D590_Final_Project_Part_2

Brian Kenney, Prabakar Mohan, Brendan Kelly

2023-04-02

The report included features analyses of closing prices for 3 separate stocks for the time period from Jan 1, 2022 through Mar 24, 2023. Each stock features the same analyses to understand how different stocks prices' closing patterns act and whether they can be fit to similar models.

1 - Analysis of AAPL

1.0 - Analysis Setup

Set up File name

```
file = "Data/AAPL_2023-03-26.csv"
```

Read the .CSV and select Date and Closing Price

```
Stock_information <- read.csv(file)

selected_stock_information <- Stock_information %>%
  select(c('Date','Close')) %>%
  mutate(row_value = row_number(),
         Date = as.Date(Date,origin="2022-01-01"))
```

Collect Stock/ETF Ticker to be displayed on future visualizations to remove confusion

```
file_name = substring(file,6,9)
```

First set

```
selected_stock_information_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

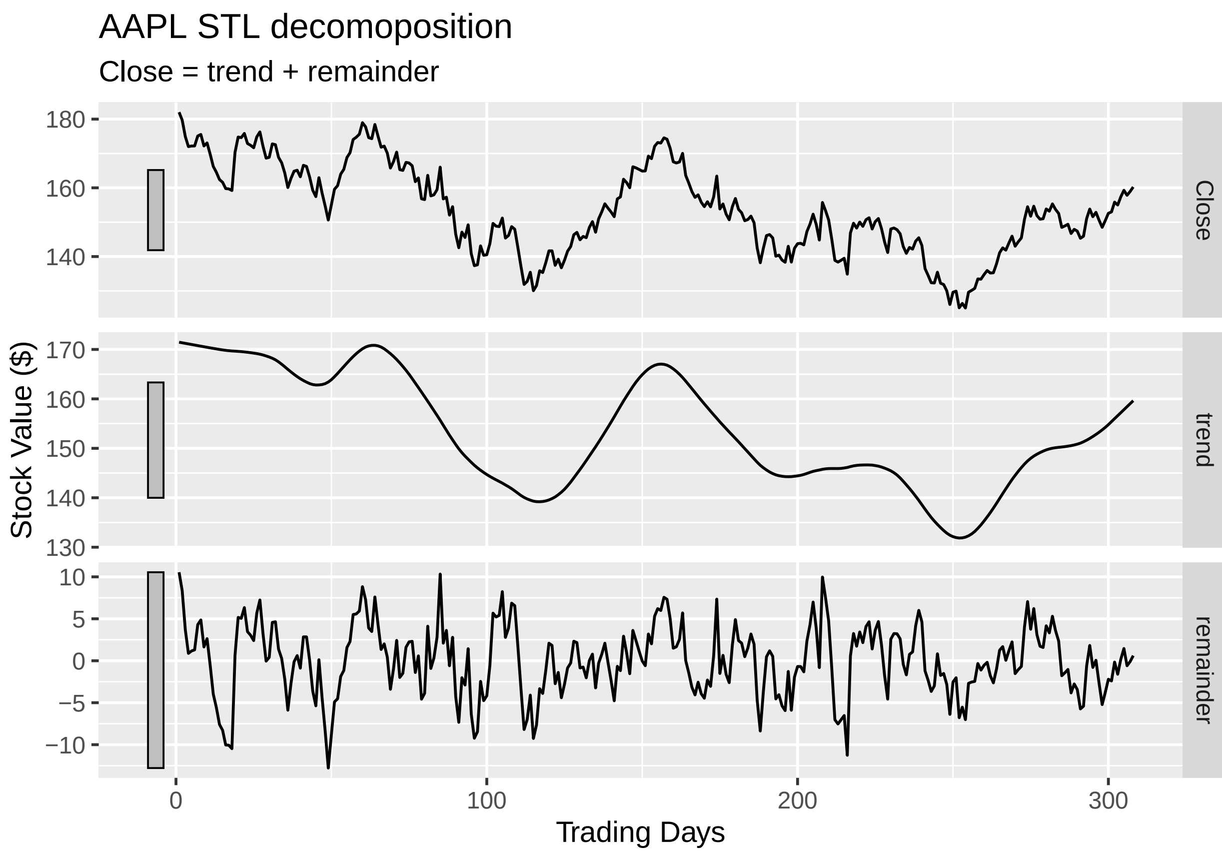
Set up for Monthly; second set. This could be useful in the future.

```
selected_stock_information_monthly <- yearmonth(selected_stock_information$`Date`, format = "%Y-%m-%d")
selected_stock_information$`Date` <- selected_stock_information_monthly
selected_stock_information_montly_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

```
selected_stock_information_montly_aggregated_tsibble <- selected_stock_information_montly_tsibble %>%
  aggregate(Close ~ Date, sum) %>%
  mutate(row_value = row_number()) %>%
  as_tsibble(index = row_value)
```

1.1 - Decomposition

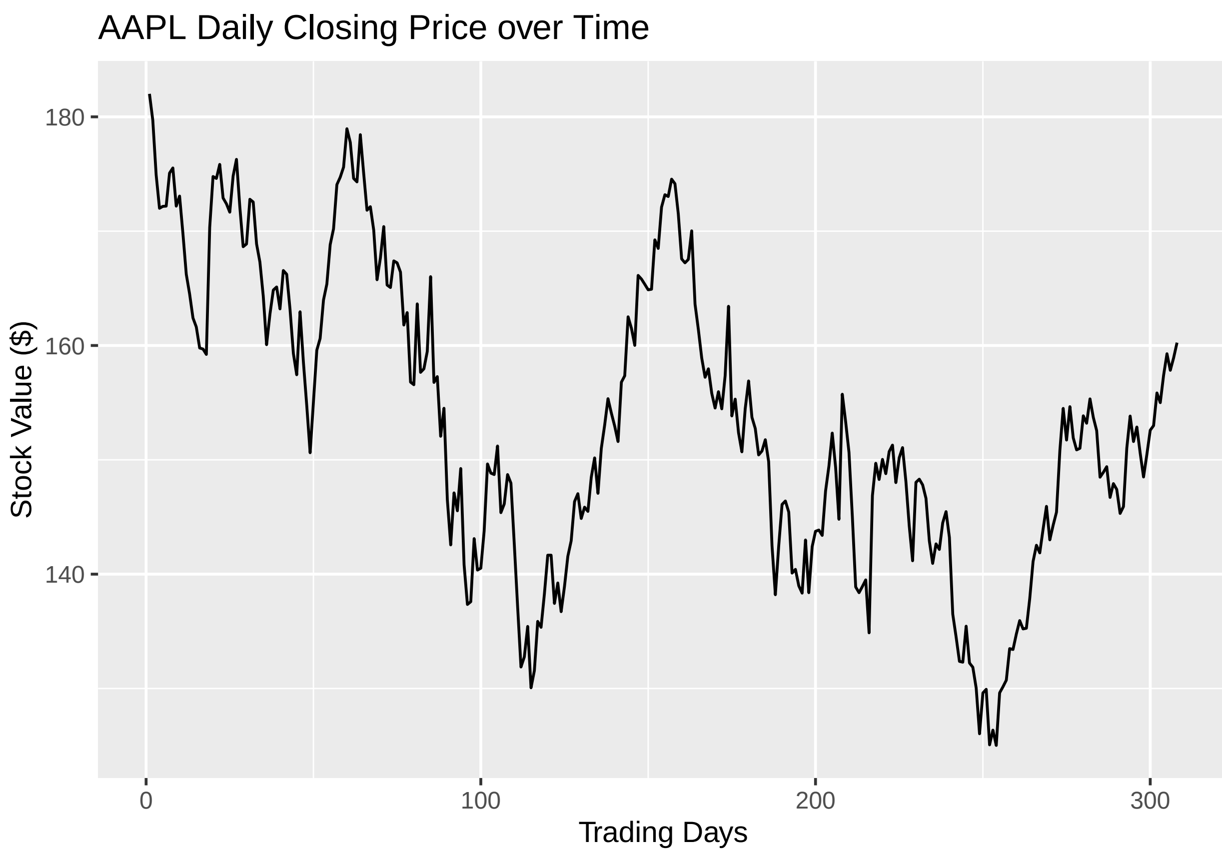
```
dcmp <- selected_stock_information_tsibble |>
  model(stl = STL(Close))

components(dcmp) |> autoplot() +
  labs(title = paste(file_name, "STL decomoposition"),
       y = "Stock Value ($)",
       x = "Trading Days")
```



1.2 - Time Series Visualization

```
autoplot(selected_stock_information_tsibble) +
  labs(y = "Stock Value ($)",
       title = paste(file_name,"Daily Closing Price over Time"),
       x = "Trading Days")
```

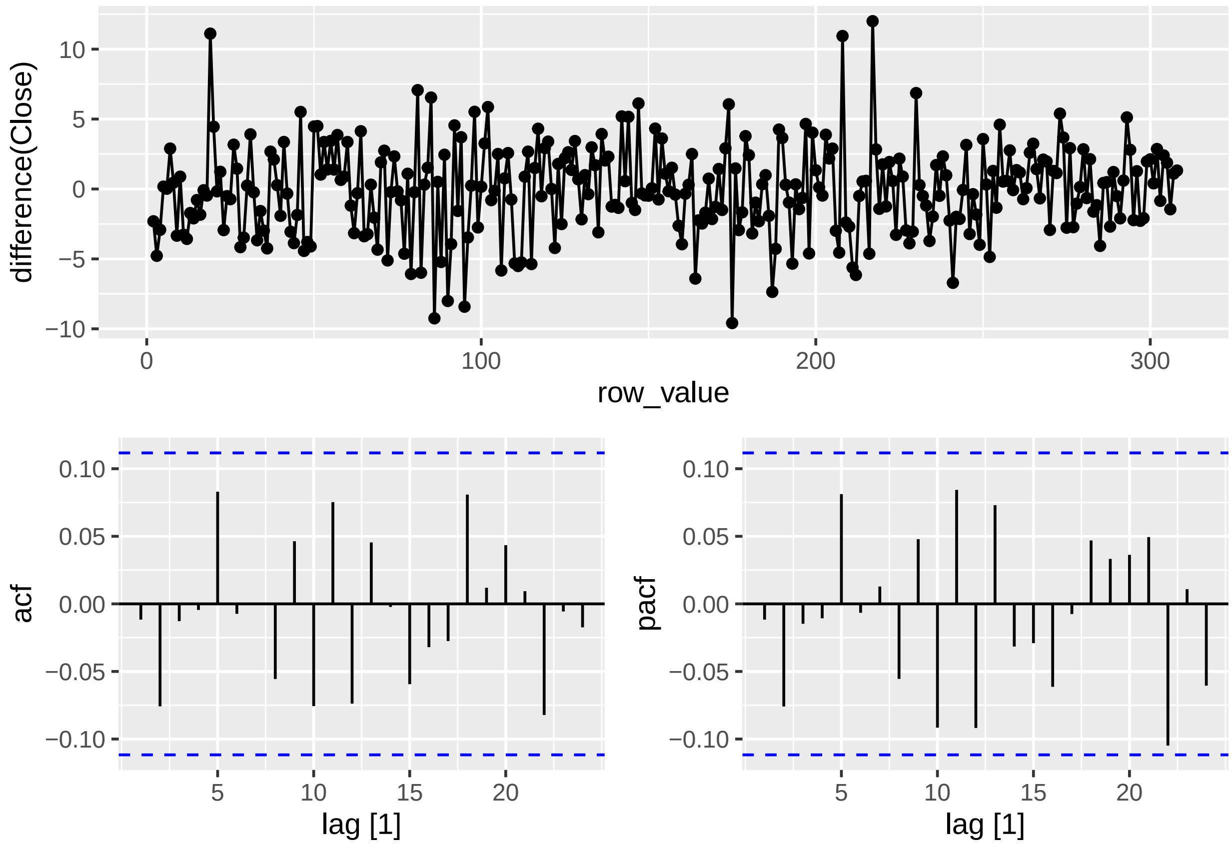
```
## Plot variable not specified, automatically selected `.vars = Close`
```



Create residual, ACF, PACF plots

```
selected_stock_information_tsibble |>
  gg_tsdisplay(difference(Close), plot_type='partial')
```

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

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



Residuals, ACF, and PACF suggest the series is a random walk.

