Stat. 474 Project

Kevin Teresi

June 30, 2021

```
library(pacman)
p_load(fpp3, tidyverse, quantmod, tidyquant)
```

Final Project - Time Series

Find a monthly (or weekly) time series data set of interest to you that contains at least two time series to work with to make forecasts for the next 12 months.

Tesla - Closing Stock

One of the more interesting stocks over the last year and a half has been Tesla, and the monstrous rise the company has made (to even make Elon Musk the richest man in the world, momentarily). I have friends who have gotten into the Stocks game as it has risen over that same time period, and they happen to treat Mr. Musk as a god. They are doing so many things right, and positive for the world (in my opinion). Using technology for amazing products that include renewable energy options is exactly what the world needs. For that reason, I want to make a projection to really see how much Tesla is projected to rise over the next twelve months.

I'm going to use the previous five years worth of data in order to really get a large amount of data usable to predict the next 12 months of stock prices. I am going to use the closing price ("Close") as my y-axis variable, and obviously use dates as my x-axis variable. My first step is to acquire Tesla stock data, which is conducted with the following code.

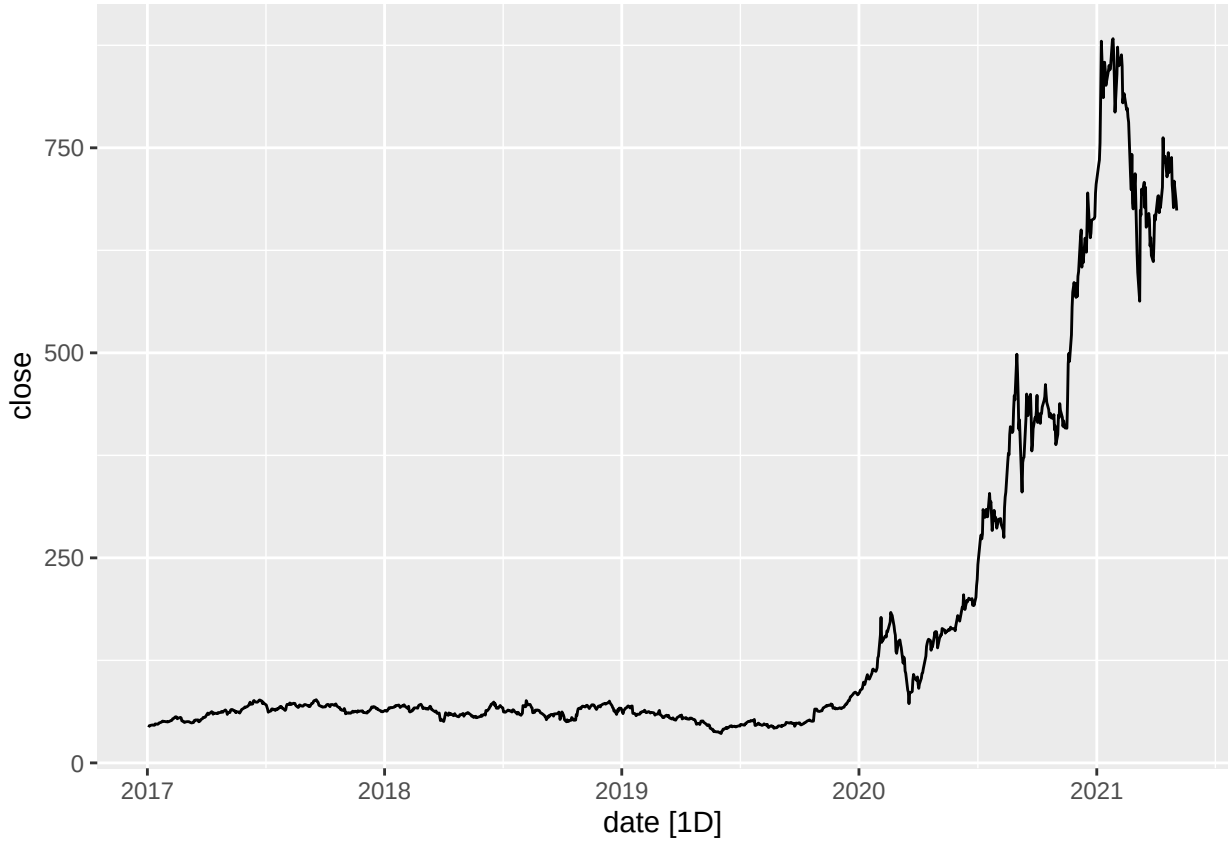
```
#Code to acquire the last five years of TSLA stock data, and turn it into a dataframe.
TSLA <- tq_get("TSLA", get = "stock.prices", from = "2017-01-01", to = "2021-05-05")
TSLA
```

```
## # A tibble: 1,091 x 8
##    symbol date        open  high   low close   volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 TSLA   2017-01-03  43.0  44.1  42.2  43.4 29616500     43.4
##  2 TSLA   2017-01-04  43.0  45.6  42.9  45.4 56067500     45.4
##  3 TSLA   2017-01-05  45.3  45.5  44.4  45.3 29558500     45.3
##  4 TSLA   2017-01-06  45.4  46.1  45.1  45.8 27639500     45.8
##  5 TSLA   2017-01-09  45.8  46.4  45.6  46.3 19897500     46.3
##  6 TSLA   2017-01-10  46.4  46.4  45.4  46.0 18300000     46.0
##  7 TSLA   2017-01-11  45.8  46.0  45.3  45.9 18254000     45.9
##  8 TSLA   2017-01-12  45.8  46.1  45.1  45.9 18951000     45.9
##  9 TSLA   2017-01-13  46    47.6  45.9  47.5 30465000     47.5
## 10 TSLA   2017-01-17  47.3  48.0  46.9  47.1 23087500     47.1
## # ... with 1,081 more rows
```

```
#Code to turn TSLA dataframe into a tsibble object.  This allows the date to be turned into an index va
TSLA_tsbl <- as_tsibble(TSLA, key = symbol)
```

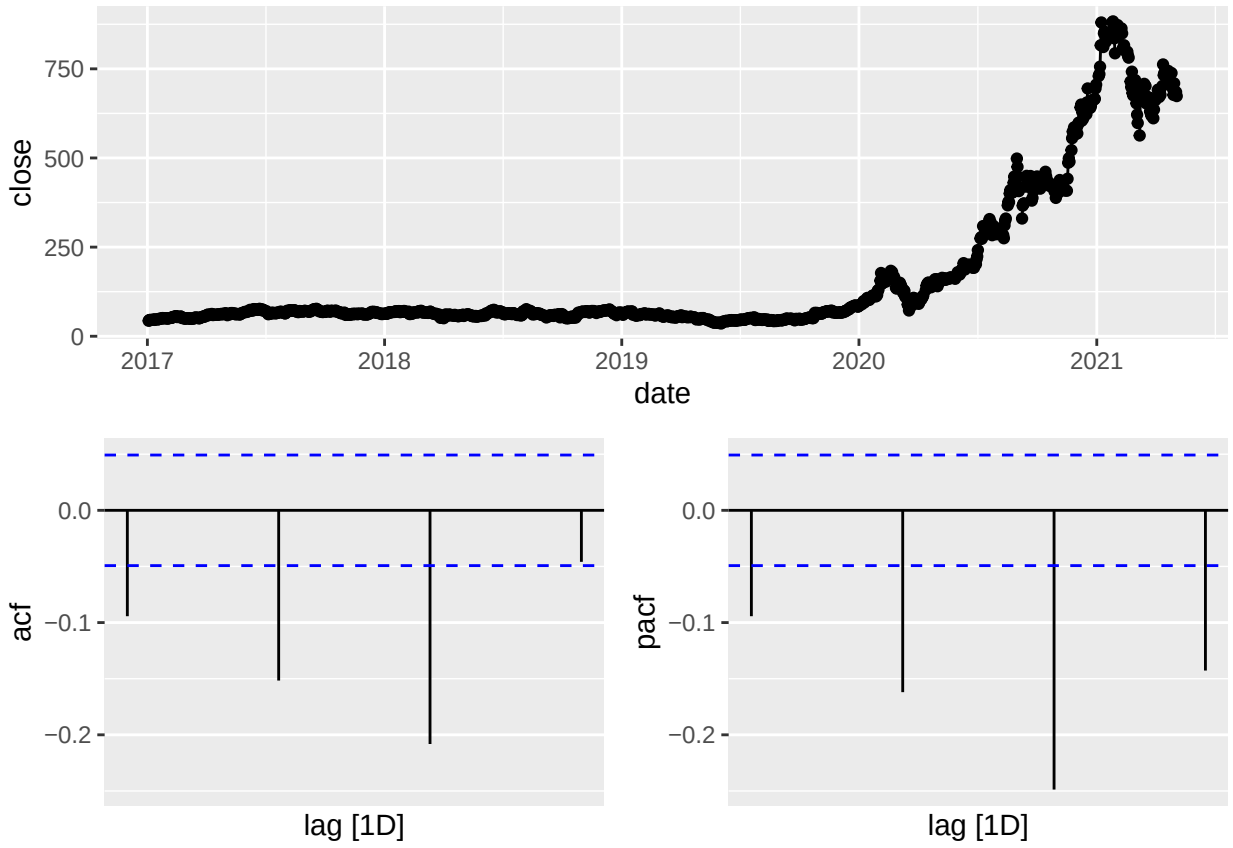
```
## Using `date` as index variable.
```

```
#Quick plot of the 'Close' variable to see how the closing price of the stock has progressed over the l
TSLA_tsbl %>%
  autoplot(close)
```

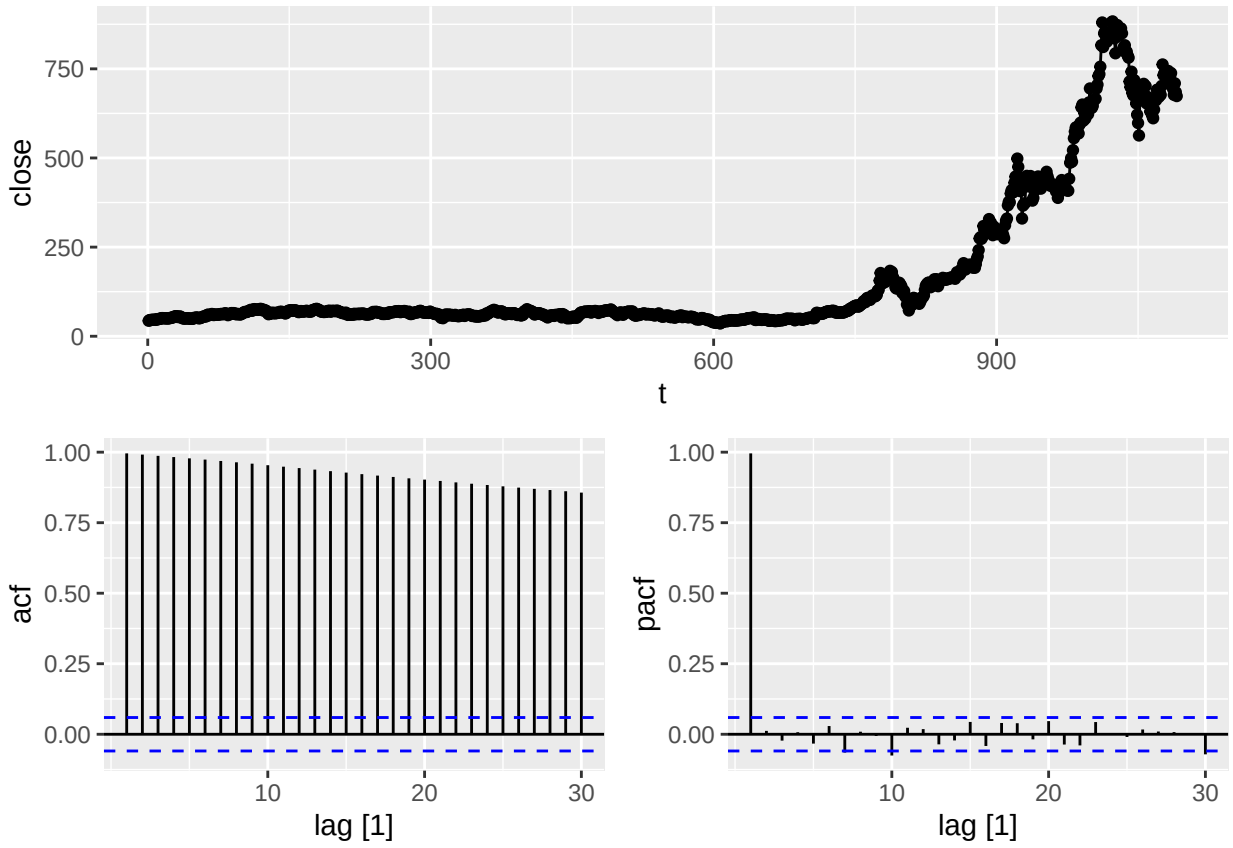



```
#Code to fill in the gaps of data (days in which the stock was closed), and display the data in a conti
TSLA_tsbl %>%
  fill_gaps() %>%
  gg_tsdisplay(close, plot_type = "partial")
```

```
## Warning: Removed 492 rows containing missing values (geom_point).
```




```
#The resulting acf and pacf from the previous plot didn't make sense, so we used different code.
# acf and pacf make more sense with this code, as there are many more points.
TSLA_tsbl %>%
  mutate(t = row_number()) %>%
  update_tsibble(index = t) %>%
  gg_tsdisplay(close, plot_type = "partial")
```



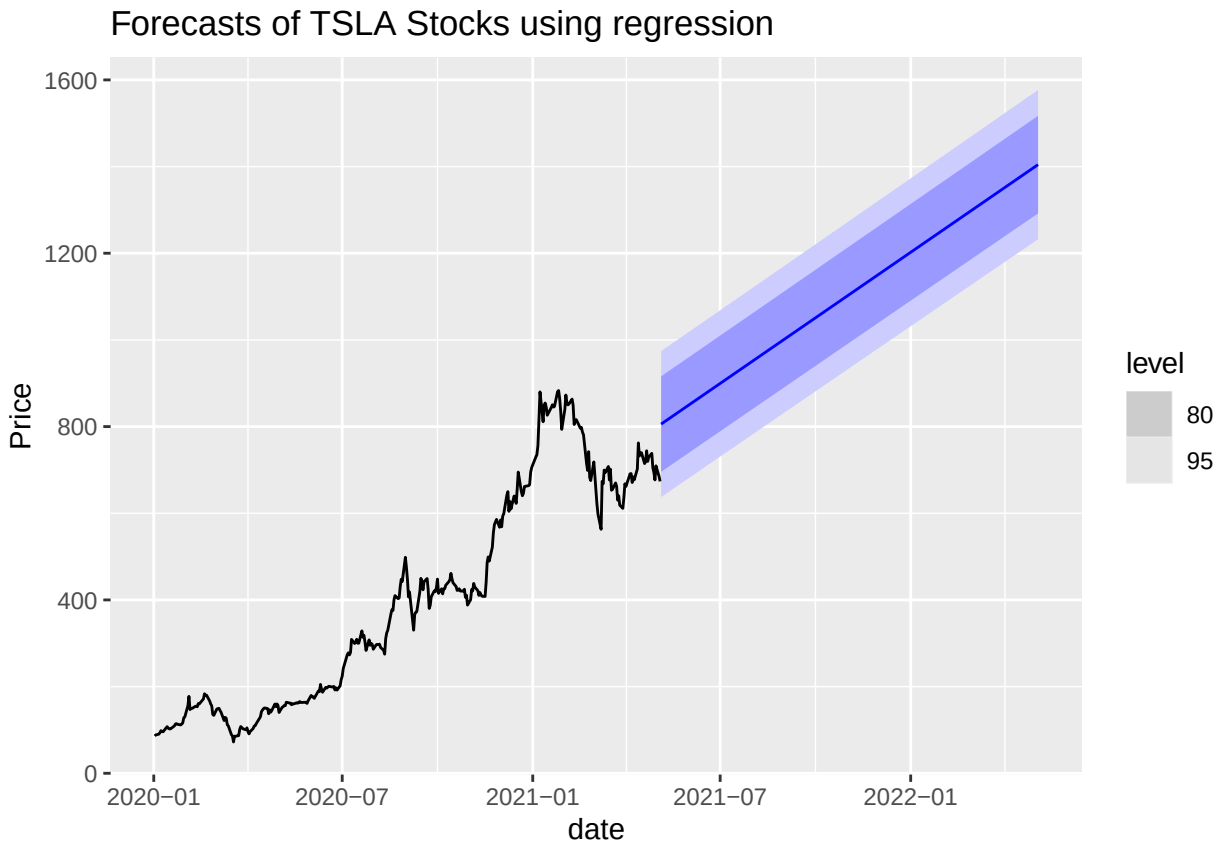
1. Build a linear regression model using TSLM().

```
recent_TSLA <- TSLA_tsbl %>%
  filter(year(date) >= 2020) #filtered 2020 onward as it is so much different from the rest of the time

fit_TSLA <- recent_TSLA %>%
  model(TSLM(close ~ trend())) #form TSLM around closing price and trend

fc_TSLA <- forecast(fit_TSLA, h = "12 months") # forecast next twelve months

fc_TSLA %>%  #plot data 2020 onward, plus 12 months of projection with TSLM() regression model.
  autoplot(recent_TSLA) +
  labs(
    title = "Forecasts of TSLA Stocks using regression",
    y = "Price"
  )
```




```
# Model to determine regression summary
recent_TSLA %>%
  model(TSLM(close ~ trend())) %>%
  report()
```

```
## Series: close
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -147.41  -60.79  -31.44   61.40  266.65
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.13166    9.32030  -0.014    0.989
## trend()      1.64478    0.03301  49.827   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.39 on 335 degrees of freedom
## Multiple R-squared: 0.8811,  Adjusted R-squared: 0.8808
## F-statistic:  2483 on 1 and 335 DF, p-value: < 2.22e-16
```

Regression line for the TSLM model since 2020 is: close = -0.13 + 1.645(day). This means that since 2020, the closing price of Tesla has grown approxamately $1.64 a trading day. That's pretty crazy. The forecasted interval looks like it could work based on this data. I decided not to use all the data since 2017 in this instance because the forecast would be too different from where the most recent data would take us. We will logarithms and differences to make the code fit our future models.

