STAT 153 Project

Group: Francie McQuarrie
Purva Kapshikar

May 6, 2020

Executive Summary:

We analyzed the stocks dataset, which includes the stock prices for Mediocre Social Network Apps Incorporated from 2015 through September 2019. To forecast their stock prices for the first ten trading days of the fourth quarter, we ultimately chose a linear model with an AR(1) fitted to its residuals as our final model. The forecast can be seen at the end of this report (Figure 7). These predictions indicate that the stocks will continue their downward trend. You can find our blog post at https://fmcquarrie3.github.io/STAT153-Final-Project/.

1 Exploratory Data Analysis

We begin the exploration by plotting the original time series stock prices process:

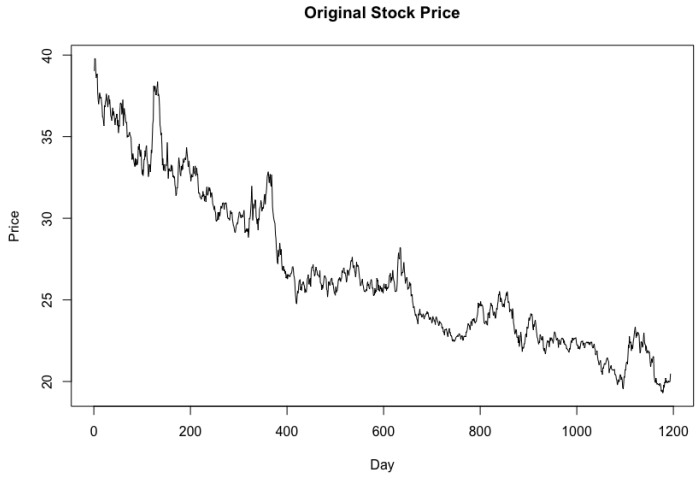

Figure 1: Mediocre Social Network Apps Incorporated stock price from 2015 to September 2019

The data shows a decreasing trend that appears to be either linear or quadratic. There seems to be seasonality: as this is stock data, we tried to take into consideration weekly, quarterly and yearly patterns, but found it difficult to work with the irregularity of having (on average) 253 trading days a year, and not knowing the best way to work with the data when there was fewer than 5 days of data for a week (given holidays). The variance seems to decrease over time and heteroscedasticity was evident in our residuals, so we decided to apply the log variance stabilizing transform on this data for all our models before doing any further fitting, filtering or modelling.

2 Models Considered

We took slightly different approaches for our individual models. Francie focused on achieving stationarity through differencing (seasonal and nonseasonal), while Purva focused on a combination of parametric and nonparametric trend modeling.

2.1 Francie's Model

I decided to focus on differencing (both seasonal and regular) to achieve stationary and then used SARIMA to model the residuals. I tried various combinations of seasonal periods and ARMA parameters. I tried seasonal periods of 12 days, 6 days, and 3 days (mistakenly thinking I was differencing by months at first), as well as 65 days (for roughly quarterly differences, which may be useful for financial data). Surprisingly, once I identified the proper ARMA components (p, q, P, Q) to go with each, all produced models with white noise residuals and high Ljung-Box pvalues. Even more surprising, the model with simply a first difference and nothing else also produced white noise residuals and high Ljung-Box p-values. Comparing the AIC/BIC and RMSE from cross validation all clearly pronounced the first difference-and-nothing-else model the winner. My final model produces the following sarima() output:

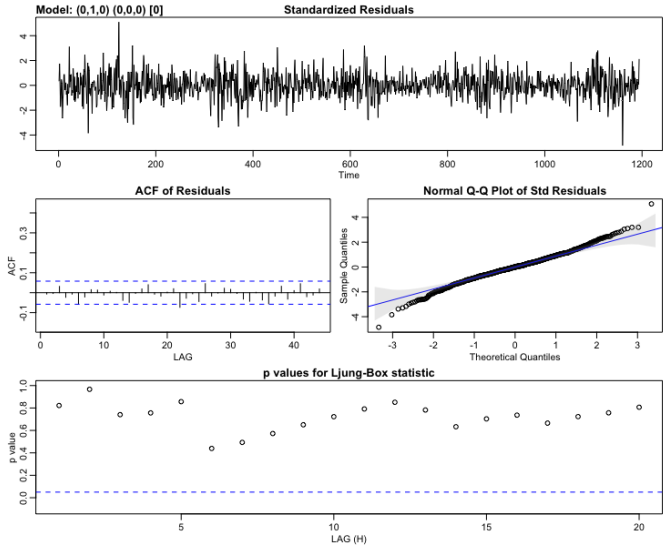

Figure 2: SARIMA output from a first difference model

We can clearly see the residuals of this simple model are white noise from the SARIMA output above.

2.2 Purva's Model

I fit a linear polynomial to the log of the stock prices (Figure 3). The resulting residuals suggested stability, as they had a constant mean of around zero and mostly constant variance.

We see from the decreasing pattern in the ACF plot and the single spike in the PACF plot (Figure 4) that AR(1) may be a reasonable model for the residuals. Below (Figure 5) is also the output from R's sarima() function with the argument $p = 1$. The resulting ACF suggests that the residuals are similar to white noise. The p-values for the Ljung-Box statistic are high, so an AR(1) model seems appropriate.

