Time Series Analysis of Mediocre Social Network Apps Incorporated Stock Data

Blog Link: https://rpubs.com/Smeet/STAT153-Blog

May 7, 2020

Executive Summary:

Our group explored and modeled Mediocre Social Network Apps Incorporated's stock data using time series analysis methods on its daily historical prices. Our main goal is to predict prices for the next ten trading days using Autoregressive Integrated Moving Average (ARIMA) and exponential smoothing models. We start out by exploring the dataset using typical time series exploratory techniques such as analyzing ACF/PACF plots, differencing, and applying a variance-stabilizing transform on the data in pursuit of homoskedasticity. After comparing individual model metrics and diagnostics, we collectively decided to implement a "random walk" ARIMA(0,1,0) model in order to make the most accurate and realistic stock price predictions based on empirical analysis and stock market theory.

1 Exploratory Data Analysis

We started exploring the dataset by plotting stock price with several other transformations of the dataset as seen below (Figure 1).

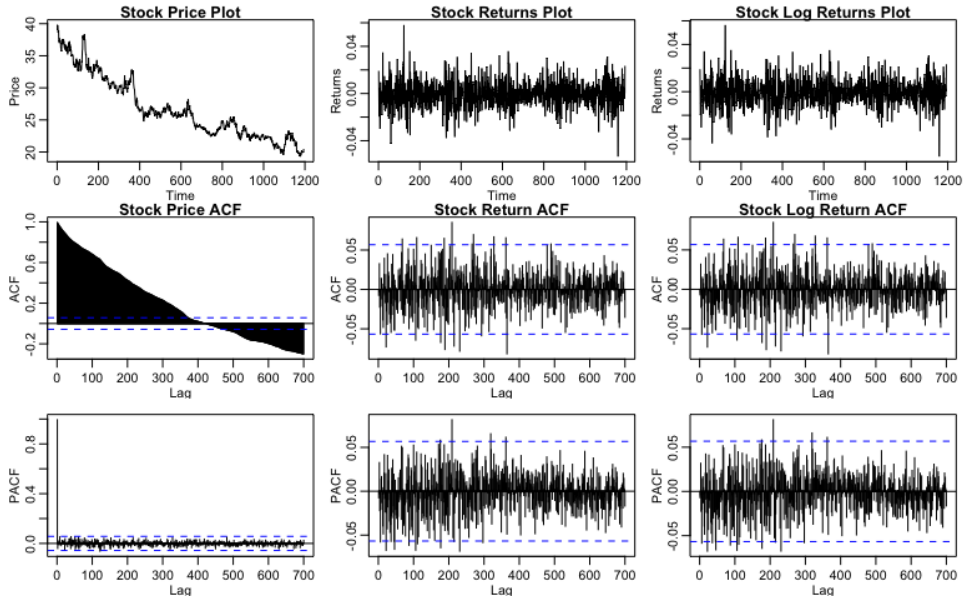

Figure 1: *Daily stock price [1,1], simple stock returns [1,2], log returns [1,3], stock price ACF [2,1], simple stock returns ACF [2,2], log returns ACF [2,3], stock price PACF [3,1], simple stock returns PACF [3,2], log returns PACF [3,3]*

Observing the plots above, we can see that the non-stationarity exhibited by the daily stock price series can be converted into a stationary series by transforming the prices into returns. Now, the series seems to have a constant mean and variance.

By taking the natural logarithm of our price data and differencing the data set, we obtain continuously compounded returns or log returns:

$$r_t = ln(1 + R_t) = ln\Big(\frac{P_t}{P_{t-1}}\Big) = p_t - p_{t-1}$$

where r_t is the log return, R_t is the simple return, P_t is the price at time t, and $p_t = ln(P_t)$.

Another approach to achieve stationarity in the series was to implement a variance-stabilizing transform by taking the log of the stock price data (Figure 2).

