STOCK PRICE ANALYSIS FOR MEDIOCRE SOCIAL NETWORK APPS INC.

Group: Robert Sweeney Blanco, Alec Kan, Jake Singleton, Jonathan Bodine

November 22, 2019

Executive Summary:
In this report, we analyze and provide forecasts for the stock price dataset, which consists of asset values in dollars for Mediocre Social Network Apps Inc. from the beginning of 2015 through the end of September 2019. In particular, we choose an ARIMA(0,1,0) model to generate forecasts for the fist 10 trading days of October 2019. Ultimately, we find that they should not expect their stock price to improve much in the first 10 trading days of October 2019.

1 Exploratory Data Analysis

We begin by plotting the raw data. Note that all prices given in this report are in dollars.

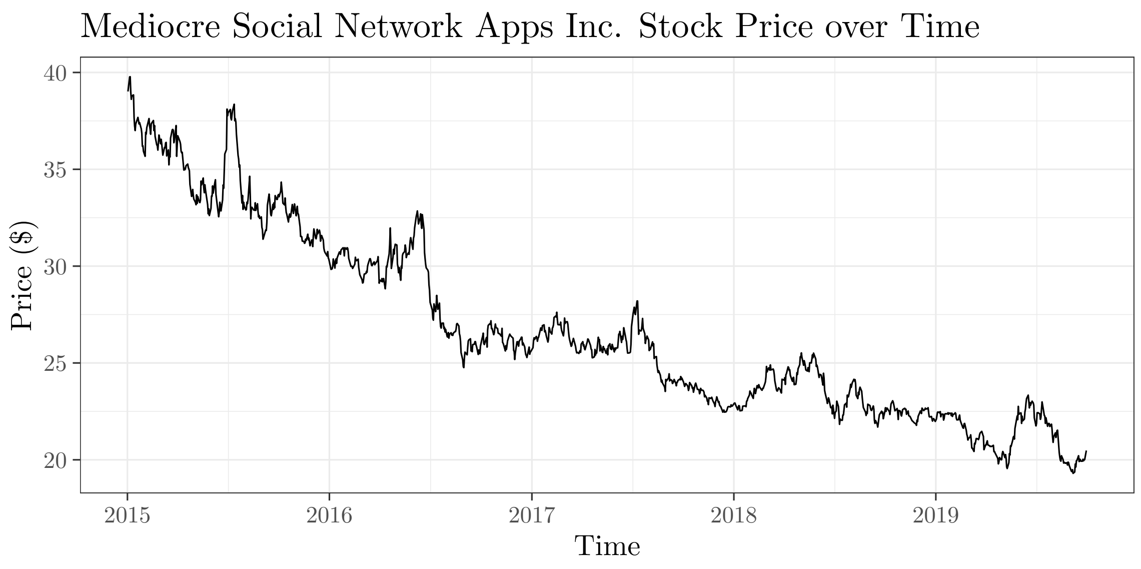

Figure 1: This figure shows Mediocre Social Network Apps Inc.'s stock price (in $) from the beginning of 2015 through September 2019.

In Figure 1 above, we see an overall decreasing trend with perhaps some yearly seasonality. In particular, it looks like a quadratic curve would fit the data well, and so above we plot its fit (2a) and residuals (2b). While the fit looks good, there is heteroscedasticity (non-constant variance) in the residuals, which we will handle in the next part of the project by performing a log transformation on the data. This transformation stabilizes, or makes constant, the variance of the time series, which enables us to make it stationary. Lastly, because of the good quadratic fit, we will consider second-order differencing in the next step.

2 Modeling a Deterministic Function of Time

In our exploration of the data, we noticed that there appears to be heteroscedasticity in the raw data, as it appears that the variance is decreasing quadratically with time. As such, we apply a variance

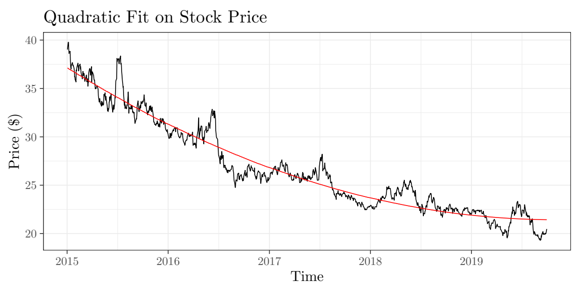

Figure 2a: The fit of the quadratic curve.

