug 0.0175
##  9 2019 Sep 0.0171
## 10 2019 Oct 0.0176
## # ... with 29 more rows
```

```
m_inflation <- monthly_inflation %>%
  mutate(month = yearmonth(month)) %>%
  as_tsibble(index = month,
             key = rate)

pharma_inflation <- merge(monthly_stocks_adj, m_inflation, by= 'month')

pharma_inflation %>%
  mutate(adjusted_with_inflation = average_adjusted / 1 + rate)
```

```
##       month average_adjusted   rate adjusted_with_inflation
## 1  2019 Jan         56.84176 0.0155                56.85726
## 2  2019 Feb         57.47217 0.0152                57.48737
## 3  2019 Mar         52.31414 0.0186                52.33274
## 4  2019 Apr         53.14889 0.0200                53.16889
## 5  2019 May         52.08178 0.0179                52.09968
## 6  2019 Jun         50.48322 0.0165                50.49972
## 7  2019 Jul         48.08661 0.0181                48.10471
## 8  2019 Aug         45.55595 0.0175                45.57345
## 9  2019 Sep         47.29869 0.0171                47.31579
## 10 2019 Oct         44.03647 0.0176                44.05407
## 11 2019 Nov         45.97240 0.0205                45.99290
## 12 2019 Dec         49.87266 0.0229                49.89556
## 13 2020 Jan         52.12159 0.0249                52.14649
## 14 2020 Feb         52.39541 0.0233                52.41871
## 15 2020 Mar         51.27689 0.0154                51.29229
## 16 2020 Apr         57.94971 0.0033                57.95301
## 17 2020 May         66.92970 0.0012                66.93090
## 18 2020 Jun         71.16934 0.0065                71.17584
## 19 2020 Jul         89.03099 0.0099                89.04089
## 20 2020 Aug         91.05554 0.0131                91.06864
## 21 2020 Sep         81.77363 0.0137                81.78733
## 22 2020 Oct         85.02441 0.0118                85.03621
## 23 2020 Nov         91.68108 0.0117                91.69278
## 24 2020 Dec        108.78397 0.0136               108.79757
## 25 2021 Jan        109.56534 0.0140               109.57934
## 26 2021 Jan        109.56534 0.0748               109.64014
## 27 2021 Feb        141.22289 0.0168               141.23969
## 28 2021 Feb        141.22289 0.0787               141.30159
## 29 2021 Mar        120.69204 0.0262               120.71824
## 30 2021 Mar        120.69204 0.0854               120.77744
## 31 2021 Apr        134.61424 0.0416               134.65584
## 32 2021 May        137.18127 0.0499               137.23117
## 33 2021 Jun        155.66667 0.0539               155.72057
## 34 2021 Jul        174.36006 0.0537               174.41376
## 35 2021 Aug        221.15303 0.0525               221.20553
## 36 2021 Sep        217.48481 0.0539               217.53871
## 37 2021 Oct        175.78027 0.0622               175.84247
## 38 2021 Nov        177.47840 0.0681               177.54650
## 39 2021 Dec        178.43921 0.0704               178.50961
```

3.2. Transformations

After some discussing, we reached the conclusion that after the adjustments made in the previous section, no further transformations - be it Box Cox or any of the mathematical transformations - are necessary. This is due to the fact that, besides it being very difficult to analyze variance (stock price data is not seasonal data in this case), stock price data is very volatile on its own and we wouldn't be able to truly report the changes made by said transformations.

3.3. Decomposition

STL Decomposition

```
dcmp1 <- monthly_stocks_adj %>%
  model(stl = STL(average_adjusted))

components(dcmp1)
```

```
## # A dable: 125 x 7 [1M]
## # Key:     .model [1]
## # :        average_adjusted = trend + season_year + remainder
##    .model    month average_adjusted trend season_year remainder season_adjust
##    <chr>     <mth>            <dbl> <dbl>       <dbl>     <dbl>          <dbl>
##  1 stl    2012 Jan             30.1  29.1      -0.668     1.69           30.8
##  2 stl    2012 Feb             30.1  29.7      -0.510     0.911          30.6
##  3 stl    2012 Mar             29.8  30.4      -0.439    -0.123          30.3
##  4 stl    2012 Apr             29.3  31.1      -1.53     -0.261          30.8
##  5 stl    2012 May             29.5  31.7      -1.45     -0.779          31.0
##  6 stl    2012 Jun             30.1  32.4       0.624    -2.96           29.4
##  7 stl    2012 Jul             35.7  33.1       4.60     -1.99           31.1
##  8 stl    2012 Aug             36.1  33.7       4.29     -1.90           31.8
##  9 stl    2012 Sep             36.1  34.4       0.799     0.905          35.3
## 10 stl    2012 Oct             38.3  35.0      -2.92      6.23           41.3
## # ... with 115 more rows
```

```
components(dcmp1) %>%
  autoplot() + xlab("Month")
```

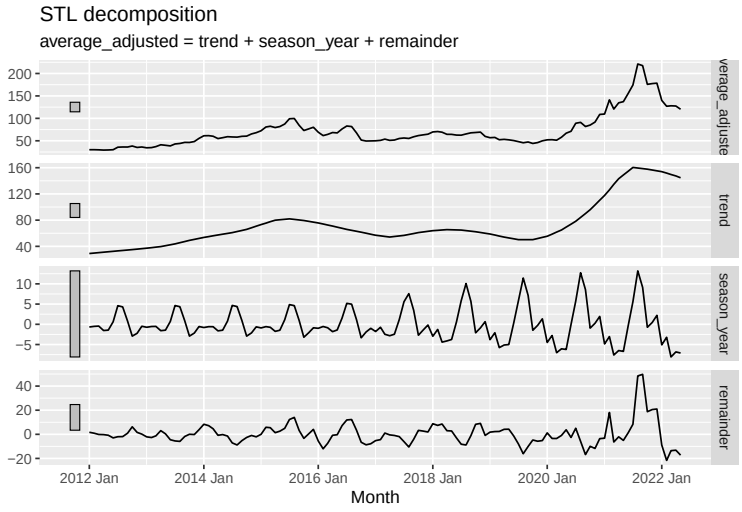

Figure 7: STL Decomposition

The STL decomposition gives us 3 different components of the average adjusted plot which together form the average adjusted close price of the stocks from 2012 to 2022.

By looking at the different components we can take several conclusions. Starting with the fact that in the last years there is an upwards trend in the close price of the pharmaceutical stocks, which started in

2019 and can be explained by the pandemic and the investments in research and vaccination done by these companies in order to fight the global virus.

Regarding the seasonalilty we can't observe representative weight in it.

The remainder presents a higher variance in the last 2 years in comparison with the previous ones much related with the uncertainty regarding the pandemic.