##2 Build an appropriate exponential smoothing model.

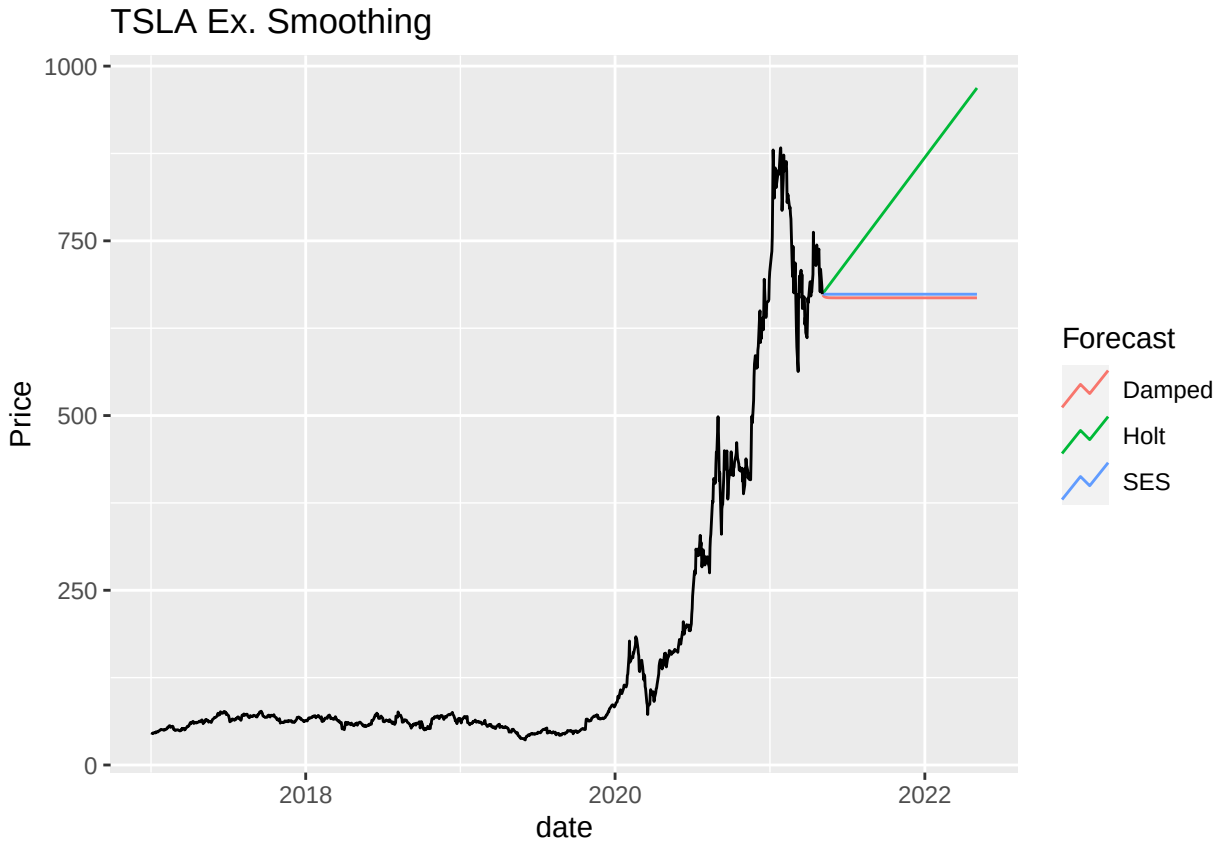
necessary package that I found to deal with the missing na values. The fill_gaps() function wouldn't work on its own, so I had to find a package that solves this issue. The imputeTS package has several different functions to fill in those blank spots. . . and I used the interpolate function, which seems to do the best job of filling in those spots with the similar values around that missing value.

```
#Time Series Missing Value Imputation
library(imputeTS)
```

```
##
## Attaching package: 'imputeTS'
```

```
## The following object is masked from 'package:zoo':
##
##     na.locf
```

```
#Code to create exponential smoothing model
models <- TSLA_tsbl %>%
  tsibble::fill_gaps() %>%
  na_interpolation() %>% # from imputeTS package.  interpolates missing values
  model(
    SES = ETS(close ~ error("A") + trend("N") + season("N")),
    Holt = ETS(close ~ error("A") + trend("A") + season("N")),
    Damped = ETS(close ~ error("A") + trend("Ad") +
                   season("N"))
  ) %>% # SES, Holt, Damped described by combo of A's and N's
  forecast(h = "1 year") %>%
  autoplot(TSLA_tsbl, level = NULL) +
  labs(title = "TSLA Ex. Smoothing",
       y = "Price") +
  guides(colour = guide_legend(title = "Forecast"))
models
```



The most appropriate exponential smoothing model here is the Holt forecast, which consists of an additive error and trend element, and is missing the seasonal element. The Holt forecast appears to be a continuation of the recent trend.

3 Build an SARIMA model

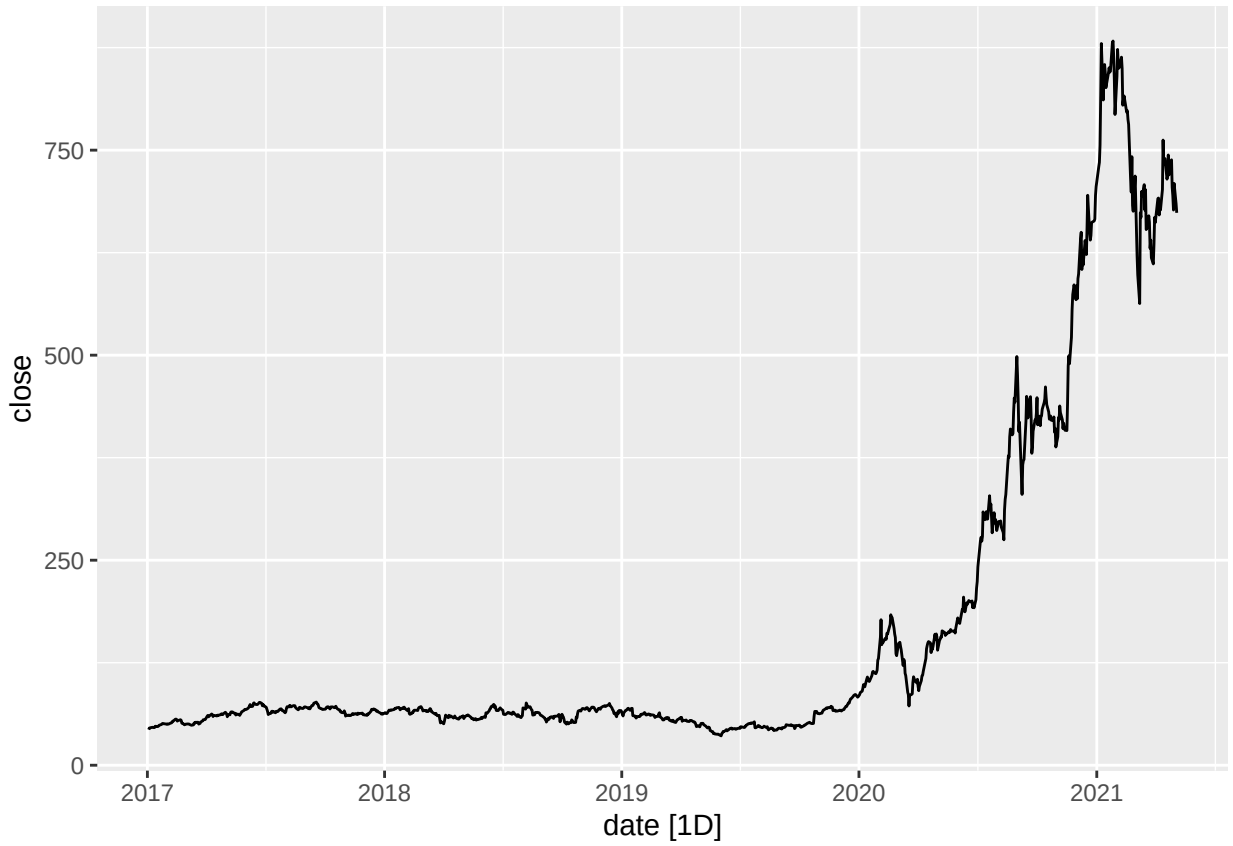
"SARIMA Model:SARIMA(p,d,q)x(P,D,Q,s)

p and seasonal P: indicate number of autoregressive terms (lags of the stationarized series) d and seasonal D: indicate differencing that must be done to stationarize series q and seasonal Q: indicate number of moving average terms (lags of the forecast errors) s: indicates seasonal length in the data"

- From https://towardsdatascience.com/time-series-forecasting-with-a-sarima-model-db051b7ae459

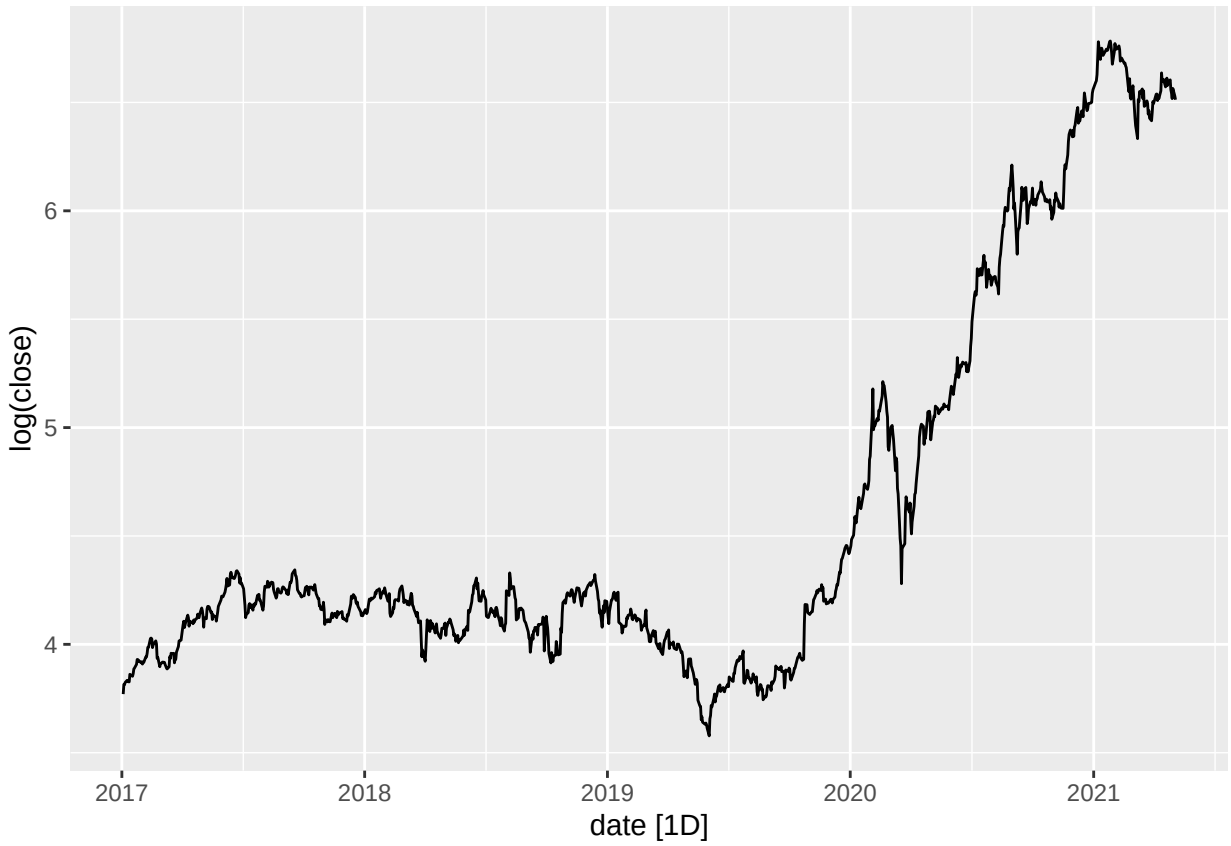
a. Produce an STL decomposition of the data and describe the trend and seasonality.

```
#autoplot without a log function applied.
TSLA_tsbl %>% autoplot(close)
```