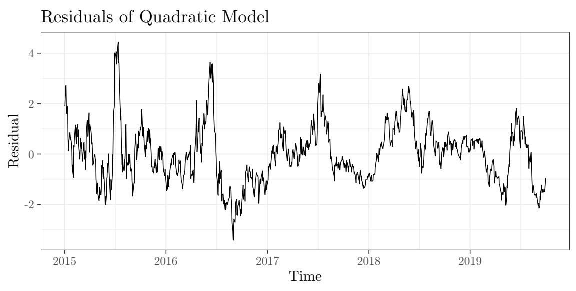

Figure 2b: The residuals of the quadratic fit, which show significant heteroscedasticity.

stabilizing transform, specifically a log transform (good for quadratically decreasing variance), to make the data homoscedastic (uniform variance). From here on out, we define X_t to be the log-transformed stock price at time t.

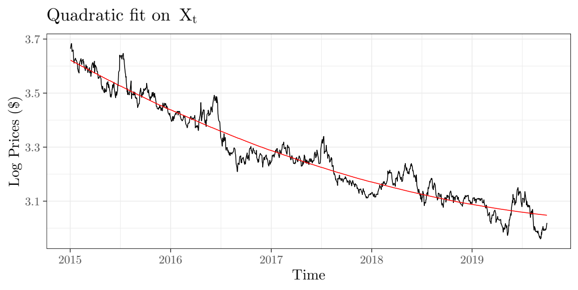

Figure 3a: The fit of the quadratic curve on X_t.

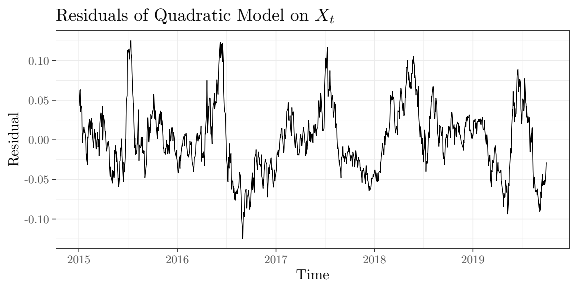

Figure 3b: The residuals of the fit in 3a.

Now, we fit a quadratic curve to X_t (3a) and plot its residuals (3b). We see that the transformed data in (3a) is homoscedastic; however, we notice that residuals ε_t still aren't stationary. Note that stationary means that the process has a mean and autocovariance function independent of t. To combat this, we take the first difference of the residuals. The results are below in Figure 4.

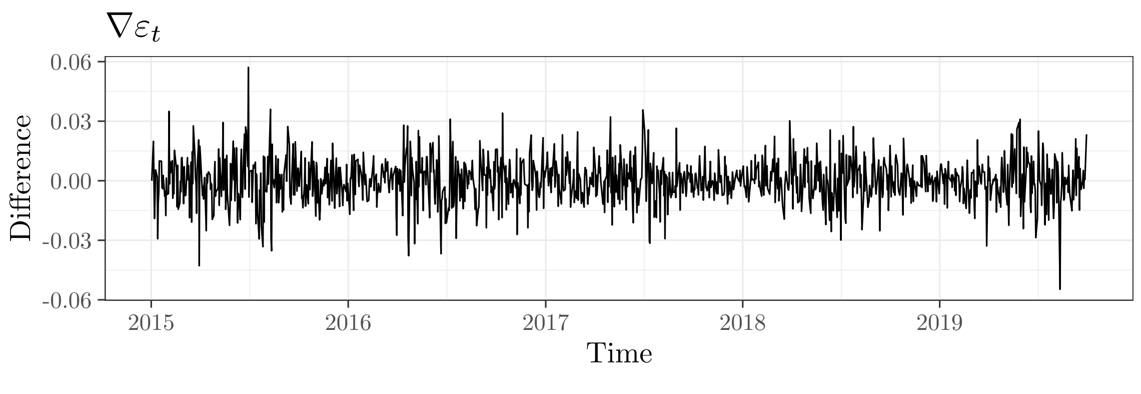

Figure 4: First difference of residuals from quadratic fit on X_t.

When we look at the first differenced residuals $\nabla\varepsilon_t$ of the quadratic fit, they now appear to be stationary. This is a success!

Now, we consider other methods of reaching stationarity. Since we had a quadratic trend in X_t earlier, we propose using a second difference in the data, $\nabla^2 X_t$, as this removes quadratic trends. The results of this transformation are below in Figure 5.