```
components(dcmp1) %>%
  as_tsibble() %>%
  autoplot(average_adjusted, colour = "gray") +
  geom_line(aes(y = season_adjust), colour = "#0072B2") +
  labs(y = "Ajusted Price ($)", x = 'Month [1M]')
```

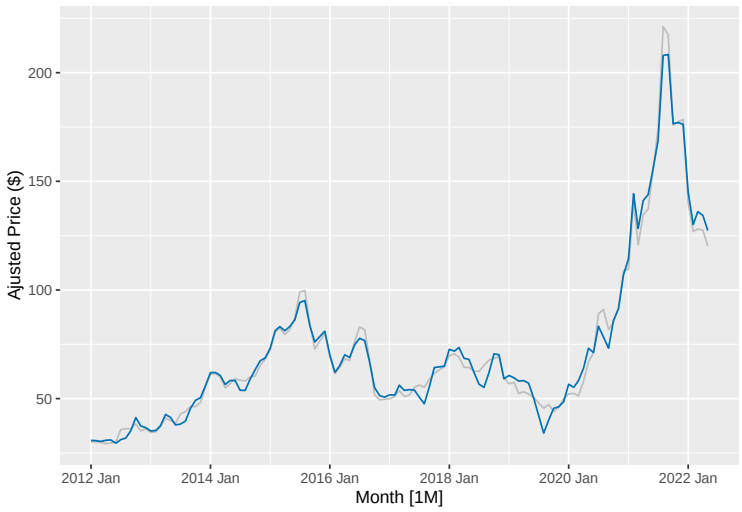

Figure 8: Stock Prices in the Pharmaceutical Industrys

As referred before, there's no evident of the presence of seasonality in the plots.

Classical Decomposition

```
monthly_stocks_adj %>%
  model(
    classical_decomposition(average_adjusted, type = "additive")
  ) %>%
  components() %>%
  autoplot() +
  xlab("Month") +
  labs(title = 'Classical Decomposition')
```

```
## Warning: Removed 6 row(s) containing missing values (geom_path).
```

Differently from the STL, and regarding the classical decomposition, shown below, we can observe an even lower variance in the seasonal component and a higher remainder in the last year which reinforces our conclusion that there's no seasonality, as well as the presence of risk and volatility due to the pandemic situation.

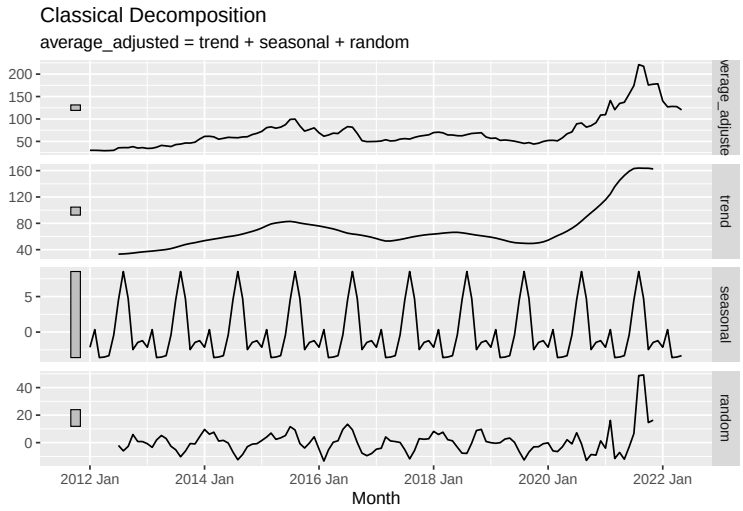

Figure 9: Classical Decomposition of Monthly Stock Prices

X-11 Method

```
x11_dcmp <- monthly_stocks_adj %>%
  model(x11 = X_13ARIMA_SEATS(average_adjusted ~ x11())) %>%
  components()
autoplot(x11_dcmp) +
  xlab("Month") +
  labs(title = 'X-11 Decomposition')
```

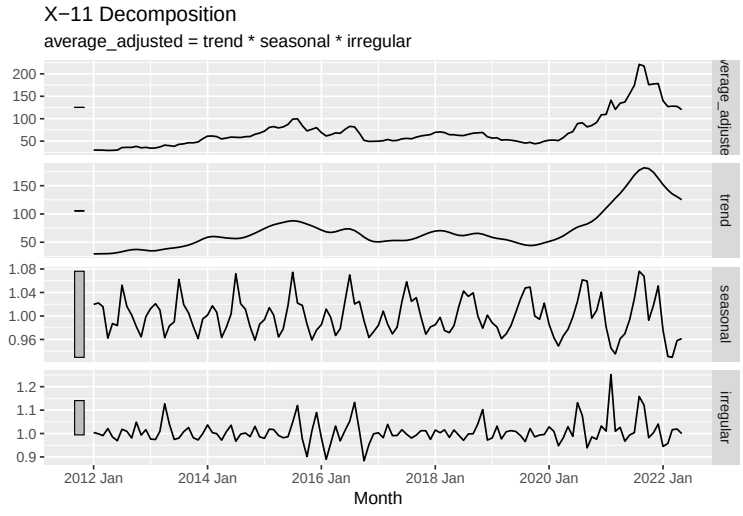

Figure 10: X-11 Decomposition of Monthly Stock Prices

SEATS Method

```
seats_dcmp <- monthly_stocks_adj %>%
  model(seats = X_13ARIMA_SEATS(average_adjusted ~ seats())) %>%
  components()

autoplot(seats_dcmp) +
  xlab("Month") +
  labs(title = "SEATS Decomposition")
```

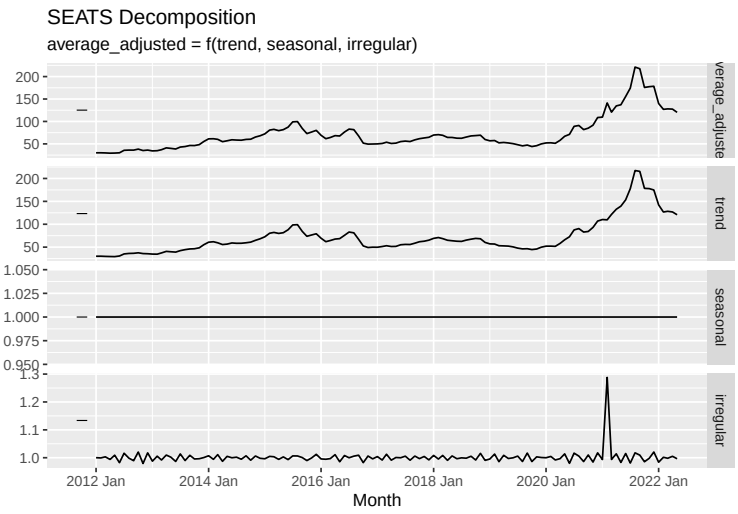

Figure 11: SEATS Decomposition of Monthly Stock Prices

From the several decompositions we can clearly conclude that there was an upward trend in the last year regarding the price of pharmaceutical stocks that we can easily relate with the pandemic situation across the world.

Since there's risk and uncertainty involved, it makes sense that all the decompositions show irregularity and a high remainder value in that period of time.

4. The Forecaster's Toolbox

4.1 Which Simple Forecasting Method is More Appropriate?

Regarding the forecasting methods that we have learned, we believe that the naïve method will be the one with more accuracy, since it's the one that is more used for financial analysis. However, we will plot three methods: mean method, naïve method and drift method

We will not plot the seasonal naive method since we concluded there's no seasonality in our sample.