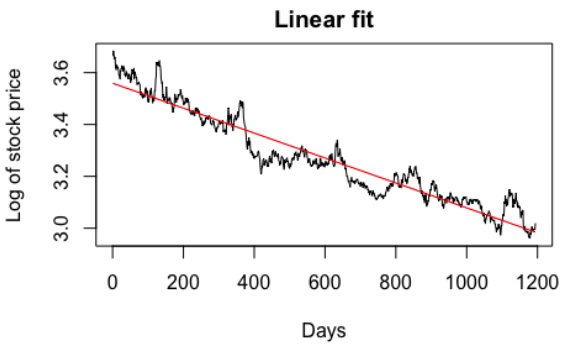

Figure 3: Fitted values from linear model in red.

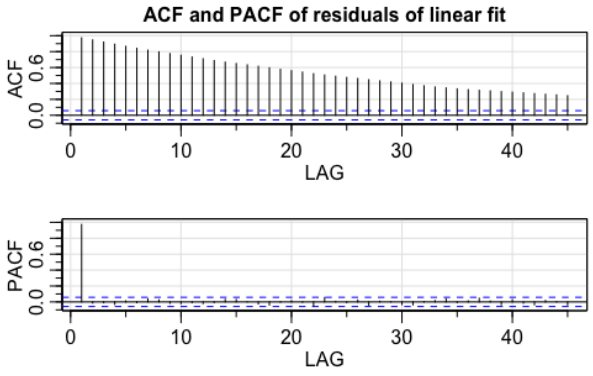

Figure 4: ACF and PACF plots for linear model.

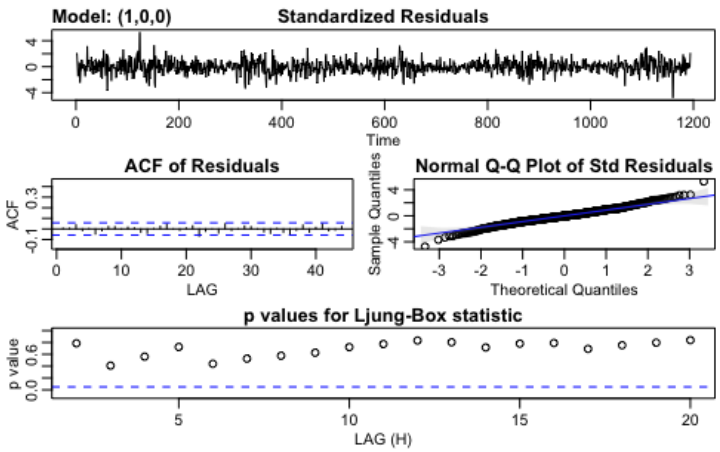

Figure 5: SARIMA output for the linear model.

I briefly incorporated indicator variables corresponding to the month of each datapoint, however, these models had lower AIC/BIC and RMSE values, so I decided to forgo the indicators. I also tried a quadratic model, which had slightly lower AIC/BIC values than the linear model, but had a greater RMSE from cross-validation, so I went with the linear model and AR(1) residuals as my final model.

2.3 The Group Model

Since Francie tried various forms of differencing for modeling, and Purva tried parametric trend fitting, we decided our group model should attempt to use an exponential smoother to fit the data (we used exponential smoothing rather than q-step smoothing so we could have a straightforward way of forecasting new points). This way, we had three different theoretical methods to model the stocks dataset.

All exponential smoothing was done on the log data. Exponential smoothing depends on some parameter α, and this must be chosen before performing the smoothing on the data set. We chose α by choosing a set of candidate α (from 0.1 to 0.9 in increments of 0.1) and performing cross validation on the data. By comparing the sum of squared errors for the different α, we found that $\alpha = 0.1$ produced the smallest squared error. Francie did try seeing if SARIMA could be used to model the residuals of the smoothing estimate. While the ACF of the residuals looked like $MA(1)$, attempting to add this SARIMA aspect to the model and forecasts resulted in a horrendous cross validation error, so the idea was quickly abandoned. Applying this smoother to the data yields the close estimate in Figure 6.

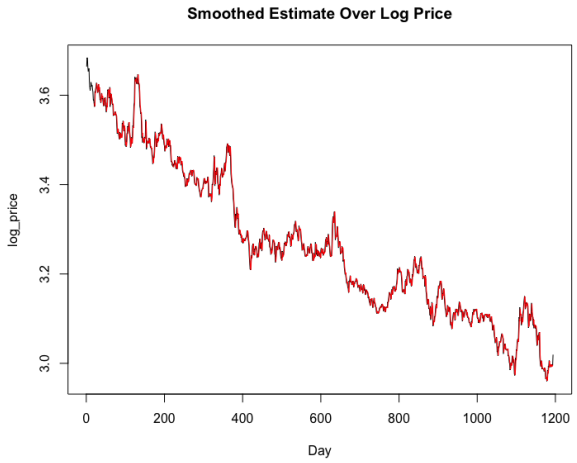

Figure 6: Exponential smoother estimate overlayed on log prices.