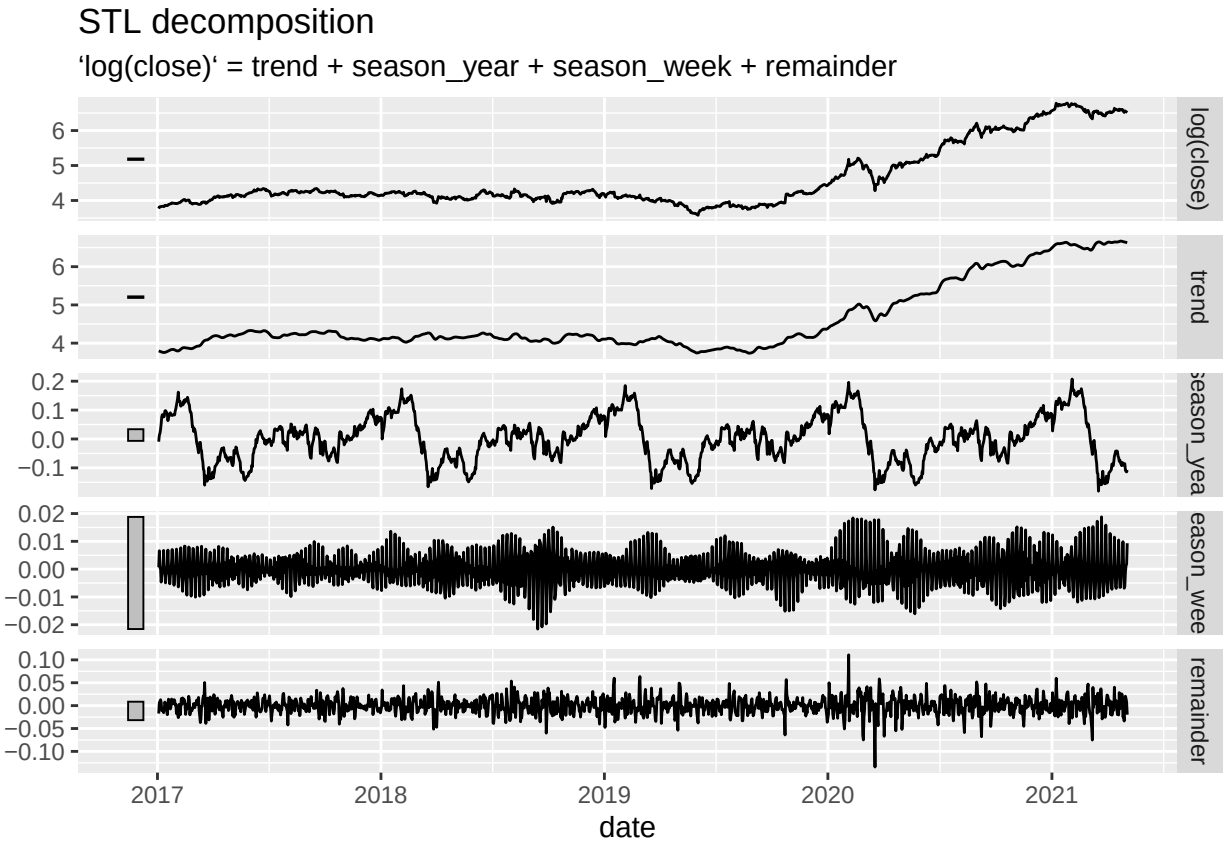

```
#new tsibble with gaps filled.  Using imputeTS's na_interpolation function
TSLA_tsbl2 <-  TSLA_tsbl %>%
  tsibble::fill_gaps() %>%
  na_interpolation()

#Autoplotting the closing prices after being log'ed
TSLA_tsbl2 %>%
  autoplot(log(close))
```



The log doesn't completely take care of the data. We will go ahead and take differences later. In the meantime, lets look at a decomposition of the logged data.

```
#Visual STL decomposition of the closing prices after being log'ed
TSLA_tsbl2 %>%
  model(STL(log(close))) %>%
  components %>%
  autoplot()
```



b) Do the data need transforming? If so, find a suitable transformation.

```
#Creating lambda from data in order to apply the appropriate box_cox transformation
lambda <- TSLA_tsbl2 %>%
  select(close) %>%
  features(close, features = guerrero) %>%
  pull(lambda_guerrero)
lambda
```

```
## [1] -0.1999901
```

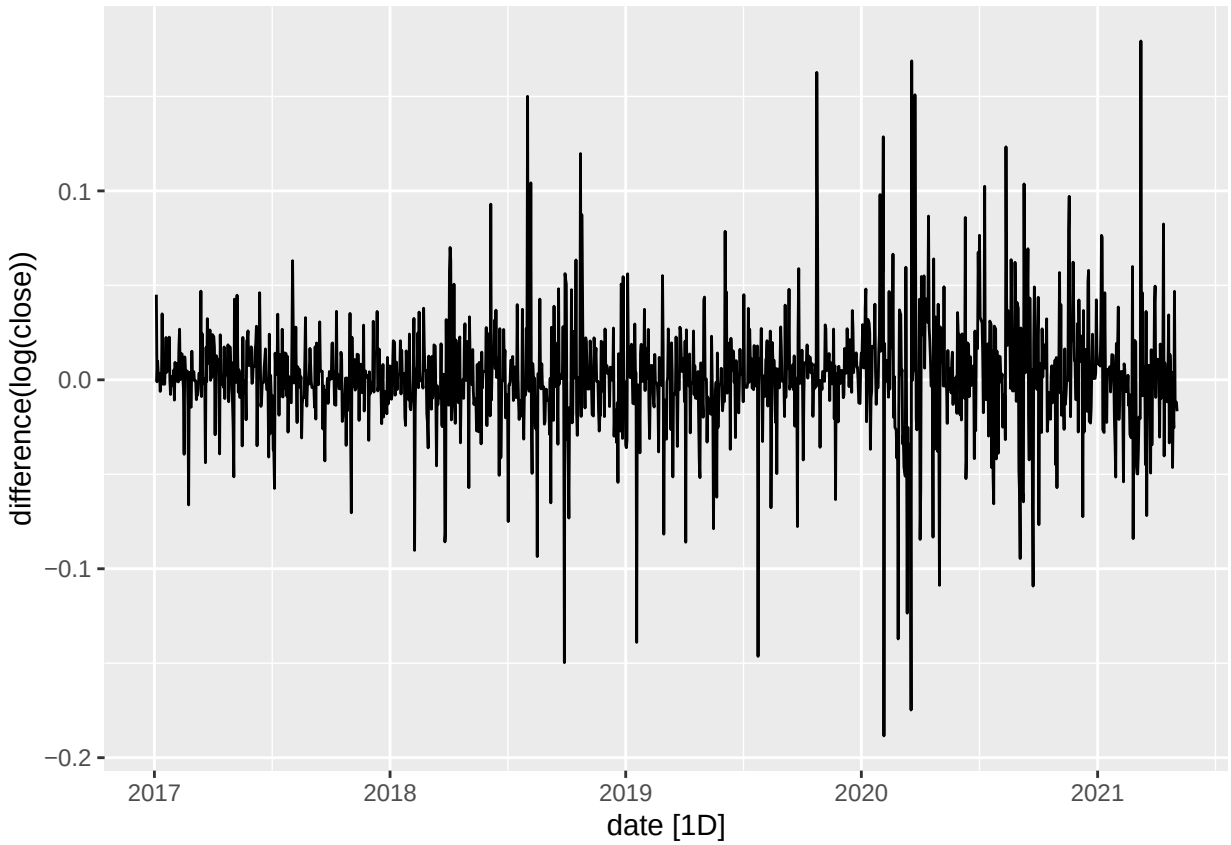
```
#visual of the box_cox transformation
TSLA_tsbl2 %>%
  select(close) %>%
  autoplot(box_cox(close, lambda))
```

It turns out that finding the box-cox is exactly the same as using a log transformation. From now on, for the sake of writing easier code, we will use Log rather than box-cox transformations.

c) Are the data stationary? If not, find an appropriate differencing which yields stationary data.

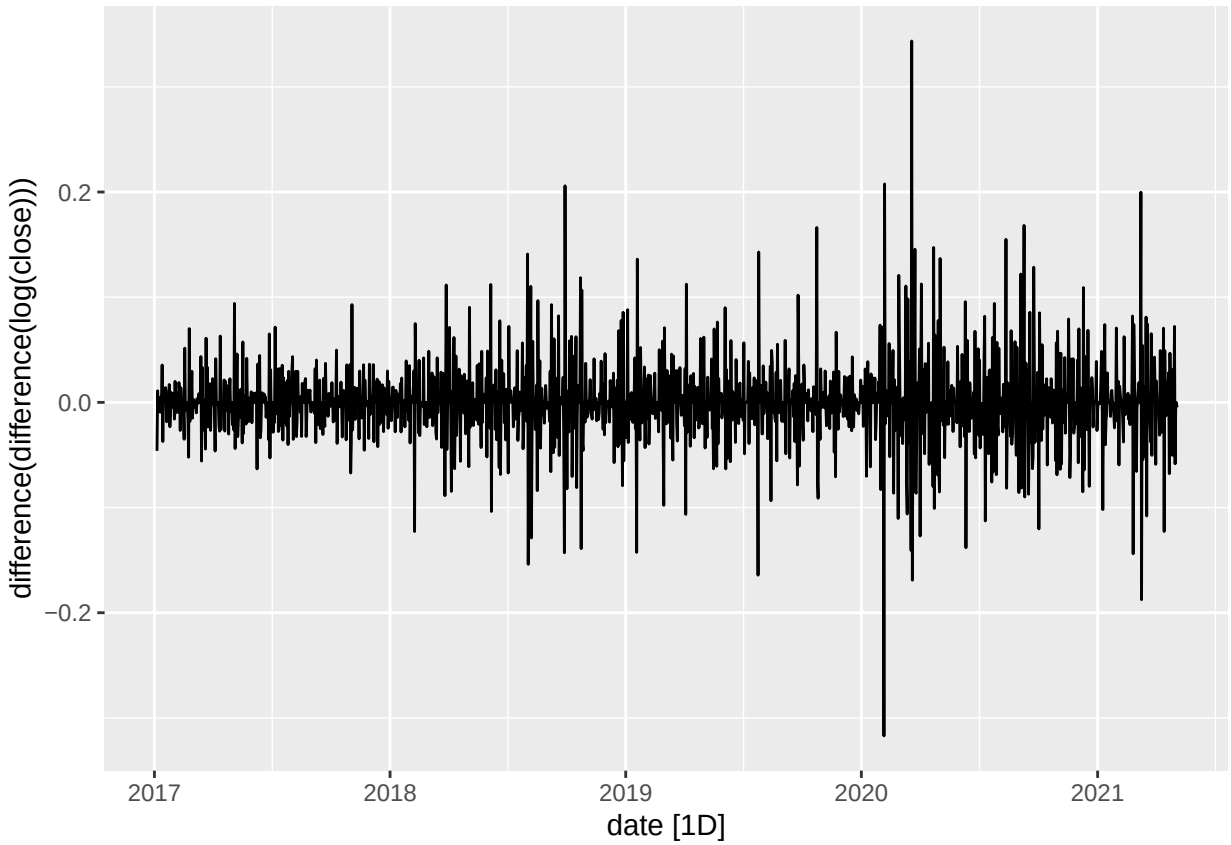
```
#visual of the logged data after a single difference.
TSLA_tsbl2 %>%
  autoplot(difference(log(close)))
```

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




```
#visual of the logged data after two differences.
TSLA_tsbl2 %>%
  autoplot(difference(difference(log(close))))
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```




```
#following codes to figure out the kpss p-values of the differenced log data.
TSLA_tsbl2 %>%
  mutate(diff_close = difference(log(close))) %>%
  features(diff_close, unitroot_kpss)#kpss p-value to determine stationarity.
```

```
## # A tibble: 1 x 3
##   symbol kpss_stat kpss_pvalue
##   <chr>      <dbl>       <dbl>
## 1 TSLA       0.383      0.0843
```

```
TSLA_tsbl2 %>%
  mutate(diff_double_close = difference(difference(log(close)))) %>%
  features(diff_double_close, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   symbol kpss_stat kpss_pvalue
##   <chr>      <dbl>       <dbl>
## 1 TSLA     0.00387         0.1
```

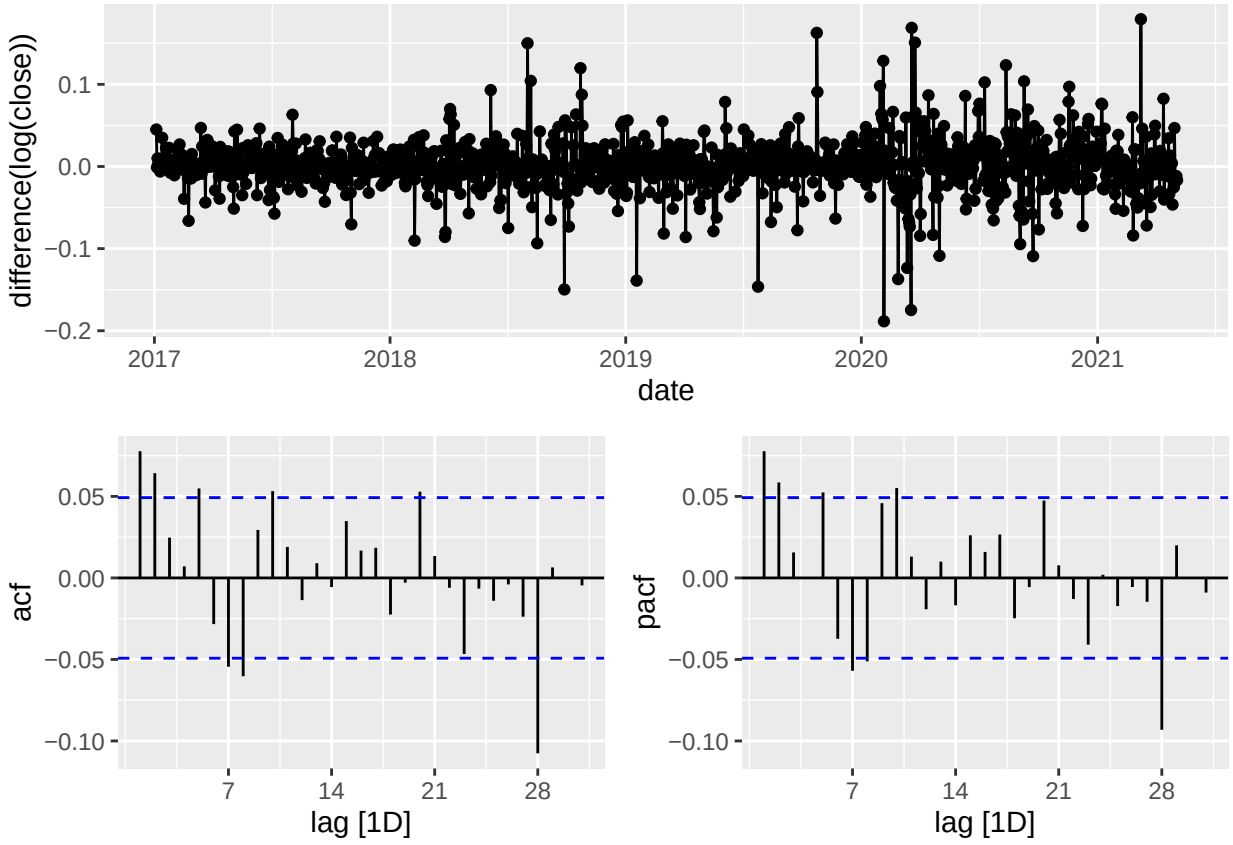
The kpss_pvalue needs to be .1 or greater to be considered stationary. Although the p-value of the single differenced log data is very close (and appears stationary), we want to use the double-differenced log data because it has a p-value of 0.1.

d) Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AICc values?