```
models_fit <-  monthly_stocks_adj %>%
  filter(!is.na(average_adjusted)) %>%
  model(
    Naive = NAIVE(average_adjusted),
    Drift = RW(average_adjusted ~ drift()),
    Mean = MEAN(average_adjusted))
```

```
models_fc <- models_fit %>%
  forecast(h = "3 years")

models_fc %>%
  autoplot(monthly_stocks_adj, level = NULL) +
  xlab("Year") + ylab("Adjusted Price ($)") +
  guides(colour = guide_legend(title = "Forecast"))
```

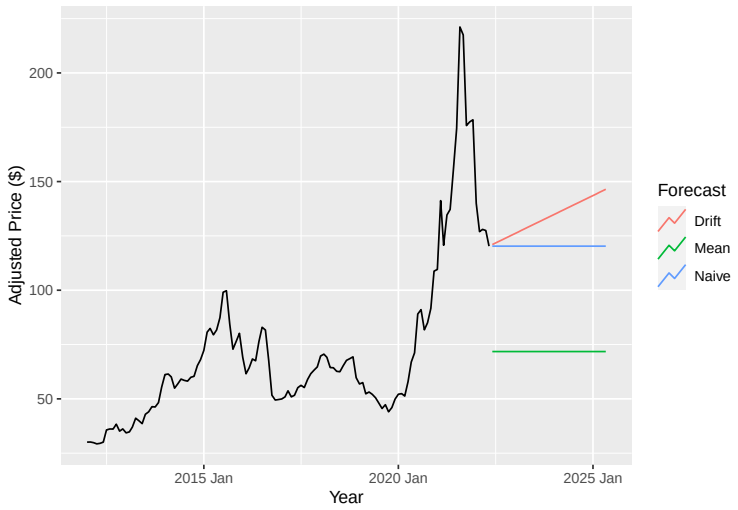

Figure 12: 3 Year Forecast for Monthly Stock Price

We can conclude that the drift and naïve method might be the most accurate one, since it follows an upward trend comparatively with the mean method, that in our opinion underpredicts this trend, as the weight of the pandemic trend is less evaluated.

4.2 Creating Training and Test Sets

```
#Train the model
train <- monthly_stocks_adj %>%
  filter_index("2012 Jan" ~ "2021 Feb")

# Fit the models
stocks_fit <- train %>%
  model(
    Mean = MEAN(average_adjusted),
    Naive = NAIVE(average_adjusted),
    Drift = RW(average_adjusted ~ drift()))
stocks_fit
```

```
## # A mable: 1 x 3
##      Mean   Naive          Drift
##   <model> <model>        <model>
## 1  <MEAN> <NAIVE> <RW w/ drift>
```

```
stocks_fc <- stocks_fit %>% forecast(h = 12)

stocks_fc %>%
  autoplot(train, level = NULL) +
  autolayer(
    filter_index(monthly_stocks_adj, "2022 Jan" ~ .),
    colour = "black") +
  labs(y = "Ajusted Price ($)", x = 'Month') +
  guides(colour = guide_legend(title = "Forecast"))
```

```
## Plot variable not specified, automatically selected `.vars = average_adjusted`
```

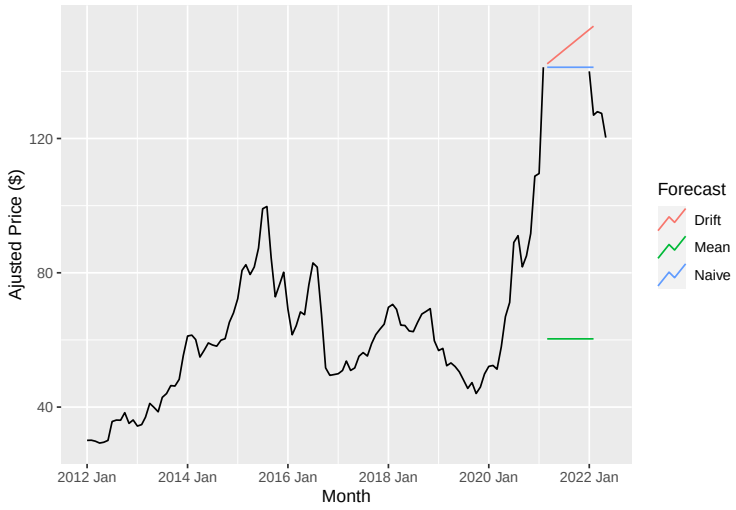

Figure 13: Train Set Forecasts for Monthly Stock Prices

4.3 Traditional and Cross Validation for Accuracy Analysis

```
#Traditional accuracy
accuracy(stocks_fc, monthly_stocks_adj)
```

```
## # A tibble: 3 x 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  Test   15.5  35.0  28.2  6.17  16.0  1.54  1.56 0.594
## 2 Mean   Test  103.  108.  103.  61.7   61.7  5.64  4.79 0.607
## 3 Naive  Test   22.1  38.8  29.9 10.3   16.5  1.64  1.73 0.607
```

```
#Cross validation accuracy
stocks_tr <- monthly_stocks_adj %>%
  stretch_tsibble(.init = 3, .step = 1) %>%
  relocate(month, average_adjusted)
stocks_tr
```

```
## # A tsibble: 7,872 x 3 [1M]
## # Key:       .id [123]
##       month average_adjusted   .id
##       <mth>            <dbl> <int>
##  1 2012 Jan             30.1     1
##  2 2012 Feb             30.1     1
##  3 2012 Mar             29.8     1
##  4 2012 Jan             30.1     2
##  5 2012 Feb             30.1     2
##  6 2012 Mar             29.8     2
##  7 2012 Apr             29.3     2
##  8 2012 Jan             30.1     3
##  9 2012 Feb             30.1     3
## 10 2012 Mar             29.8     3
## # ... with 7,862 more rows
```

```
# TSCV accuracy
 stocks_tr %>%
  model(Drift = RW(average_adjusted ~ drift())) %>%
  forecast(h = 12) %>%
  accuracy(monthly_stocks_adj)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 12 observations are missing between 2022 Jun and 2023 May
```

```
## # A tibble: 1 x 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  Test   2.28  26.4  16.9 -1.29  20.4 0.690 0.757 0.852
```

```
# Training set accuracy
 monthly_stocks_adj %>%
  model(RW(average_adjusted ~ drift())) %>%
  accuracy()
```

```
## # A tibble: 1 x 10
##   .model              .type          ME  RMSE   MAE    MPE  MAPE  MASE RMSSE  ACF1
##   <chr>               <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 RW(average_adjust~ Train~ -1.20e-15  9.34  5.04 -0.477  6.05 0.206 0.268 0.161
```

```
# TSCV2 accuracy
stocks_tr %>%
  model(Naive = NAIVE(average_adjusted)) %>%
  forecast(h = 12) %>%
  accuracy(monthly_stocks_adj)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 12 observations are missing between 2022 Jun and 2023 May
```

```
## # A tibble: 1 x 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Naive  Test   6.49  26.4  16.9  4.84  19.9 0.690 0.757 0.846
```

```
# Training set accuracy2
monthly_stocks_adj %>%
  model(Naive = NAIVE(average_adjusted)) %>%
  accuracy()
```

```
## # A tibble: 1 x 10
##   .model .type       ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Naive  Training 0.727  9.37  5.16 0.742  6.17 0.211 0.269 0.161
```

We can conclude from these values that the drift method is still better and that the cross validation gives us better results regarding the error values according to the original forecast.

4.4 White Noise of the Residuals

Now we will present and analyze the white noise of the residuals of the two models that performed better with our data: the drift model and the naïve model

```
WN_naive <- monthly_stocks_adj %>% model(NAIVE(average_adjusted)) %>% augment()
autoplot(WN_naive, .innov) + labs(y = "Adjusted Price ($)", x = 'Month')
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