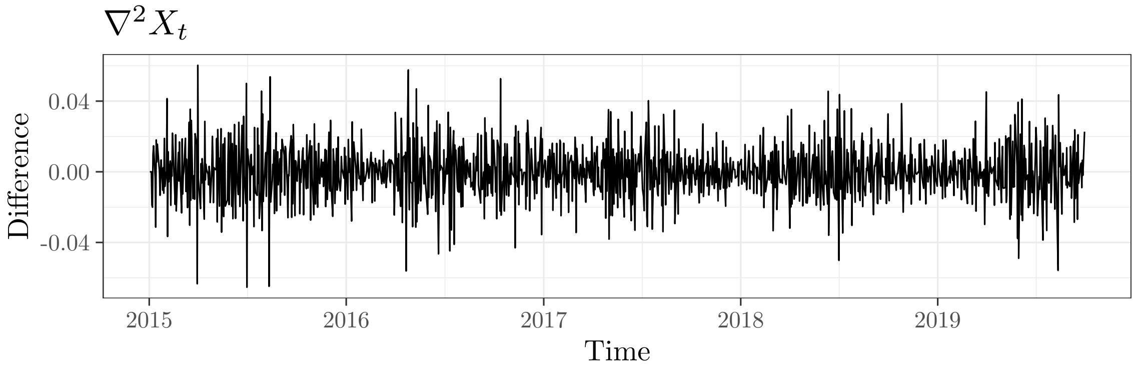

Figure 5: The second difference of X_t.

Our transformation $\nabla^2 X_t$ also appears to be stationary. This is another success!

Now, since a second-order difference appears to produce stationarity and X_t appears to weakly follow a linear trend as seen in Figure 3a, we decide to try the simpler model ∇X_t. The results are as follows in Figure 6.

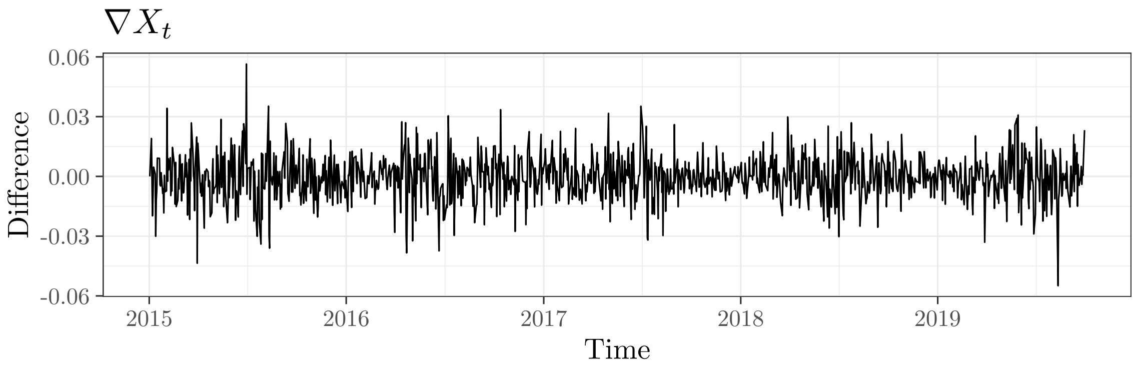

Figure 6: The first difference of X_t.

This again results in a stationary time series. So, now that we have three different ways of reaching stationarity, we move on to using ARIMA models to fit these stationary processes.

3 ARIMA Model Selection

We first consider fitting an ARMA model to $\nabla\varepsilon_t$, the differenced residuals from the quadratic fit on X_t. To do so, we first look at the the ACF (autocorelation function) and PACF (partial autocorrelation function). These charts are below in Figures 7a and b.

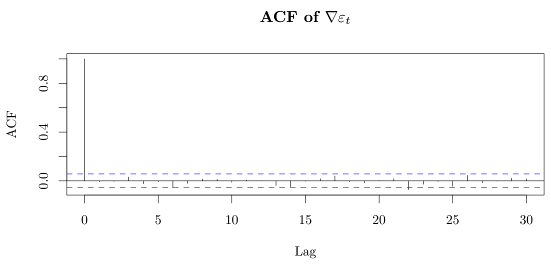

Figure 7a: The ACF of $\nabla\varepsilon_t$

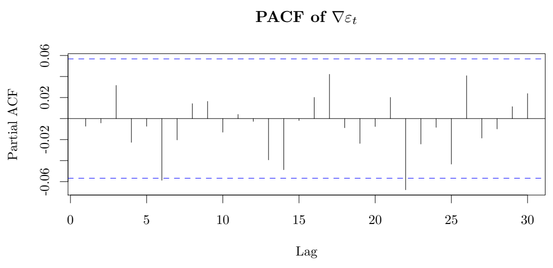

Figure 7b: The PACF of $\nabla\varepsilon_t$

These plots allow us to see whether the fitted model results in white noise. Specifically, white noise implies that our model is complete and that there is no longer any more information to extract from the stationary process. Because all of the spikes (correlation to X_{t-h}) are within or barely outside the blue lines, we conclude that the remaining residuals are indeed white noise. This means that the best model for us to use for ε_t is an ARIMA(0,1,0) model.

