Apple Stock Price timeseries data ARIMA model fitting

Here we are doing a time-series analysis of the closing stock prices of the Apple stock for the last 5 years (from 1st Jan. 2016 to 31st Dec. 2020).

getting the data

```
start <- as.Date("2016-01-01")
end <- as.Date("2020-12-31")
AAPL <- tq_get("AAPL", from = start, to = end)
```

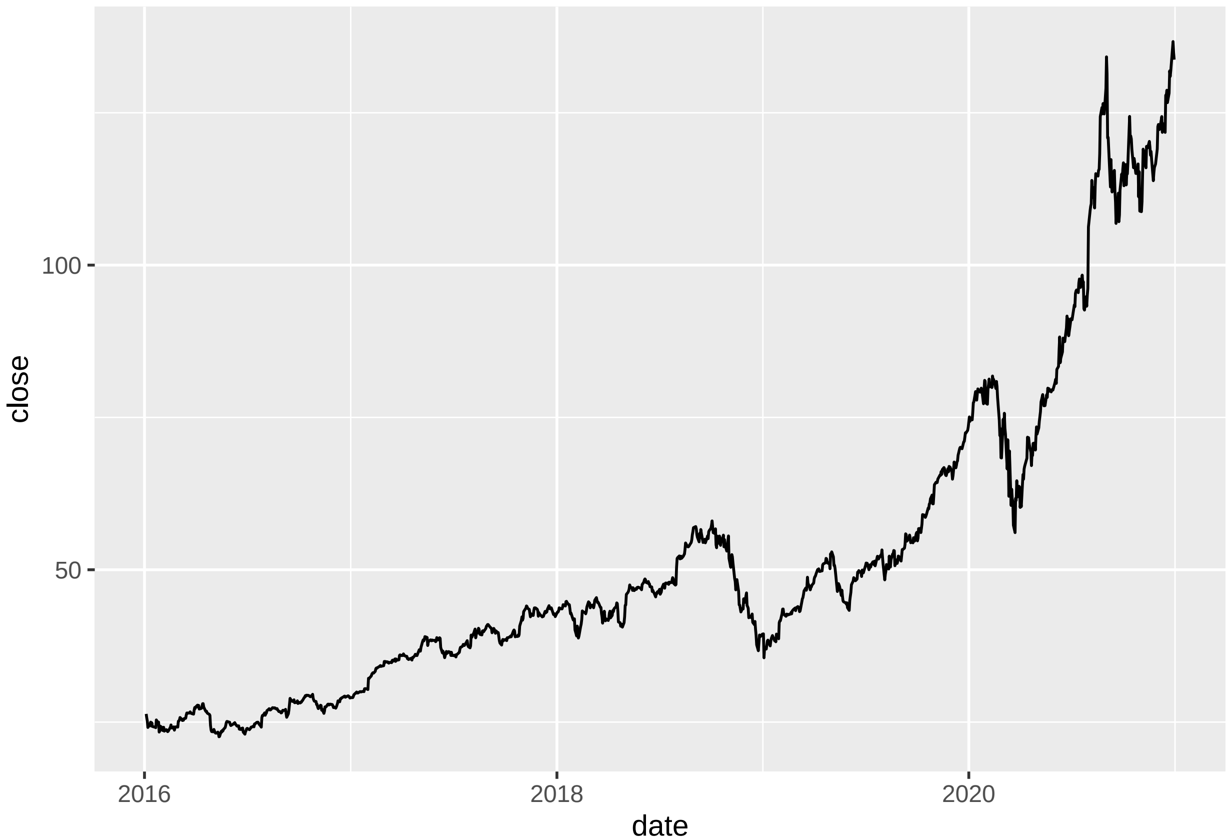
Viewing the data

```
head(AAPL)
```

```
## # A tibble: 6 x 8
##   symbol date        open  high   low close    volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 AAPL   2016-01-04  25.7  26.3  25.5  26.3 270597600     24.4
## 2 AAPL   2016-01-05  26.4  26.5  25.6  25.7 223164000     23.8
## 3 AAPL   2016-01-06  25.1  25.6  25.0  25.2 273829600     23.3
## 4 AAPL   2016-01-07  24.7  25.0  24.1  24.1 324377600     22.3
## 5 AAPL   2016-01-08  24.6  24.8  24.2  24.2 283192000     22.4
## 6 AAPL   2016-01-11  24.7  24.8  24.3  24.6 198957600     22.8
```

Closing stock price vs. Date plot

```
AAPL %>% ggplot(aes(x = date, y = close)) + geom_line()
```



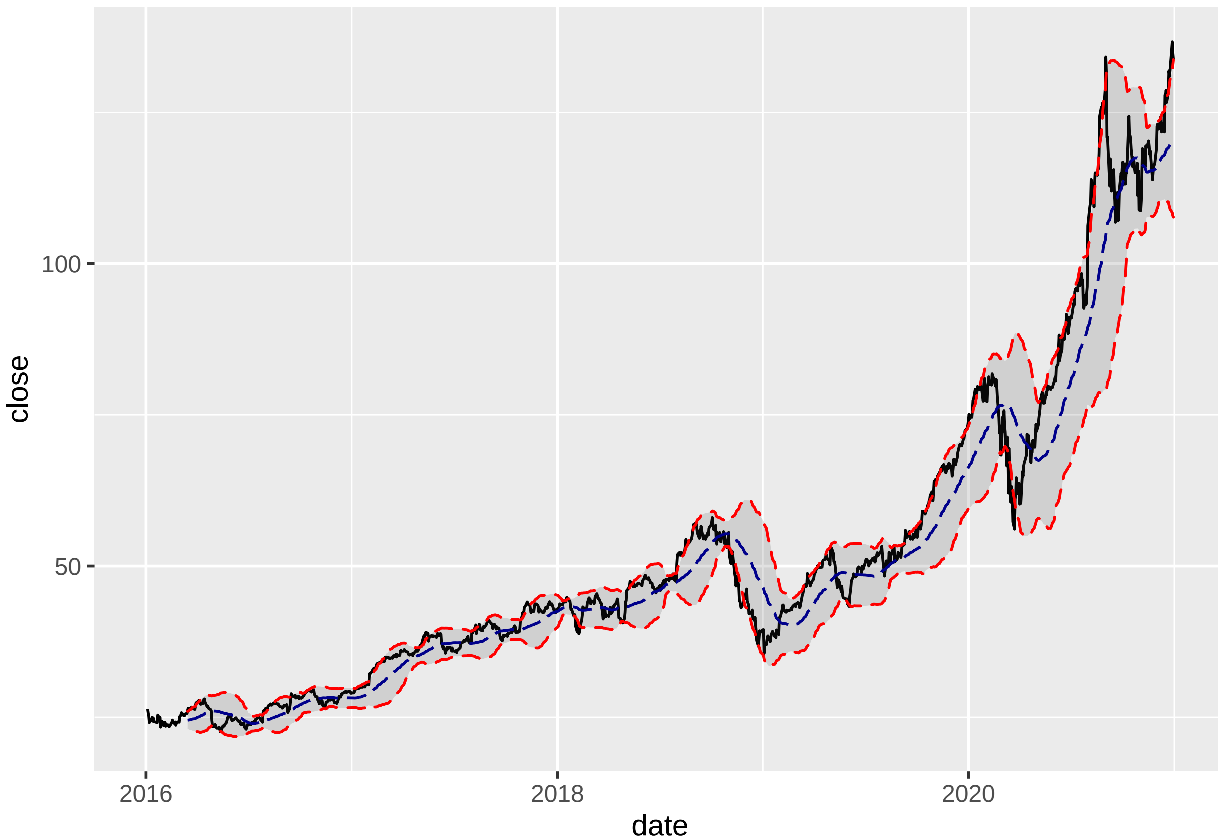
The data shows clear trend and possibly some seasonality.