1.3 - Description of the Time Series:

As is frequently noted, stocks tend to resemble random walk time series. The AAPL stock time series is no different; the stock's closing value appears to fluctuate up or down on each given day with no particular pattern, with some periods featuring larger downwards movements and some periods featuring larger upwards movements.

The period of time displayed may feature a slight downwards drift for AAPL stock - the ARIMA search method can be applied to determine how to best model the AAPL time series, but the residuals, ACF, and PACF as well as the fluctuations within the time series itself suggest a model with a random walk.

There is no apparent element of seasonality, and the time series is likely not stationary.

1.4 - TS Models (transformations)

Checking if the data is stationary. If kpss_stat > kpss_pvalue, then we reject null hypothesis and claim the data is non-stationary. Differencing will be required.

```
selected_stock_information_tsibble |>
  features(Close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      2.13        0.01
```

Output is the number of differences make the data stationary

```
selected_stock_information_tsibble |>
  features(Close, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

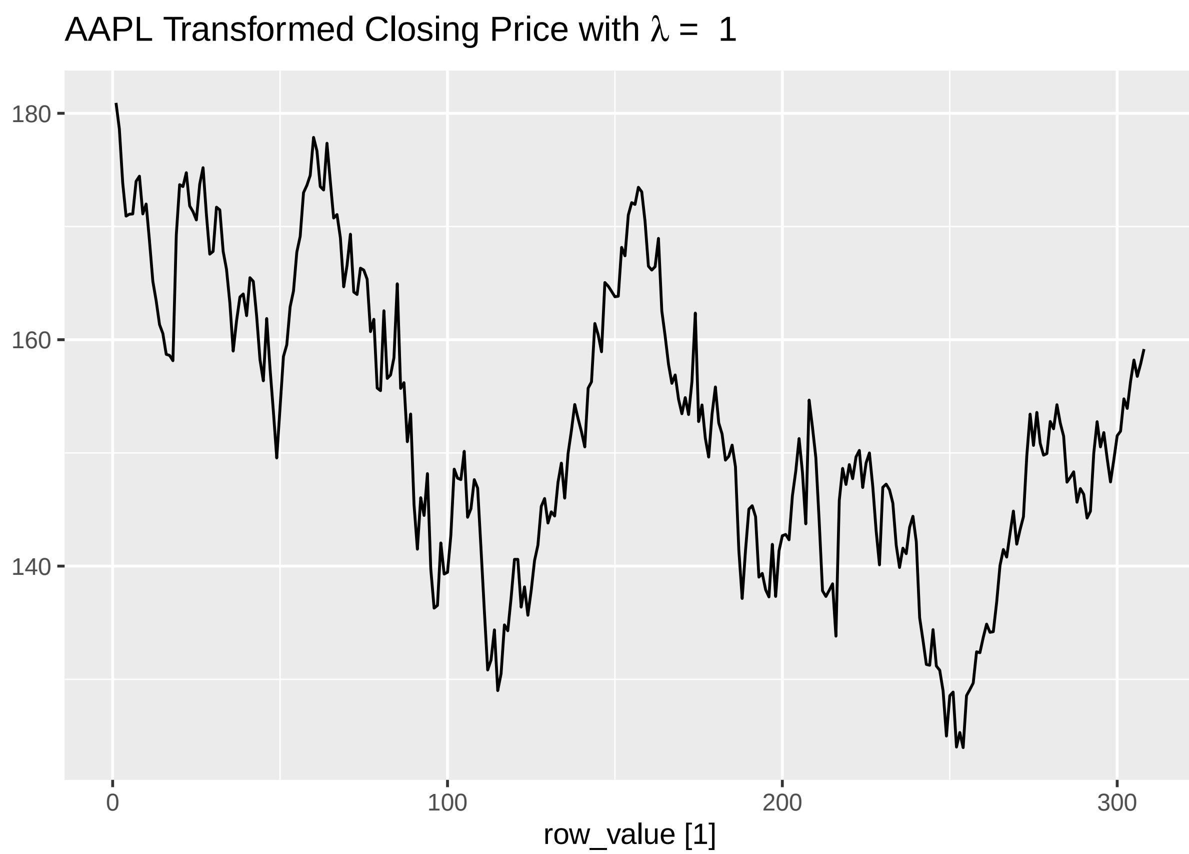
The difference

```
selected_stock_information_tsibble |>
  mutate(diff_close = difference(Close)) |>
  features(diff_close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.150         0.1
```

Using Box-Cox Transformation does not change the shape of the Closing Price with a lambda value of 1.

```
lambda <- selected_stock_information_tsibble |>
  features(Close, features = guerrero) |>
  pull(lambda_guerrero)
selected_stock_information_tsibble |>
  autoplot(box_cox(Close, lambda)) +
  labs(y = "",
       title = latex2exp::TeX(paste(file_name,
         "Transformed Closing Price with $\\lambda$ = ",
         round(lambda,2))))
```



First modeling approach will be to generate a few ARIMA orders and compare results.

```
selected_stock_fit <- selected_stock_information_tsibble |>
  model(arima010 = ARIMA(Close ~ pdq(0,1,0)),
        arima110 = ARIMA(Close ~ pdq(1,1,0)),
        arima011 = ARIMA(Close ~ pdq(0,1,1)),
        arima111 = ARIMA(Close ~ pdq(1,1,1)),
        search = ARIMA(Close, stepwise=FALSE))

print(selected_stock_fit$search)
```

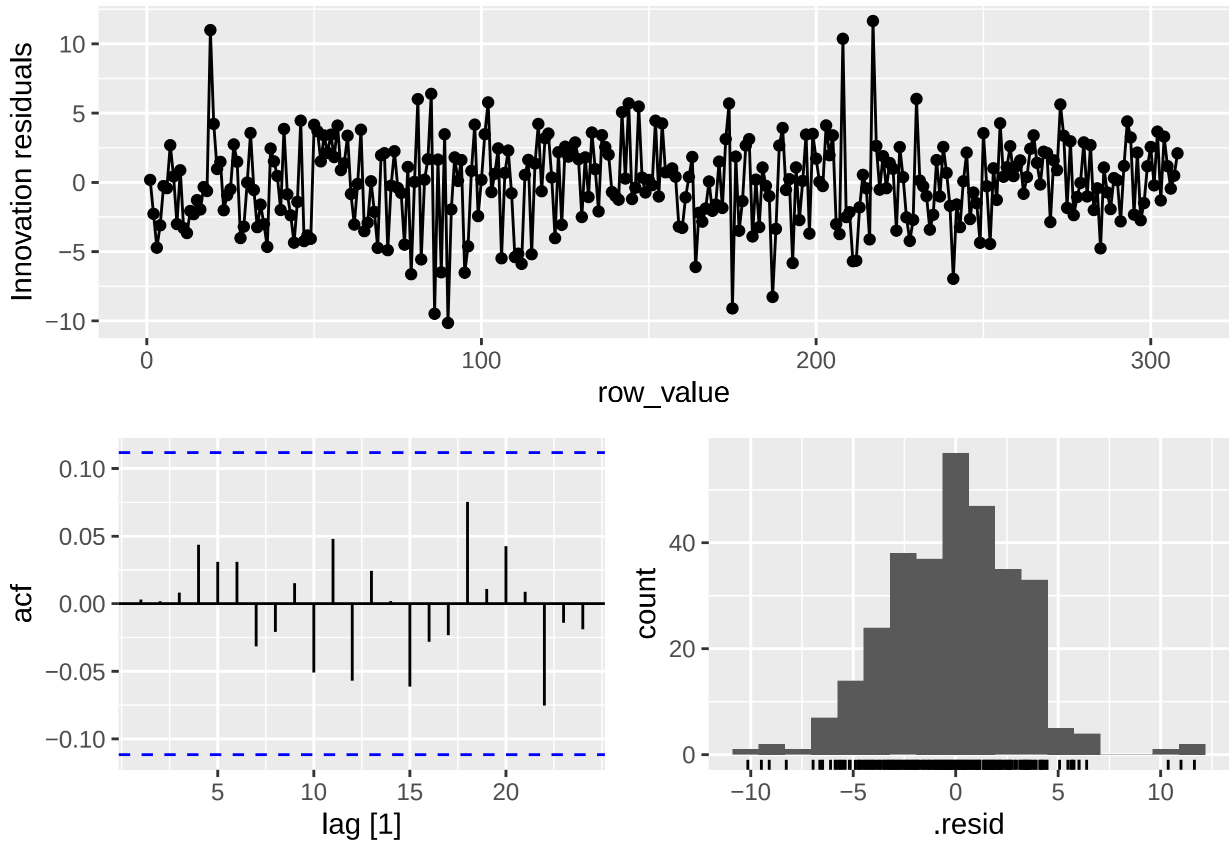
```
## <lst_mdl[1]>
## [1] <ARIMA(4,1,1)>
```

```
glance(selected_stock_fit) |> arrange(AICc) |> select(.model:BIC)
```

```
## # A tibble: 5 x 6
##   .model   sigma2 log_lik   AIC  AICc   BIC
##   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 search     10.2   -790. 1592. 1592. 1614.
## 2 arima010   10.4   -796. 1593. 1593. 1597.
## 3 arima011   10.5   -796. 1595. 1595. 1603.
## 4 arima110   10.5   -796. 1595. 1595. 1603.
## 5 arima111   10.5   -795. 1597. 1597. 1608.
```

Show the residuals using the 'search' ARIMA

```
selected_stock_fit |>
  select(search) |>
  gg_tsresiduals()
```



Portmanteau test shows a large P-value, suggesting the residuals are simliar to white noise