```
#ACF's, PACF's and Time Series plots of the differenced log data's.
TSLA_tsbl2 %>%
  gg_tsdisplay(difference(log(close)), plot_type="partial")
```

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

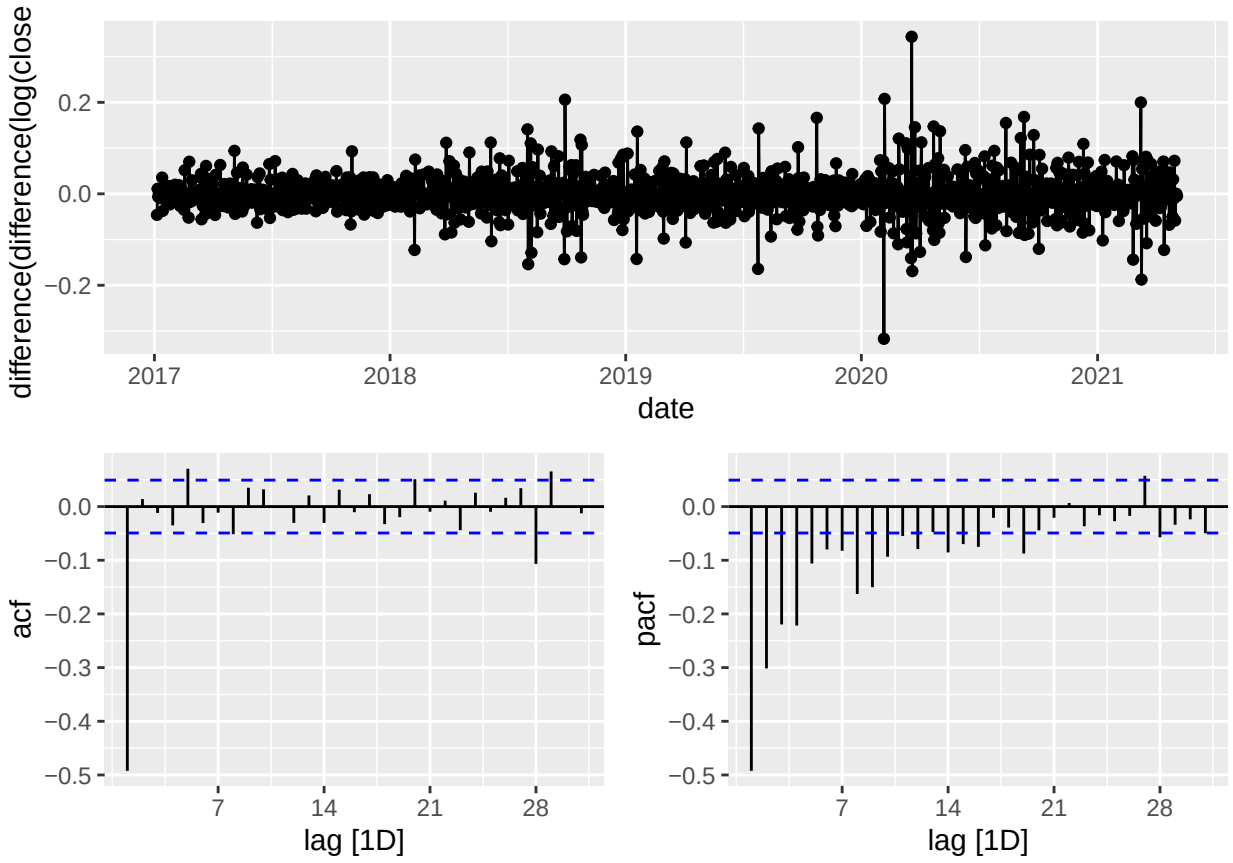
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
TSLA_tsbl2 %>%
  gg_tsdisplay(difference(difference(log(close))), plot_type="partial")
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



Looking at the ACF and PACF of the two different differencing log models, it actually looks like we are better off using the single difference. There is a lag at the seventh spot (which is what we want to see), meaning that there is a single difference in the transformation. We think there will be a zero for the difference in the seasonal part because the second difference made the graphs look worse. Some models to consider: * ARIMA(0,1,1)(0,0,1) See the ACF at lag 7

- ARIMA(0,1,2)(0,0,1) See the ACF at lag 7
- ARIMA(0,1,2)(2,0,1) See the ACF at lag 7
- ARIMA(3,1,0)(2,0,0) See the PACF at lag 7
- ARIMA(3,1,0)(0,0,1) See the PACF at lag 7

```
TSLA_fit <- TSLA_tsbl2 %>%
  model(
    arima011011 = ARIMA(log(close) ~ pdq(0,1,1) + PDQ(0,0,1)),
    arima012011 = ARIMA(log(close) ~ pdq(0,1,2) + PDQ(0,0,1)),
    arima310210 = ARIMA(log(close) ~ pdq(3,1,0) + PDQ(2,0,0)),
    arima310011 = ARIMA(log(close) ~ pdq(3,1,0) + PDQ(0,0,1)),
    arima012210 = ARIMA(log(close) ~ pdq(0,1,2) + PDQ(2,0,0))
  )
#AICc will determine our best model based on my estimates.
glance(TSLA_fit)
```

```
## # A tibble: 5 x 9
##   symbol .model        sigma2 log_lik    AIC   AICc    BIC ar_roots   ma_roots
##   <chr>  <chr>          <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <list>     <list>
## 1 TSLA   arima011011 0.000815   3383. -6758. -6758. -6736. <cpl [0]>  <cpl [8]>
```

```
## 2 TSLA   arima012011 0.000812   3386. -6763. -6763. -6736. <cpl [0]>  <cpl [9]>
## 3 TSLA   arima310210 0.000812   3387. -6760. -6760. -6723. <cpl [17]> <cpl [0]>
## 4 TSLA   arima310011 0.000812   3387. -6762. -6762. -6730. <cpl [3]>  <cpl [7]>
## 5 TSLA   arima012210 0.000812   3386. -6761. -6761. -6729. <cpl [14]> <cpl [2]>
```

The lowest AICC of these models is the ARIMA (0,1,2) (0,1,1) model. Next, we will use the ARIMA function to see if it is, in fact, the best model.

e) Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.

```
#ARIMA function will automatically determine the best ARIMA model of the logged closing price data.
TSLA_fit2 <- TSLA_tsbl2 %>%
  model(
    auto = ARIMA(log(close))
  )
#Stats of best model
glance(TSLA_fit2)
```

```
## # A tibble: 1 x 9
##   symbol .model   sigma2 log_lik    AIC   AICc    BIC ar_roots  ma_roots
##   <chr>  <chr>     <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <list>    <list>
## 1 TSLA   auto   0.000811   3387. -6764. -6764. -6737. <cpl [1]> <cpl [8]>
```

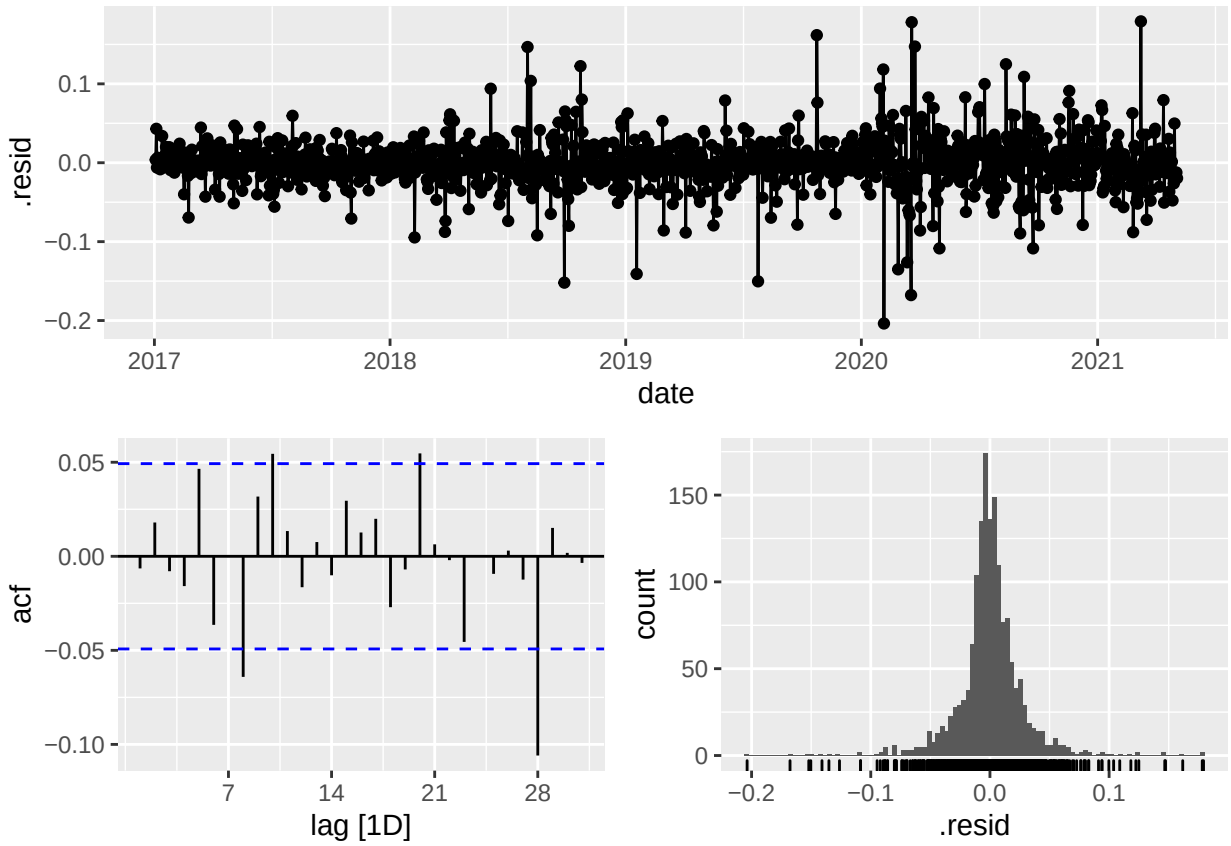
```
#Will reveal the best model
TSLA_fit2 %>% select(auto) %>% report()
```

```
## Series: close
## Model: ARIMA(1,1,1)(0,0,1)[7] w/ drift
## Transformation: log(close)
##
## Coefficients:
##          ar1      ma1     sma1  constant
##       0.6765  -0.6037  -0.0611     6e-04
## s.e.  0.1519   0.1641   0.0269     3e-04
##
## sigma^2 estimated as 0.0008111:  log likelihood=3386.9
## AIC=-6763.8   AICc=-6763.76   BIC=-6736.97
```

The best model is the ARIMA(1,1,1)(0,0,1)[7] model with drift.

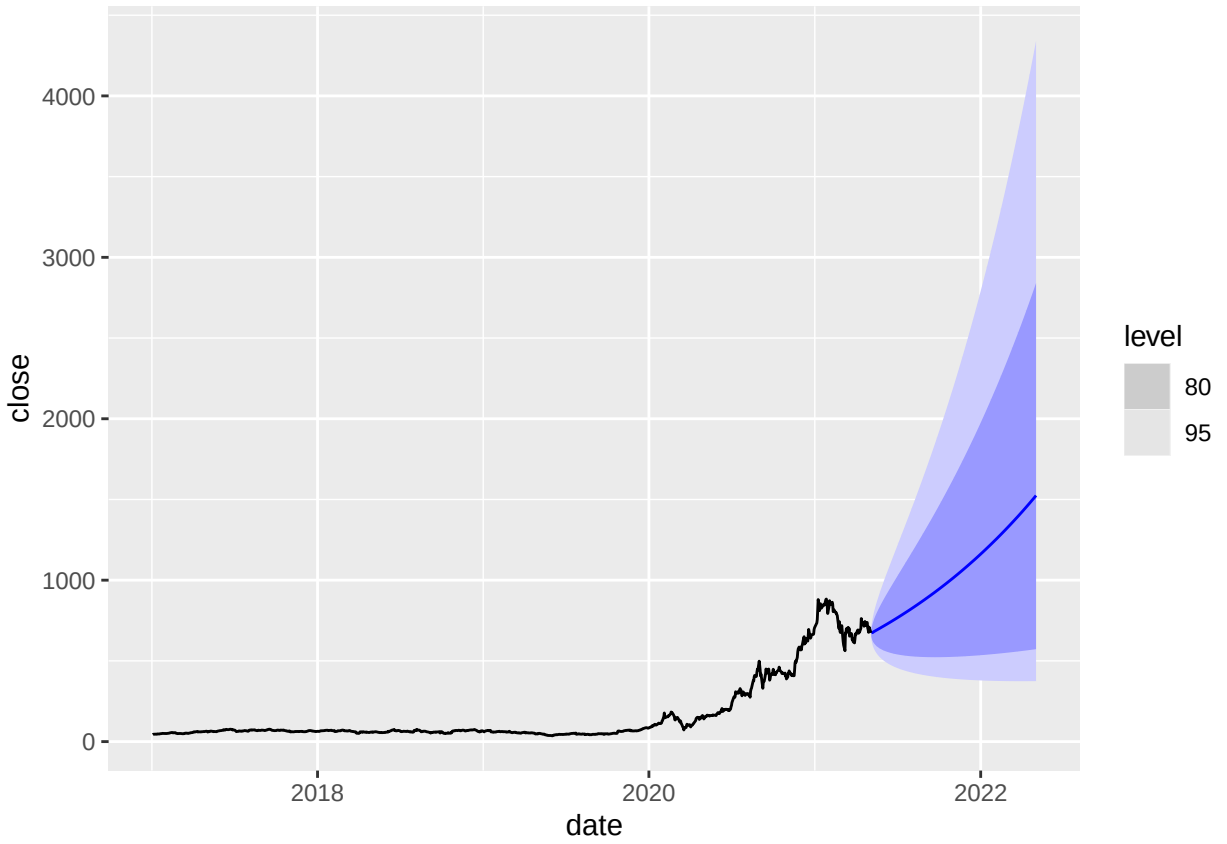
The residuals for this model are normal, only have a few significant langs, and the the pattern of the residuals appear to be very random. This is the best SARIMA model for the closing TSLA stock data.

```
TSLA_fit2 %>%
  select(auto) %>%
  gg_tsresiduals()
```



f) Forecast the next 52 weeks of data.

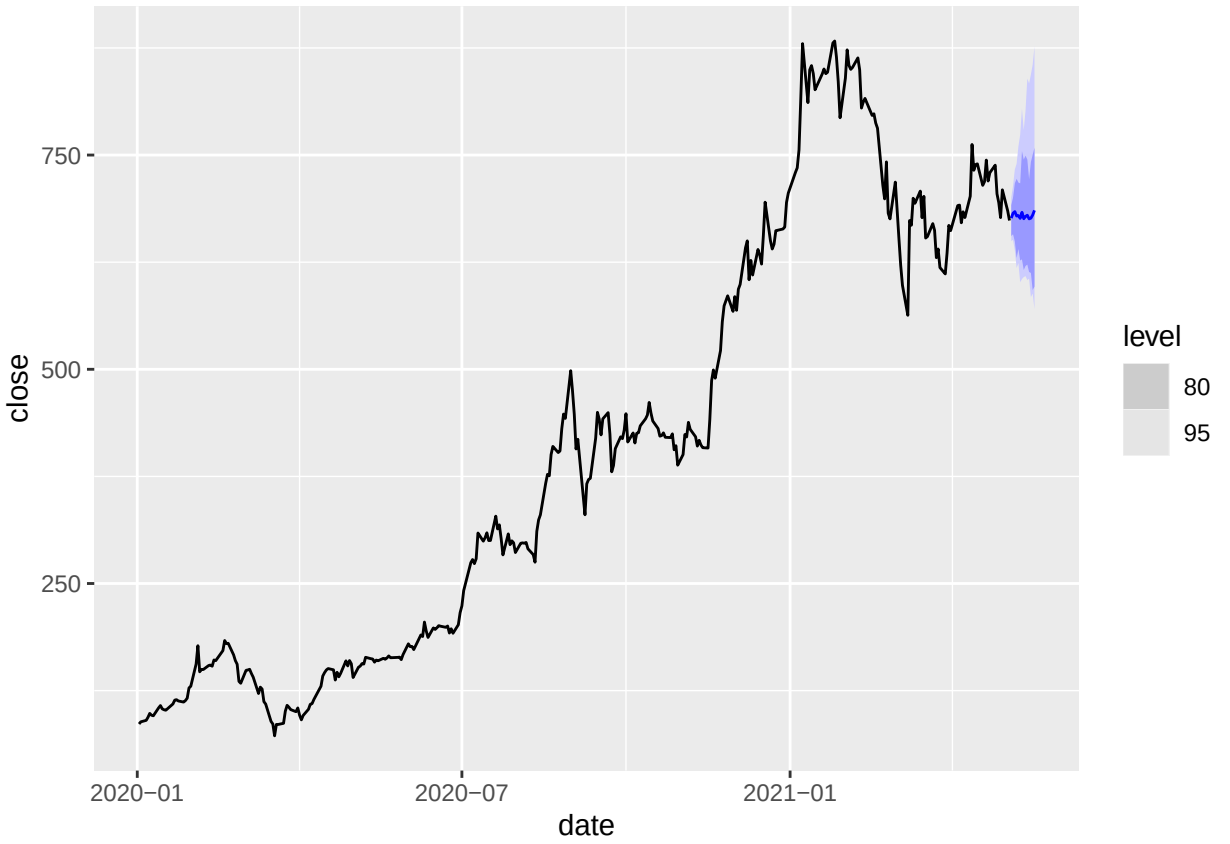
```
#code for a forecast of the next 52 weeks of data
forecast_TSLA <- TSLA_fit2 %>%
  forecast(h = "52 weeks")
#fitting the forecast into the existing tsibble.
forecast_TSLA %>%
  filter(.model=="auto") %>%
  autoplot(TSLA_tsbl2)
```



Forecasting the next year of data spreads a wide confidence interval net. However, this does seem to capture the exponential growth that TSLA has had over the last year, and does a great job projecting that growth into the future. This, of course, is the best modelling we have been able to use so far.

4 Build a neural network.