Bollinger- Band plots

Moving average plot along with Moving Average +- standard deviation plots

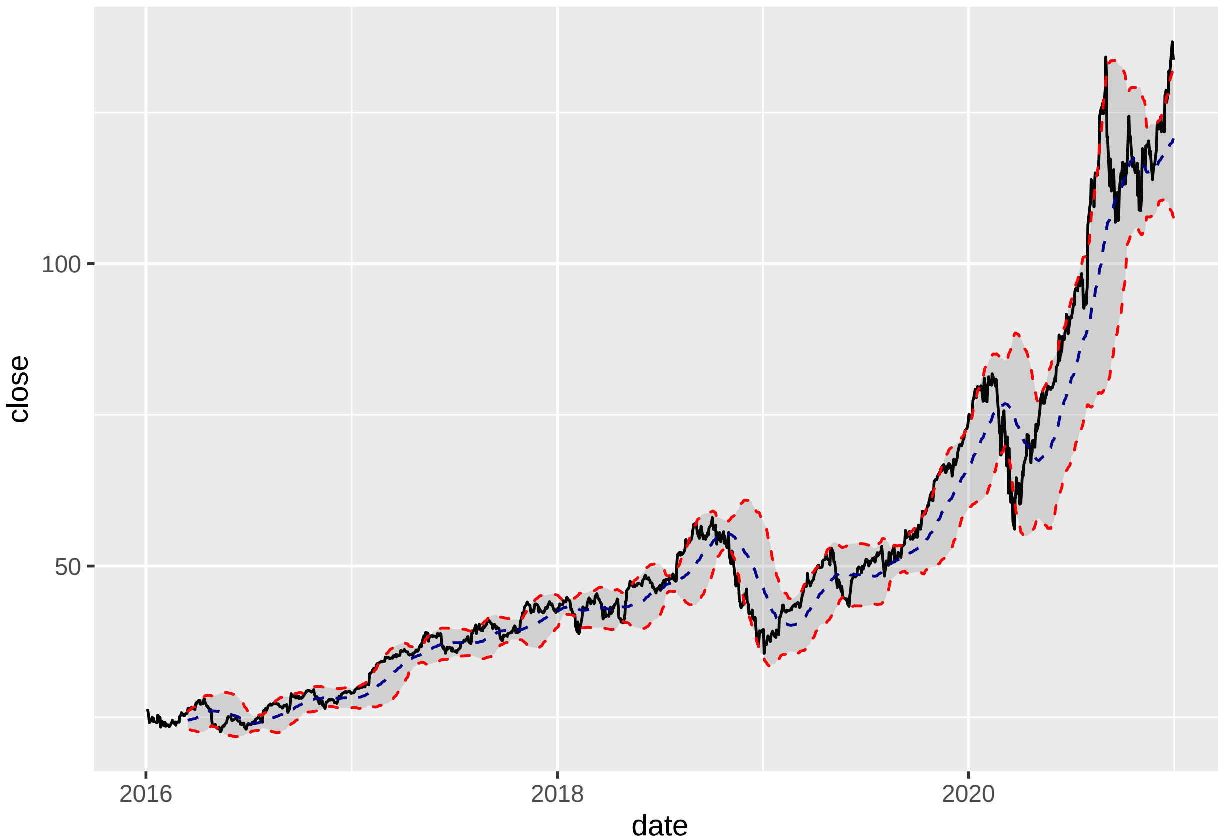
1. Simple Moving Average (SMA; window = 50 days)

```
# SMA
AAPL %>%
    ggplot(aes(x = date, y = close)) +
    geom_line() +
    geom_bbands(aes(high = high, low = low, close = close), ma_fun = SMA, n = 50, linetype=5) +
    coord_x_date(xlim = c(start, end))
```