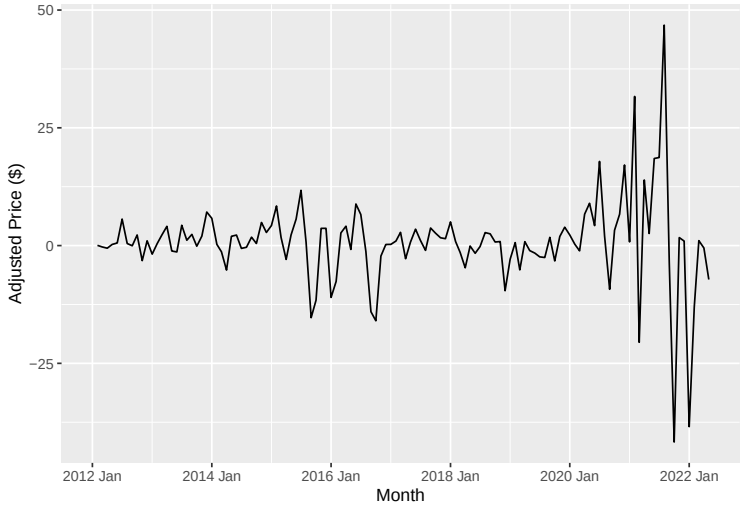

Figure 14: Residuals from the Naïve Method

```
WN_naive %>% ACF(.innov) %>% autoplot() + labs(y = 'ACF')
```

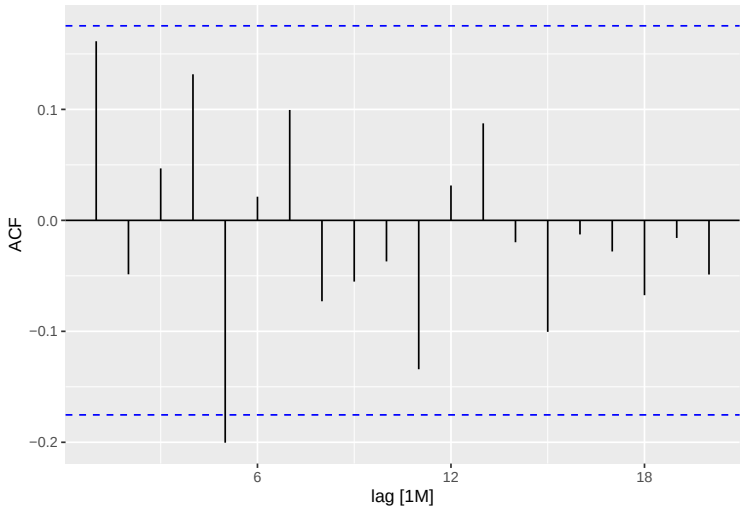

Figure 15: ACF: Residuals from the Naïve Method

```
WN_drift <- monthly_stocks_adj %>%
  model(RW(average_adjusted ~ drift())) %>%
  augment()
autoplot(WN_drift, .innov) +
  labs(y = "Adjusted Price ($)", x = 'Month')
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

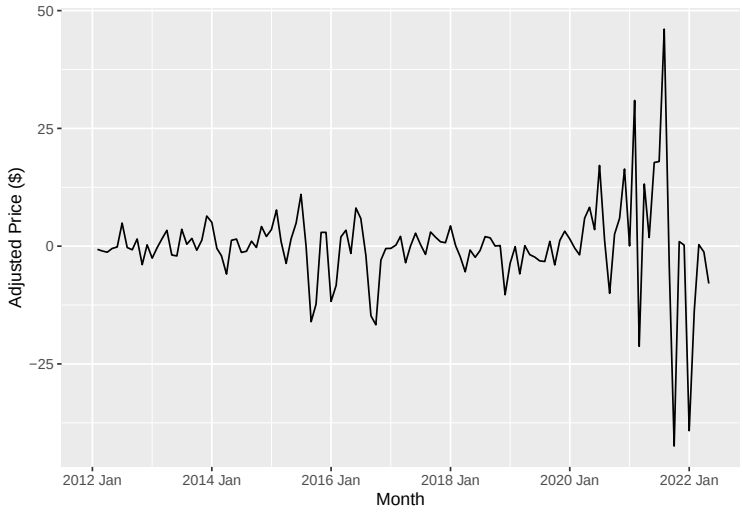

Figure 16: Residuals from the Drift Method

```
WN_drift %>%
  ACF(.innov) %>%
  autoplot() + labs(y = "ACF")
```

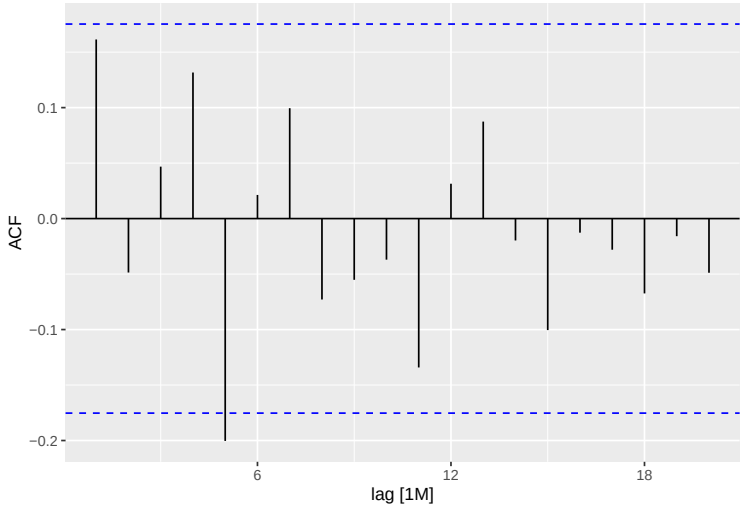

Figure 17: ACF: Residuals from the Drift Method

By these plots, which are really similar for each one of the models, it is possible to understand that the lack of correlation of the resiuals suggest that the models are accurate in the predictions of the data.

5. Exponential Smoothing

Exponential smoothing methods are a great choice as they produce very reliable forecasts for time series. When choosing an ETS model, we need to have in mind the evolution of the data throughout time, more specifically, have a look at the presence or not of seasonality and trend, as well as the variation of the error term.

5.1. ETS Model Selection by Main Features

If we try to look at the main features of our data through the graphs we have shown in other chapters, we may or may not be able to select the best and most appropriate model. This way, we decided to select 3 of the stock companies present in our dataset.

By looking at the visualizations developed in previous chapters, we've seen an increasing trend throughout the years in all the stock companies in analysis. On the other hand, some are now, in 2022, decreasing, which can be explained, as mentioned before, by the fact we've passed the most critical period of the pandemic, where the investors felt the importance of these enterprises, but now are moving towards other types of investments.