3 Model Comparison and Selection

Model Name	Description	RMSE	AIC	BIC
Francie	First Difference	64.0690	-6.142	-6.134
Purva	Linear trend and AR(1)	62.8348	-6.149	-6.136
Group	Exponential smoother	62.9541	N/A	N/A

Table 1: These are the RMSE's for our models of interest from cross-validation, as well as the AIC and BIC values from the sarima() output. We used a log variance stabilizing transform for all.

We computed the root mean squared error above through cross validation. For each model, we iterated through the dataset 100 points at a time. The model was trained on the previous points (accumulating over time) and then predicted the next ten points. The RMSE was computed on the 10 points (which was possible because we were predicting data that we had access to but not trained on) and summed over all iterations. Root mean squared error was computed by taking the difference between our predictions and the true values, squaring them, taking the average, and then the square root.

Note that we were unable to figure out how to calculate AIC/BIC for our exponential smoother model. We compute this model and estimates by hand, and thus could not pull out these criterion from a model object. We were unsure how to compute AIC/BIC by hand from the likelihood.

While the AIC/BIC values of our linear and first differencing model are roughly similar, the linear model performed better in cross validation when predicting out of sample points. While the exponential smoother and linear model had similar cross validation values, the linear model was slightly better and had AIC/BIC to back it up. Thus our final model is the linear trend with AR(1) residuals.

4 Results

Based on our results from Section 3, our model is summarized in the following equations, where Y_t represents the original stock price data:

$$V_t = log(Y_t) \tag{1}$$

$$V_t = mt + b + X_t \tag{2}$$

$$(X_t - \mu) = \phi(X_{t-1} - \mu) + W_t \tag{3}$$

Combining these together, we have

$$Y_t = exp(mt + b + (1 - \phi)\mu + \phi X_{t-1} + W_t) \tag{4}$$

4.1 Estimation of model parameters

With the above model, we estimated the parameters as:

Parameter	Estimate	(s.e)
m	-4.807e-04	4.239e-06
b	3.559	2.924e-03
ϕ	0.9764	0.0063
μ	0.0033	0.0132

Table 2: The parameter estimates and corresponding standard errors for the model given by equation 4.

The estimates and standard errors were determined from the summary and output of the lm() and sarima() functions, respectively.

4.2 Prediction

To compute our predictions, we fit our linear model and SARIMA model on our entire data set (all 1194 values), and then predicted the next 10 timestep prices (accounting for the fact that our models are in terms of log). The forecasts are shown in the plot (Figure 7) below, which only displays the original data from March 2018 to September 2019 – this is 18 months, and includes enough datapoints to make the downward linear trend still evident – in addition to our ten forecasted points representing MSNAI's stock price for the first ten trading days in October 2019.

The forecasts are: 20.45512, 20.43062, 20.40650, 20.38275, 20.35935, 20.33630, 20.31359, 20.29121, 20.26915, 20.24740.

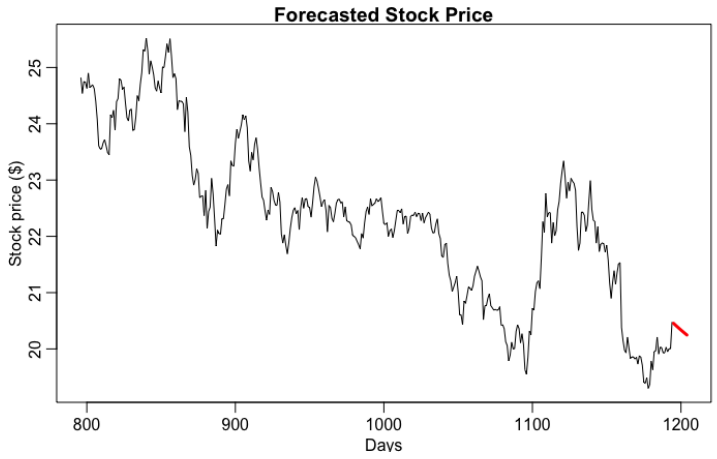

Figure 7: MSNAI stock price from March 2018 to October 2019 in black and forecasted stock prices for the first ten trading days in October 2019 in red.

We are also interested in the uncertainty of our estimates. Figure 8 displays the confidence intervals computed by sarima.for(). Note, however, that since we only used this function to model the residuals, this uncertainty doesn't account for the fitted trend parameters.

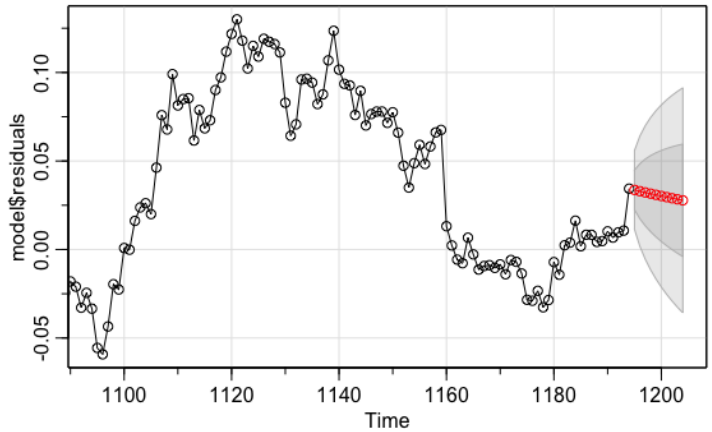

Figure 8: The plot above shows the uncertainty interval from the sarima.for() output, which we used to model the residuals. It only includes the last 100 datapoints for clarity.