```
augment(selected_stock_fit) |>
  filter(.model=='search') |>
  features(.innov, ljung_box, lag = 10, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 search    2.59     0.920
```

Second approach, creating a Train set and a Test set with stepwise ARIMA and Prophet models

```
length_df = nrow(selected_stock_information_tsibble)
Train_number = round(length_df * 0.98, 0)


Train <- selected_stock_information_tsibble %>%
```

```
  filter(row_value <= Train_number) %>%
  mutate(Date = as.Date(Date,origin="2022-01-01")) %>%
  as_tsibble(index=Date) %>%
  fill_gaps()
Test <- selected_stock_information_tsibble[Train_number:length_df,]
```

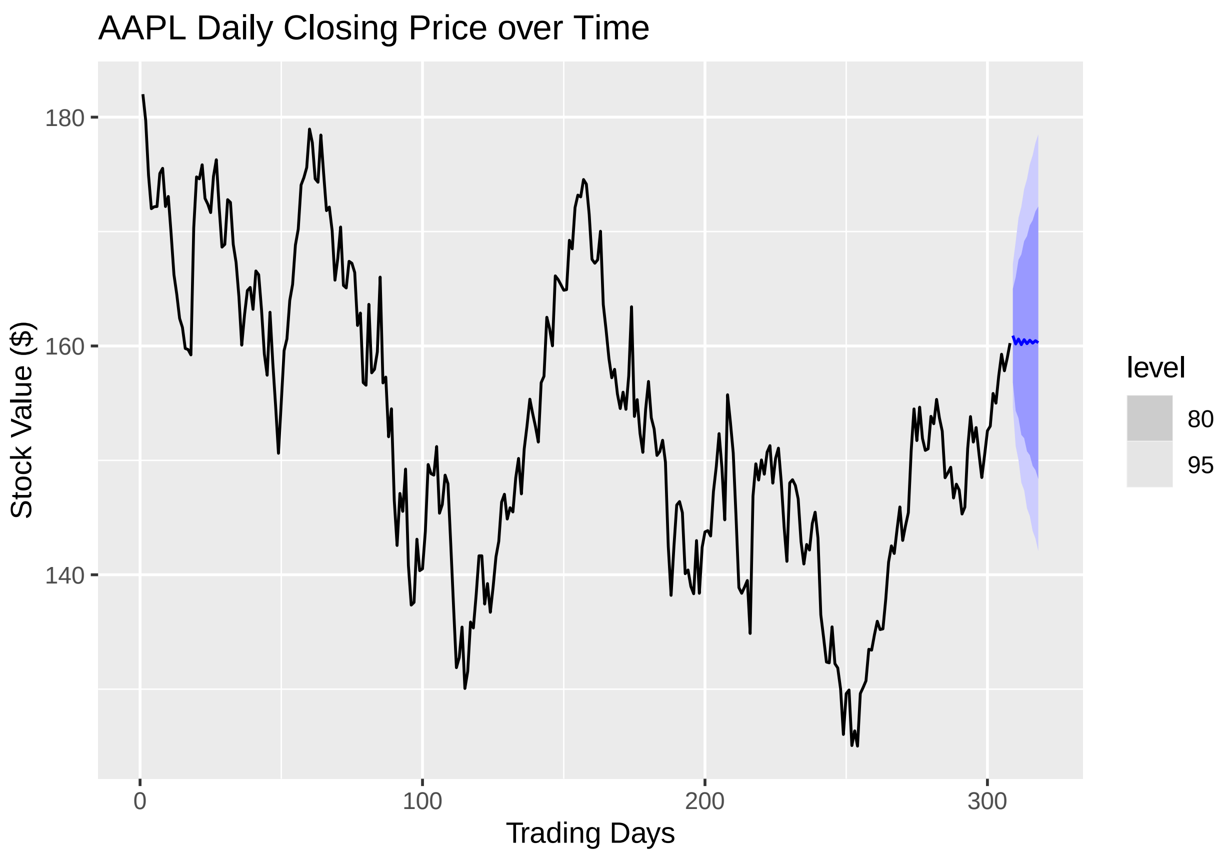
Period is set to twelve for the months, and order = 1 since data is non-seasonal

```
selected_stock_information_tsibble_fit <- Train |>
  model(
    arima = ARIMA(Close),
    prophet = prophet(Close ~ season(period = 12, order = 1,
                                    type = "additive"))
  )
```

1.5 - Predictions

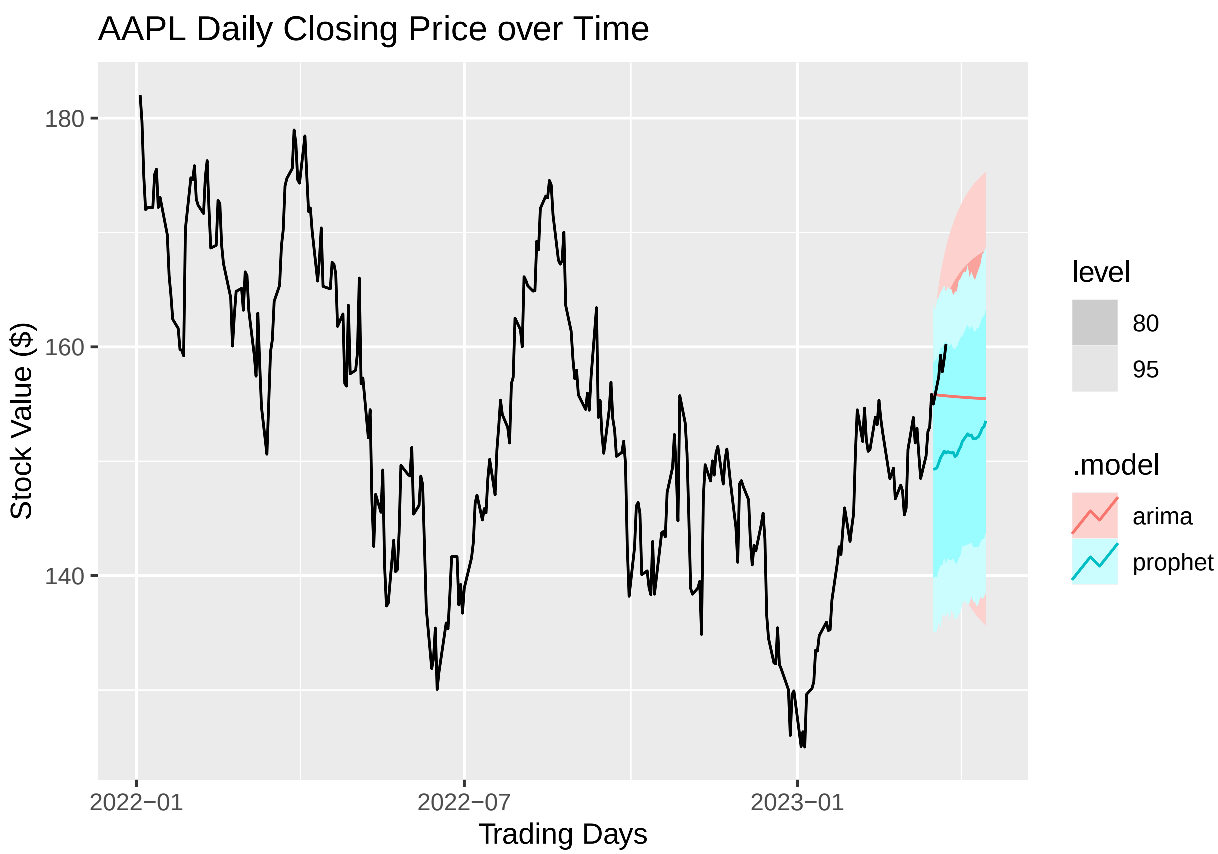
First Prediction - uses selected ARIMA

```
selected_stock_fit |>
  forecast(h=10) |>
  filter(.model=='search') |>
  autoplot(selected_stock_information_tsibble)  +
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```



Second Prediction set - Prophet + stepwise ARIMA

```
selected_stock_information_tsibble_fit_fc <- selected_stock_information_tsibble_fit |> forecast(h = 30)
selected_stock_information_tsibble_fit_fc |> autoplot(selected_stock_information_tsibble %>% as_tsibble
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```


AAPL Daily Closing Price over Time
Stock Value ($)
180
160
140
2022−01
2022−07
2023−01
Trading Days
level
80
95
.model
arima
prophet

2 - Analysis of MSFT

2.0 - Analysis Setup

Set up File name

```
file = "Data/MSFT_2023-03-26.csv"
```

Read the .CSV and select Date and Closing Price

```
Stock_information <- read.csv(file)

selected_stock_information <- Stock_information %>%
  select(c('Date','Close')) %>%
  mutate(row_value = row_number(),
         Date = as.Date(Date,origin="2022-01-01"))
```

Collect Stock/ETF Ticker to be displayed on future visualizations to remove confusion

```
file_name = substring(file,6,9)
```

First set

```
selected_stock_information_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

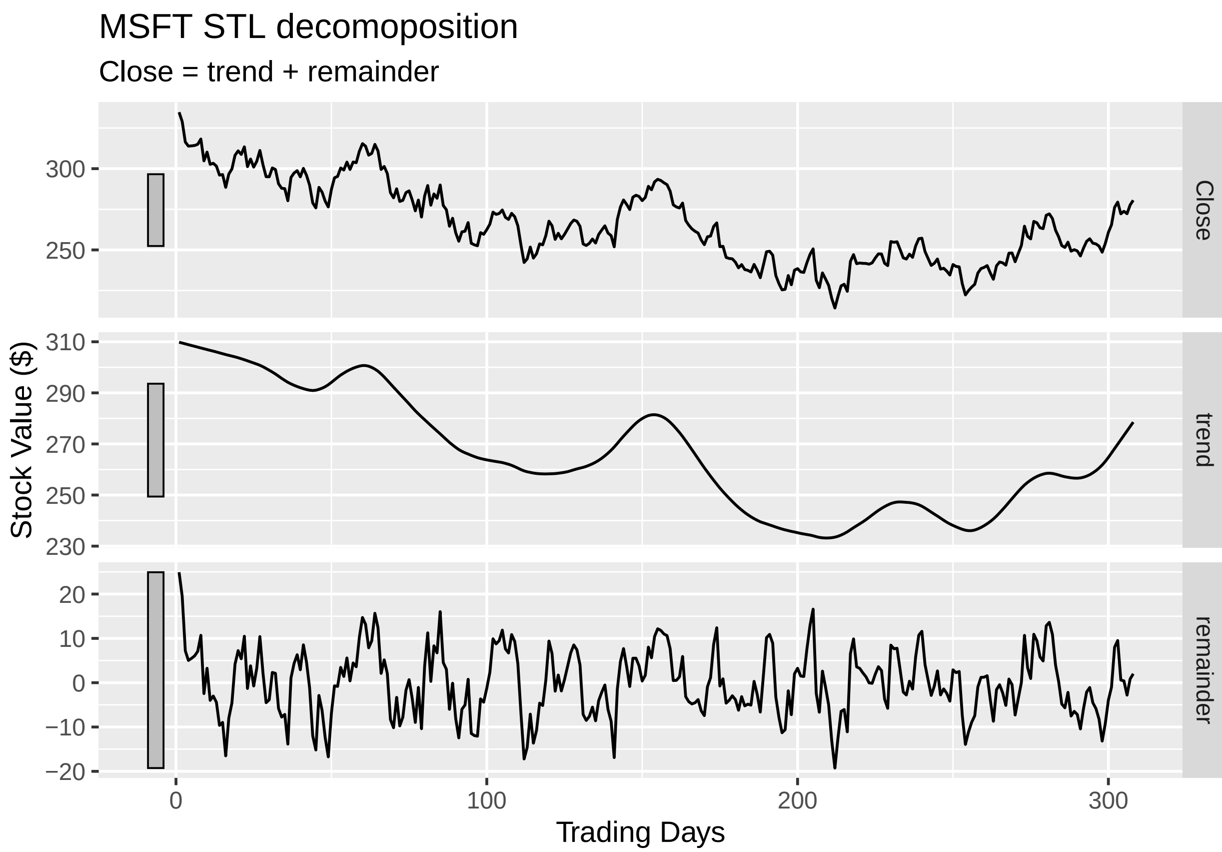
Set up for Monthly; second set. This could be useful in the future.

```
selected_stock_information_monthly <- yearmonth(selected_stock_information$`Date`, format = "%Y-%m-%d")
selected_stock_information$`Date` <- selected_stock_information_monthly
selected_stock_information_montly_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