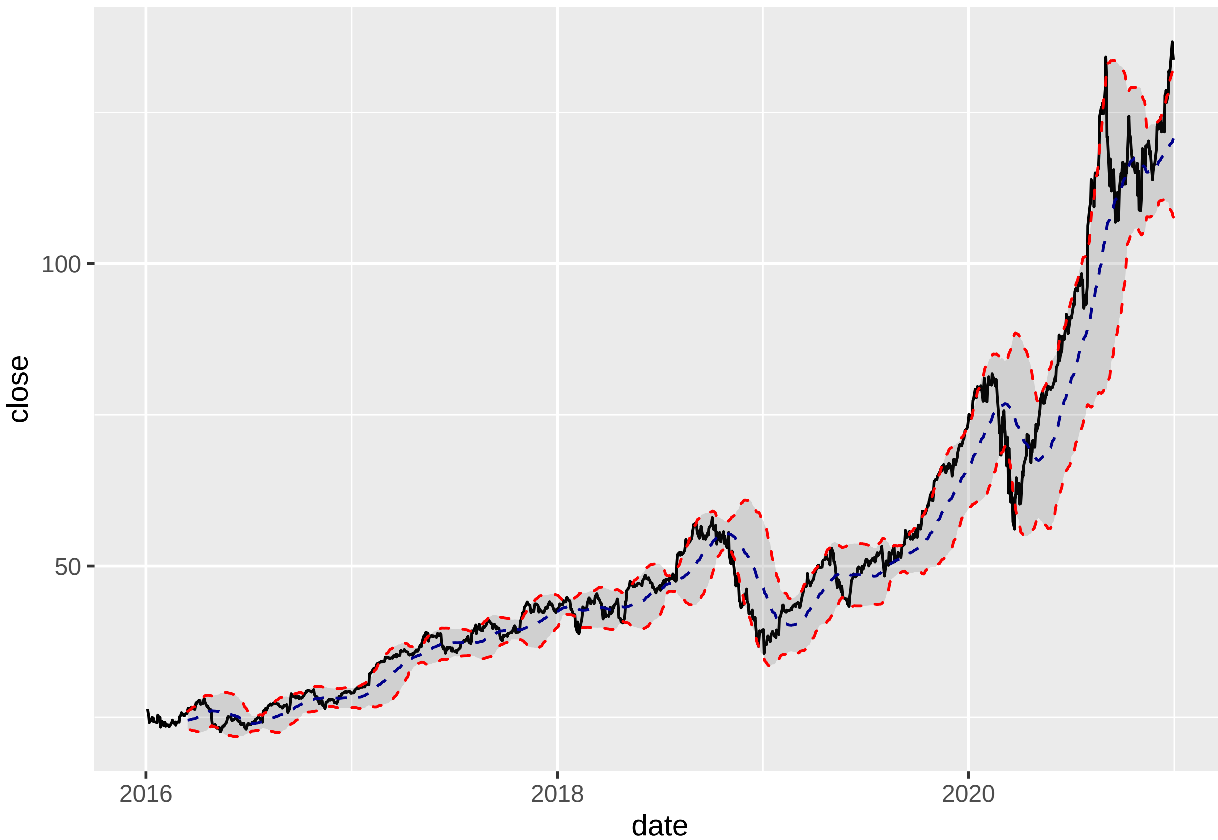
2. Exponential moving average (EMA; window = 50 days)

```
# EMA
AAPL %>%
   ggplot(aes(x = date, y = close)) +
   geom_line() +            # Plot stock price
   geom_bbands(aes(high = high, low = low, close = close),
                  ma_fun = EMA, wilder = TRUE, ratio = NULL, n = 50) +
   coord_x_date(xlim = c(start, end))
```



3. Volume Weighted moving average (VWMA; window = 50 days)

```
# VWMA
AAPL %>%
    ggplot(aes(x = date, y = close)) +
    geom_line() +            # Plot stock price
    geom_bbands(aes(high = high, low = low, close = close, volume = volume),
                   ma_fun = VWMA, n = 50) +
    coord_x_date(xlim = c(start,end))
```



Conducting ADF test for the Closing Price

```
print(adf.test(AAPL$close))
```

```
## Warning in adf.test(AAPL$close): p-value greater than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  AAPL$close
## Dickey-Fuller = -0.048662, Lag order = 10, p-value = 0.99
## alternative hypothesis: stationary
```

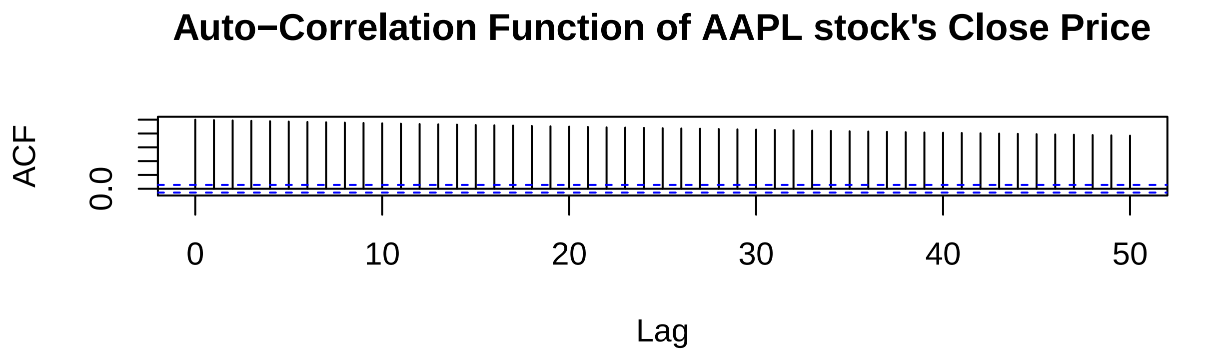
The high p-value indicates that the data is non-stationary

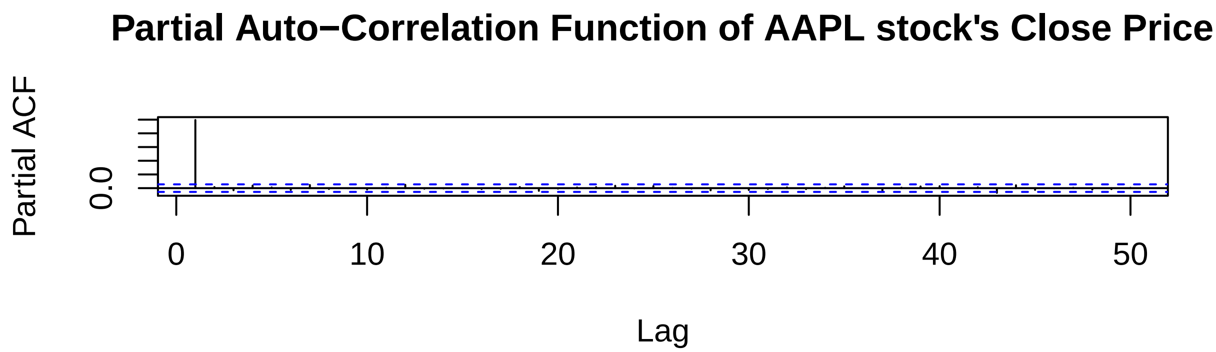
ACF and PACF plots

Let's first look at the auto-correlation function and the partial auto-correlation function plots.

```
data <- AAPL$close

par(mfrow=c(2,1))
acf(data,main="Auto-Correlation Function of AAPL stock's Close Price",50)
pacf(data,main="Partial Auto-Correlation Function of AAPL stock's Close Price",50);
```