As analysts, interpreting the results of Figure 7, we would give the Mediocre Social Networks App company the bad news that their stock prices will continue to decrease in the fourth quarter, with some small fluctuations.

Final Project

Francie McQuarrie

4/25/2020

#Preliminaries

```
library(astsa)
```

Load in the Data

```
stocks <- read.csv("stocks.csv")
head(stocks)
price <- stocks$Price
```

Exploratory Data Analysis

Note: This section will walk through the many models we tried before settling on our final three models. The last section contains the code for our final three models.

Removing Trend and Variance

```
time <- 1:nrow(stocks)
plot(time, price, type = 'l', xlab = "Day", ylab="Price")
title("Original Stock Price")
```

From this plot, we can clearly see a negative linear trend. What happens when we take one difference of the plot?

```
stocks_1_diff <- diff(price, differences = 1)
plot(stocks_1_diff, type = 'l', xlab = "Day", ylab=  "Difference")
title("First Difference of Original Price")
```

While this removes the trend, it also gives us evidence for some uneven variance - the variance is much larger in the beginning of time, probably when the company was first starting out and its market position was precarious.

It's hard to tell from the original picture which type of variance transformation would be best, so I'll try both.

First up is the square root transformation - I'll plot the resulting process and its first difference.

```
sqrt_stock <- sqrt(price)
plot(sqrt_stock, type = 'l')
title("Square Root Prices")
```

```
sqrt_first_diff <- diff(sqrt_stock,  differences = 1)
plot(sqrt_first_diff, type = 'l')
title("First Difference Square Root")
```

Hmm this seemed to have no effect, so let's try logging the data:

```
log_price <- log(stocks$Price)
plot(log_price, type = 'l')
title("Log Price")
```

```
log_diff_1 <- diff(log_price, differences =  1)
plot(log_diff_1, type = 'l', xlab = "Day", ylab = "Difference")
title("First Difference of Log Price")
```

```
acf2(log_diff_1)
```

Ok while this is barely any improvement, the log has slightly less variability, so I'll stick with the logged data from now on.

Out of curiosity, what happens if I take a second difference of the logged data?

```
log_diff_2 <- diff(log_price, differences = 2)
plot(log_diff_2, type =  'l')
title("Second Difference Log")
```

This doesn't seem like it does that much so I'm going to stick with one difference for now.

Modeling Seasonality

There's definitely some seasonal affects going in the data, but it's not super clear from the picture, so I'll try a few periods.

What happens when we try a 12 month seasonality on the first differenced logged data?

```
log_seas_12_diff_1 <- diff(log_diff_1, lag = 12)
plot(log_seas_12_diff_1, type='l', xlab = "Day", ylab = "Difference ")
title("12 month Seasonal Difference of First Difference of Log Price")
```

Ok this looks promising, but let's try a few other different periods:

```
log_seas_6_diff_1 <- diff(log_diff_1, lag = 6)
plot(log_seas_6_diff_1, type='l')
title("6 Seasonal Diff of 1st Diff Log Price")
```

And let's try a quarterly difference as well:

```
log_seas_4_diff_1 <- diff(log_diff_1, lag = 4)
plot(log_seas_4_diff_1, type='l')
title("4 Seasonal Diff of Log Price")
```

It's difficult to tell from the above visuals which works best, so let's try creating acf plots for all three to distinguish them.

```
acf2(log_seas_12_diff_1,
     main = "ACF/PACF of 12 month Seasonal Difference of 1st Difference of Log Prices")
```

From this, we see the clear presence of some seasonal MA and AR components. Let's check out the other differences as well.

```
acf2(log_seas_6_diff_1,  main = "ACF/PACF of 6 month Seasonal Difference of 1st Difference of Log Price
```

Exponetial decreasing in pacf indicates MA, and the one spike at 6 indicates seasonal MA.

Finally,

```
acf2(log_seas_4_diff_1)
```

More exponetial decreasing in pacf which again indicates seasonal MA.

```
log_seas_3_diff_1 <- diff(log_diff_1, lag = 3)
plot(log_seas_3_diff_1, type='l')
title("3 Seasonal Diff of Log Price")
```

```
acf2(log_seas_3_diff_1, main = "ACF/PACF of 3 Month Seasonal Difference of 1st Difference of Log Data")
```

Francie Model Building

Exploring the seasonality didn't give me a clear indication of which one was best, so I'll try models for all of them and see what happens.

First up, I'll try a p = 0, d = 1, q = 0, S = 11, P = 3, D = 1, Q = 1 ,model. The d is to remove the trend, the S = 11 is because the spikes of acf are at 11, and the P = 3 is for the 3 pacf spikes and the Q = 1 is for one MA spike. The D = 1 is for the one seasonal difference I did

```
m1  <- sarima(log_price, p = 0, d = 1,  q = 0, S = 12, P = 3, D = 1, Q = 1 )
```

Oh wow these are some great results! The quick glance at the acf shows no significant spikes, and the pvalues for the Ljung-Box test are all high.