```
pharma_stocks %>%
  filter(symbol == 'MRNA' | symbol == 'JNJ' | symbol == 'PFE') %>%
  autoplot(adjusted) +
  xlab("Day") + ylab("Adjusted Price ($)")
```

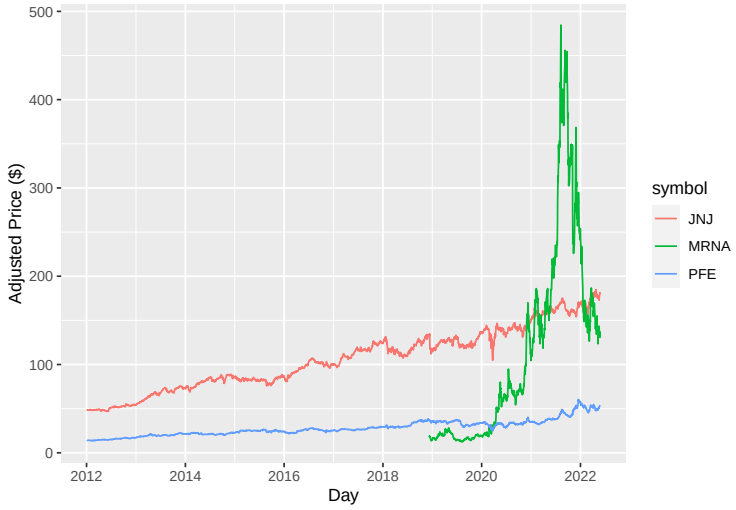

Figure 18: Pharmaceutical Industry Adjusted Stock Price Sample

This way, in order to do an analysis of just some of the variables that we considered most representative, we decided to choose the stocks from: Johnson & Johnson, which has continued its growth, even in 2022; Moderna that has had a big increase but now that has changed; and Pfizer given that it hasn't had very visible changes.

```
pharma_stocks2 <- pharma_stocks %>%
  filter(symbol == 'MRNA' | symbol == 'PFE' | symbol == 'JNJ') %>%
  index_by(month = yearmonth(date)) %>%
  summarise(average_adjusted = mean(adjusted))
```

```
pharma_stocks2 %>% autoplot(average_adjusted) +
  xlab("Month") + ylab("Adjusted Price ($)")
```

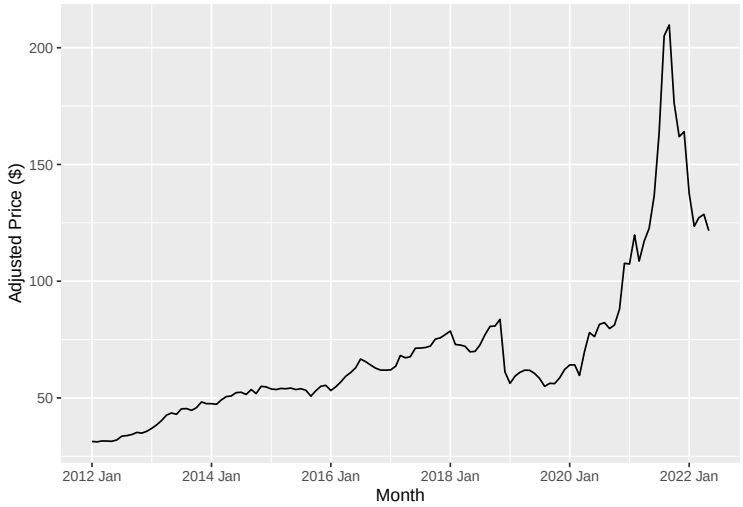

Figure 19: Pharmaceutical Industry Monthly Adjusted Stock Price Sample

As mentioned previously, we chose these stocks, since we feel it is a good representative sample of the dataset in question, as it is possible to visualize in the graph above.

Having in mind the previously found trend in this data set and the not present seasonality, we knew we would have a model where Season would be 'None'. As for Trend, we believe that we should choose 'Additive' since we roughly see the same size ups and downs throughout the time series except for the last 2 years. To finish with the Error, our doubts were between the 'Additive' or 'Multiplicative' models and given the just mentioned changes, as there's an increasing trend from 2015 to 2021 and a sudden steep decrease after that, we believe the error is 'Multiplicative'.

This way, and given the justifications above, we believe the most appropriate model for the dataset in analysis is: ETS(average_adjusted ~ error("M") + trend("A") + season("N")

5.2. ETS and SES Model Selection

ETS

After the model selection just based on the visualization of the main features of our data, we decided to analyze this topic using an ETS() function, in order to see if we get the same or a similar model, but this time using all the series available in the present dataset.

```
fit_all <- monthly_stocks_adj %>%
    model(ets = ETS(average_adjusted))
report(fit_all)
```

```
## Series: average_adjusted
## Model: ETS(M,Ad,N)
##   Smoothing parameters:
##     alpha = 0.9998999
##     beta  = 0.205204
##     phi   = 0.8000026
##
##   Initial states:
##      l[0]      b[0]
##  27.13262 0.9386299
##
##   sigma^2:  0.0074
##
##      AIC     AICc      BIC
## 1037.701 1038.413 1054.671
```

```
components(fit_all) %>% autoplot() + xlab("Month")
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

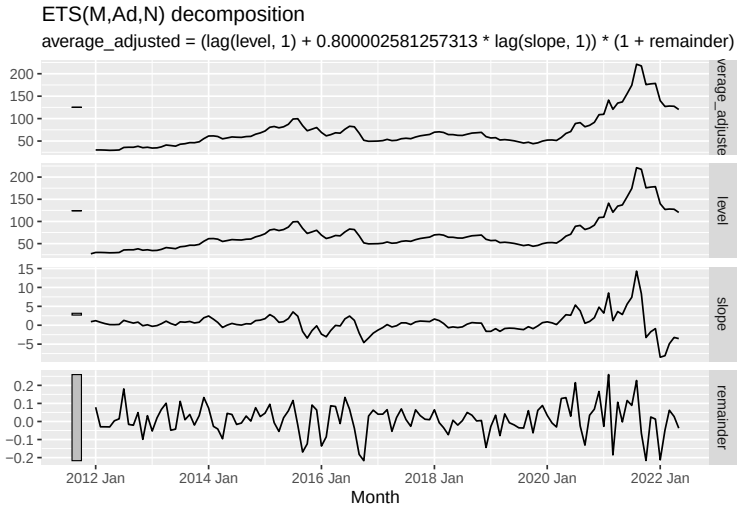

Figure 20: ETS(M,A,N) Decompostion

```
forecast_all <- fit_all %>% forecast(h = '2 years')
forecast_all %>% autoplot(monthly_stocks_adj) + xlab("Month") + ylab("Adjusted Price ($)")
```

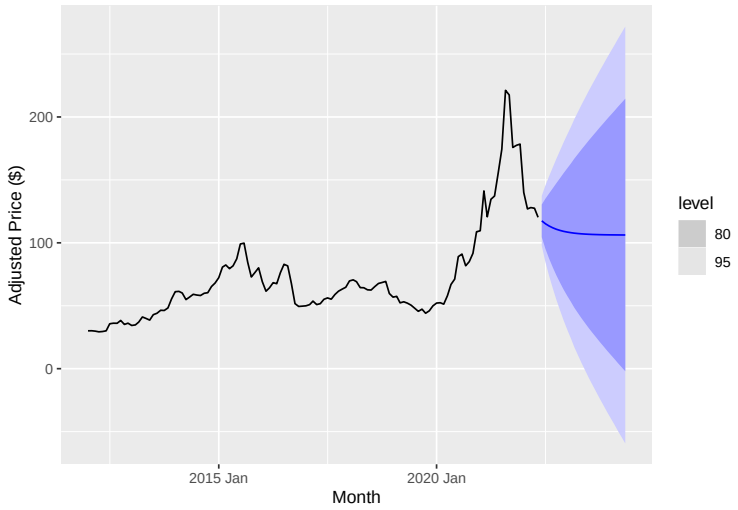

Figure 21: ETS Monthly Adjusted Stock Price 2 Year Forecast

It is possible to verify that the model obtained did coincide with the one from the sample of the data previously done.