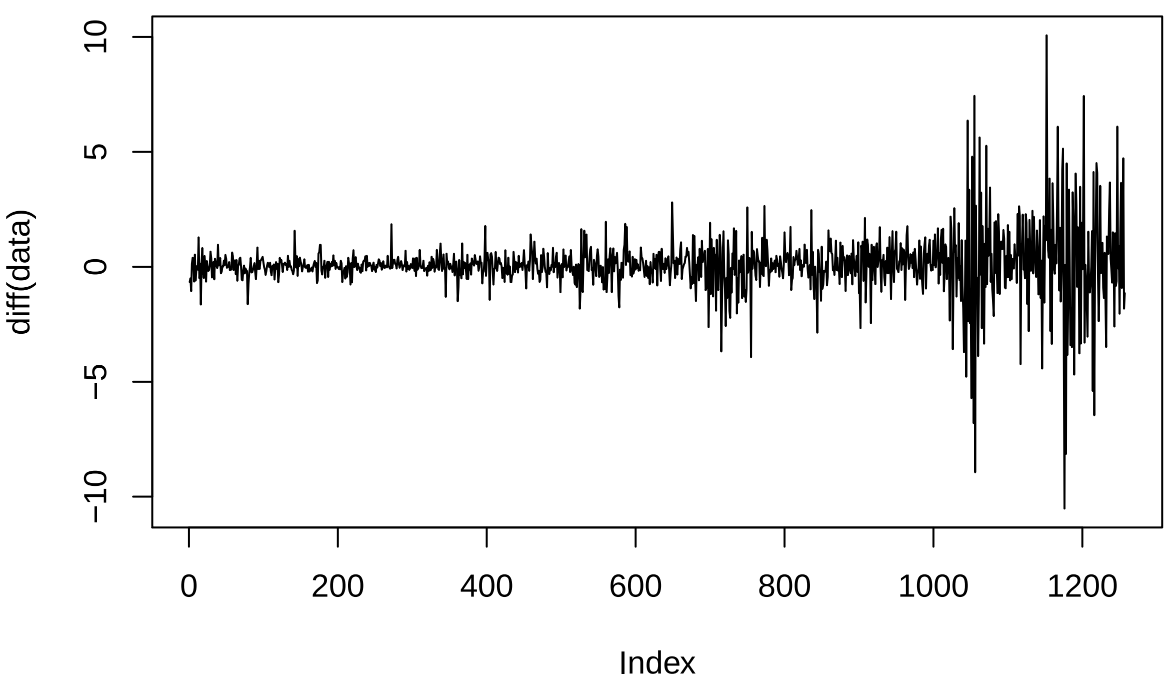


We see that the ACF plot shows lots of correlation for a very large number of lags.

Guessing the right orders for ARIMA model fitting

1. Differncing orders d

```
plot(diff(data),type="l")
```



So we got rid of the trend but some seasonality and a clear trend in the variation can be easily seen.

Let's do ADF test on the differenced data

```
diff_data <- diff(data)
print(adf.test(diff_data))
```

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

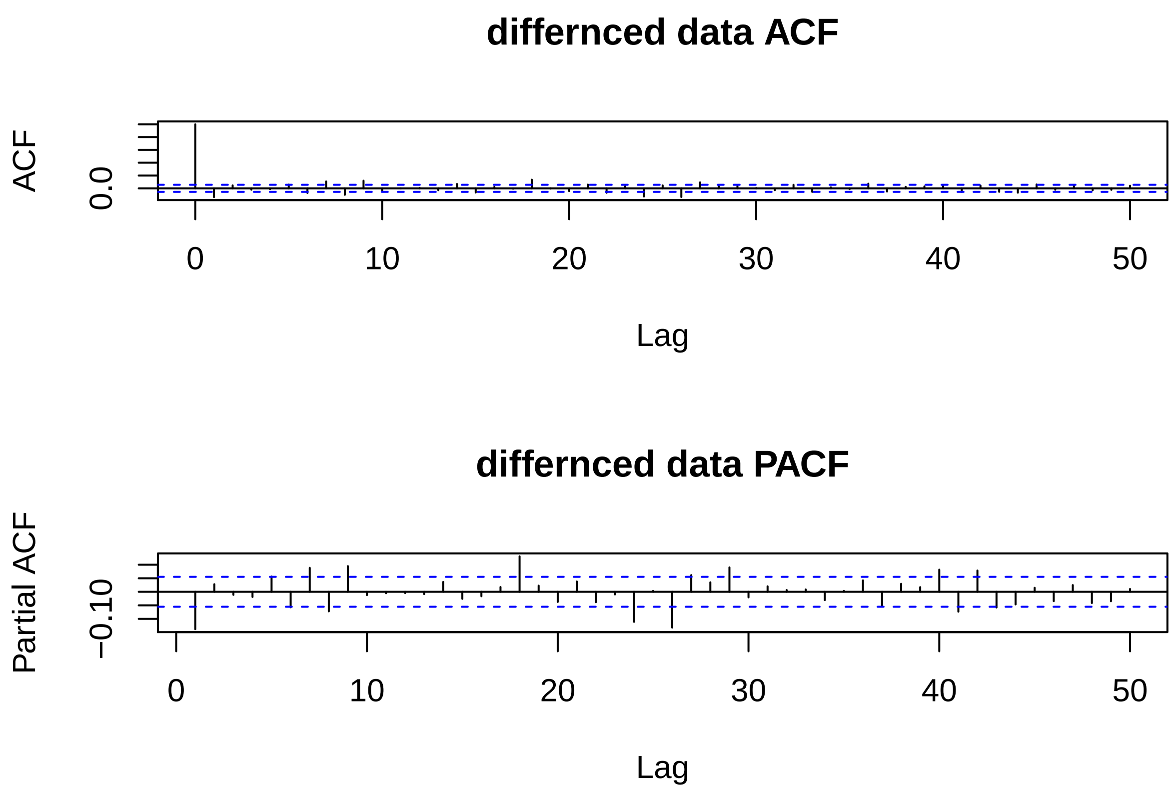
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_data
## Dickey-Fuller = -10.305, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

The test confirms that the differenced data is (weakly) stationary.

2. orders for the auro-regressive (AR) and Moving Average (MA) terms i.e. p and q

ACF and PACF for differenced data

```
par(mfrow=c(2,1))
acf(diff_data,main='differnced data ACF',50)
pacf(diff_data,main='differnced data PACF',50);
```



Now, the ACF plot does not show much correlation but PACF shows lots of correlation at several lags. But there is no abrupt drop and it's very difficult to guess the orders.