Next, we consider the stationary time series $\nabla^2 X_t$. Its ACF and PACF are below in Figures 8a and b.

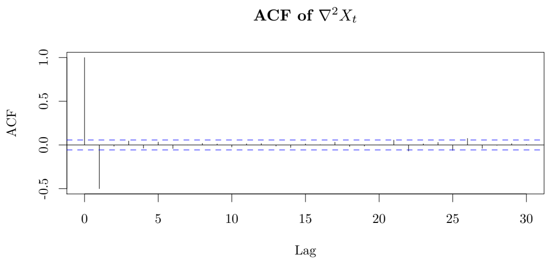

Figure 8a: The ACF of $\nabla^2 X_t$

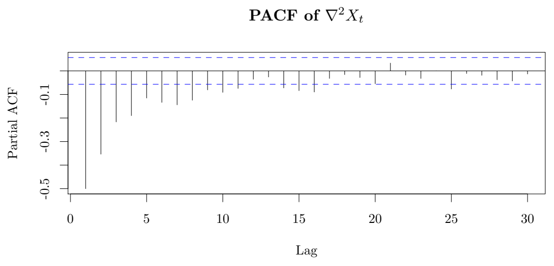

Figure 8b: The PACF of $\nabla^2 X_t$

There is a distinct spike in the ACF graph at lag one and exponentially decaying spikes in the PACF graph. This pattern is indicative of an MA(1) model, and so we fit an ARIMA(0,2,1) model to X_t. The results of this fit are below in Figure 9.

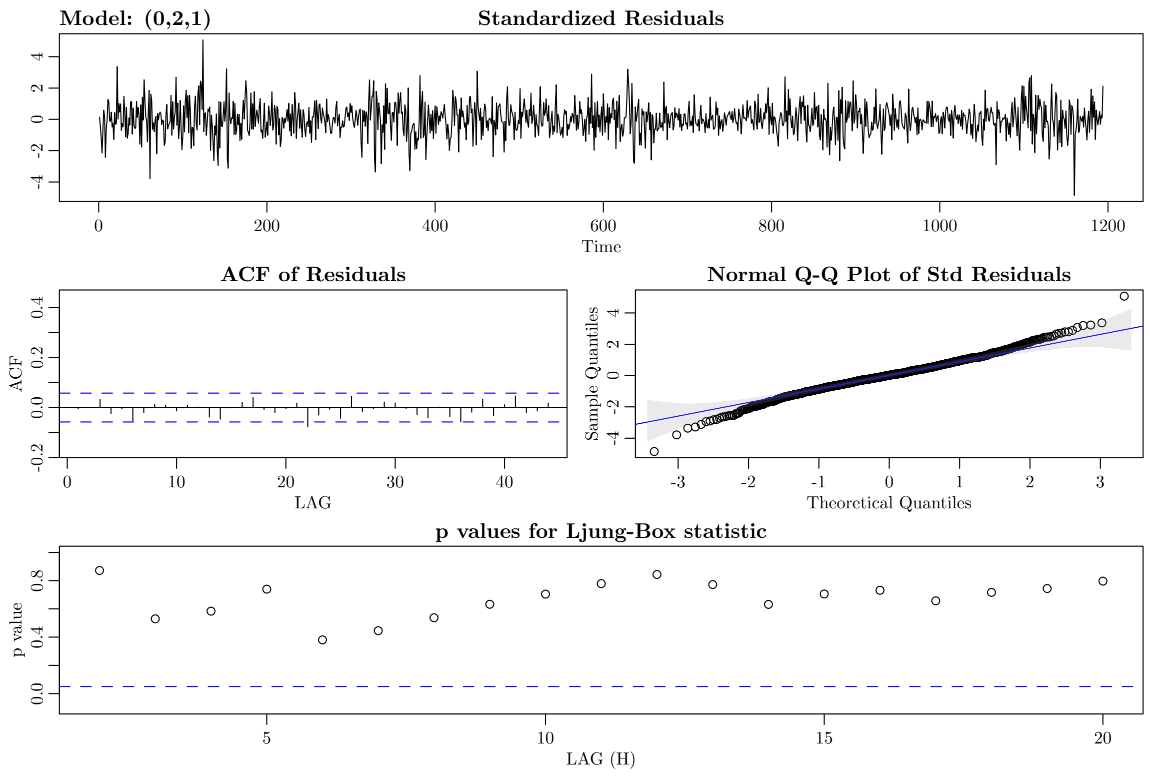

Figure 9: The diagnostic plots for the ARIMA(0, 2, 1) fit.