When computing the ETS(average_adjusted ~ error("M") + trend("A") + season("N")) model, we obtain some measures, to then make a decision regarding this model.

Starting with the AIC, the Akaike's Information Criterion, with a value of 1037.696, as well as, the Corrected AIC, with a value of 1038.408. Finally, we also obtained a Bayesian Information Criterion with a value of 1054.666.

```
residuals(fit_all) %>% ACF(.resid) %>% autoplot() + ylab('ACF')
```

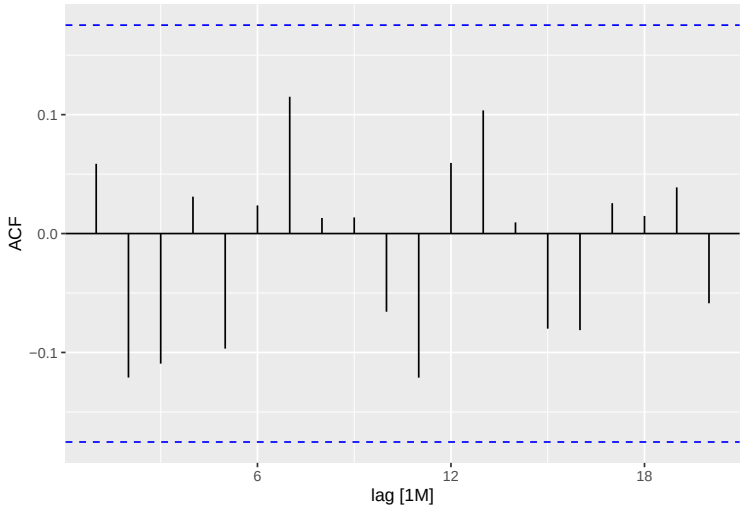

Figure 22: ACF: ETS Model Residuals

Studying the ACF of the residuals of the ETS model obtained, we can visualize through the ACF plot above, that, as expected, it appears to be uncorrelated, and this way, we are in the presence of white noise.

SES

```
fit_all_ses <- monthly_stocks_adj %>%
    model(ses = ETS(average_adjusted ~ error("A") + trend("N") + season("N")))
report(fit_all_ses)
```

```
## Series: average_adjusted
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##      l[0]
##  30.12749
##
##   sigma^2:  88.5413
##
##      AIC     AICc      BIC
## 1167.957 1168.155 1176.442
```

```
components(fit_all_ses) %>% autoplot() +
    xlab("Month")
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

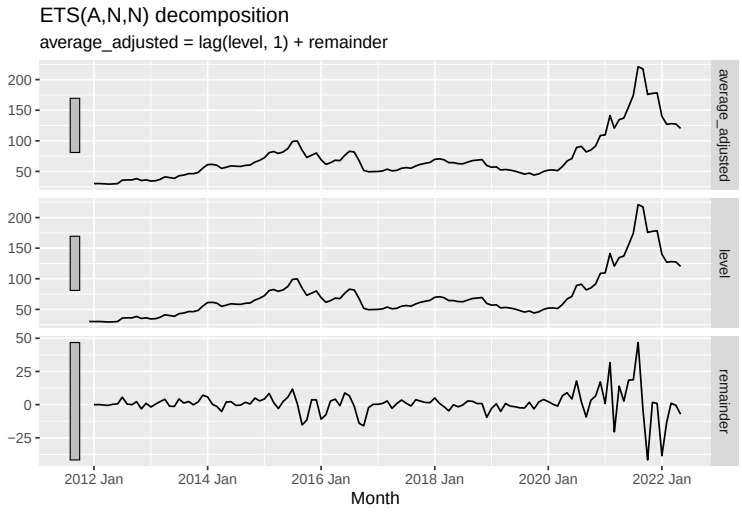

Figure 23: SES Decompostion

```
forecast_all_ses <- fit_all_ses %>%
  forecast(h = '2 years')

forecast_all_ses %>%
```

```
autoplot(monthly_stocks_adj) +
xlab("Month") +
ylab("Adjusted Price ($)")
```

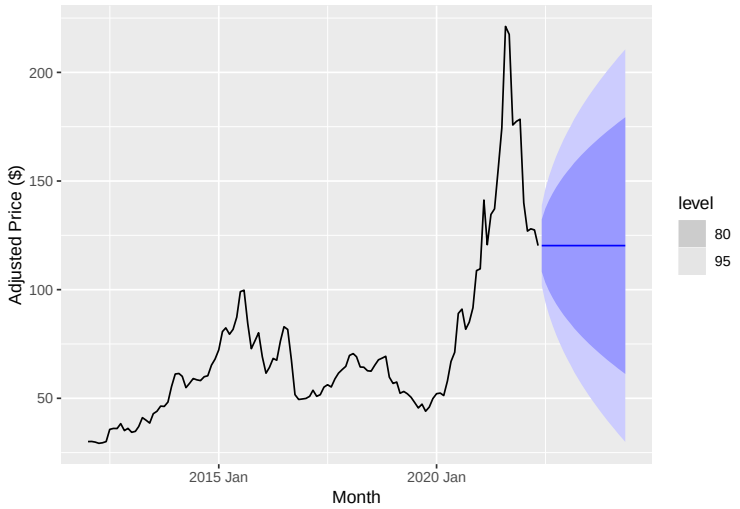

Figure 24: SES Monthly Adjusted Stock Price 2 Year Forecast

Next, we thought it would be interesting to also compare the model we got previously to a simple exponential smoothing one, as to understand which one is more appropriate for out dataset. This way, we select the function SES(average_adjusted ~ error("A") + trend("N") + season("N")).

As before, we obtain the same measures AIC, the Akaike's Information Criterion, with a value of 1167.946 and the Corrected AIC with a value of 1168.145. As well as the Bayesian Information Criterion with a value of 1176.431.

In order to choose the best model, we have to analyze which of these models had the lowest values in the mentioned measures. As it is possible to compare, the ETS one got the smallest amount in all three, and this way, without a doubt, it appears to be the most appropriate one to perform forecasts in the dataset in analysis.

6. ARIMA

Similarly to the section about Exponential Smoothing, we will use the stocks from Johnson & Johnson, Pfizer and Moderna.

6.1. Box-Jenkins Methodology

Firstly, we should define if we are making any Box-Cox transformations to our data in order to reduce variances in it, but as we discussed earlier, such changes will not be necessary.

As for stationarity and differencing, we are using the following definition made by Hyndman and Athanasopoulos (2021): 'A stationary time series is one whose statistical properties do not depend on the time at which the series is observed'. This implies that if the data is affected by a trend or seasonality, it will not be stationary.

It's not so difficult to understand that our stock data is not stationary due to the trend it presents. Given that, we can make it like so, by using a method called differencing.