Finding best parameters using

1. Grid Search

Trying for different values of p,q,P,Q and note down AIC, SSE and p-value (for Ljun-box-test). We want high p-values and small AIC and SSE using parsimony principle (simpler the better) while searching. Let's

```r
for(p in 1:5){
  for(d in 1:2){
    for(q in 1:5){
      if(p+d+q<=10){

        model<- arima(x=data, order = c(p-1,d,q-1))

        pval<-Box.test(model$residuals, lag=log(length(model$residuals)))

        sse<-sum(model$residuals^2)

        cat(p-1,d,q-1, 'AIC=', model$aic,' p-VALUE=', pval$p.value,'\n')
      }

    }
  }
}
```

```
## 0 1 0 AIC= 4240.346  p-VALUE= 4.259606e-09
## 0 1 1 AIC= 4221.526  p-VALUE= 0.004673906
## 0 1 2 AIC= 4220.769  p-VALUE= 0.02630195
## 0 1 3 AIC= 4222.214  p-VALUE= 0.04505514
## 0 1 4 AIC= 4224.04  p-VALUE= 0.05168168
## 0 2 0 AIC= 5265.214  p-VALUE= 0
## 0 2 1 AIC= 4240.085  p-VALUE= 2.550414e-09
## 0 2 2 AIC= 4218.041  p-VALUE= 0.0091442
## 0 2 3 AIC= 4218.368  p-VALUE= 0.03032378
## 0 2 4 AIC= 4219.215  p-VALUE= 0.06780968
## 1 1 0 AIC= 4219.83  p-VALUE= 0.01491896
## 1 1 1 AIC= 4220.512  p-VALUE= 0.03573231
## 1 1 2 AIC= 4222.373  p-VALUE= 0.03815019
## 1 1 3 AIC= 4224.183  p-VALUE= 0.04652894
## 1 1 4 AIC= 4222.183  p-VALUE= 0.09446043
## 1 2 0 AIC= 4749.115  p-VALUE= 0
## 1 2 1 AIC= 4216.583  p-VALUE= 0.02540967
## 1 2 2 AIC= 4217.837  p-VALUE= 0.04618131
## 1 2 3 AIC= 4221.689  p-VALUE= 0.01060686
## 1 2 4 AIC= 4222.35  p-VALUE= 0.02937024
## 2 1 0 AIC= 4220.42  p-VALUE= 0.03642116
## 2 1 1 AIC= 4222.401  p-VALUE= 0.03774638
## 2 1 2 AIC= 4218.643  p-VALUE= 0.04254989
```

```
## Warning in arima(x = data, order = c(p - 1, d, q - 1)): possible convergence
## problem: optim gave code = 1
```

```
## 2 1 3 AIC= 4198.677  p-VALUE= 0.8693383
## 2 1 4 AIC= 4224.16  p-VALUE= 0.09461257
## 2 2 0 AIC= 4594.835  p-VALUE= 0
## 2 2 1 AIC= 4217.911  p-VALUE= 0.04318682
## 2 2 2 AIC= 4208.044  p-VALUE= 0.1436806
## 2 2 3 AIC= 4176.686  p-VALUE= 0.4133833
## 2 2 4 AIC= 4171.071  p-VALUE= 0.9976173
## 3 1 0 AIC= 4222.379  p-VALUE= 0.03931329
```

```
## 3 1 1 AIC= 4224.37  p-VALUE= 0.03978806
## 3 1 2 AIC= 4217.33  p-VALUE= 0.009496059

## Warning in arima(x = data, order = c(p - 1, d, q - 1)): possible convergence
## problem: optim gave code = 1

## 3 1 3 AIC= 4177.999  p-VALUE= 0.4107303

## Warning in arima(x = data, order = c(p - 1, d, q - 1)): possible convergence
## problem: optim gave code = 1

## 3 1 4 AIC= 4171.91  p-VALUE= 0.9378829
## 3 2 0 AIC= 4516.423  p-VALUE= 0
## 3 2 1 AIC= 4219.617  p-VALUE= 0.0539585
## 3 2 2 AIC= 4221.574  p-VALUE= 0.05586328

## Warning in arima(x = data, order = c(p - 1, d, q - 1)): possible convergence
## problem: optim gave code = 1

## 3 2 3 AIC= 4218.86  p-VALUE= 0.01072795
## 4 1 0 AIC= 4224.142  p-VALUE= 0.04493521
## 4 1 1 AIC= 4205.131  p-VALUE= 0.8468612
## 4 1 2 AIC= 4175.827  p-VALUE= 0.9926715
## 4 1 3 AIC= 4171.884  p-VALUE= 0.9391635
## 4 2 0 AIC= 4431.317  p-VALUE= 2.220446e-16
## 4 2 1 AIC= 4220.917  p-VALUE= 0.07028125
## 4 2 2 AIC= 4201.597  p-VALUE= 0.9260231
```

2. Using auto.arima()

```
#auto.arima( data, d = 1, D = 1,  max.p = 5,  max.q = 5,  max.P = 5,  max.Q = 5, max.order = 10,  start
model <- auto.arima(data, lambda = "auto")
model
```

```
## Series: data
## ARIMA(4,1,1) with drift
## Box Cox transformation: lambda= -0.2499176
##
## Coefficients:
##           ar1      ar2      ar3      ar4     ma1  drift
##       -0.9529  -0.0729  -0.0188  -0.0858  0.8687  5e-04
## s.e.   0.0504   0.0392   0.0391   0.0296  0.0427  2e-04
##
## sigma^2 estimated as 4.861e-05:  log likelihood=4461.44
## AIC=-8908.88   AICc=-8908.79   BIC=-8872.93
```