```
selected_stock_information_montly_aggregated_tsibble <- selected_stock_information_montly_tsibble %>%
  aggregate(Close ~ Date, sum) %>%
  mutate(row_value = row_number()) %>%
  as_tsibble(index = row_value)
```

2.1 - Decomposition

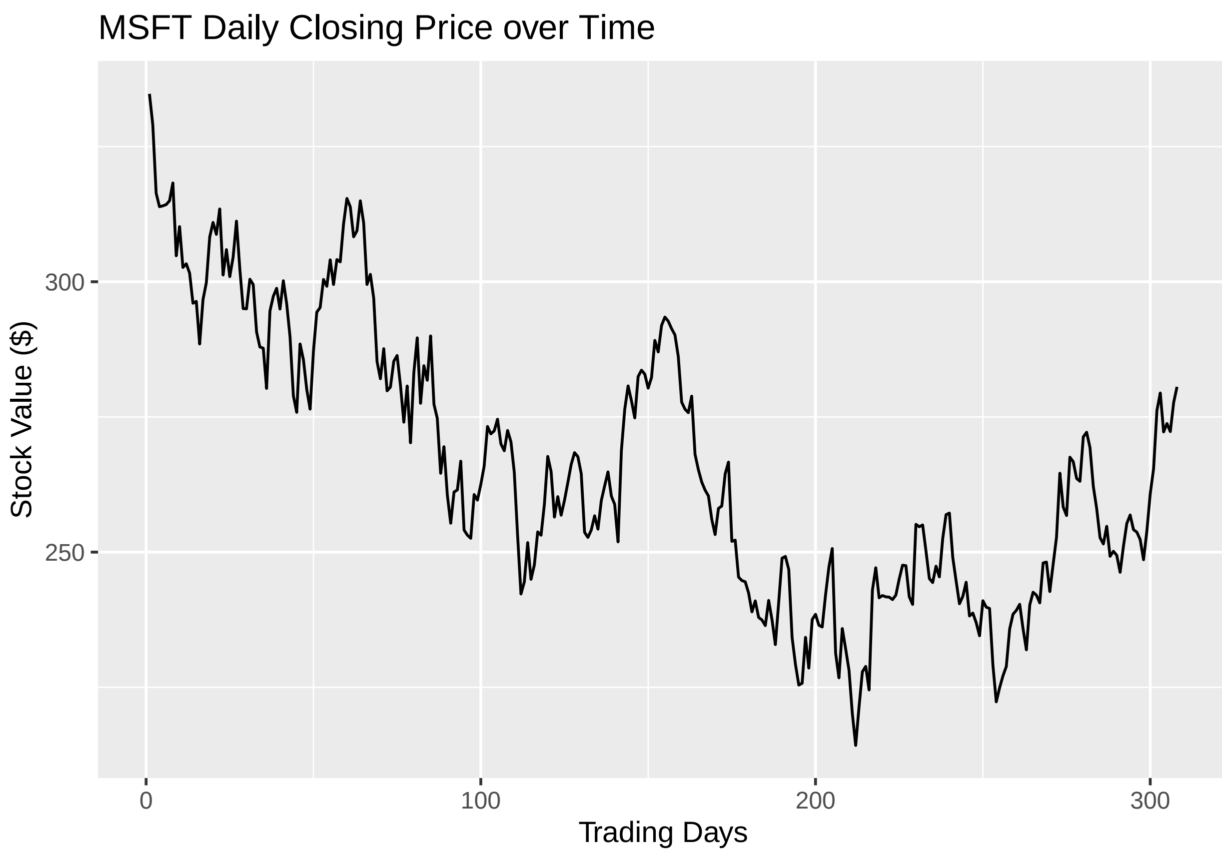
```
dcmp <- selected_stock_information_tsibble |>
  model(stl = STL(Close))

components(dcmp) |> autoplot() +
  labs(title = paste(file_name, "STL decomoposition"),
       y = "Stock Value ($)",
       x = "Trading Days")
```



2.2 - Time Series Visualization

```
autoplot(selected_stock_information_tsibble) +
  labs(y = "Stock Value ($)",
       title = paste(file_name,"Daily Closing Price over Time"),
       x = "Trading Days")
```

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

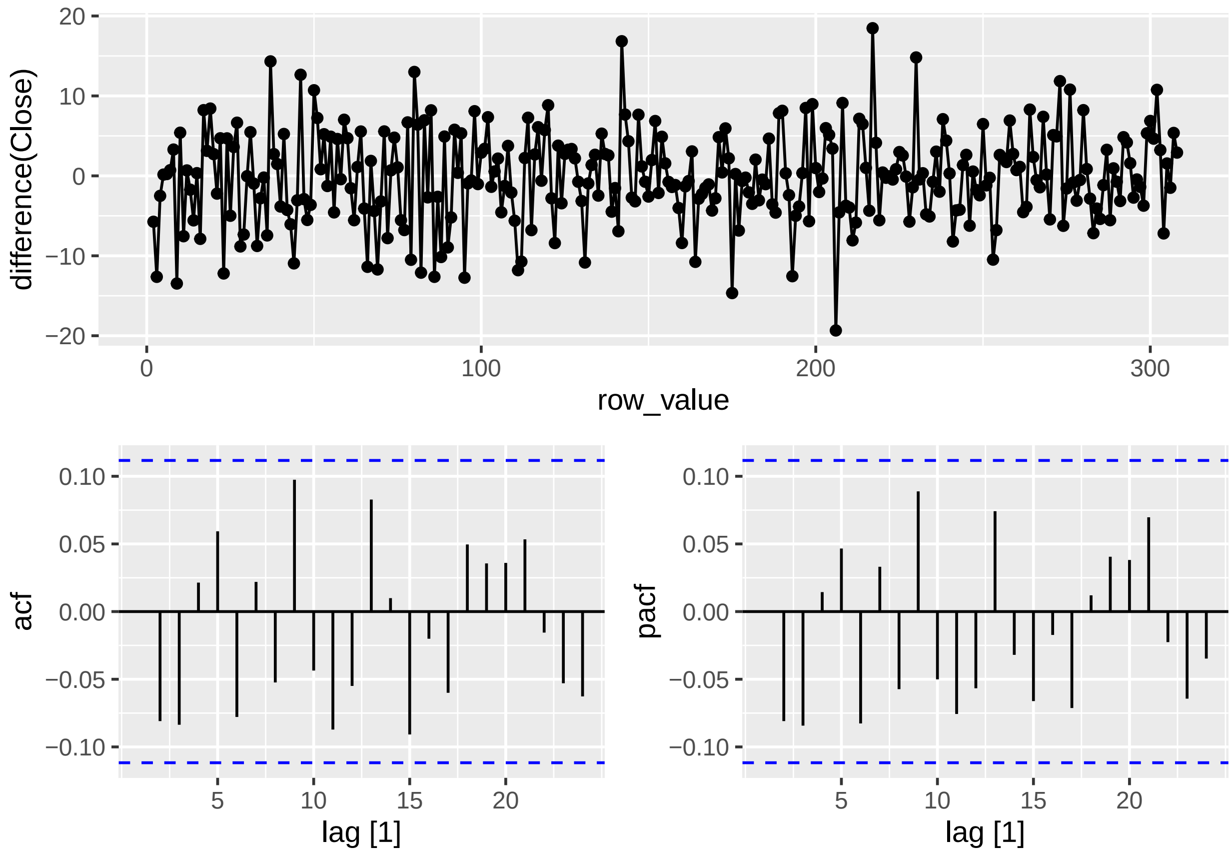


create residual, ACF, PACF plots

```
selected_stock_information_tsibble |>
  gg_tsdisplay(difference(Close), plot_type='partial')
```

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

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



Residuals, ACF, and PACF suggest the series is similar to white noise.

2.3 - Description of the Time Series:

Much like the AAPL stock, MSFT stock represents a random walk in terms of day-to-day fluctuations featuring high levels of variability. The extended upward and downward patterns in the MSFT closing price mirror the periods in which AAPL exhibited the same patterns, suggesting that broader market forces impact both stocks.

Additionally, the MSFT stock may also feature a downwards drift, although the MSFT decline over the period from \$330.81 to \$280.57 per share (-15.2%) represents a larger decline than AAPL's decline from \$180.68 to \$160.25 per share (-11.3%). This suggests that further analysis may uncover a larger downward drift in MSFT's price over the period than AAPL's after accounting for market forces.

There is no apparent element of seasonality, and the time series is likely not stationary.

2.4 - TS Models (transformations)

Checking if the data is stationary. If kpss_stat > kpss_pvalue, then we reject null hypothesis and claim the data is non-stationary. Differencing will be required.