Just for fun, let's see what happens when I try the other models

```
m2 <- sarima(log_price, p = 0, d = 1, q = 0, S = 6, P = 0, D= 1, Q = 1)
```

```
m3 <-  sarima(log_price, p = 0, d = 1, q = 0, S = 4, P = 0, D= 1, Q = 1)
```

```
m4 <-  sarima(log_price, p = 0, d = 1, q = 0, S = 3, P = 0, D= 1, Q = 1)
```

Oooo hmm this is a good model as well. How to choose?

Let's check AIC, AICc, BIC

```
m1$AIC
m2$AIC
m3$AIC
m4$AIC
```

```
m1$AICc
m2$AICc
m3$AICc
m4$AICc
```

```
m1$BIC
m2$BIC
m3$BIC
m4$BIC
```

We always want the smallest one, so all three candidates say that the second and third model are best. But the models with different seasonality have the same parameters, so would it be better to choose a seasonality of 4 to make more sense with quarterly stocks?

```
m3$fit$residuals
```

I realize that all of my stuff above was assumming a 12 seasonal difference was 12 month, but that's not true it's 12 days. Now I will try 65 days, which is about a quarter of a year in a days.

```
seas_diff_65 <- diff(log_diff_1, lag = 65)
plot(seas_diff_65, type='l')
```

```
acf2(seas_diff_65)
```

I will also try Monday-Friday (5 day) differencing

```
mo_fri <- 365*(5/7)
seas_mo_fri <- diff(log_diff_1, lag <- mo_fri)
plot(seas_mo_fri, type = 'l')
```

```
acf2(seas_mo_fri)
```