Best-model

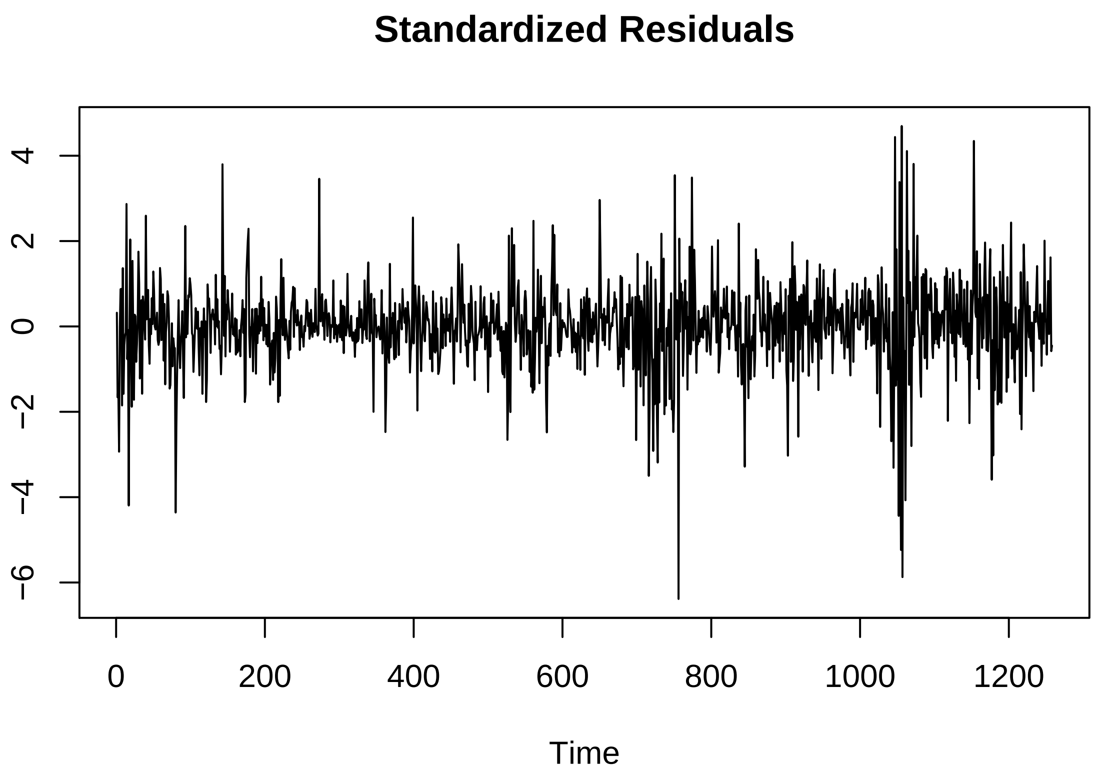
The orders selected for the minimum AIC values (~4171) in the gird search method are 2,2,4 and 4,1,3. With auto.arima we found the order 4,1,1 correspnding to AIC~4205 which is only slightly large and has fewer parameters. All three models show siginificant p-values for the Ljung-Box test. We will proceed with the order 4,1,1 for the rest of the analysis.

Train-test split

```
N = length(data)
n = 0.7*N
train = data[1:n]
test  = data[(n+1):N]
```

ARIMA(4,1,1) fitting results

```
standard_residuals<- model$residuals/sd(model$residuals)
plot(standard_residuals,ylab='',main='Standardized Residuals')
```




```
print(adf.test(standard_residuals))
```