```
selected_stock_information_tsibble |>
  features(Close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      3.37        0.01
```

Output is the number of differences make the data stationary

```
selected_stock_information_tsibble |>
  features(Close, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

The difference

```
selected_stock_information_tsibble |>
  mutate(diff_close = difference(Close)) |>
  features(diff_close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.238         0.1
```

Using Box-Cox Transformation does not change the shape of the Closing Price with a lambda value of 1.36

```
lambda <- selected_stock_information_tsibble |>
  features(Close, features = guerrero) |>
  pull(lambda_guerrero)
selected_stock_information_tsibble |>
  autoplot(box_cox(Close, lambda)) +
  labs(y = "",
       title = latex2exp::TeX(paste(file_name,
         "Transformed Closing Price with $\\lambda$ = ",
         round(lambda,2))))
```

First modeling approach will be to generate a few ARIMA orders and compare results.

```
selected_stock_fit <- selected_stock_information_tsibble |>
  model(arima010 = ARIMA(Close ~ pdq(0,1,0)),
        arima110 = ARIMA(Close ~ pdq(1,1,0)),
        arima011 = ARIMA(Close ~ pdq(0,1,1)),
        arima111 = ARIMA(Close ~ pdq(1,1,1)),
        search = ARIMA(Close, stepwise=FALSE))

print(selected_stock_fit$search)
```

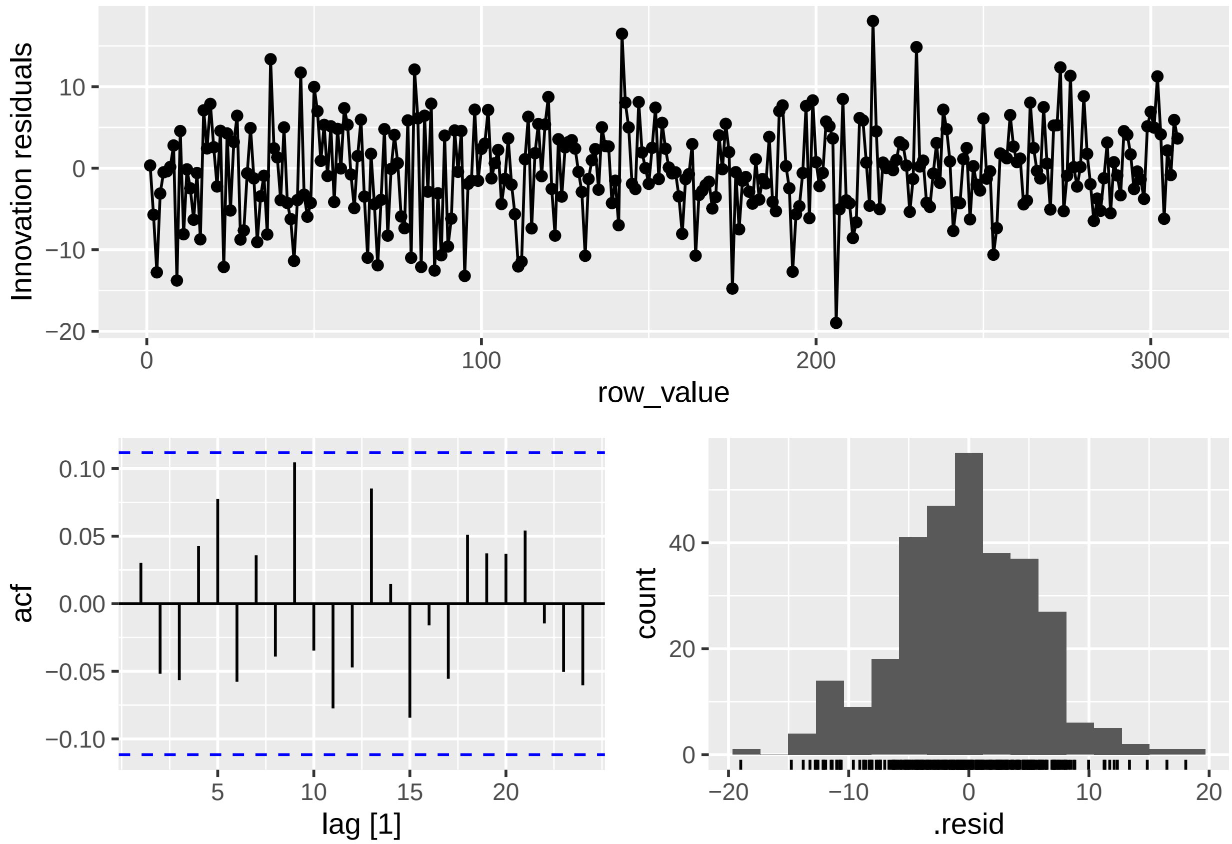
```
## <lst_mdl[1]>
## [1] <ARIMA(1,1,1)>
```

```
glance(selected_stock_fit) |> arrange(AICc) |> select(.model:BIC)
```

```
## # A tibble: 5 x 6
##   .model   sigma2 log_lik   AIC  AICc   BIC
##   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 arima010   33.1   -973. 1948. 1948. 1952.
## 2 arima111   33.1   -972. 1950. 1950. 1961.
## 3 search     33.1   -972. 1950. 1950. 1961.
## 4 arima011   33.2   -973. 1950. 1950. 1957.
## 5 arima110   33.2   -973. 1950. 1950. 1957.
```

Show the residuals using the 'search' ARIMA

```
selected_stock_fit |>
  select(search) |>
  gg_tsresiduals()
```



Portmanteau test shows a large P-value, suggesting the residuals are simliar to white noise

```
augment(selected_stock_fit) |>
  filter(.model=='search') |>
  features(.innov, ljung_box, lag = 10, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 search    10.4     0.166
```

Second approach, creating a Train set and a Test set with stepwise ARIMA and Prophet models

```
length_df = nrow(selected_stock_information_tsibble)
Train_number = round(length_df * 0.98, 0)


Train <- selected_stock_information_tsibble %>%
```

```r
  filter(row_value <= Train_number) %>%
  mutate(Date = as.Date(Date,origin="2022-01-01")) %>%
  as_tsibble(index=Date) %>%
  fill_gaps()
Test <- selected_stock_information_tsibble[Train_number:length_df,]
```

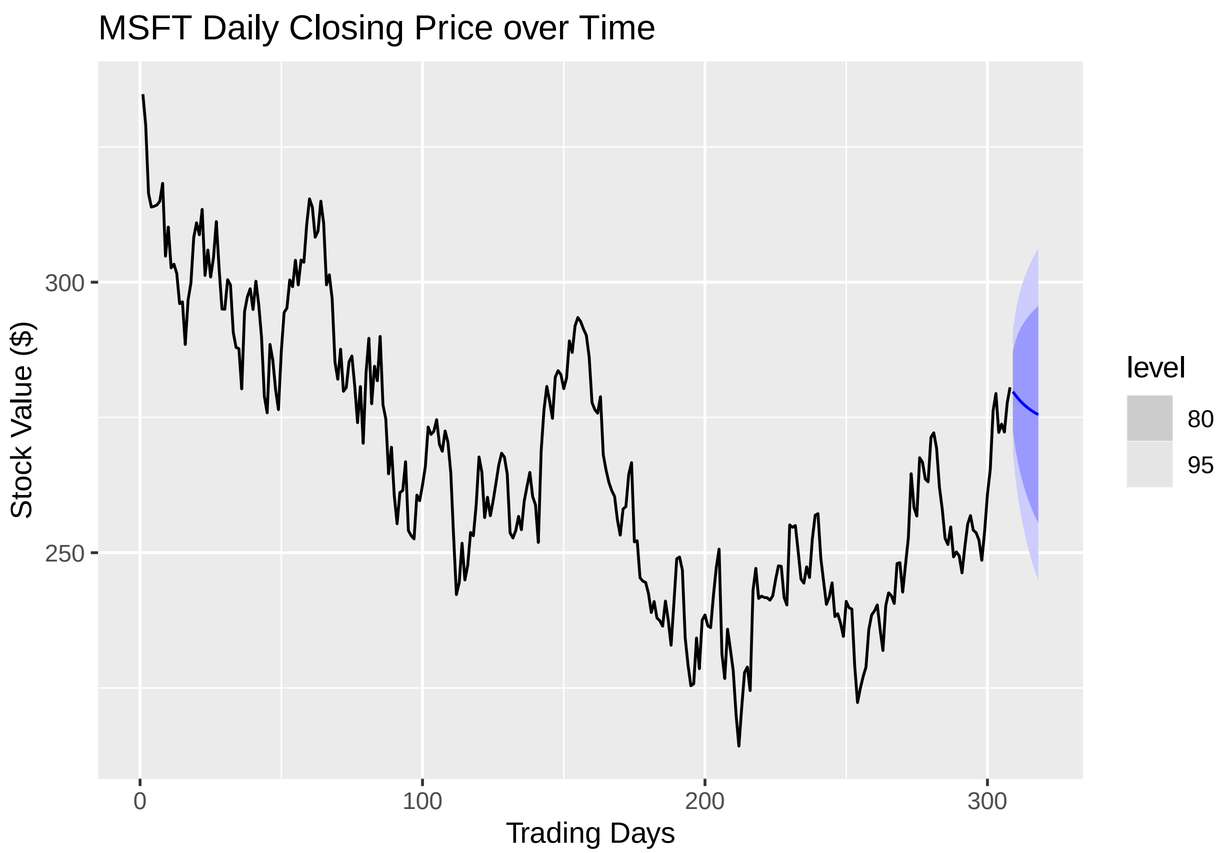
Period is set to twelve for the months, and order = 1 since data is non-seasonal

```r
selected_stock_information_tsibble_fit <- Train |>
  model(
    arima = ARIMA(Close),
    prophet = prophet(Close ~ season(period = 12, order = 1,
                                     type = "additive"))
  )
```