Purva Model Building

```
date = stocks$Date
# Quadratic trend + AR(1) residuals:
quad_m = lm(log_price ~ time + I(time^2))
plot(time, log_price, type="l", main="Quadratic fit", xlab="Days")
lines(quad_m$fitted.values, type="l", col="red", lwd=2)
plot(time, quad_m$residuals, type="l", xlab="Days", main="Residuals")
acf2(quad_m$residuals)
sarima(quad_m$residuals, p=1,d=0,q=0)

# Monthly indicators (ugly code and bad coding practices but...it...works)
jan = rep(0, 1194)
feb = rep(0, 1194)
mar = rep(0, 1194)
apr = rep(0, 1194)
may = rep(0, 1194)
jun = rep(0, 1194)
jul = rep(0, 1194)
aug = rep(0, 1194)
sep = rep(0, 1194)
oct = rep(0, 1194)
nov = rep(0, 1194)
for (i in time) {
  jan[i] = ifelse(substr(toString(date[i]), 6, 7)=="01", 1, 0)
  feb[i] = ifelse(substr(toString(date[i]), 6, 7)=="02", 1, 0)
  mar[i] = ifelse(substr(toString(date[i]), 6, 7)=="03", 1, 0)
  apr[i] = ifelse(substr(toString(date[i]), 6, 7)=="04", 1, 0)
  may[i] = ifelse(substr(toString(date[i]), 6, 7)=="05", 1, 0)
  jun[i] = ifelse(substr(toString(date[i]), 6, 7)=="06", 1, 0)
  jul[i] = ifelse(substr(toString(date[i]), 6, 7)=="07", 1, 0)
  aug[i] = ifelse(substr(toString(date[i]), 6, 7)=="08", 1, 0)
  sep[i] = ifelse(substr(toString(date[i]), 6, 7)=="09", 1, 0)
  oct[i] = ifelse(substr(toString(date[i]), 6, 7)=="10", 1, 0)
  nov[i] = ifelse(substr(toString(date[i]), 6, 7)=="11", 1, 0)
}


# Linear trend + monthly indicators + AR(1) residuals
linear_ind_m = lm(log_price ~ time + jan + feb + mar + apr + may + jun + jul + aug + sep + oct + nov)
```

```r
plot(time, log_price, type="l", ylab="Log of stock price", xlab="Days", main="Linear + monthly indicato
lines(time, linear_ind_m$fitted.values, col="red", lwd=3)
plot(time, linear_ind_m$residuals, type="l", ylab="Residuals", xlab="Days", main="Residuals of linear i
sarima(linear_ind_m$residuals, p=1, d=0, q=0)

# Quadratic trend + monthly indicators + AR(1) residuals
quad_ind_m = lm(log_price ~ time + I(time^2) + jan + feb + mar + apr + may + jun + jul + aug + sep + oc
plot(time, log_price, type="l", ylab="Log of stock price", xlab="Days", main="Quadratic + monthly indic
lines(time, quad_ind_m$fitted.values, col="red", lwd=3)
plot(time, quad_ind_m$residuals, type="l", ylab="Residuals", xlab="Days", main="Residuals of linear ind
sarima(quad_ind_m$residuals, p=1, d=0, q=0)

## Cross validation and RMSE for explored models
ten_steps <-seq(0, 1080, by = 10)
root_mean_squared_errors <- c(quad_ind = 0, quad = 0, linear_ind = 0)
for (i in ten_steps)  {
  training_set_prices <- log_price[1:(100+i)]
  test_set_prices <- log_price[(100+i+1):(100+i+10)]

  # Seasonal dummy matrix for training_set
  training_matrix <- matrix(0L, nrow=(100+i), ncol=11, byrow=TRUE)
  for (j in 1:(100+i)) {
    month = as.numeric(substr(toString(date[j]), 6,7))
    if (0 < month && month < 12) {
      training_matrix[(j-1)*(11) + month] = 1
    }
  }
  training_matrix = matrix(training_matrix, nrow=(100+i), byrow=TRUE)

  # Seasonal dummy matrix for test_set
  testing_matrix <- matrix(0L, nrow=10, ncol=11, byrow=TRUE)
  for (j in 1:10) {
    month = as.numeric(substr(toString(date[j+100+i]), 6,7))
    if (0 < month && month < 12) {
      testing_matrix[(j-1)*(11) + month] = 1
    }
  }
  testing_matrix = matrix(testing_matrix, nrow=10, byrow=TRUE)

  # Second order polynomial + monthly indicators + AR(1) residuals
  quad_ind_model <- lm(training_set_prices ~ poly(1:(100+i), degree=2, raw=TRUE) + training_matrix)
  test_matrix <- model.matrix( ~ poly((100+i+1):(100+i+10), degree=2, raw=TRUE) + testing_matrix)
  coef = quad_ind_model$coefficients
  coef[is.na(coef)] <- 0
  quad_ind_ar1_forecast <- test_matrix %*% coef + sarima.for(quad_ind_model$residuals, n.ahead = 10, p =

  # Second order polynomial + AR(1) residuals
  quad_model <- lm(training_set_prices ~ poly(1:(100+i), degree=2, raw=TRUE))
  test_matrix <- model.matrix( ~ poly((100+i+1):(100+i+10), degree=2, raw=TRUE))
  quad_ar1_forecast <- test_matrix %*% quad_model$coefficients + sarima.for(quad_model$residuals, n.ahea

  # First order polynomial + monthly indicators + AR(1) residuals
  linear_ind_model <- lm(training_set_prices ~ poly(1:(100+i), degree=1, raw=TRUE) + training_matrix)
```

```
  test_matrix <- model.matrix( ~ poly((100+i+1):(100+i+10), degree=1, raw=TRUE) + testing_matrix)
  coef = linear_ind_model$coefficients
  coef[is.na(coef)] <- 0
  linear_ind_ar1_forecast <- test_matrix %*% coef + sarima.for(linear_ind_model$residuals, n.ahead = 10

  root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(quad_ind_ar1_forecast) - e
  root_mean_squared_errors[2] = root_mean_squared_errors[2] + sqrt(mean((exp(quad_ar1_forecast) - exp(t
  root_mean_squared_errors[3] = root_mean_squared_errors[3] + sqrt(mean((exp(linear_ind_ar1_forecast) -
}

# Last 4 points
training_set_prices <- log_price[1:1190]
test_set_prices <- log_price[1191:1194]

# Seasonal dummy matrix for training_set
training_matrix <- matrix(0L, nrow=1190, ncol=11, byrow=TRUE)
for (j in 1:1190) {
  month = as.numeric(substr(toString(date[j]), 6,7))
  if (0 < month && month < 12) {
    training_matrix[(j-1)*(11) + month] = 1
  }
}
training_matrix = matrix(training_matrix, nrow=1190, byrow=TRUE)

# Seasonal dummy matrix for test_set
testing_matrix <- matrix(0L, nrow=4, ncol=11, byrow=TRUE)
for (j in 1:4) {
  month = as.numeric(substr(toString(date[j+1190]), 6,7))
  if (0 < month && month < 12) {
    testing_matrix[(j-1)*(11) + month] = 1
  }
}
testing_matrix = matrix(testing_matrix, nrow=4, byrow=TRUE)

# Second order polynomial + monthly indicators + AR(1) residuals
quad_ind_model <- lm(training_set_prices ~ poly(1:1190, degree=2, raw=TRUE) + training_matrix)
test_matrix <- model.matrix( ~ poly(1191:1194, degree=2, raw=TRUE) + testing_matrix)
coef = quad_ind_model$coefficients
coef[is.na(coef)] <- 0
quad_ind_ar1_forecast <- test_matrix %*% coef + sarima.for(quad_ind_model$residuals, n.ahead = 4, p = 1

# Second order polynomial + AR(1) residuals
quad_model <- lm(training_set_prices ~ poly(1:1190, degree=2, raw=TRUE))
test_matrix <- model.matrix( ~ poly(1191:1194, degree=2, raw=TRUE))
quad_ar1_forecast <- test_matrix %*% quad_model$coefficients + sarima.for(quad_model$residuals, n.ahead

# First order polynomial + monthly indicators + AR(1) residuals
linear_ind_model <- lm(training_set_prices ~ poly(1:1190, degree=1, raw=TRUE) + training_matrix)
test_matrix <- model.matrix( ~ poly(1191:1194, degree=1, raw=TRUE) + testing_matrix)
coef = linear_ind_model$coefficients
coef[is.na(coef)] <- 0
linear_ind_ar1_forecast <- test_matrix %*% coef + sarima.for(linear_ind_model$residuals, n.ahead = 4, p
```

```
root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(quad_ind_ar1_forecast) - exp
root_mean_squared_errors[2] = root_mean_squared_errors[2] + sqrt(mean((exp(quad_ar1_forecast) - exp(tes
root_mean_squared_errors[3] = root_mean_squared_errors[3] + sqrt(mean((exp(linear_ind_ar1_forecast) - e

root_mean_squared_errors
```