```
#NNETAR formula has a "single hidden layer and lagged inputs for forecasting univariate time series"
TSLA_tsbl2 %>%
  model(nn = NNETAR(log(close))) %>%
  forecast(times=20) %>%
  autoplot(recent_TSLA)
```



I can't seem to figure out how to extend this forecast over the next year. Every bit of code that I try to extend the forecast over the course of several weeks gets stuck on 'load' for several minutes. I'm not sure if I should kill the code, or let it roll for a while. But what I will provide is the default forecast. This appears to be a working Neural Network, which is defined by the book as:

"This is known as a multilayer feed-forward network, where each layer of nodes receives inputs from the previous layers. The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output."

Tesla - Opening Stock

Using the same data set from before, we are going to use the opening price as opposed to the closing stock. I can only imagine that everything will look awfully similar to the closing stock info... but I shall use a different regression (fitting close on open).

```
TSLA <- tq_get("TSLA", get = "stock.prices", from = "2017-01-01", to = "2021-05-05")
TSLA
```

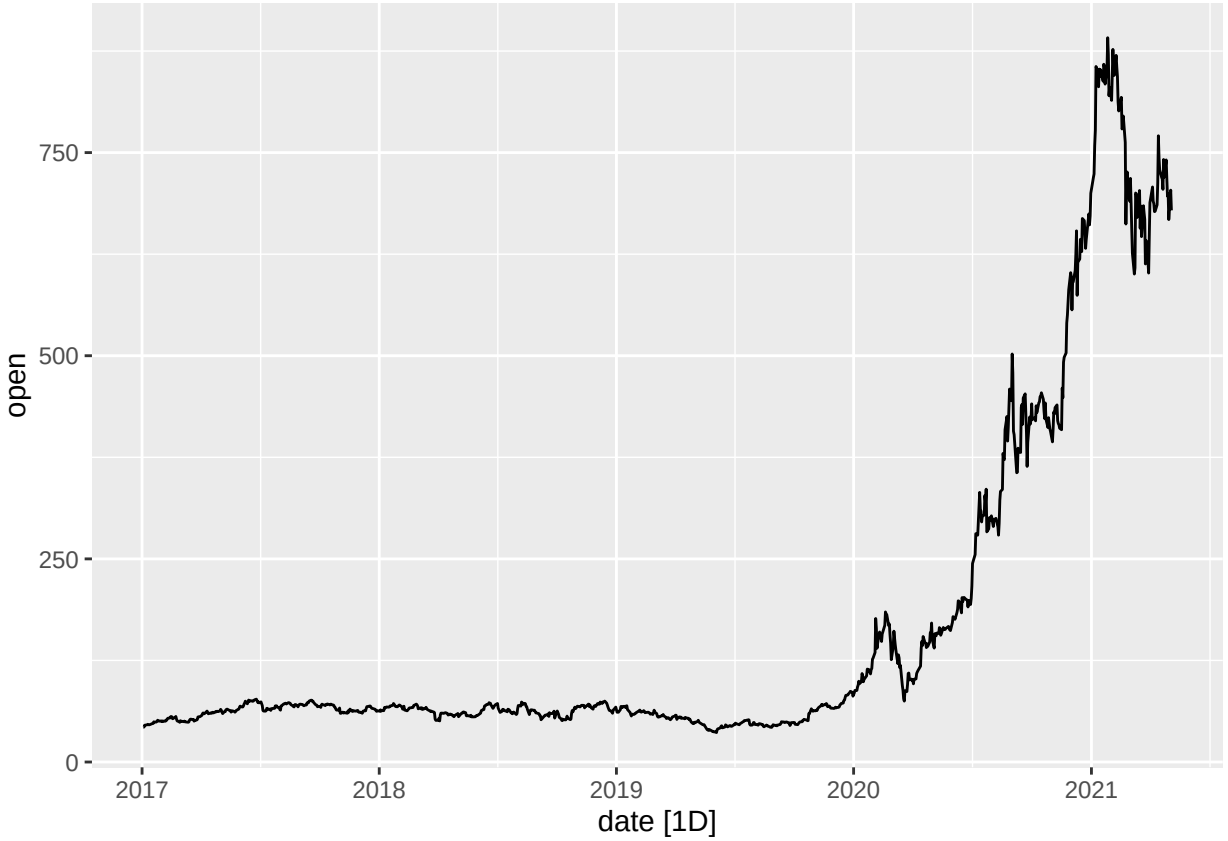
```
## # A tibble: 1,091 x 8
##    symbol date        open  high   low close   volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 TSLA   2017-01-03  43.0  44.1  42.2  43.4 29616500     43.4
##  2 TSLA   2017-01-04  43.0  45.6  42.9  45.4 56067500     45.4
##  3 TSLA   2017-01-05  45.3  45.5  44.4  45.3 29558500     45.3
##  4 TSLA   2017-01-06  45.4  46.1  45.1  45.8 27639500     45.8
##  5 TSLA   2017-01-09  45.8  46.4  45.6  46.3 19897500     46.3
##  6 TSLA   2017-01-10  46.4  46.4  45.4  46.0 18300000     46.0
```

```
##  7 TSLA   2017-01-11  45.8  46.0  45.3  45.9 18254000     45.9
##  8 TSLA   2017-01-12  45.8  46.1  45.1  45.9 18951000     45.9
##  9 TSLA   2017-01-13  46    47.6  45.9  47.5 30465000     47.5
## 10 TSLA   2017-01-17  47.3  48.0  46.9  47.1 23087500     47.1
## # ... with 1,081 more rows
```

```
#added an 'o' at the end of the variable name to indicate 'open'
TSLA_tsbl_o <- as_tsibble(TSLA, key = symbol)
```

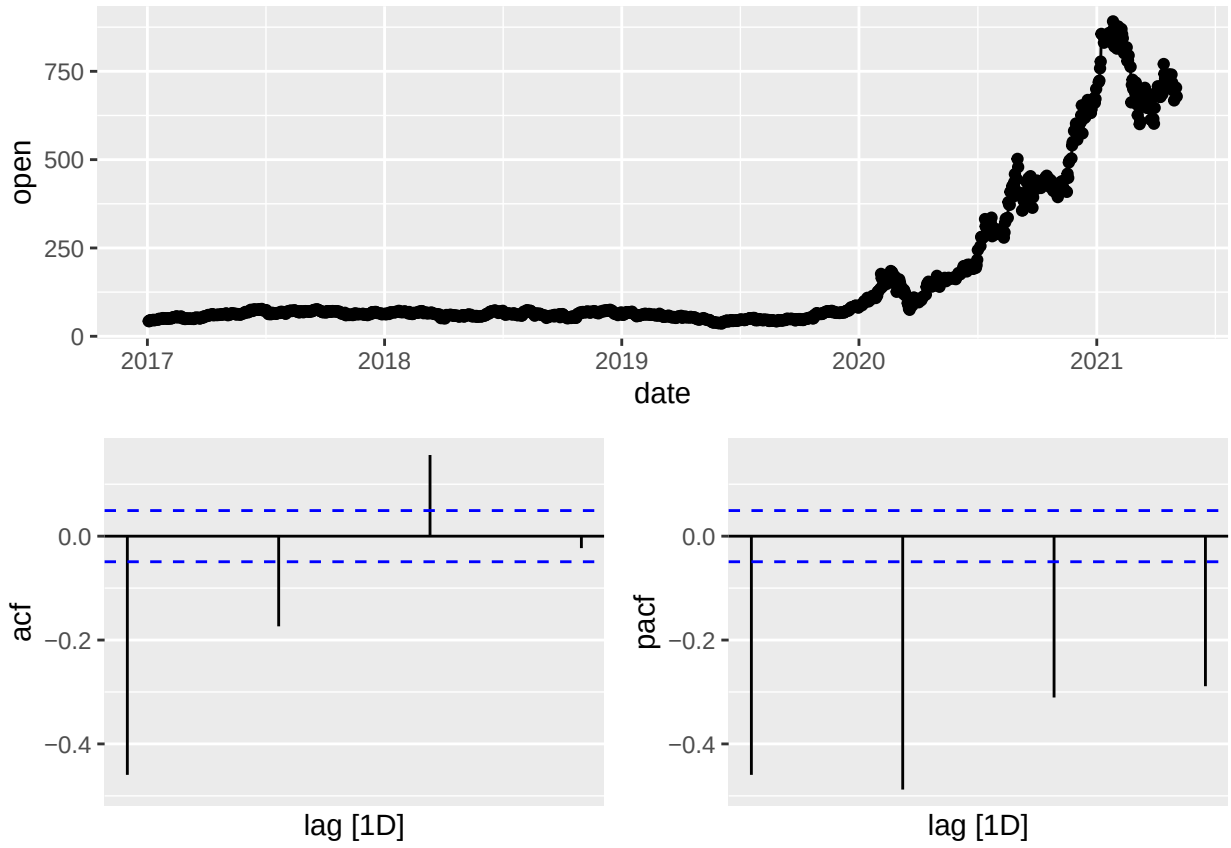
```
## Using `date` as index variable.
```

```
TSLA_tsbl_o %>%
  autoplot(open)
```

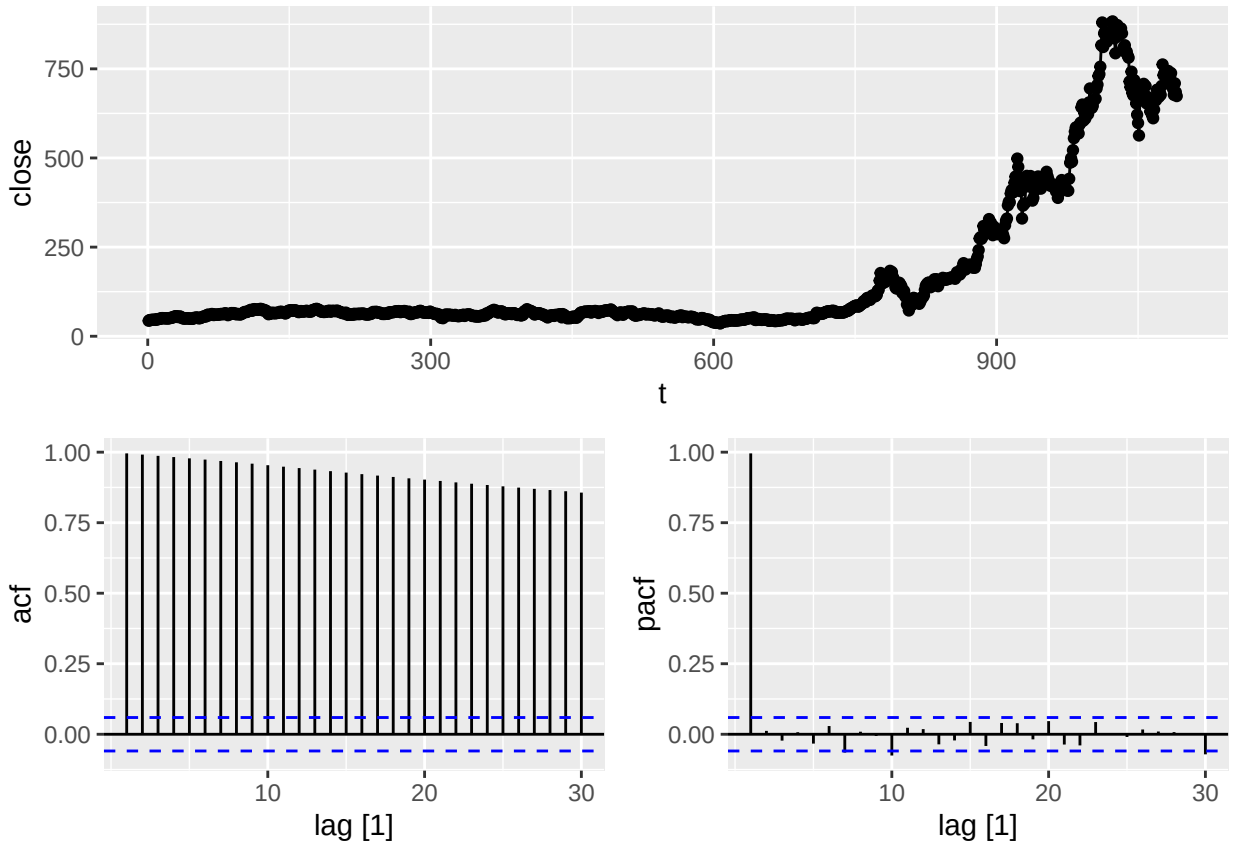



```
TSLA_tsbl_o %>%
  fill_gaps() %>%
  gg_tsdisplay(open, plot_type = "partial")
```

```
## Warning: Removed 492 rows containing missing values (geom_point).
```



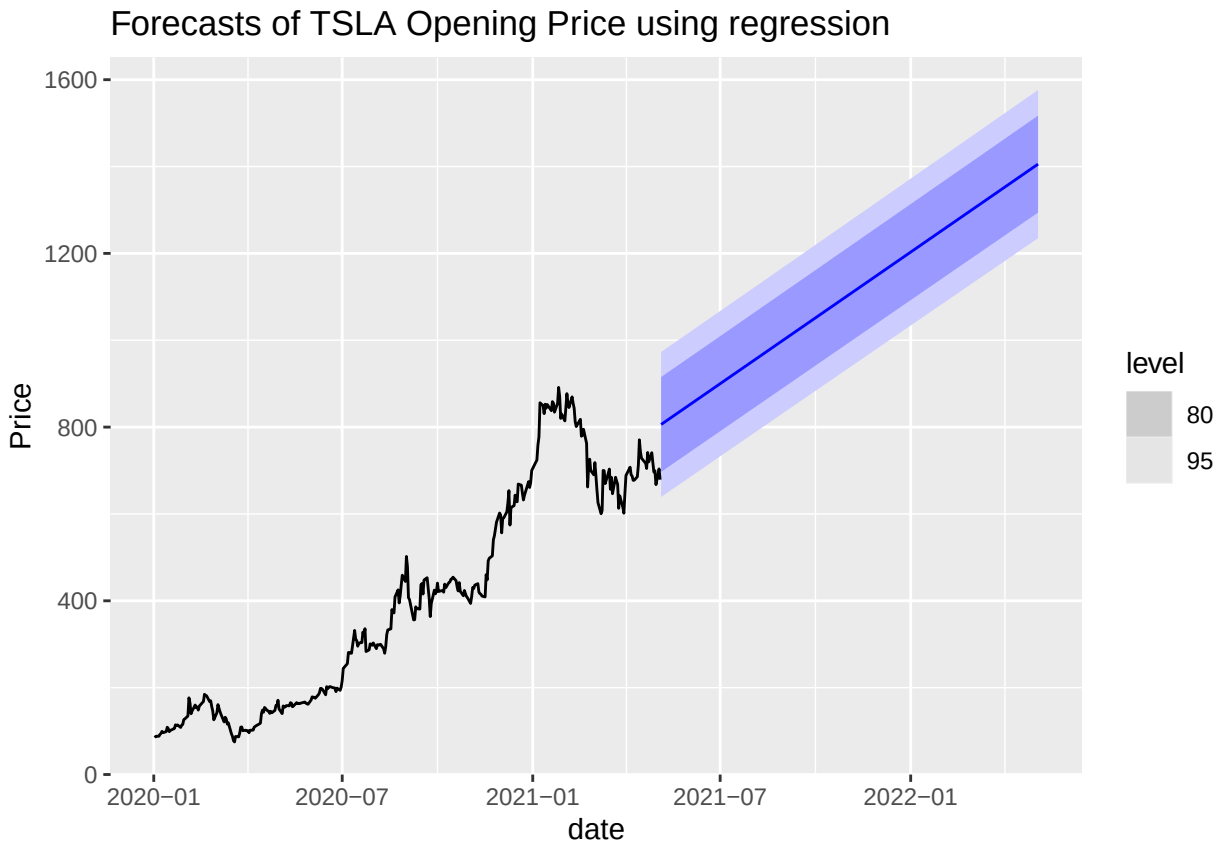

```
TSLA_tsbl_o %>%
  mutate(t = row_number()) %>%
  update_tsibble(index = t) %>%
  gg_tsdisplay(close, plot_type = "partial")
```



The residuals of the data for an autoplot (without any changes) looks almost exactly like the residuals for Close.

1. Build a linear regression model using TSLM().