We notice that the residuals are white noise from the ACF and Standardized Residuals plots. The Normal Q-Q Plot of the Standardized Residuals tells us that our white noise is not perfectly normally distributed; however, it is close enough. Finally, the p values for the Ljung-Box statistic show us that the model fits well, as the null hypothesis H_0 is that the model is a good fit. Overall, this appears to be a good fit for X_t.

Lastly, we examine our ∇X_t residuals to fit an ARIMA model. The ACF and PACF for ∇X_t are below in Figures 10a and b.

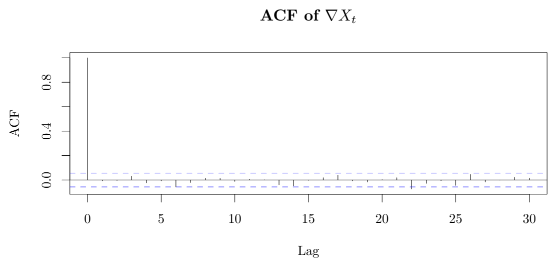

Figure 10a: The ACF of ∇X_t

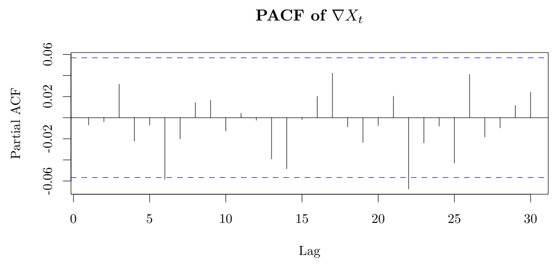

Figure 10b: The PACF of ∇X_t

Again, because all of the correlations are within or barely outside the 95% white noise confidence bounds given by the blue lines, we conclude that we indeed have white noise. This indicates that an ARIMA(0, 1, 0) model is reasonable for X_t.

At this point, all three of these models seem to fit well, and their residuals appear to be white noise. Thus, we can move on to selecting the best model via cross validation. Cross validation is a method by which we estimate the out-of-sample error of the models. It works by first fitting the model on a set of time points in the past, and then generating forecasts for future time points already existing in the data. The forecast error is then calculated by taking the difference between the model's predicted forecasts and the actual values.

3.1 Cross Validation Model Comparison

Model	RMSE
Log Stabilization Transformed + Quadratic Trend + ARIMA(0, 1, 0)	0.6315534
Log Stabilization Transformed + ARIMA(0, 2, 1)	0.6052125
Log Stabilization Transformed + ARIMA(0, 1, 0)	0.6052122

Table 1: These are the out-of-sample RMSEs for our models of interest. Note that the units of the RMSE values are in dollars.

Since the log-stabilized + ARIMA(0, 1, 0) has the lowest RMSE, it is our final model that we will use to make predictions. In addition to having the lowest cross validation RMSE, it is quite simple which is nice.

4 Results

So, our final model is

$$\nabla X_t = \mu + Z_t, \tag{1}$$

where $\mu = \overline{\nabla X_t}$, the average of the differenced X_t. We estimate its lone parameter μ below.

4.1 Estimation of model parameters

Parameter	Estimate (SE)
μ	-5.4 $\times 10^{-4}$ (3×10^{-4})

Table 2: These are our estimates for μ and its corresponding standard error (SE) for the ARIMA model in equation (1).

Although it appears that $\hat{\mu}$ and its standard error are very close to zero, recall that these are on a log scale and so are not as small as they seem.

4.2 Prediction

Given our final model defined in equation (1), we wish to generate forecasts $\hat{Y}_t$ for the next 10 time points. We do so through the following calculations:

$$\begin{aligned}\nabla X_t &= \mu + Z_t \\ X_t - X_{t-1} &= \mu + Z_t \\ \hat{X}_t &= \mu + X_{t-1} \\ e^{\hat{X}_t} &= e^{\mu + X_{t-1}} \\ \hat{Y}_t &= Y_{t-1} e^{\mu}\end{aligned}$$

This is how we generate our predictions! Note that we exponentiate in order to make the units of our predictions dollars, like in the raw data. The predictions (red) for the first 10 trading days of October 2019 are plotted below in Figure 11, along with $\pm 1 \times SE$ (light blue) and $\pm 1.96 \times SE$ (dark blue) prediction intervals. Note that the raw data below is only of the previous 100 trading days of 2019.