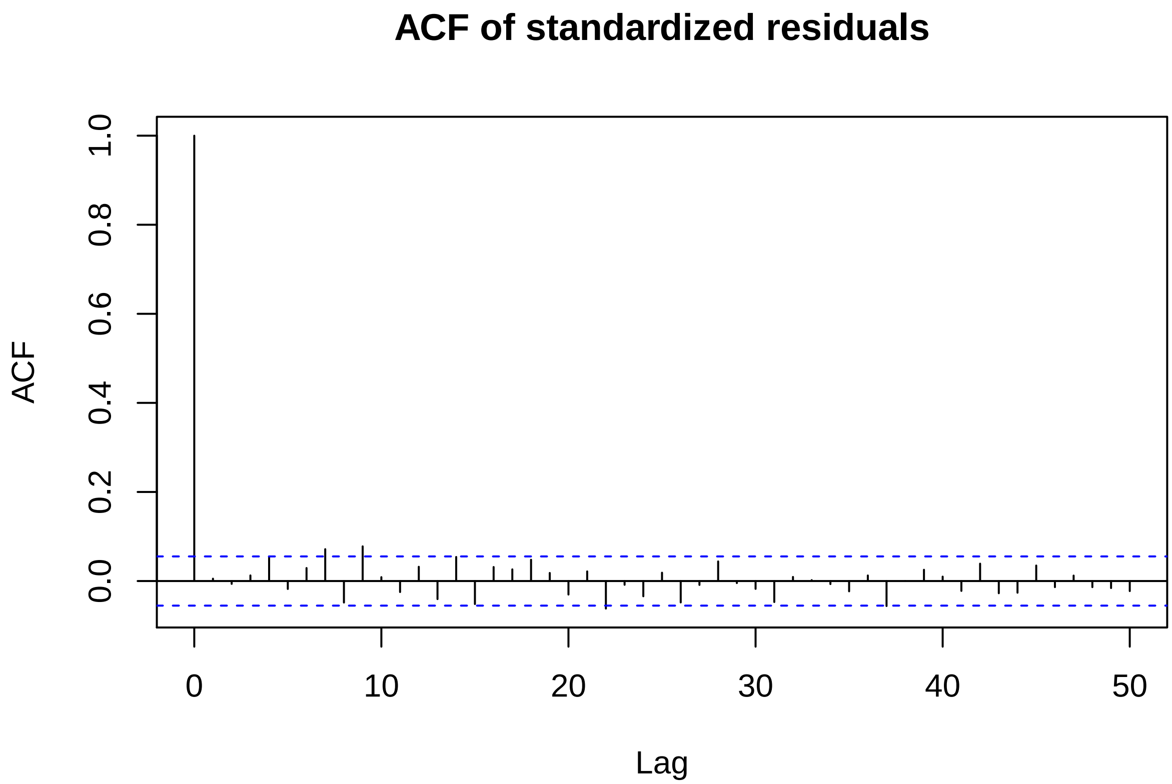
```
## Warning in adf.test(standard_residuals): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  standard_residuals
## Dickey-Fuller = -9.8592, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

We see that the residuals look almost stationary which we also confirmed with the ADF test

Let's check for correlations in the residual using the ACF plot

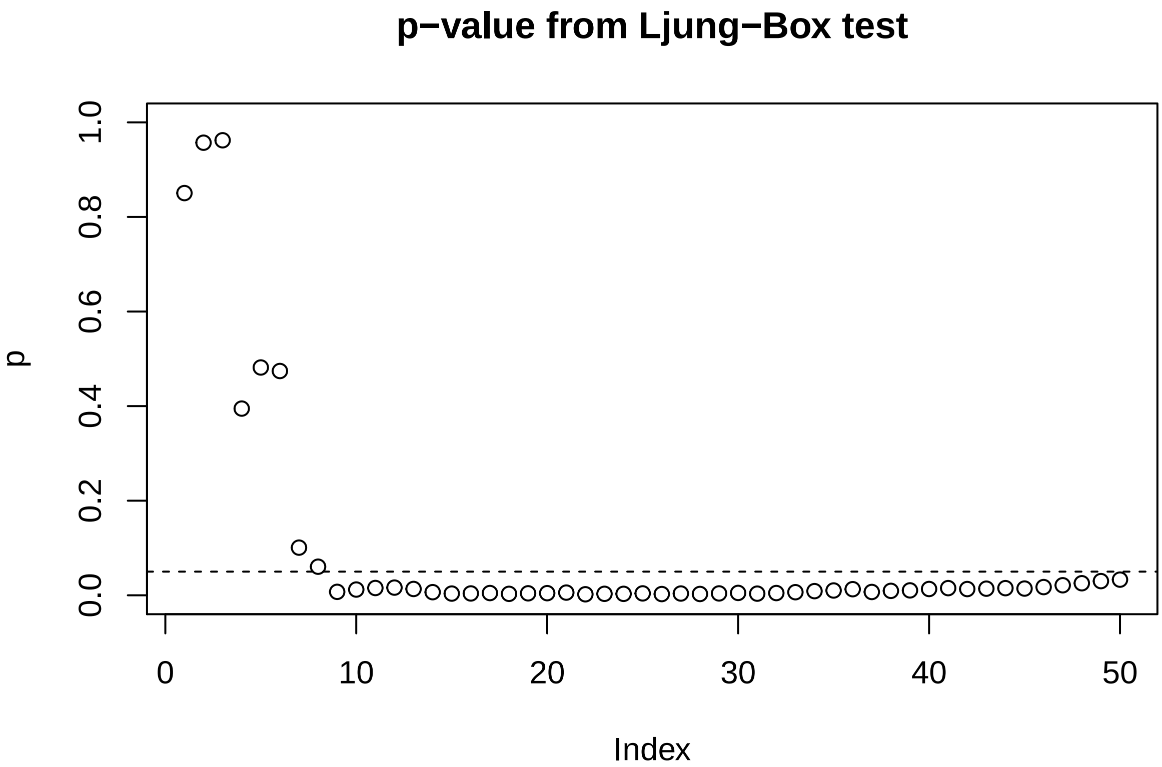
```
acf(standard_residuals,50,main='ACF of standardized residuals');
```



The correlations at all lags seem to be insignificant for the residuals.

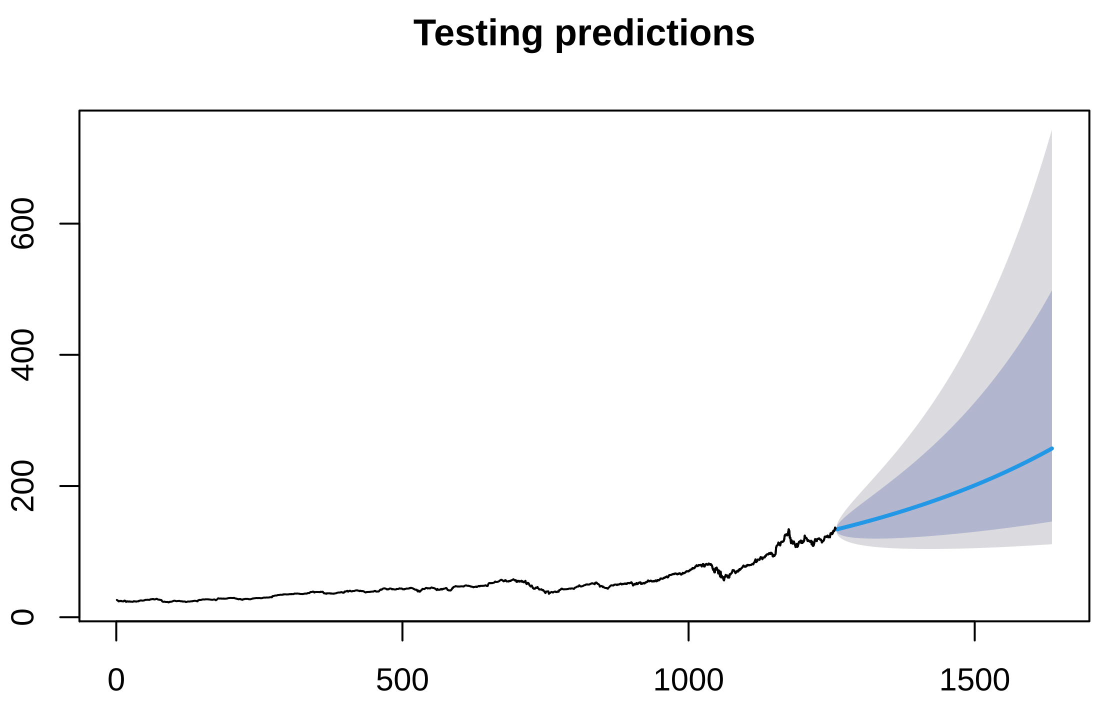
Next, we will perform a Ljung-Box test on the residuals. The null hypotheis for the test is:
H0: The dataset points are independently distributed (not correlated).
where a p-value of greater than 0.05 will be insifficient to reject the null hypothesis.

```
for (lag in seq(1:50)){
  pval<-Box.test(model$residuals, lag=lag)
  p[lag]=pval$p.value
}
plot(p,ylim = (0.0:1), main='p-value from Ljung-Box test')
abline(h=0.05,lty=2)
```



Any value above the dashed line (at y=0.05) is significant. We see that the p-values of the Ljung-Box test at the lags < 17 are all significant and therefore the hypothesis that the residuals are not correlated cannot be rejected.

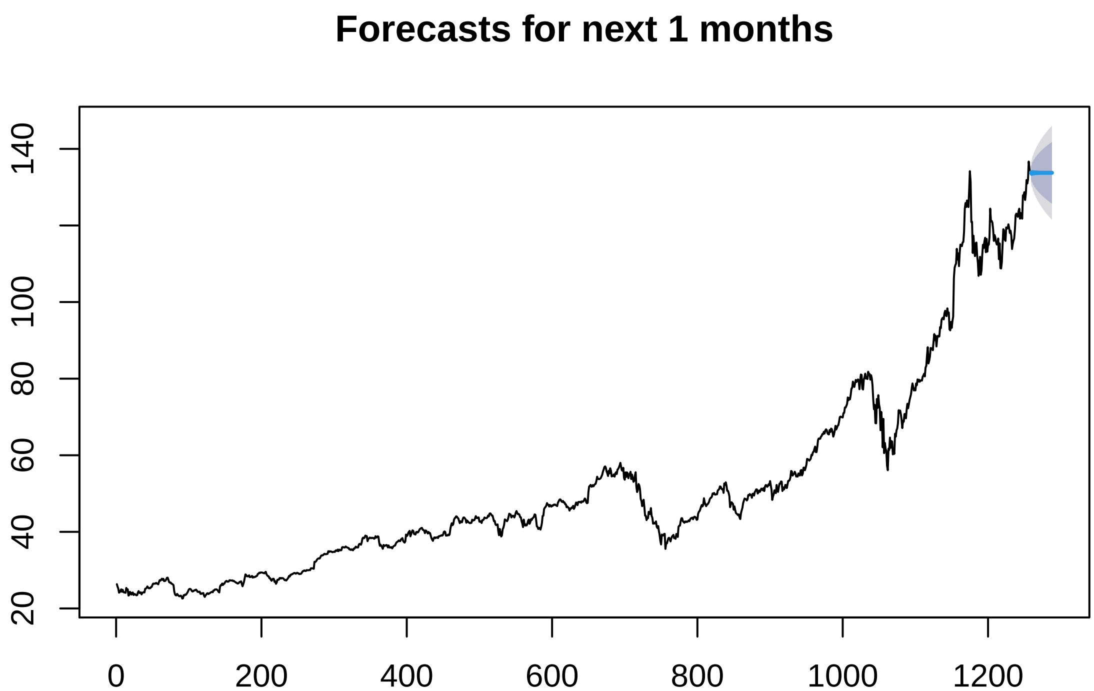
```
pred_len=length(test)
plot(forecast(model, h=pred_len),main='Testing predictions')
train_x = seq(length(train)+1,length(train)+length(test))
lines(train_x,test)
```



Here the black line in the first left shows the training data. The blue line on the right showing the predictions from our model. The small shaded region on the blue lines which seem to cover the test data on the right completely shows the confidence interval of the predictions; it consists of two different dark and light shaded regions showing the 80% and 95% confidence regions.

Forecasting using the best-model