```
recent_TSLA_o <- TSLA_tsbl_o %>%
  filter(year(date) >= 2020)
fit_TSLA_o <- recent_TSLA_o %>%
  model(TSLM(open ~ trend()))
fc_TSLA_o <- forecast(fit_TSLA_o, h = "12 months")
fc_TSLA_o %>%
  autoplot(recent_TSLA_o) +
  labs(
    title = "Forecasts of TSLA Opening Price using regression",
    y = "Price"
  )
```




```
# Model to determine regression summary
recent_TSLA_o %>%
  model(TSLM(open ~ trend())) %>%
  report()
```

```
## Series: open
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -145.05  -60.77  -29.56   62.61  248.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.96381    9.23766  -0.104    0.917
## trend()      1.64705    0.03272  50.342   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 84.64 on 335 degrees of freedom
## Multiple R-squared: 0.8832,  Adjusted R-squared: 0.8829
## F-statistic:  2534 on 1 and 335 DF, p-value: < 2.22e-16
```

(FOR CLOSING) Regression line for the TSLM model since 2020 is: close = -0.13 + 1.645(day). (FOR OPENING): Regression line for the TSLM model since 2020 is: close = -0.96 + 1.647(day) This means that since 2020, the opening price of Tesla has grown approxamately $1.65 a trading day. The regression line has the opening price grow by 2 tenths of cent higher than that of the closing price. I guess that maybe means that its better to sell your stock right before close, but it doesn't really matter too much unless you

are trading a ton of TSLA stock.

```
TSLA_tsbl_o %>%
  ggplot(aes(x = open, y = close)) +
  labs(y = "Closing Price",
       x = "Opening Price of Tesla Stock") +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```


```
TSLA_tsbl_o %>%
  model(TSLM(close ~ open)) %>%
  report()
```

```
## Series: close
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -49.0012  -1.3458  -0.1295   1.0230  65.5786
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.224944   0.357268    0.63    0.529
## open        0.999336   0.001338  747.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

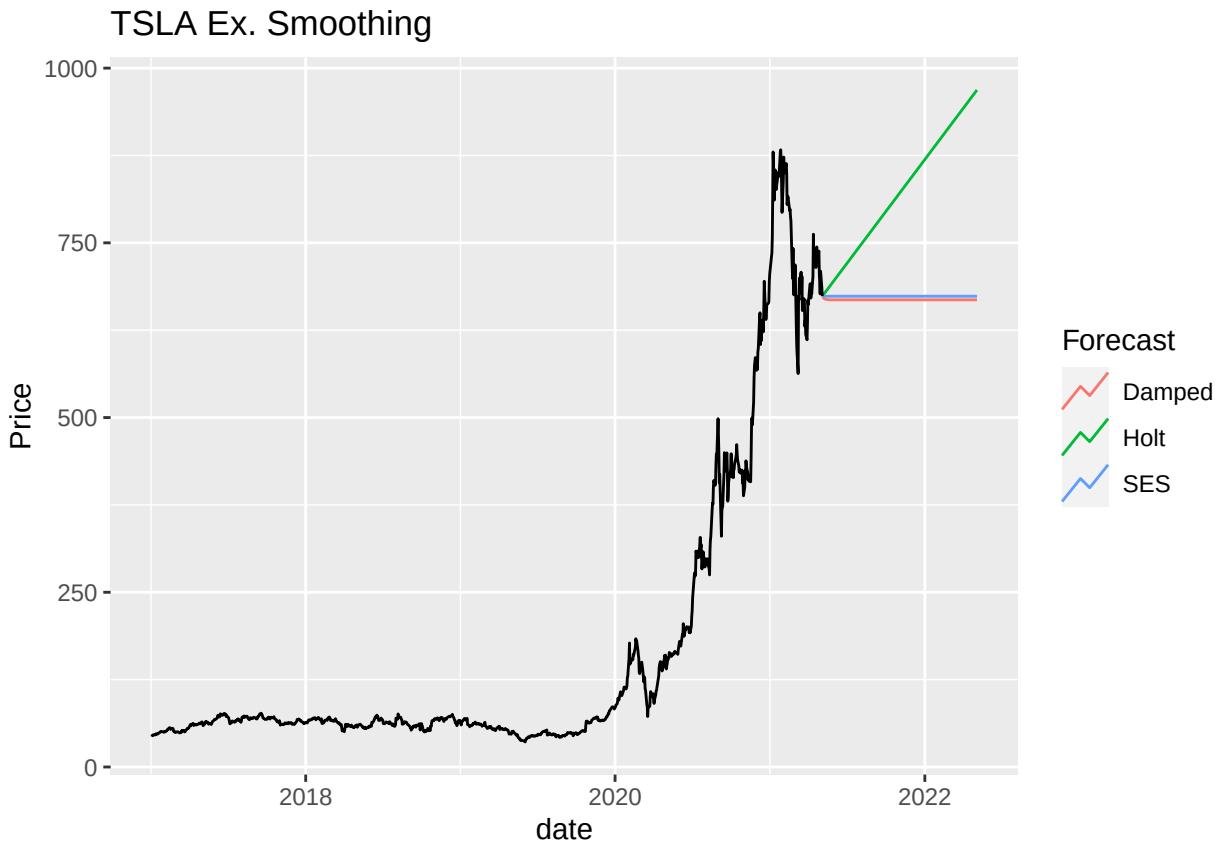
```
##
## Residual standard error: 9.25 on 1089 degrees of freedom
## Multiple R-squared: 0.9981,  Adjusted R-squared: 0.9981
## F-statistic: 5.581e+05 on 1 and 1089 DF, p-value: < 2.22e-16
```

The coefficient for open is 0.999. This shows how correlated the closing and opening prices for the Tesla stock are. This makes sense... in a data set of such a long span of time, the differences in stock price on any given day is going to look like chump change. This proves that investing is a long-term game. Invest now, and look smart down the line. For those who love to gamble, play the day trading game. But this data doesn't help those types of people.

##2 Build an appropriate exponential smoothing model.

```
library(imputeTS)
```

```
models_o <- TSLA_tsbl_o %>%
  tsibble::fill_gaps() %>%
  na_interpolation() %>%
  model(
    SES = ETS(open ~ error("A") + trend("N") + season("N")),
    Holt = ETS(open ~ error("A") + trend("A") + season("N")),
    Damped = ETS(open ~ error("A") + trend("Ad") +
                   season("N")),
  ) %>%
  forecast(h = "1 year") %>%
  autoplot(TSLA_tsbl_o, level = NULL) +
  labs(title = "TSLA Ex. Smoothing",
       y = "Price") +
  guides(colour = guide_legend(title = "Forecast"))
models
```

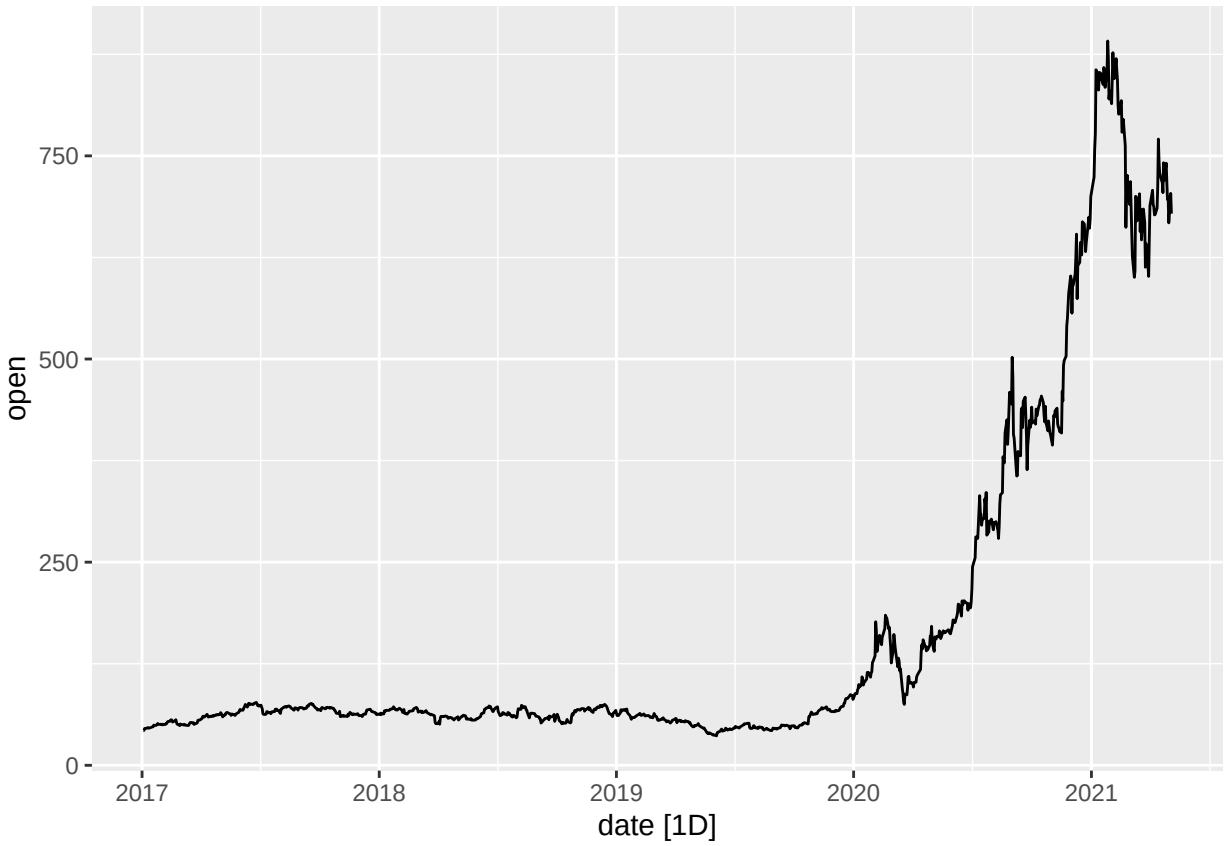


Like the closing price data, the Holt model is easily the best forecasting exponential model to use, and its not even close.

3 Build an SARIMA model

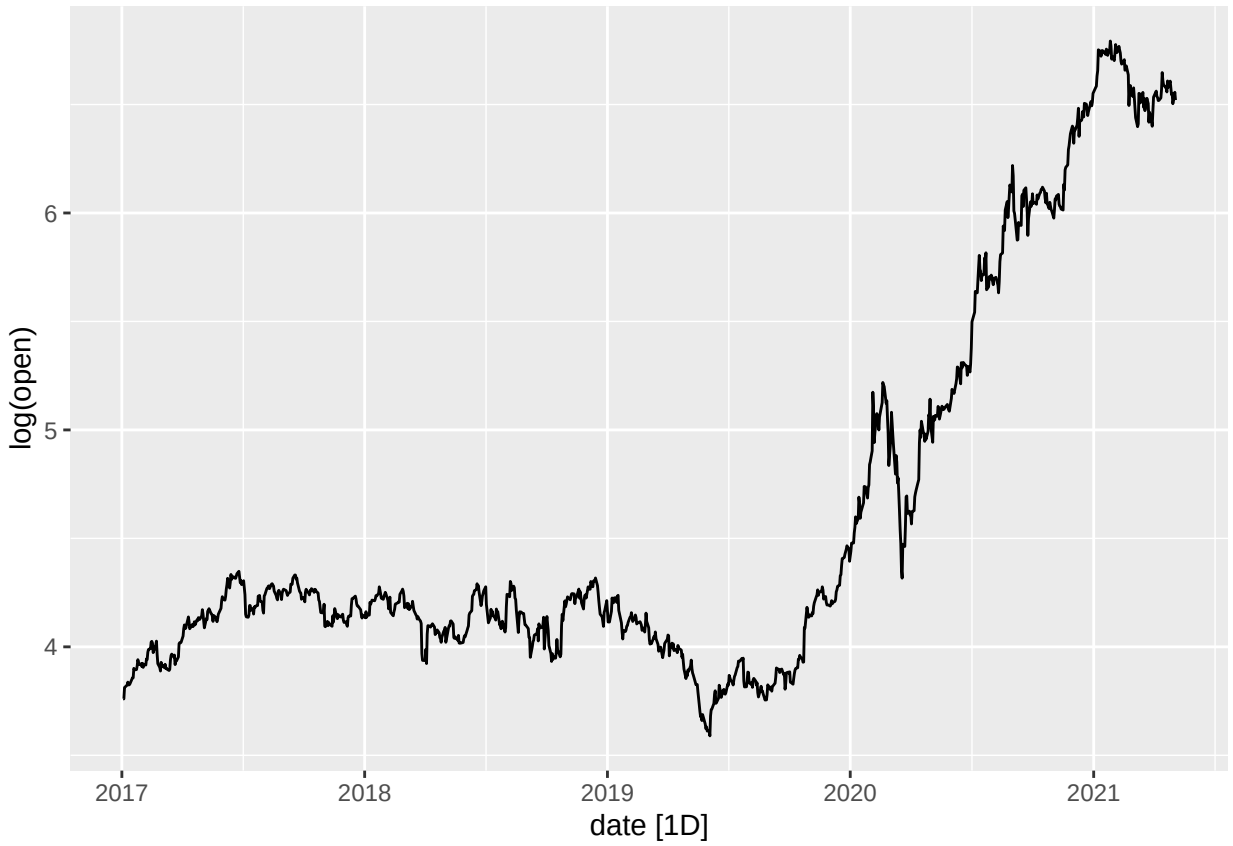
a. Produce an STL decomposition of the data and describe the trend and seasonality.