Testing Exponential Smoother

I want to try an exponential smoother but need to use cross validation to find the best value of

```
a_range <- seq(0.1, 0.99, by = 0.1)
sse<- c()
for (a in a_range) {
  train <- stocks[1:1190, 'Price']
  log_train <- log(train)
  test <- stocks[1191:1194, 'Price']
  weights = (1-a)/a * (a^(1:1190))
  weights = weights/sum(weights)
  # Reverse data since we'll be chopping off the earliest data points
  data = rev(log_train)
  for (i in 1:4) {
    # not including some last points as we add new forecasts
    y_i <- sum(weights %*% (data[1:1190]))
    # Add forcasted point to next part for next forcase
    data <- c(y_i, data)
  }
  forecast <- data[1:4]
  error <- sum((test - exp(forecast))^2)
  sse <- c(sse, error)
}
sse
```

```
best_a <- a_range[which.min(sse)]
best_a
```

```
#Make a little function to quickly forecast for later on
expo_forecast <- function(train, num_test, a) {
  # Make weights the length of current training length (since it changes)
  weights = (1-a)/a * (a^(1:length(train)))
  weights = weights/sum(weights)

  #print(length(weights))
  #print(length(data[1:length(train)]))
  # Reverse since we'll be chopping of the earliest time points
  log_train <- log(train)
  data = rev(log_train)
  for (i in 1:num_test) {
    # Compute smoother estimate
    y_i <- sum(weights %*% data[1:length(train)])
    # add to front of data to be used in the next forecast
    data <- c(y_i, data)
  }
  # Pull out just the forecasts
```

```
  forecast <- data[1:num_test]
  # Log will be undo later
  return (forecast)
}
```

```
# For plot
a = 0.1
weights = (1-a)/a * (a^(1:20))
weights = weights/sum(weights)

m = filter(log_price,sides = 1,filter=c(0,weights))
plot(log_price, type = 'l', xlab= "Day", main = "Smoothed Estimate Over Log Price")
lines(m, col='red', lwd  = 1)
```

What happens when I try to use SARIMA to model the residuals?

```
res <- log_price - m
acf2(res)
```

```
sarima(res, p =  0, d = 0, q = 1, S = 0, P = 0, D= 0, Q = 0)
```

This looks promising - we can test it with cross validation later on to see if it works.

Final Models to Test

```
# white noise acf, good pvalues
m1 <- sarima(log_price,  p = 0, d = 1, q  = 0, S = 0, P = 0, D = 0, Q  = 0, details = FALSE)
# white noise acf, good pvalues,theta estimate is 1 bad
m2 <- sarima(log_price,  p = 0, d = 2, q  = 1, S = 0, P = 0, D = 0, Q  = 0, details = FALSE)
# # white noise acf, good pvalues, theta estimate close to 1 bad
m3 <- sarima(log_price, p = 0, d = 1, q = 0, S = 5,  P =0, D = 1, Q  = 1, details = FALSE)
```

Purva:

```
### Purva's model: Linear trend + AR(1) residuals
stocks = read.csv("stocks.csv")
time = 1:length(stocks$Price)
date = stocks$Date
#stocks = stocks$Price
#log_stocks = log(stocks)

linear_m = lm(log_price ~ time)
plot(time, log_price, type="l", main="Linear fit", xlab="Days", ylab="Log of stock price")
lines(linear_m$fitted.values, type="l", col="red")
plot(time, linear_m$residuals, type="l", main="Residuals of linear fit")
acf2(linear_m$residuals, main="ACF and PACF of residuals of linear fit")
sarima(linear_m$residuals, p=1,d=0,q=0, no.constant=TRUE)

## Cross-validation and RMSE
ten_steps <-seq(0, 1080, by = 10)
root_mean_squared_errors <- c(linear = 0)
for (i in ten_steps)  {
  training_set_prices <- log_price[1:(100+i)]
  test_set_prices <- log_price[(100+i+1):(100+i+10)]
```

```
  linear_model <- lm(training_set_prices ~ poly(1:(100+i), degree=1, raw=TRUE))
  test_matrix <- model.matrix( ~ poly((100+i+1):(100+i+10), degree=1, raw=TRUE))
  linear_ar1_forecast <- test_matrix %*% linear_model$coefficients + sarima.for(linear_model$residuals,

  root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(linear_ar1_forecast) - exp
}

# Last 4 points
training_set_prices <- log_price[1:1190]
test_set_prices <- log_price[1191:1194]

linear_model <- lm(training_set_prices ~ poly(1:1190, degree=1, raw=TRUE))
test_matrix <- model.matrix( ~ poly(1191:1194, degree=1, raw=TRUE))
linear_ar1_forecast <- test_matrix %*% linear_model$coefficients + sarima.for(linear_model$residuals, n

root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(linear_ar1_forecast) - exp(t
```

```
root_mean_squared_errors
```