```r
model<-arima(x=data, order = c(4,1,1))
par(mfrow=c(1,1))
h=30 # forecasting for the next 1 month after the end of the dataset
plot(forecast(model,h), main='Forecasts for next 1 months');
```



The uncertainty on the prediction seems to be pretty big for the ARIMA model.
We will now use the *prophet* model for modeling the data and make predictions.

Using Prophet for modeling

```r
df <- data.frame(ds = list(AAPL[,'date']), y = list(AAPL[,'close'])) %>% rename(ds=date,y=close)
head(df)
```

```
##           ds       y
## 1 2016-01-04 26.3375
## 2 2016-01-05 25.6775
## 3 2016-01-06 25.1750
## 4 2016-01-07 24.1125
## 5 2016-01-08 24.2400
## 6 2016-01-11 24.6325
```

```r
len_train = nrow(df)*0.9
len_test = nrow(df) - len_train
df_train = df[1:len_train,]
```

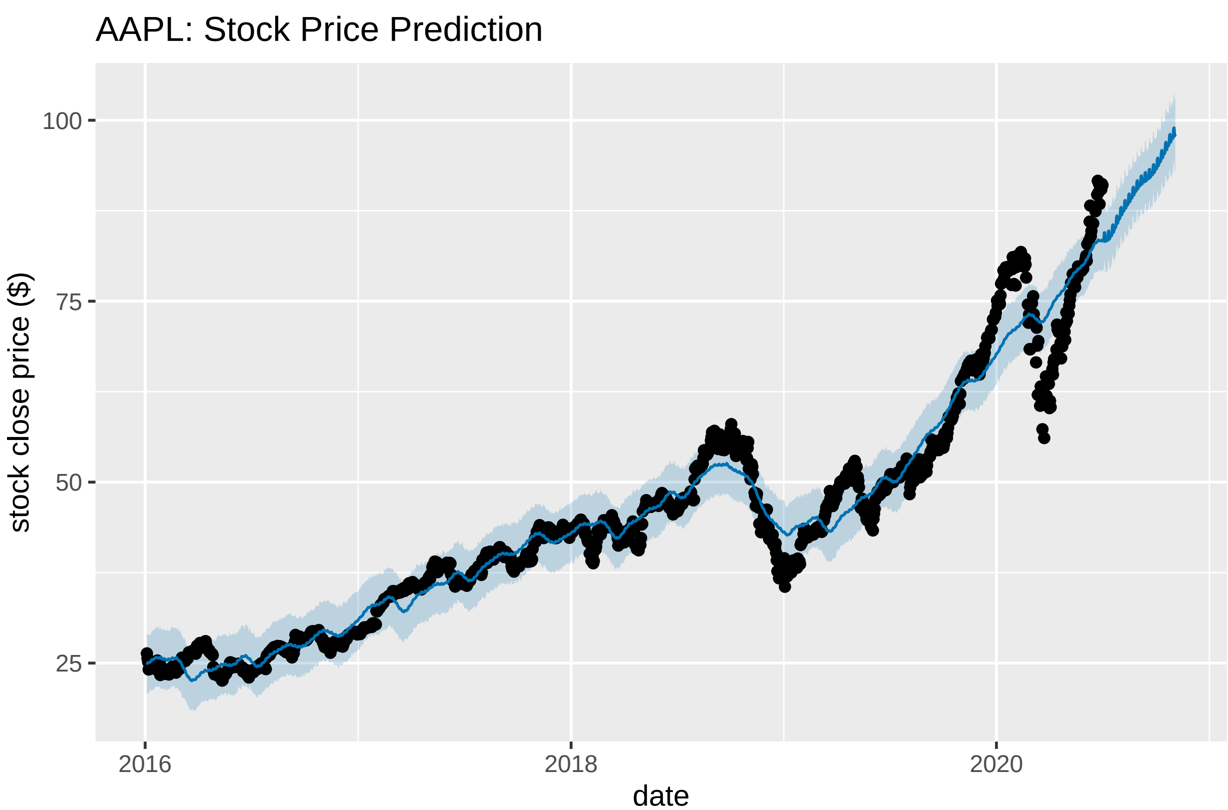
```r
df_test = df[len_train+1:nrow(df),]
m <- prophet(df_train)
```

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

```r
future <- make_future_dataframe(m, periods = len_test)
forecast <- predict(m, future)
head(forecast[c('ds','yhat','yhat_lower','yhat_upper')])
```

```
##           ds     yhat yhat_lower yhat_upper
## 1 2016-01-04 24.79345   20.95182   28.94855
## 2 2016-01-05 25.00143   20.83883   28.90352
## 3 2016-01-06 25.09444   20.76883   29.05638
## 4 2016-01-07 25.07370   20.80113   28.97473
## 5 2016-01-08 25.06195   21.07501   28.82420
## 6 2016-01-11 25.21837   21.38489   29.01686
```

```r
plot(m, forecast, xlabel = "date", ylabel = "stock close price ($)") + ggtitle("AAPL: Stock Price Predi
```



We see that the uncertainties in the predections obtained with the prophet model are quite small in comparison to the ARIMA model.