```
TSLA_tsbl_o %>% autoplot(open)
```



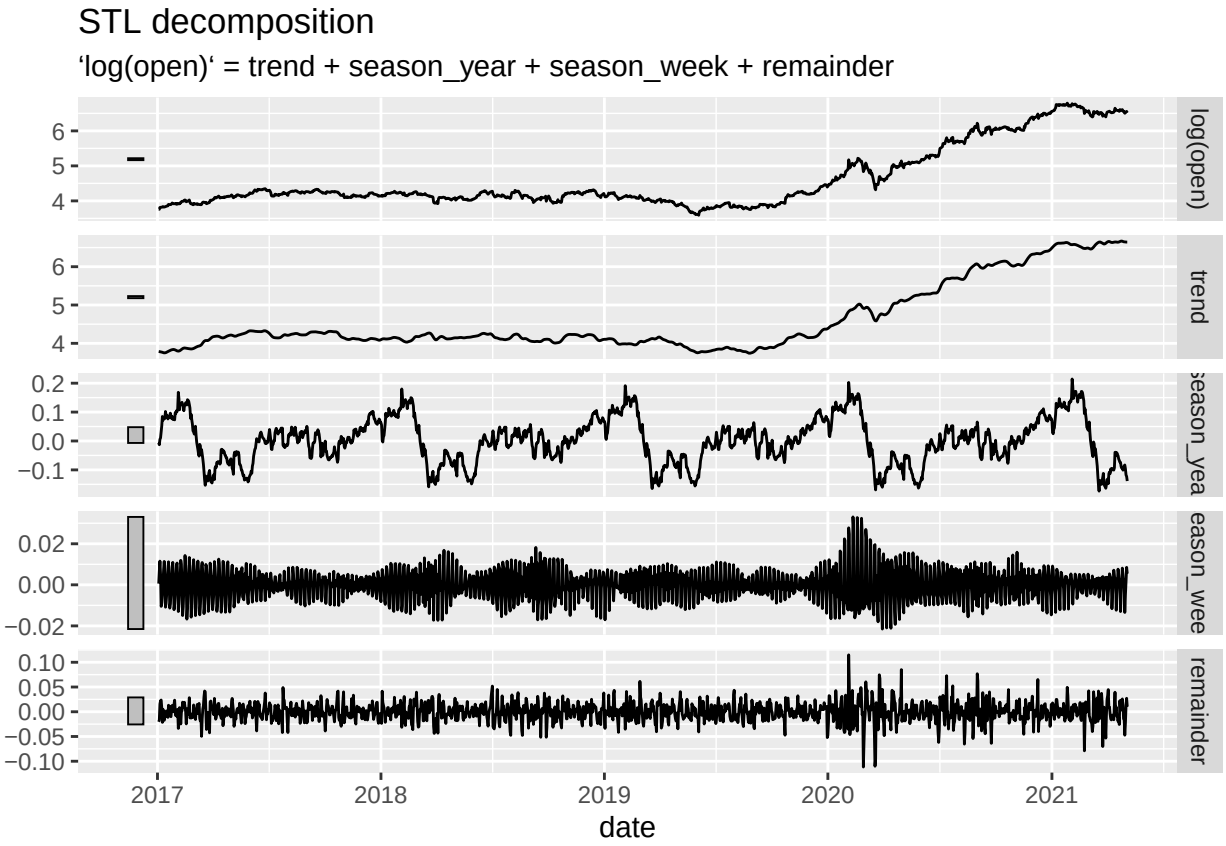

```
TSLA_tsbl2_o <-  TSLA_tsbl_o %>%
  tsibble::fill_gaps() %>%
  na_interpolation()

TSLA_tsbl2_o %>%
  autoplot(log(open))
```



The log doesn't completely take care of the data. We will go ahead and take differences later. In the meantime, lets look at a decomposition of the logged data.

```
TSLA_tsbl2_o %>%
  model(STL(log(open))) %>%
  components %>%
  autoplot()
```



b) Do the data need transforming? If so, find a suitable transformation.

```
lambda <- TSLA_tsbl2_o %>%
  select(open) %>%
  features(open, features = guerrero) %>%
  pull(lambda_guerrero)
lambda
```

```
## [1] -0.2158834
```

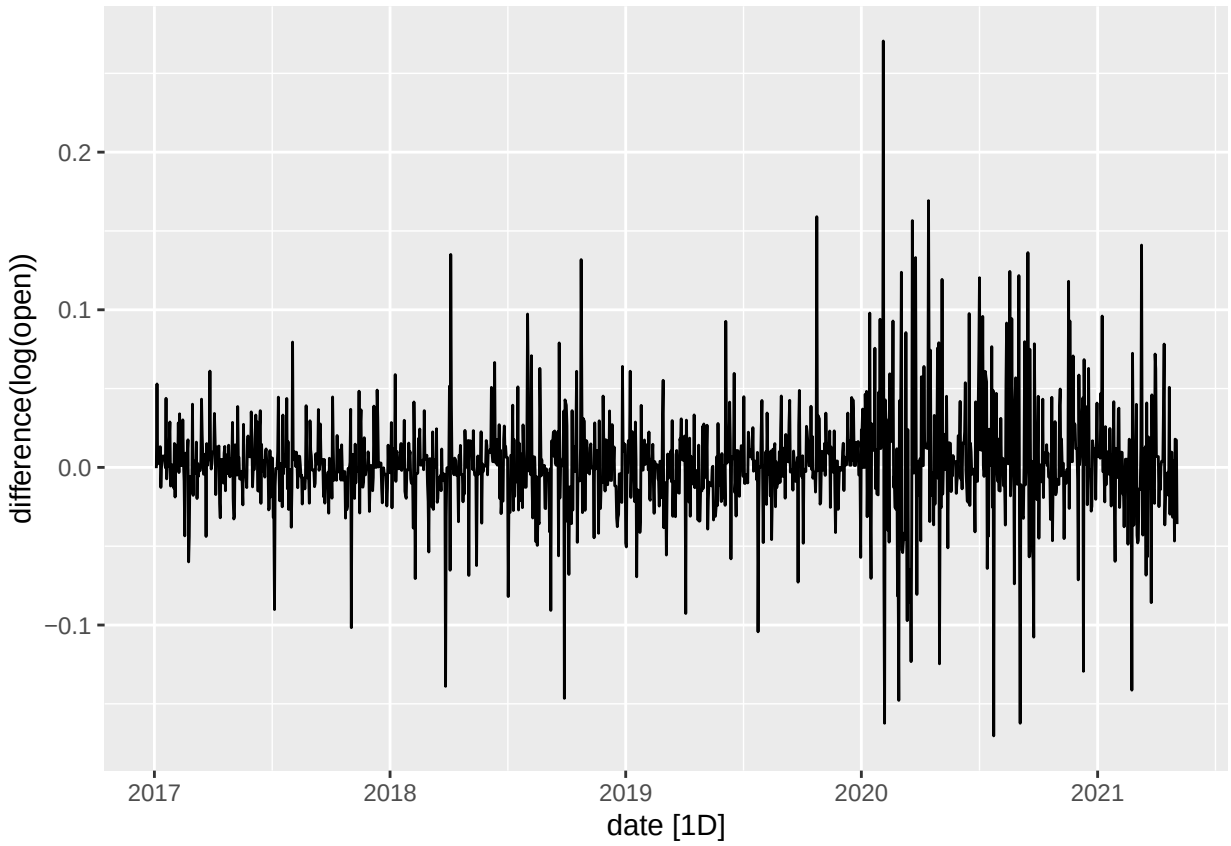
```
TSLA_tsbl2 %>%
  select(open) %>%
  autoplot(box_cox(open, lambda))
```

It turns out that finding the box-cox is exactly the same as using a log transformation. From now on, for the sake of writing easier code, we will use Log rather than box-cox transformations. The lambda for the opening data is -0.25, while the lambda for the closing data was -.20.

c) Are the data stationary? If not, find an appropriate differencing which yields stationary data.

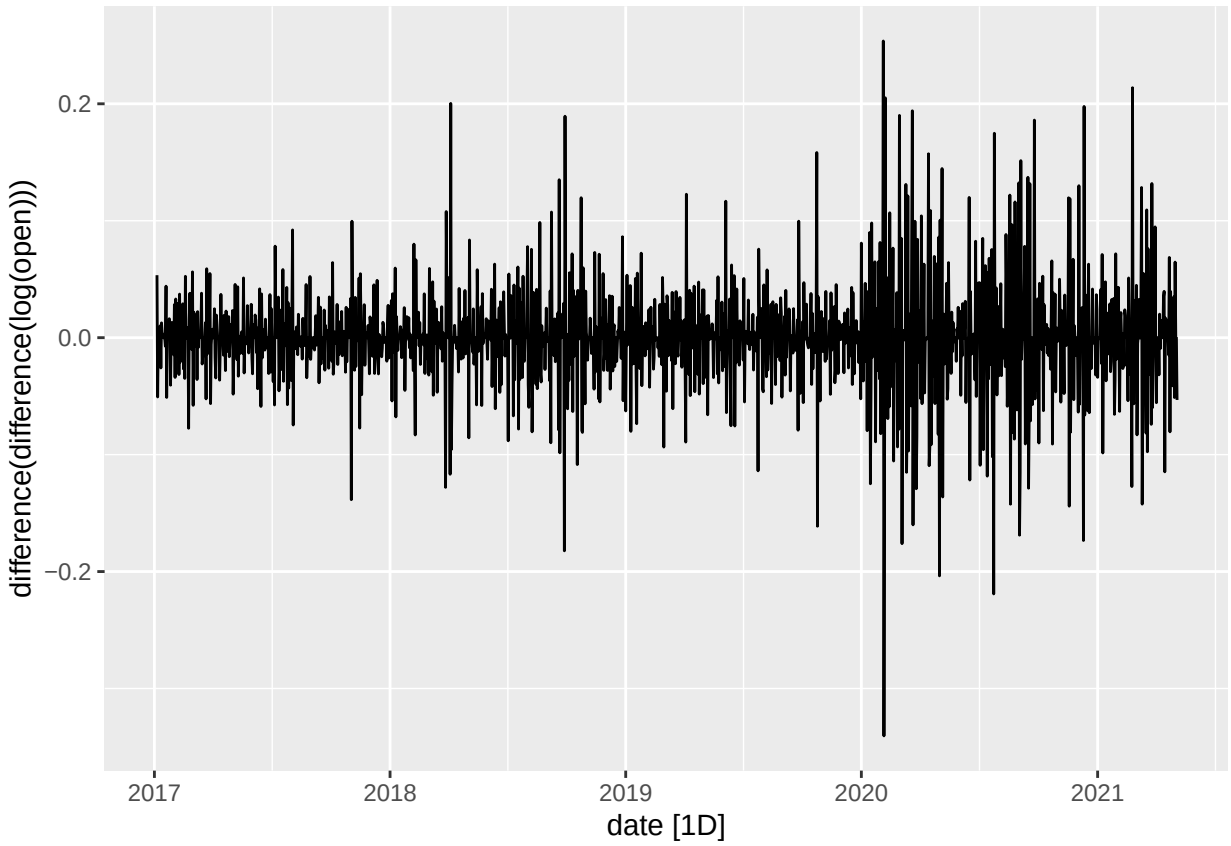
```
TSLA_tsbl2_o %>%
  autoplot(difference(log(open)))
```

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




```
TSLA_tsbl2_o %>%
  autoplot(difference(difference(log(open))))
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```




```
TSLA_tsbl2_o %>%
  mutate(diff_open = difference(log(open))) %>%
  features(diff_open, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   symbol kpss_stat kpss_pvalue
##   <chr>      <dbl>       <dbl>
## 1 TSLA       0.396      0.0787
```

```
TSLA_tsbl2_o %>%
  mutate(diff_double_open = difference(difference(log(open)))) %>%
  features(diff_double_open, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   symbol kpss_stat kpss_pvalue
##   <chr>      <dbl>       <dbl>
## 1 TSLA     0.00388         0.1
```

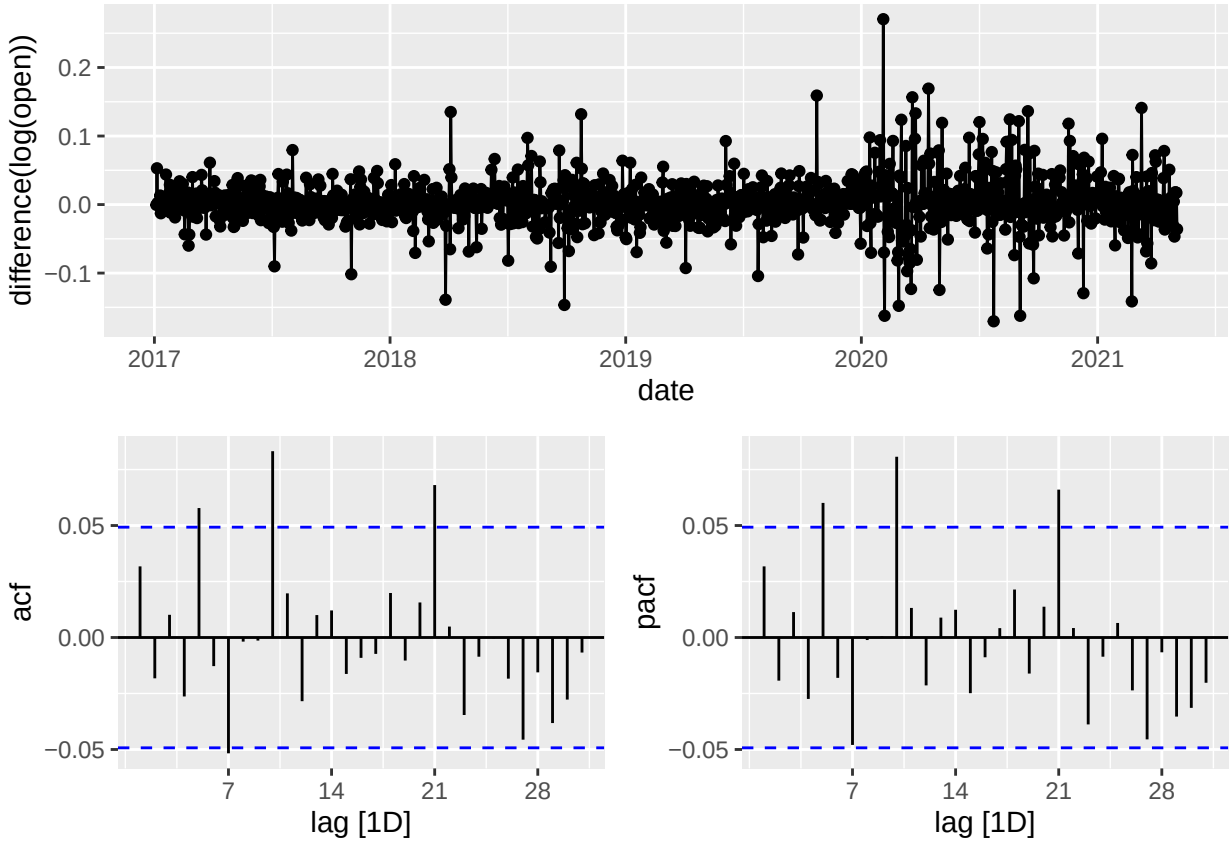
The kpss_pvalue needs to be .1 or greater to be considered stationary. Although the p-value of the single differenced log data is very close (and appears stationary), we want to use the double-differenced log data because it has a p-value of 0.1. HOWEVER, in the last dataset, this logic backfired. I feel like maybe a p-value of under 1 might actually be the better move. We will find out shortly if this is true again for the opening stock data.

d) Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AICc values?

```
TSLA_tsbl2_o %>%
  gg_tsdisplay(difference(log(open)), plot_type="partial")
```

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

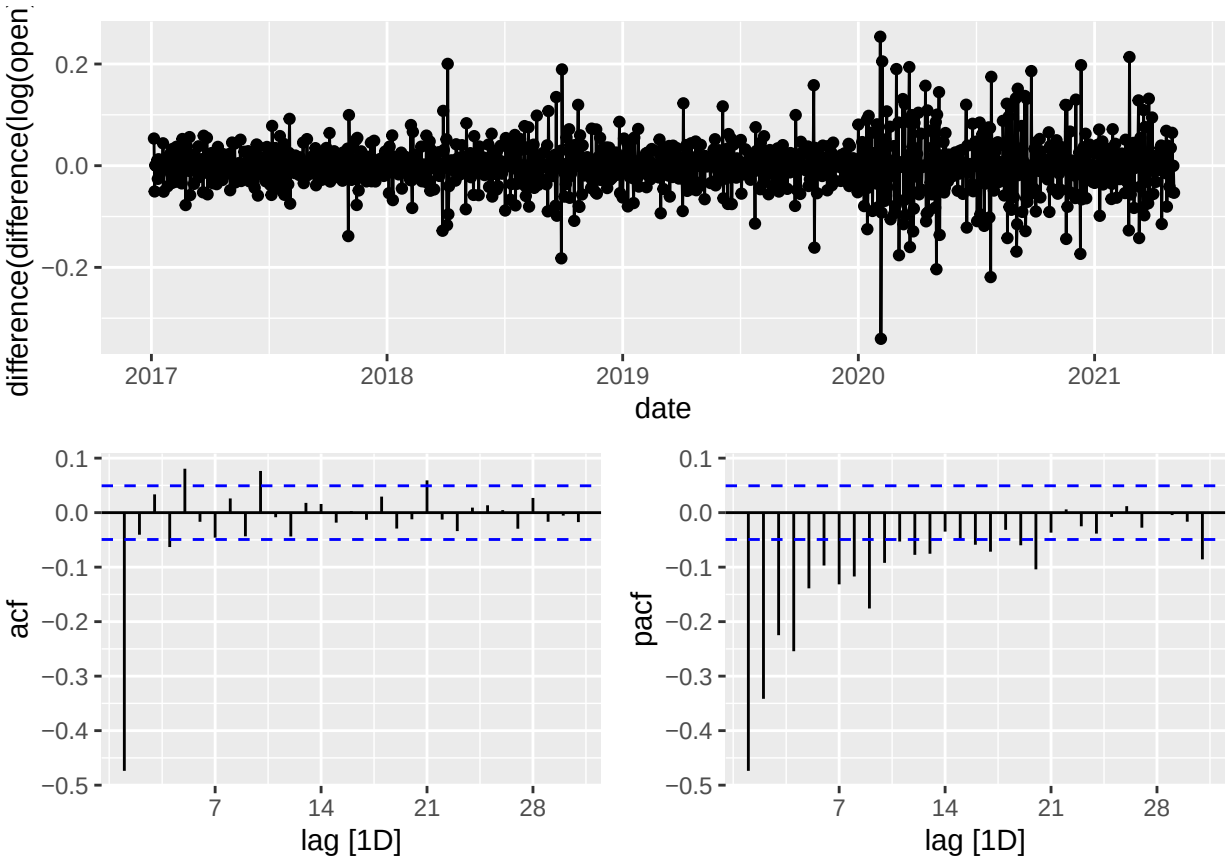
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
TSLA_tsbl2_o %>%
  gg_tsdisplay(difference(difference(log(open))), plot_type="partial")
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



We will once again use the single difference data as our reference here. Just like before, it appears that the single difference data fits much better. We will once again try some ARIMA models, the same as up above, and see how those fit. Then, we will later use the ARIMA function to come up with the actual best model.