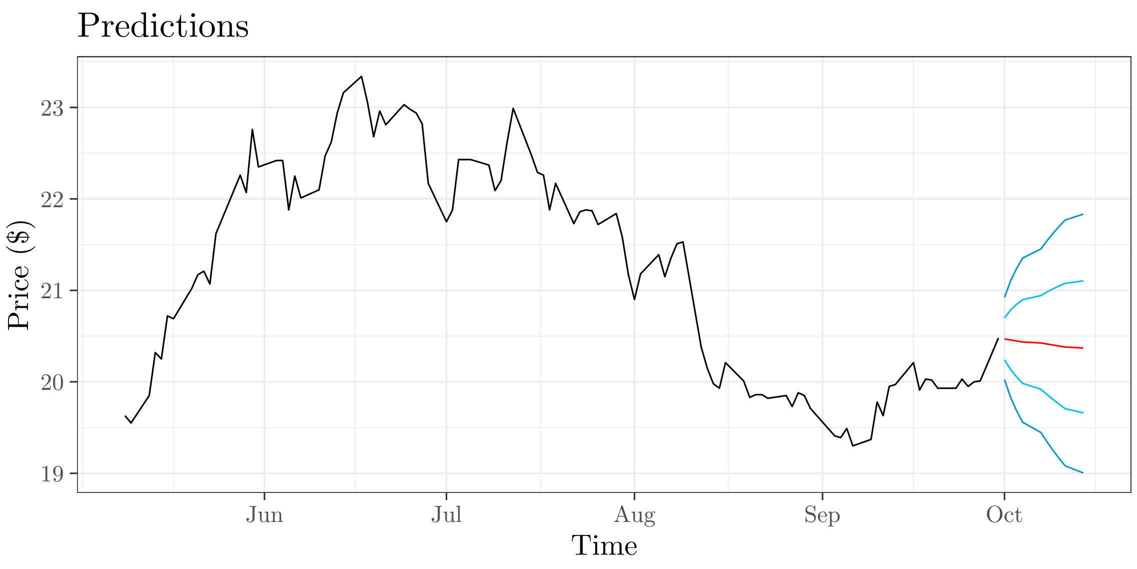

Figure 11: Predictions and Prediction Intervals for Stock Price Oct. 1-14, 2019

So, we see that the stock price for Mediocre Social Network Apps Inc. is predicted to go down over the next 10 trading days. Unfortunately, they should not expect their fortunes to turn around so soon. However, all hope is not lost, as an increase is possible as shown by the prediction intervals.

That's it! Our appendix with all relevant code is attached below.

Appendix

Import libraries

```r
library(forecast)
library(astsa)
library(ggplot2)
```

Import data. Have the csv in the same working directory as the code.

```r
stocks <- read.csv('stocks.csv') # Read in the data
stocks$Date <- as.Date(stocks$Date) # Change Dates to be date types
```

Exploratory Data Analysis

Plotting data [Figure 1]

```r
title1 <- 'Mediocre Social Network Apps Inc. Stock Price over Time'
figure1 <- ggplot(stocks) + geom_line(aes(x=Date, y=Price))
figure1 <- figure1 + labs(x = 'Time', y = 'Price ($)')
figure1 <- figure1 + ggtitle(title1)
figure1 <- figure1 + theme_bw()
figure1
```

Fitting quadratic curve and find its residuals

```r
prices <- stocks$Price
time <- stocks$X
time_2 <- time ** 2
mod <- lm(prices~time + time_2)
stocks$resid <- mod$residuals
stocks$fitted <- mod$fitted.values
```

Plot the quadratic curve [Figure 2 (a)]

```r
title2_a <- 'Quadratic Fit on Stock Price'
figure2_a <- ggplot(stocks) + geom_line(aes(x=Date, y=Price))
figure2_a <- figure2_a + geom_line(aes(x=Date, y=fitted), color = 'red')
figure2_a <- figure2_a + labs(x = 'Time', y = 'Price ($)')
figure2_a <- figure2_a + ggtitle(title2_a)
figure2_a <- figure2_a + theme_bw()
figure2_a
```