2.5 - Predictions

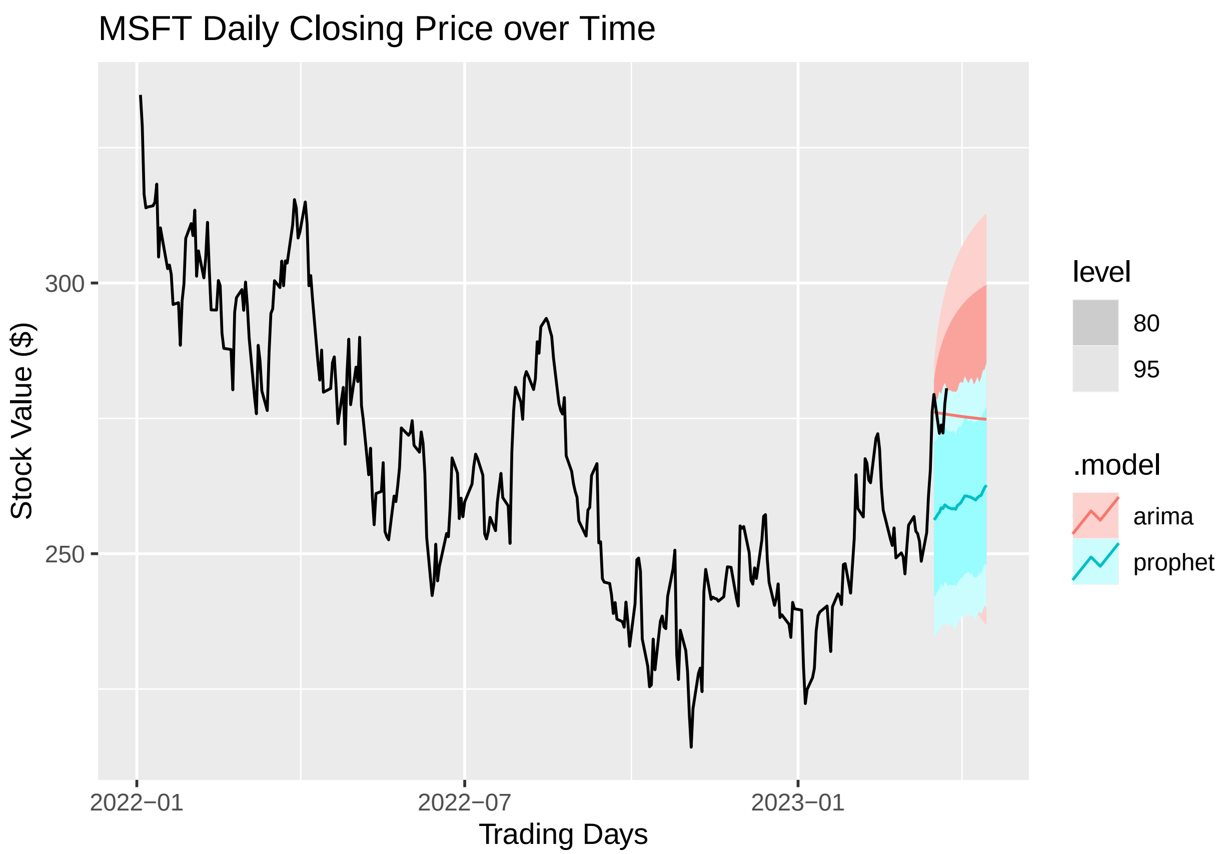
First Prediction - uses selected ARIMA

```
selected_stock_fit |>
  forecast(h=10) |>
  filter(.model=='search') |>
  autoplot(selected_stock_information_tsibble)  +
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```



Second Prediction set - Prophet + stepwise ARIMA

```
selected_stock_information_tsibble_fit_fc <- selected_stock_information_tsibble_fit |> forecast(h = 30)
selected_stock_information_tsibble_fit_fc |> autoplot(selected_stock_information_tsibble %>% as_tsibble
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```


MSFT Daily Closing Price over Time
Stock Value ($)
300
250
2022−01
2022−07
2023−01
Trading Days
level
80
95
.model
arima
prophet

3 - Analysis of TSLA

3.0 - Analysis Setup

Set up File name

```
file = "Data/TSLA_2023-03-26.csv"
```

Read the .CSV and select Date and Closing Price

```
Stock_information <- read.csv(file)

selected_stock_information <- Stock_information %>%
  select(c('Date','Close')) %>%
  mutate(row_value = row_number(),
         Date = as.Date(Date,origin="2022-01-01"))
```

Collect Stock/ETF Ticker to be displayed on future visualizations to remove confusion

```
file_name = substring(file,6,9)
```

First set

```
selected_stock_information_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

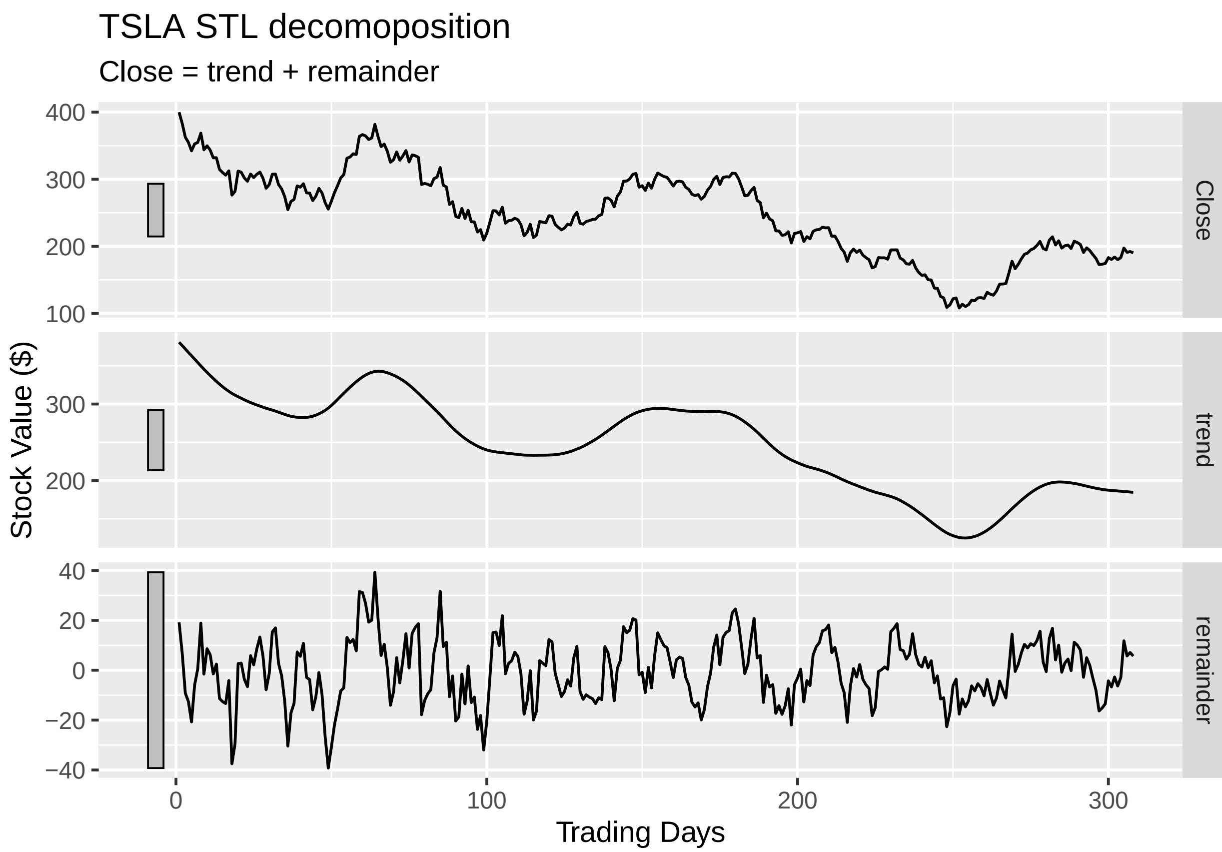
Set up for Monthly; second set. This could be useful in the future.

```
selected_stock_information_monthly <- yearmonth(selected_stock_information$`Date`, format = "%Y-%m-%d")
selected_stock_information$`Date` <- selected_stock_information_monthly
selected_stock_information_montly_tsibble <- as_tsibble(selected_stock_information, index = row_value)
```

```
selected_stock_information_montly_aggregated_tsibble <- selected_stock_information_montly_tsibble %>%
  aggregate(Close ~ Date, sum) %>%
  mutate(row_value = row_number()) %>%
  as_tsibble(index = row_value)
```

3.1 - Decomposition

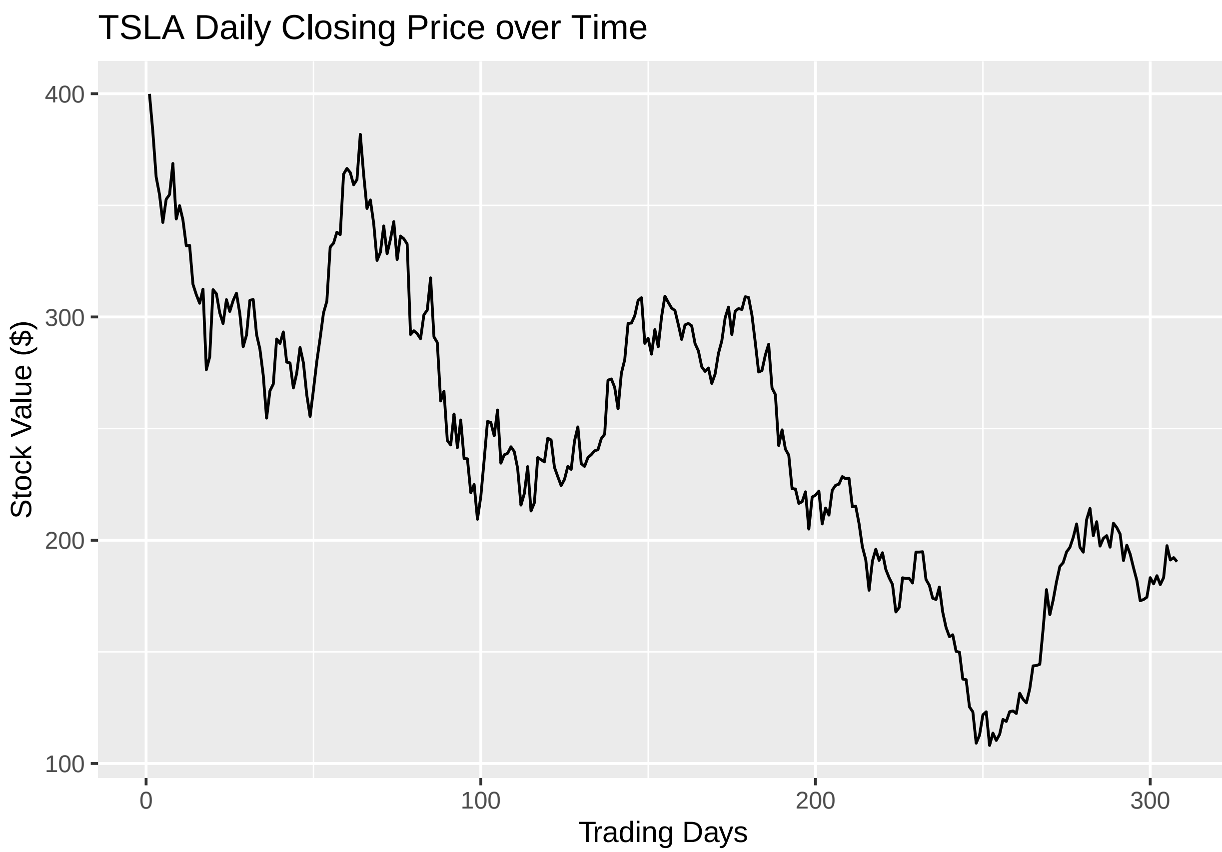
```
dcmp <- selected_stock_information_tsibble |>
  model(stl = STL(Close))

components(dcmp) |> autoplot() +
  labs(title = paste(file_name, "STL decomoposition"),
       y = "Stock Value ($)",
       x = "Trading Days")
```



3.2 - Time Series Visualization