- ARIMA(0,1,1)(0,0,1) See the ACF at lag 7
- ARIMA(0,1,2)(0,0,1) See the ACF at lag 7
- ARIMA(0,1,2)(2,0,0) See the ACF at lag 7
- ARIMA(3,1,0)(2,0,0) See the PACF at lag 7
- ARIMA(3,1,0)(0,0,1) See the PACF at lag 7

```
TSLA_fit <- TSLA_tsbl2 %>%
  model(
    arima011011 = ARIMA(log(close) ~ pdq(0,1,1) + PDQ(0,0,1)),
    arima012011 = ARIMA(log(close) ~ pdq(0,1,2) + PDQ(0,0,1)),
    arima310200 = ARIMA(log(close) ~ pdq(3,1,0) + PDQ(2,0,0)),
    arima310001 = ARIMA(log(close) ~ pdq(3,1,0) + PDQ(0,0,1)),
    arima012200 = ARIMA(log(close) ~ pdq(0,1,2) + PDQ(2,0,0))
  )
glance(TSLA_fit)
```

```
## # A tibble: 5 x 9
##   symbol .model        sigma2 log_lik    AIC   AICc    BIC ar_roots   ma_roots
##   <chr>  <chr>          <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <list>     <list>
## 1 TSLA   arima011011 0.000815   3383. -6758. -6758. -6736. <cpl [0]>  <cpl [8]>
## 2 TSLA   arima012011 0.000812   3386. -6763. -6763. -6736. <cpl [0]>  <cpl [9]>
## 3 TSLA   arima310200 0.000812   3387. -6760. -6760. -6723. <cpl [17]> <cpl [0]>
```

```
## 4 TSLA   arima310001 0.000812   3387. -6762. -6762. -6730. <cpl [3]>  <cpl [7]>
## 5 TSLA   arima012200 0.000812   3386. -6761. -6761. -6729. <cpl [14]> <cpl [2]>
```

The lowest AICC of these models is the ARIMA (0,1,2) (0,1,1) model, which has a different non-seasonal part of the ARIMA. Next, we will use the ARIMA function to see if it is, in fact, the best model.

5) Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.

```
TSLA_fit2_o <- TSLA_tsbl2_o %>%
  model(
    auto = ARIMA(log(open))
  )

glance(TSLA_fit2_o)
```

```
## # A tibble: 1 x 9
##   symbol .model   sigma2 log_lik    AIC   AICc    BIC ar_roots  ma_roots
##   <chr>  <chr>     <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <list>    <list>
## 1 TSLA   auto   0.000981   3236. -6466. -6466. -6449. <cpl [0]> <cpl [7]>
```

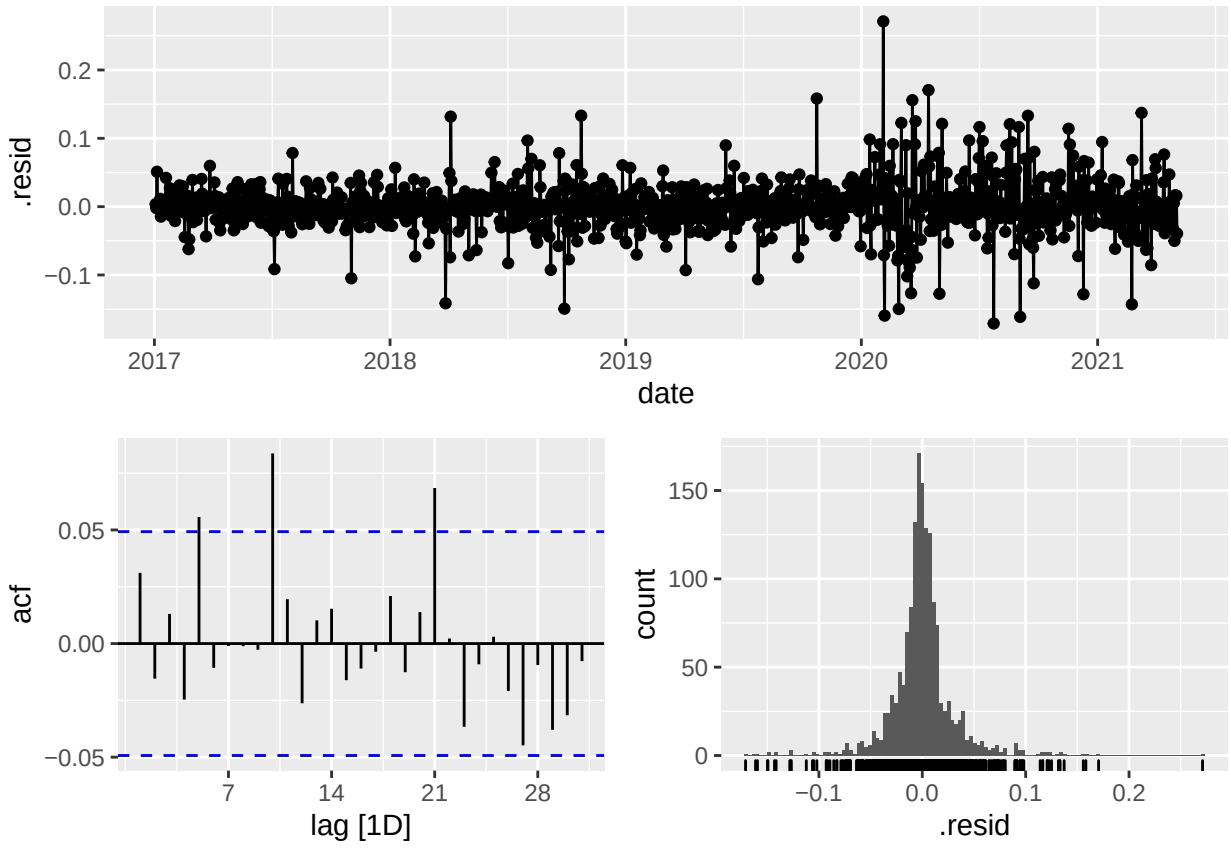
```
TSLA_fit2 %>% select(auto) %>% report()
```

```
## Series: close
## Model: ARIMA(1,1,1)(0,0,1)[7] w/ drift
## Transformation: log(close)
##
## Coefficients:
##          ar1      ma1     sma1  constant
##       0.6765  -0.6037  -0.0611     6e-04
## s.e.  0.1519   0.1641   0.0269     3e-04
##
## sigma^2 estimated as 0.0008111:  log likelihood=3386.9
## AIC=-6763.8   AICc=-6763.76   BIC=-6736.97
```

The best model is the ARIMA(0,1,0)(0,0,1)[7] model with drift. This is different from the model for the closing model. For the non-seasonal part of that model, the numbers were (1,1,1). It's surprising that its this much different.

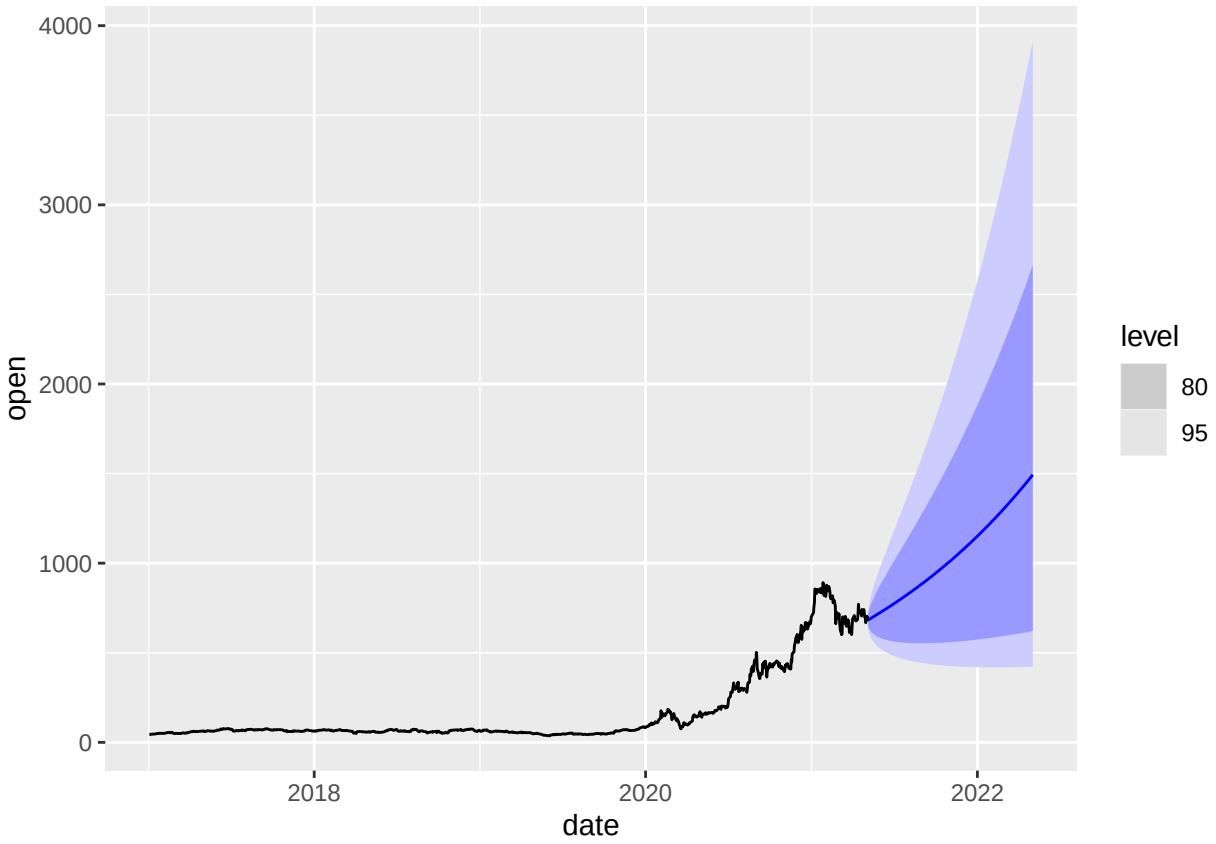
The residuals for this model are normal, only have a few significant langs, and the the pattern of the residuals appear to be very random. This is the best SARIMA model for the closing TSLA stock data.

```
TSLA_fit2_o %>%
  select(auto) %>%
  gg_tsresiduals()
```



f) Forecast the next 52 weeks of data.

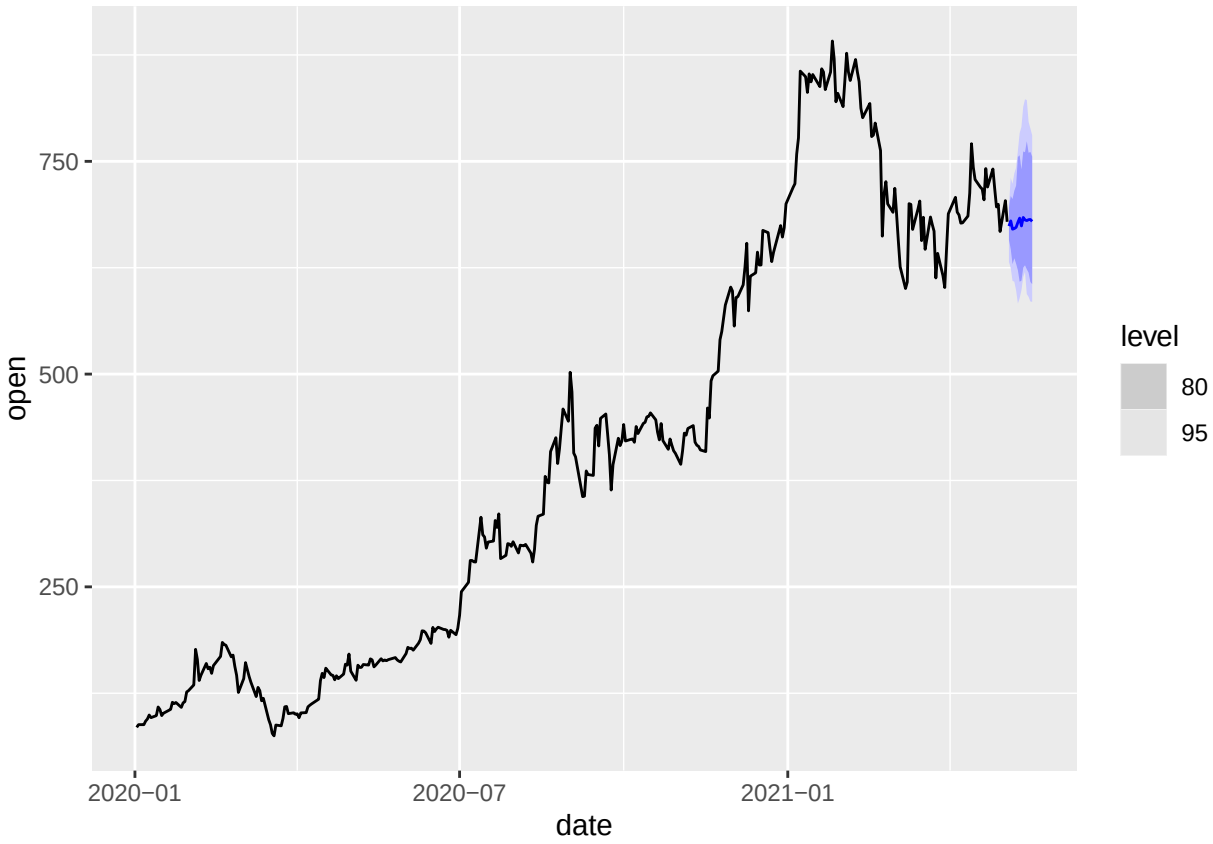
```
forecast_TSLA_o <- TSLA_fit2_o %>%
  forecast(h = "52 weeks")
forecast_TSLA_o %>%
  filter(.model=="auto") %>%
  autoplot(TSLA_tsbl2_o)
```



Forecasting the next year of data spreads a wide confidence interval net. However, this does seem to capture the exponential growth that TSLA has had over the last year, and does a great job projecting that growth into the future. This, of course, is the best modelling we have been able to use so far.

4 Build a neural network.

```
TSLA_tsbl2_o %>%
  model(nn = NNETAR(log(open))) %>%
  forecast(times=20) %>%
  autoplot(recent_TSLA)
```



See part 4 of the closing data for the same analysis.