Plot the residuals of the quadratic curve [Figure 2 (b)]

```r
title2_b <- 'Residuals of Quadratic Model'
figure2_b <- ggplot(stocks) + geom_line(aes(x=Date, y=resid))
figure2_b <- figure2_b + labs(x = 'Time', y = 'Residual')
figure2_b <- figure2_b + ggtitle(title2_b)
figure2_b <- figure2_b + theme_bw()
figure2_b
```

2 Modeling a Deterministic Function of Time

Take the log of the data, fit the quadratic, and find the residuals

```
stocks$transform <- log(stocks$Price)
mod <- lm(stocks$transform~time + time_2)
stocks$transform_resid <-  mod$residuals
stocks$transform_fitted <- mod$fitted.values
```

Plot the quadratic fit on X_t [Figure 3 (a)]

```
title3_a <- 'Quadratic fit on'~X[t]
figure3_a <- ggplot(stocks) + geom_line(aes(x=Date, y=transform))
figure3_a <- figure3_a + geom_line(aes(x=Date, y=transform_fitted), color = 'red')
figure3_a <- figure3_a + labs(x = 'Time', y = 'Price (S)') + ggtitle(title3_a) + theme_bw()
figure3_a
```

Plot the residuals for the quadratic fit on X_t [Figure 3 (b)]

```
title3_b <- 'Residuals of Quadratic Model on'~X[t]
figure3_b <- ggplot(stocks) + geom_line(aes(x=Date, y=transform_resid))
figure3_b <- figure3_b + labs(x = 'Time', y = 'Residual') + ggtitle(title3_b) + theme_bw()
figure3_b
```

Find and plot the first difference of X_t

```
stocks$diff_poly <- c(0, diff(stocks$transform_resid))
```

Plot the first difference of X_t [Figure 4]

```
title4 <- '$\\nabla(\\epsilon_{t})$'
figure4 <- ggplot(stocks) + geom_line(aes(x=Date, y=diff_poly))
figure4 <- figure4 + labs(x = 'Time', y = 'Difference') + ggtitle(title4) + theme_bw()
figure4
```

Find $\nabla^2 X_t$

```
stocks$diff_2_log <- c(0, 0, diff(stocks$transform, differences = 2))
```

Plot $\nabla^2 X_t$ [Figure 5]

```
title5 <- '$\\nabla^{2}(X_{t})$'
figure5 <- ggplot(stocks) + geom_line(aes(x=Date, y=diff_2_log))
figure5 <- figure5 + labs(x = 'Time', y = 'Difference') + ggtitle(title5)
figure5 <- figure5 + theme_bw()
figure5
```

Find ∇X_t

```
stocks$diff_log <- c(0,diff(stocks$transform))
```

Plot of the first difference of the log data [Figure 6]

```
title6 <- '$\\nabla(X_{t})$'
figure6 <- ggplot(stocks) + geom_line(aes(x=Date, y=diff_log))
figure6 <- figure6 + labs(x = 'Time', y = 'Difference') + ggtitle(title6)
figure6 <- figure6 + theme_bw()
figure6
```

ARIMA Model Selection

Plot the ACF of $\nabla(\epsilon)$ [Figure 7 (a)]

```r
title7_a <- 'ACF of $\\nabla$($\\epsilon_{t}$)'
acf(stocks$diff_poly, main = title7_a)
```

Plot the PACF of $\nabla(\epsilon)$ [Figure 7 (b)]

```r
title7_b <- 'PACF of $\\nabla$($\\epsilon_{t}$)'
pacf(stocks$diff_poly, main = title7_b)
```

Plot the ACF of $\nabla(\epsilon)$ [Figure 8 (a)]

```r
title8_a <- 'ACF of $\\nabla^{2}(X_{t})$'
acf(stocks$diff_2_log, main=title8_a)
```

Plot the PACF of $\nabla(\epsilon)$ [Figure 8 (b)]

```r
title8_b <- 'PACF of $\\nabla^{2}(X_{t})$'
pacf(stocks$diff_2_log, main=title8_b)
```

The diagionistics of ARIMA(0,2,1) for the log data [Figure 9]

```r
sarima(stocks$transform, 0, 2, 1)
```

The ACF and PACF of ∇X_t [Figure 10]

```r
title10_a <- "ACF of $\\nabla(X_{t})"
acf(stocks$diff_log, main=title10_a)
```

The ACF and PACF of ∇X_t [Figure 10]

```r
title10_b <- "PACF of $\\nabla(X_{t})"
pacf(stocks$diff_log, main=title10_b)
```