Cross Validation + Model Selection

We'll want to decide betwee models so I'll try practicing how to cross validate

```
# From post @277
# loop on i = 0,10,20,30,... through the end of your data.
# training set = data[1:(100+i)]
# test set = data[(100+i+1):(100+i+10) ]
# fit model on training data
# forecast next 10 days.
# record error on those 10 days: test set - forecast[1:10]


#our data is 1194 rows
# We go 100 at a time
# Will do the last four points separately
ten_steps <-seq(0, 1080, by = 10)
root_mean_squared_errors <- c(model1=0, model2=0, model3=0, model4=0)

# For exponential smoother
a = 0.1

for (i in ten_steps)  {
  training_set_prices <- stocks[1:(100+i), 'Price']
  test_set_prices <-  stocks[(100+i+1):(100+i+10),  'Price']
  #  Fit and prepare for forceast
  forecast1 <- sarima.for(log(training_set_prices), n.ahead = 10, p = 0, d = 1, q = 0, S = 0, P = 0, D =
  forecast2 <- sarima.for(log(training_set_prices), n.ahead = 10, p = 0, d = 2, q = 1, S = 0, P = 0, D =
  forecast3 <- sarima.for(log(training_set_prices), n.ahead = 10, p = 0, d = 1, q = 0, S = 5, P = 0, D =
  forecast4 <- expo_forecast(training_set_prices, length(test_set_prices), a)

  # Seeing if adding sarima modeling to residuals does any good
  # DID NOT WORK
```

```
  #res <- exp(forecast4)  - log(test_set_prices)
  #forecast5 <- expo_forecast(training_set_prices, length(test_set_prices), a) + sarima.for(res, n.ahea

  # Since model built on log data, must exp forecasts to compare to test set
  root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(forecast1) - test_set_price
  root_mean_squared_errors[2] = root_mean_squared_errors[2] + sqrt(mean((exp(forecast2) - test_set_price
  root_mean_squared_errors[3] = root_mean_squared_errors[3] + sqrt(mean((exp(forecast3) - test_set_price
  root_mean_squared_errors[4] = root_mean_squared_errors[4] + sqrt(mean((exp(forecast4) - test_set_price
  #root_mean_squared_errors[5] = root_mean_squared_errors[5] + sqrt(mean((exp(forecast5) - test_set_pri


}
# Get the last four datapoints
training_set_prices <- stocks[1:1190, 'Price']
test_set_prices <-  stocks[1191:1194,  'Price']
#  Fit and prepare for forceast
forecast1 <- sarima.for(log(training_set_prices), n.ahead = 4, p = 0, d = 1, q = 0, S = 0, P = 0, D = 0
forecast2 <- sarima.for(log(training_set_prices), n.ahead = 4, p = 0, d = 2, q = 1, S = 0, P = 0, D = 0
forecast3 <- sarima.for(log(training_set_prices), n.ahead = 4, p = 0, d = 1, q = 0, S = 5, P = 0, D = 1
forecast4 <- expo_forecast(training_set_prices, length(test_set_prices), a)

# Seeing if adding sarima modeling does any good
# DID NOT WORK
#res <- exp(forecast4)  - log(test_set_prices)
#forecast5 <- expo_forecast(training_set_prices, length(test_set_prices), a) + sarima.for(res, n.ahead

# Since model built on log data, must exp forecasts to compare to test set
root_mean_squared_errors[1] = root_mean_squared_errors[1] + sqrt(mean((exp(forecast1) - test_set_prices
root_mean_squared_errors[2] = root_mean_squared_errors[2] + sqrt(mean((exp(forecast2) - test_set_prices
root_mean_squared_errors[3] = root_mean_squared_errors[3] + sqrt(mean((exp(forecast3) - test_set_prices
root_mean_squared_errors[4] = root_mean_squared_errors[4] + sqrt(mean((exp(forecast4) - test_set_prices
#root_mean_squared_errors[5] = root_mean_squared_errors[5] + sqrt(mean((exp(forecast5) - test_set_price
```

```
root_mean_squared_errors
```

Forcasting

```
## Forecasting next 10 datapoints
old_data <- rep(NA, 10)
old_data = c(price, old_data)

# Linear trend + AR(1) residuals
model <- lm(log_price ~ poly(1:1194, degree=1, raw=TRUE))
test_matrix <- model.matrix( ~ poly(1195:1204, degree=1, raw=TRUE))
forecast <- exp(test_matrix %*% model$coefficients +  sarima.for(model$residuals, n.ahead = 10, p = 1, d

# Plotting only March 2018 through September 2019 + the 10 points of October 2019
plot(796:1204, old_data[796:1204], type='l', col = "black", xlab="Days", ylab="Stock price ($)", main="
lines(1195:1204, forecast, col="red", lwd=3)
forecast
```