```
pharma_stocks2 %>% gg_tsdisplay(average_adjusted, plot_type = 'partial') +
  labs(y = 'Adjusted Price ($)', x = 'Month')
```

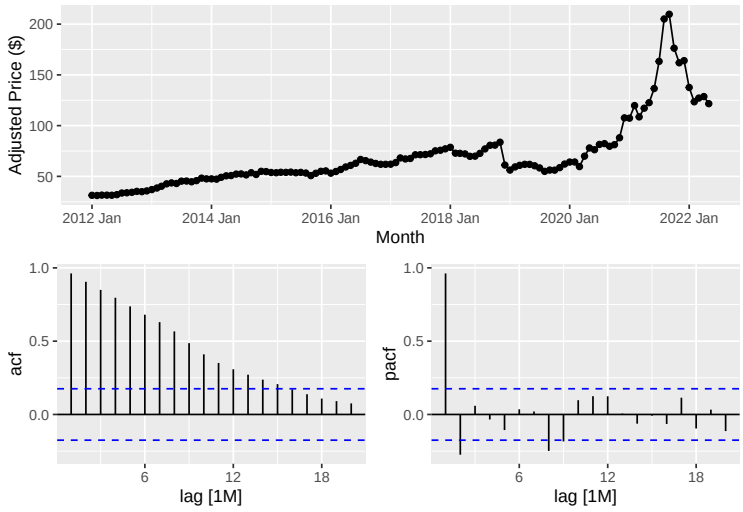

Figure 25: Time Series Display

Let's now apply differencing.

```
pharma_stocks2 %>% mutate(diff = difference(average_adjusted)) %>%
  gg_tsdisplay(diff, plot_type = 'partial') +
  labs(y = 'Diff Adjusted Price ($)', x = 'Month')
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

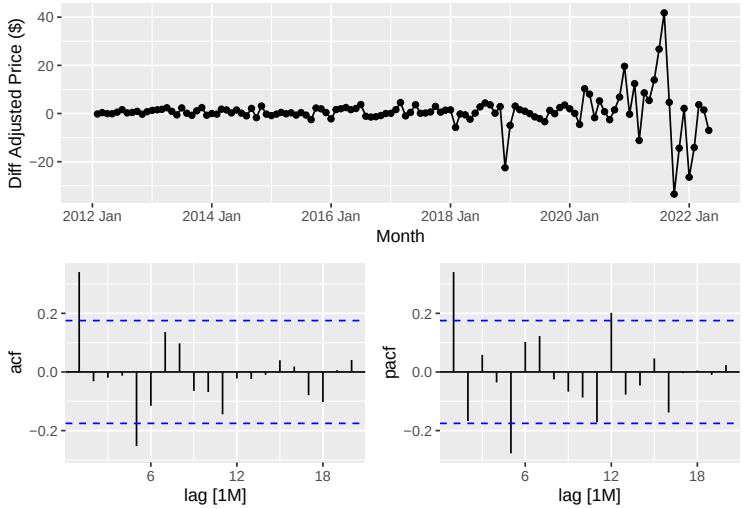

Figure 26: First Differencing Time Series Display

Having turned our time series into a stationary one, we now have do deal with the rest of the ARIMA model - i.e., the autoregressive and moving average part of the model. ARIMA(p,d,q) has 3 components: p, which is the order of the autoregressive part of the model; d, the number of nonseasonal differences needed for stationarity, which in our case was 1; and q, the order of the moving average part.

Below we performed a trial and error approach to the candidate models and we will use, among other measures, the AICc to see which one is the better one.

```
fit_arima <- pharma_stocks2 %>%
    model(ar1 = ARIMA(average_adjusted ~ pdq(1, 1, 0) + PDQ(0,0,0)),
        ar2 = ARIMA(average_adjusted ~ pdq(2, 1, 0) + PDQ(0,0,0)),
        ma1 = ARIMA(average_adjusted ~ pdq(0, 1, 1) + PDQ(0,0,0)),
        ma2 = ARIMA(average_adjusted ~ pdq(0, 1, 2) + PDQ(0,0,0)),
        arima11 = ARIMA(average_adjusted ~ pdq(1, 1, 1) + PDQ(0,0,0)),
        arima12 = ARIMA(average_adjusted ~ pdq(1, 1, 2) + PDQ(0,0,0)),
        arima21 = ARIMA(average_adjusted ~ pdq(2, 1, 1) + PDQ(0,0,0)),
        arima22 = ARIMA(average_adjusted ~ pdq(2, 1, 2) + PDQ(0,0,0)),
        arima32 = ARIMA(average_adjusted ~ pdq(3, 1, 2) + PDQ(0,0,0)),
        arima23 = ARIMA(average_adjusted ~ pdq(2, 1, 3) + PDQ(0,0,0)),
        arima33 = ARIMA(average_adjusted ~ pdq(3, 1, 3) + PDQ(0,0,0)))
```

```
## Warning: It looks like you're trying to fully specify your ARIMA model but have not said if a constar
## You can include a constant using `ARIMA(y~1)` to the formula or exclude it by adding `ARIMA(y~0)`.
```

```
## Warning: It looks like you're trying to fully specify your ARIMA model but have not said if a constar
## You can include a constant using `ARIMA(y~1)` to the formula or exclude it by adding `ARIMA(y~0)`.
```

```
## Warning: 1 error encountered for arima32
## [1] Could not find an appropriate ARIMA model.
## This is likely because automatic selection does not select models with characteristic roots that may
## For more details, refer to https://otexts.com/fpp3/arima-r.html#plotting-the-characteristic-roots

## Warning: 1 error encountered for arima33
## [1] Could not find an appropriate ARIMA model.
## This is likely because automatic selection does not select models with characteristic roots that may
## For more details, refer to https://otexts.com/fpp3/arima-r.html#plotting-the-characteristic-roots
```

```
fit_arima %>% glance() %>% arrange(AIC)
```

```
## # A tibble: 9 x 8
##   .model  sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 arima22   43.3   -408.  829.  830.  846. <cpl [2]> <cpl [2]>
## 2 ma1       47.7   -415.  834.  834.  840. <cpl [0]> <cpl [1]>
## 3 arima12   47.4   -414.  836.  836.  847. <cpl [1]> <cpl [2]>
## 4 ar2       48.0   -415.  836.  836.  845. <cpl [2]> <cpl [0]>
## 5 ma2       48.1   -415.  836.  836.  845. <cpl [0]> <cpl [2]>
## 6 arima11   48.1   -415.  836.  836.  845. <cpl [1]> <cpl [1]>
## 7 ar1       49.0   -417.  838.  838.  843. <cpl [1]> <cpl [0]>
## 8 arima21   48.3   -415.  838.  838.  849. <cpl [2]> <cpl [1]>
## 9 arima23   47.7   -413.  839.  839.  855. <cpl [2]> <cpl [3]>
```

As we can see, ARIMA(2,1,2) outperformed the other candidate models, so through the Box-Jenkins we conclude that that is the better one.

```
boxj_arima <- pharma_stocks2 %>%
    model(ARIMA(average_adjusted ~ pdq(2, 1, 2) + PDQ(0,0,0)))
boxj_arima %>% forecast(h = '2 years') %>% autoplot(pharma_stocks2) +
  labs(y = 'Adjusted Price ($)', x = 'Month')
```