```
autoplot(selected_stock_information_tsibble) +
  labs(y = "Stock Value ($)",
       title = paste(file_name,"Daily Closing Price over Time"),
       x = "Trading Days")
```

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

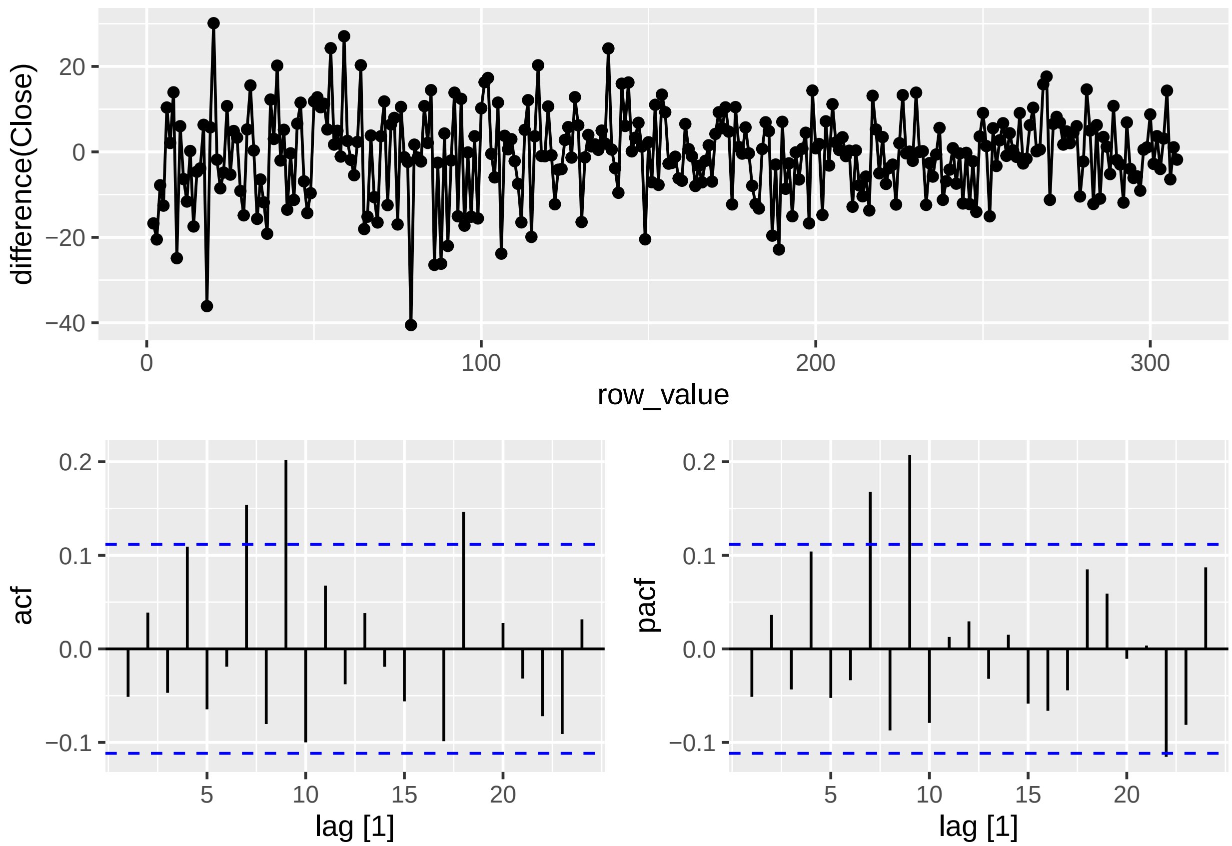


create residual, ACF, PACF plots

```
selected_stock_information_tsibble |>
  gg_tsdisplay(difference(Close), plot_type='partial')
```

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

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



Residuals, ACF, and PACF suggest the series is similar to white noise.

3.3 - Description of the Time Series:

Despite the fact that TSLA's closing price mirrors the pattern of a random walk, the closing price at the end of the period of $190.41 per share is a 52.4% decline from the first closing price per share of $399.93.

The high levels of variation in TSLA's stock price in periods of both upwards and downwards movement may make it more difficult to create an adequate random walk model, despite the downwards drift being more pronounced in TSLA's stock price.

There is no apparent element of seasonality, and the time series is certainly not stationary based on the large decline in value over the period relative to other stocks.

3.4 - TS Models (transformations)

Checking if the data is stationary. If kpss_stat > kpss_pvalue, then we reject null hypothesis and claim the data is non-stationary. Differencing will be required.

```
selected_stock_information_tsibble |>
  features(Close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      3.58        0.01
```

Output is the number of differences make the data stationary

```
selected_stock_information_tsibble |>
  features(Close, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

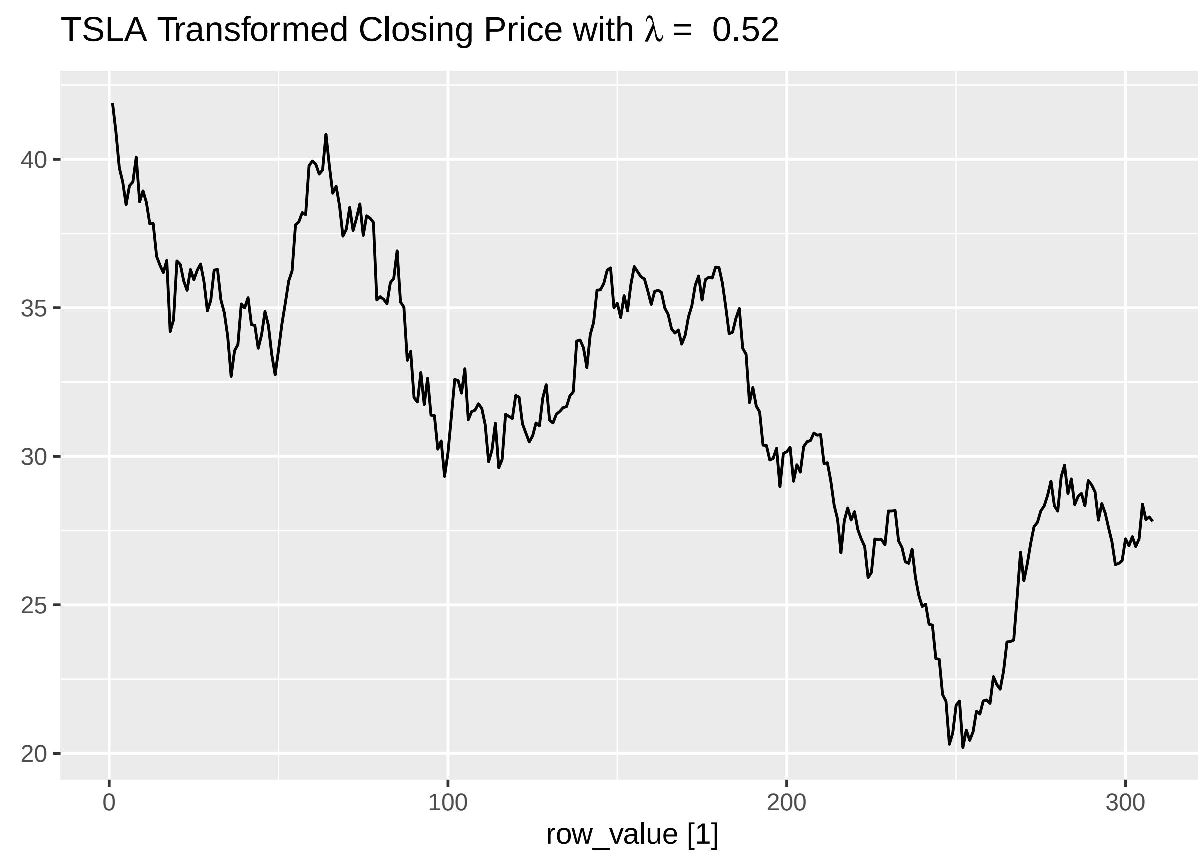
The difference

```
selected_stock_information_tsibble |>
  mutate(diff_close = difference(Close)) |>
  features(diff_close, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.127         0.1
```

Using Box-Cox Transformation does not change the shape of the Closing Price with a lambda value of 1.

```
lambda <- selected_stock_information_tsibble |>
  features(Close, features = guerrero) |>
  pull(lambda_guerrero)
selected_stock_information_tsibble |>
  autoplot(box_cox(Close, lambda)) +
  labs(y = "",
       title = latex2exp::TeX(paste(file_name,
         "Transformed Closing Price with $\\lambda$ = ",
         round(lambda,2))))
```



First modeling approach will be to generate a few ARIMA orders and compare results.

```
selected_stock_fit <- selected_stock_information_tsibble |>
  model(arima010 = ARIMA(Close ~ pdq(0,1,0)),
        arima110 = ARIMA(Close ~ pdq(1,1,0)),
        arima011 = ARIMA(Close ~ pdq(0,1,1)),
        arima111 = ARIMA(Close ~ pdq(1,1,1)),
        search = ARIMA(Close, stepwise=FALSE))

print(selected_stock_fit$search)
```

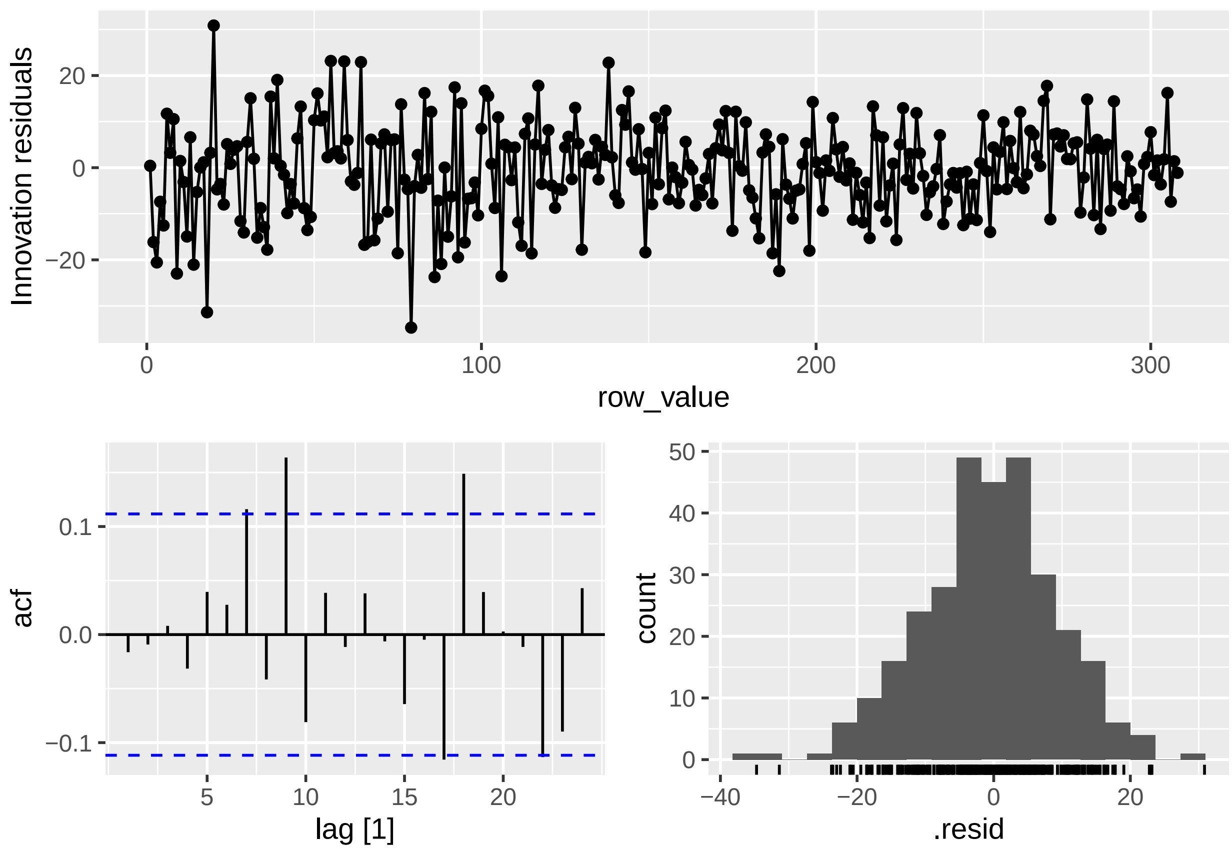
```
## <lst_mdl[1]>
## [1] <ARIMA(0,1,5)>
```

```
glance(selected_stock_fit) |> arrange(AICc) |> select(.model:BIC)
```

```
## # A tibble: 5 x 6
##   .model   sigma2 log_lik   AIC  AICc   BIC
##   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 search     102.  -1143. 2297. 2298. 2320.
## 2 arima010   105.  -1150. 2301. 2301. 2305.
## 3 arima110   105.  -1149. 2302. 2302. 2310.
## 4 arima011   105.  -1149. 2302. 2302. 2310.
## 5 arima111   105.  -1149. 2305. 2305. 2316.
```