Cross Validation Model Comparison

A function like sarmia.for but doesn't generate plots. Credit: https://stackoverflow.com/questions/55871256/can-i-prevent-sarima-for-from-plotting

```r
sarima.noplot = function(x, ...) {
  png(tf<-tempfile())
  out <- sarima.for(x, ...)
  dev.off()
  file.remove(tf)
  return(out)
}
```

Calculates the MSE with cross-validation

```r
mse <- function(p, d, q, P, D, Q, S) {
  start_year <- 254
  end_year <- length(stocks$Price)
  sum_squared_errors = 0
  for (year in seq(start_year, end_year, 10)) {
    train_set <- window(stocks$Price, end=year-1)
    test_set <- window(stocks$Price, start=year, end=year + 9)
    forecast <- exp(sarima.noplot(log(train_set),
                                  n.ahead=10, p = p, d = d, q = q, P = P, D = D, Q = Q, S = S)$pred)
    sum_squared_errors <- sum_squared_errors + sum((forecast - test_set)^2)
  }
  mse <- sum_squared_errors / (10*length(seq(start_year, end_year, 10)))
  return(mse)
}
```

Log Stabilization Transformed + ARIMA(0, 2, 1)

```
sqrt(mse(0,2,1,0,0,0,0))
```

Log Stabilization Transformed + ARIMA(0, 1, 0)

```
sqrt(mse(0,1,0,0,0,0,0))
```

Log Stabilization Transformed + Quadratic Trend + ARIMA(0, 1, 0)

```
start_year <- 254
end_year <- length(stocks$Price)
sum_squared_errors <- 0
for (year in seq(start_year, end_year, 10)) {
    train_set <- window(stocks$Price, end=year-1)
    test_set <- window(stocks$Price, start=year, end=year + 9)

    model4 <- lm(log(train_set) ~ poly(1:length(train_set), degree=2, raw=TRUE))
    test_matrix <- model.matrix( ~ poly((length(train_set) + 1):(length(train_set) + 10),
                                       degree=2, raw=TRUE))
    trendforecast4 <- test_matrix %*% model4$coefficients
    temp <- sarima.noplot(model4$residuals,
                         n.ahead = 10, p = 1, d = 0, q = 0, P = 0, D = 0, Q = 0, S = 0)$pred
    forecast4 <- exp(trendforecast4 + temp)
    sum_squared_errors <- sum_squared_errors + sum((forecast4 - test_set)^2)
}

sqrt(sum_squared_errors / (10*length(seq(start_year, end_year, 10))))
```

Information for ARIMA(0,1,0)

```
sarima(stocks$transform, 0, 1, 0)
```

Create predictions

```
last <- tail(stocks, 100)
forecast <- sarima.for(stocks$transform, n.ahead = 10, 0, 1, 0)
predictions <- exp(forecast$pred[1:10])
lower_bound_1 <- exp(forecast$pred[1:10] - forecast$se[1:10])
upper_bound_1 <- exp(forecast$pred[1:10] + forecast$se[1:10])
lower_bound_2 <- exp(forecast$pred[1:10] - 1.96*forecast$se[1:10])
upper_bound_2 <- exp(forecast$pred[1:10] + 1.96*forecast$se[1:10])
last <- last[c('Date', 'Price')]
last$Key <- 'Observations'
new_dates <- c("2019-10-01", "2019-10-02", "2019-10-03", "2019-10-04",
               "2019-10-07", "2019-10-08", "2019-10-09", "2019-10-10",
               "2019-10-11", "2019-10-14")
preds <- data.frame(Date = new_dates, Price = predictions, Key = 'Predictions')
upper_2 <- data.frame(Date = new_dates, Price = upper_bound_2,  Key = 'Upper_2')
lower_2 <- data.frame(Date = new_dates, Price = lower_bound_2,  Key = 'Lower_2')
```

```
upper_1 <- data.frame(Date = new_dates, Price = upper_bound_1,  Key = 'Upper_1')
lower_1 <- data.frame(Date = new_dates, Price = lower_bound_1,  Key = 'Lower_1')
df <- rbind(last, preds, upper_2, lower_2, upper_1, lower_1)
```

Plot prediction [Figure 11]

```
figure11 <- ggplot(df) + geom_line(aes(x=Date, y=Price, color = Key))
figure11 <- figure11 + labs(x = 'Time', y = 'Difference') + ggtitle('Predictions')
figure11 <- figure11 + theme_bw() + theme(legend.position = 'none')
figure11 <- figure11 + scale_colour_manual(values=c(
Observations='black', Predictions='red', Upper_1='deepskyblue', Lower_1='deepskyblue',
Upper_2='deepskyblue3', Lower_2='deepskyblue3'))
figure11
```