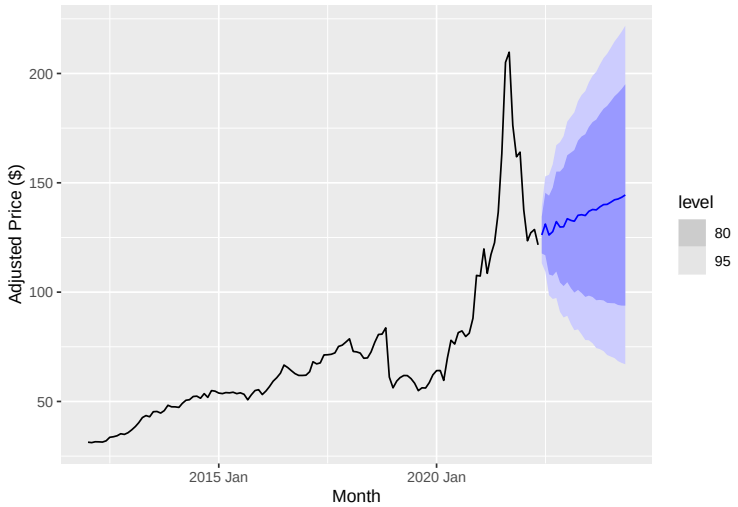

Figure 27: Box Jenkins ARIMA 2 Year Forecast

6.2. Automatic Modelling Procedure with ARIMA()

Now, performing the same analysis but without an automated ARIMA function, in order to see if we get the same or different result in comparison to the Box-Jenkins methodology.

```
auto_arima <- pharma_stocks2 %>%
  model(arima = ARIMA(average_adjusted ~ pdq(d=1)))
report(auto_arima)
```

```
## Series: average_adjusted
## Model: ARIMA(2,1,2)(0,0,2)[12]
##
## Coefficients:
##           ar1      ar2     ma1     ma2    sma1     sma2
##       -0.6637  -0.7564  1.0709  0.9498  0.2208  -0.3704
## s.e.   0.0701   0.0715  0.0355  0.0472  0.1443   0.1352
##
## sigma^2 estimated as 39.21:  log likelihood=-404.24
## AIC=822.49   AICc=823.45   BIC=842.23
```

```
auto_arima %>% forecast(h = '2 years') %>% autoplot(pharma_stocks2) +
  labs(y = 'Adjusted Price ($)', x = 'Month')
```

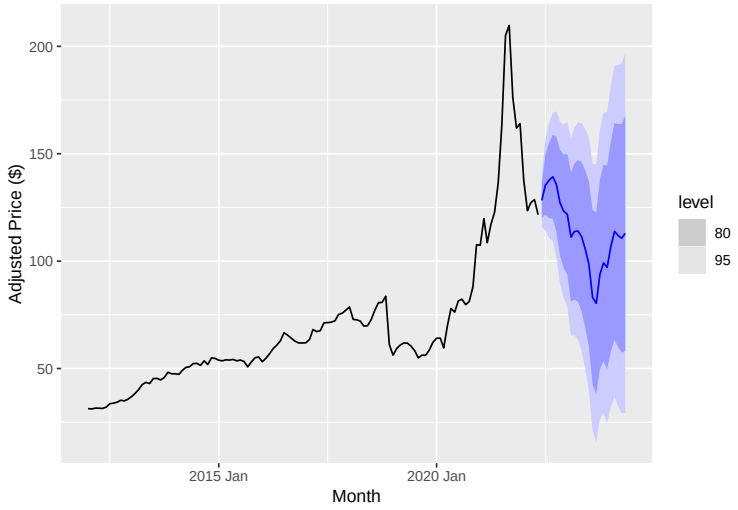

Figure 28: Automated ARIMA 2 Year Forecast

Having now plotted both models (Box Jenkins and the ARIMA()), we can conclude that this forecasting method looks superior at a first glance. But let's do a proper comparison between these 2 models and the others we tested in the previous sections of the report.

6.3. Comparing the Forecasts

For this comparison, we will include both our ARIMA models (Box Jenkins and Automated), the ETS and Drift. We will not include neither Naive nor Mean, as we have already done the comparison between these two and the Drift method and concluded that the latter was superior. The main parameter we are using to compare is RMSE (Root Mean Squared Error)l which we believe to be the most simple to interpret. We also found the addition of MAE (Mean Absolute Error), MPE (Mean Percentage Error) and MAPE (Mean Absolute Percentage Error) to be relevant to the discussion, maybe to be some kind of 'tiebreaker' or even to strengthen a 'winner'.

```
model_comparison <- pharma_stocks2 %>%
  slice(-n()) %>%
  model(
    ETS = ETS(average_adjusted),
    auto_ARIMA = ARIMA(average_adjusted),
    boxj_ARIMA = ARIMA(average_adjusted ~ pdq(2, 1, 2) + PDQ(0,0,0)),
    drift = RW(average_adjusted ~ drift())) %>%
  forecast(h = '2 years')

model_comparison %>%
  accuracy(pharma_stocks2) %>%
  select(.model, RMSE:MAPE)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 23 observations are missing between 2022 Jun and 2024 Apr
```

```
## # A tibble: 4 x 5
##   .model      RMSE   MAE    MPE  MAPE
##   <chr>      <dbl> <dbl>  <dbl> <dbl>
## 1 auto_ARIMA  3.13  3.13  2.57  2.57
## 2 boxj_ARIMA  1.17  1.17  0.958 0.958
## 3 drift       7.80  7.80 -6.41  6.41
## 4 ETS         5.61  5.61 -4.61  4.61
```

The results turned out to be quite interesting, as the ARIMA model using Box Jenkins methodology returned the lowest RMSE (~ 1.17), followed by the automated ARIMA (~ 3.13), then by the ETS (~ 5.6) and finally by the Drift forecast (~ 7.8), which doesn't come as a surprise. To reinforce the result, the Box Jenkins ARIMA also had a lower MAE (~ 1.17), MPE (~ 0.96) and MAPE (~ 0.96)), becoming quite clear it was the strongest candidate forecasting model given our data.

Now, having analyzed all of the important statistical parameters of the models, a good visualization can be putting the 4 methods in the same plot in order to get a good idea of what the actual forecasts they predict.

```
model_comparison %>%
  autoplot(pharma_stocks2) +
  labs(y = 'Adjusted Price ($)', x = 'Month')
```

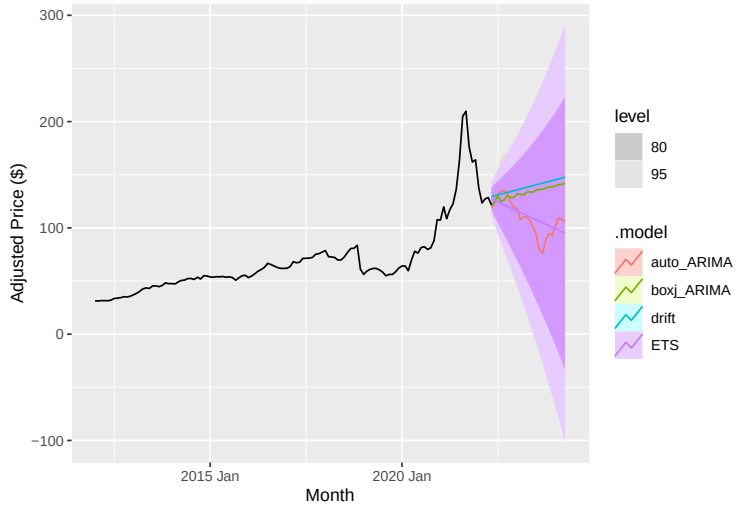

Figure 29: Model Comparison 2 Year Forecast