Show the residuals using the 'search' ARIMA

```
selected_stock_fit |>
  select(search) |>
  gg_tsresiduals()
```



Portmanteau test shows a large P-value, suggesting the residuals are simliar to white noise

```
augment(selected_stock_fit) |>
  filter(.model=='search') |>
  features(.innov, ljung_box, lag = 10, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 search    16.7    0.0197
```

Second approach, creating a Train set and a Test set with stepwise ARIMA and Prophet models

```
length_df = nrow(selected_stock_information_tsibble)
Train_number = round(length_df * 0.98, 0)


Train <- selected_stock_information_tsibble %>%
```

```
  filter(row_value <= Train_number) %>%
  mutate(Date = as.Date(Date,origin="2022-01-01")) %>%
  as_tsibble(index=Date) %>%
  fill_gaps()
Test <- selected_stock_information_tsibble[Train_number:length_df,]
```

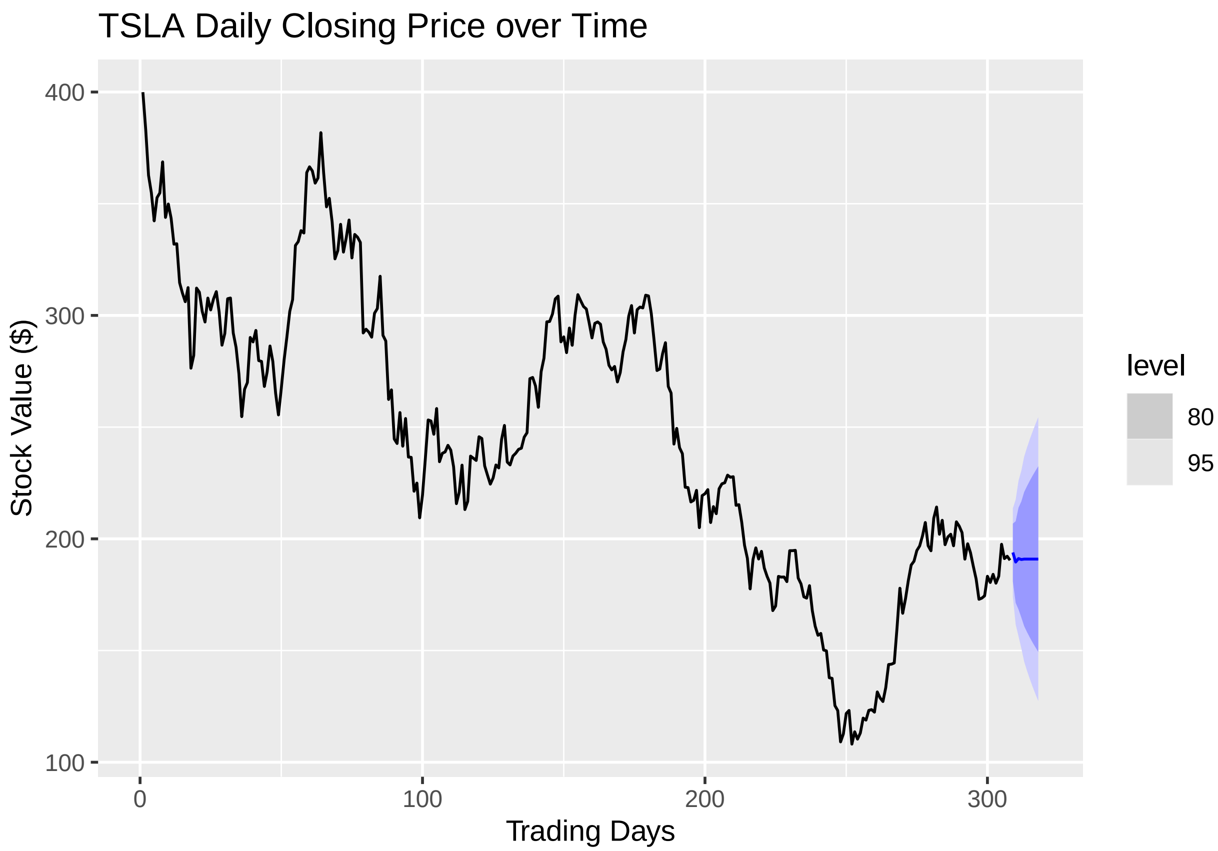
Period is set to twelve for the months, and order = 1 since data is non-seasonal

```
selected_stock_information_tsibble_fit <- Train |>
  model(
    arima = ARIMA(Close),
    prophet = prophet(Close ~ season(period = 12, order = 1,
                                    type = "additive"))
  )
```

3.5 - Predictions

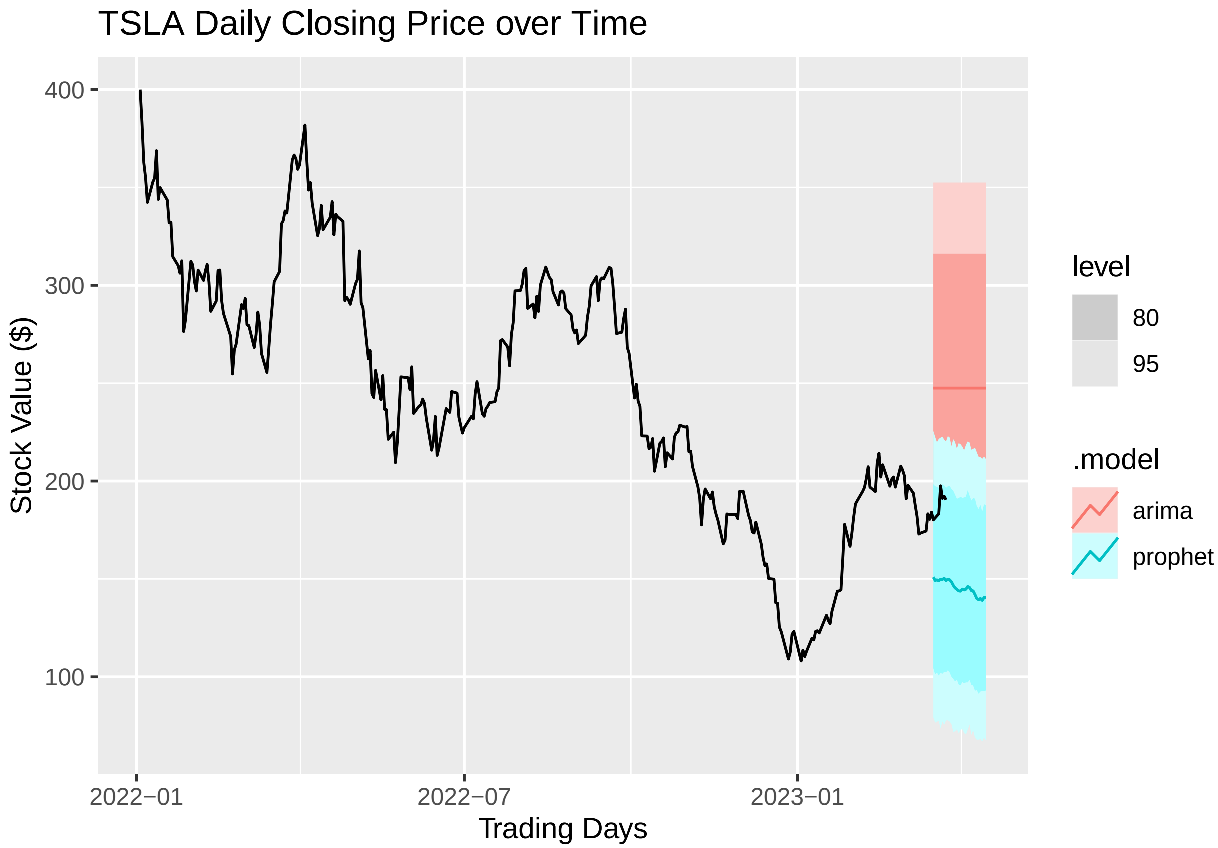
First Prediction - uses selected ARIMA

```
selected_stock_fit |>
  forecast(h=10) |>
  filter(.model=='search') |>
  autoplot(selected_stock_information_tsibble)  +
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```



Second Prediction set - Prophet + stepwise ARIMA

```
selected_stock_information_tsibble_fit_fc <- selected_stock_information_tsibble_fit |> forecast(h = 30)
selected_stock_information_tsibble_fit_fc |> autoplot(selected_stock_information_tsibble %>% as_tsibble
  labs(y = "Stock Value ($)",
       title = paste(file_name, "Daily Closing Price over Time"),
       x = "Trading Days")
```



4 - Conclusions

Each of the stocks appears to be best modeled by random walk models, either with or without drift. This aligns with conventional wisdom of stock prices following a random walk pattern, although the timeframe selected (Jan 1, 2022 through Mar 24, 2023) features a downwards drift for many cases as the broader market saw a large pullback in stock prices.

Including a different timeframe, or different stocks, may result in random walks without drift or random walks with upwards drift providing a better model fit for stock prices.

Team contributions

Brian: Completed rough draft of the Part 2 submission.

Mohan: Generated Github repo and place Datasets. Worked ahead on web app dashboard draft.

Brendan: Formatted Part 2 document, provided model descriptions and rough draft bug fixes.