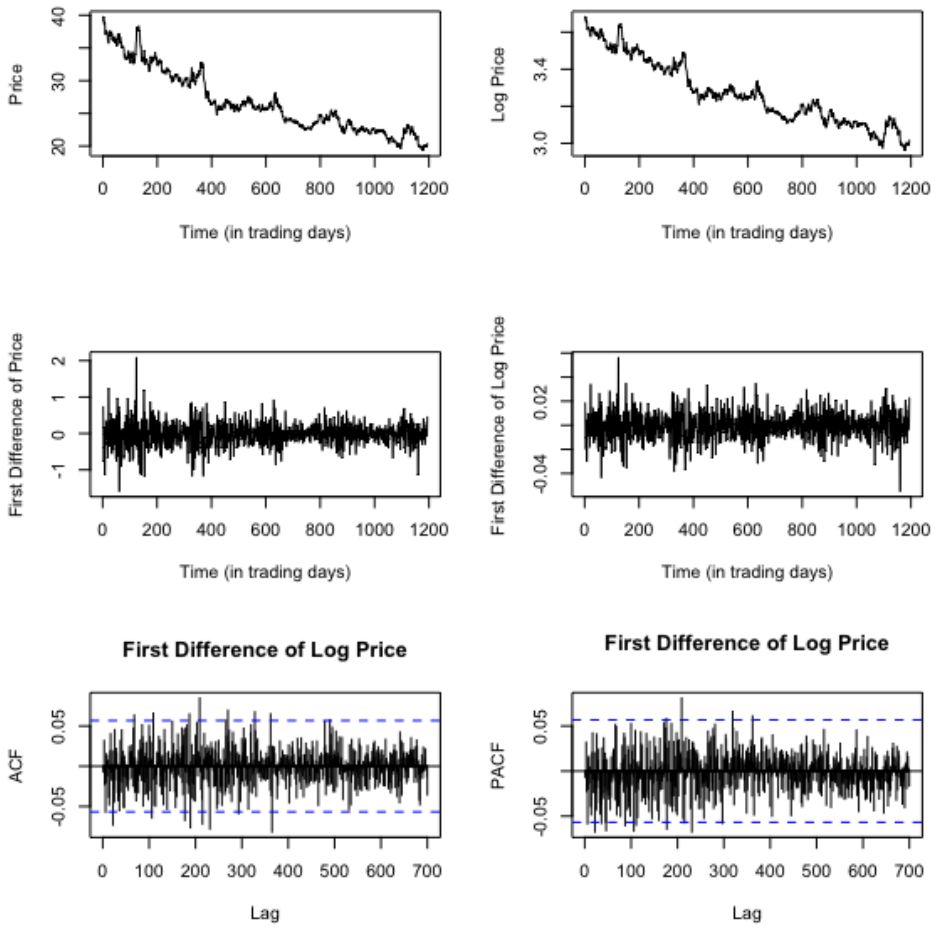

Figure 2: *Original and log stock price plots (top), first difference of original and log price plots (middle), and ACF/PACF of differenced log price (bottom)*

It is clear looking at the plots above that the differenced log data achieves better homoskedasticity through a simple shoebox test. When comparing the left and right halves of the differenced stock data, the left half presents much larger spikes, or variances. This is not the case when comparing the left and right halves of the differenced log data, where the two halves appear far more symmetrical and hence more homoskedastic. As for the ACF and PACF plots of the differenced log price data, though there are a few particularly significant spikes, the noise appears to generally behave like white noise since we can expect to see five autocorrelation spikes per one hundred lags and still deem the noise to be white noise. Recall that the dashed blue lines indicate 95% confidence intervals for white noise.

2 Models Considered

Smeet and Pedram decided to take similar approaches in their model design by taking the log transform of the stock price data and then assigning their own preferred ARIMA models to the variance-stabilized data. Rex decided to fit an exponential smoothing model, taking into account many past values and speculating that Mediocre's stock prices are correlated with several past values and hence hold predictive power.

2.1 Smeet's Model

Log Price + ARIMA(1,1,1):

$$f(X_t) = \mu + \phi_1 f(X_{t-1}) + \theta_1 W_{t-1} + W_t$$

$f(X_t)$	μ	ϕ_1	θ_1
$log(X_t)$	-0.0005	-0.0023	-0.0047

Table 1: Model parameter specifications and estimates

I have used log returns because the series is stationary and applying cross-validation to several models becomes an easier task. In the dataset given using log returns makes sense since they are approximately equal to simple returns.

From the formula above we can connect logarithmic and simple returns $r_t = ln(1 + R_t)$ shows that as long as R_t is not too large then logarithmic and simple returns are very similar in size.

After exploring the dataset, it was clear the popular assumption of efficient market hypothesis is valid (at least in this scenario and time granularity) so a random walk model should be the only valid prediction that supports the theory.

When modelling and deciding between several models, I observed that the addition of a constant in the model has a dramatic impact on predictions and can increase the volatility of my cross-validation scores. Intuitively a random walk with drift in the model makes sense since the overall trajectory of the price has been going down and a negative drift will further push down the predicted prices.

2.2 Pedram's Model

Log Price + ARIMA(1,1,0):

$$X_t = \mu + X_{t-1} + \phi_1(X_{t-1} - X_{t-2}) + W_t$$

μ	ϕ_1
-0.0005	-0.0070

Table 2: Model parameter estimates

I chose a differenced first-order autoregressive model to account for any autocorrelated errors in the random walk. This problem can be fixed by regressing the first difference of X_t, the log price, on itself and differencing once more. Adding a constant improved the various information criteria provided below while slightly decreasing the MSE compared to various other ARIMA models tested.

2.3 Rex's Model

Exponential smoothing:

$$X_t = \frac{1-\alpha}{\alpha}(\alpha X_{t-1} + \alpha^2 X_{t-2} + ...\alpha^n X_{t-n})$$

Exponential smoothing involves assigning weights to past values in order to forecast future values. The number of past values to choose and the magnitude of the weights are related parameters which need to be determined. n is the number of past terms used in the prediction, α is the weighting parameter, and $\frac{1-\alpha}{\alpha}$ is the normalizing factor. I chose the weighting parameter to be .99, meaning the stock price today depends on a large amount of past terms since the weighting is only decreasing by 1 percent each term. I decided on a large weighting parameter because I speculated that the price of a stock is correlated with many past values. This model can be thought of as an AR process without white noise.

2.4 Group Model

After comparing individual models, we as a group decided to pursue a pure "random walk" ARIMA(0,1,0) on the log price data and in doing so believe that stationarity, and thus predictive power, is best achieved given various model diagnostics and testing criteria (Figure 3).

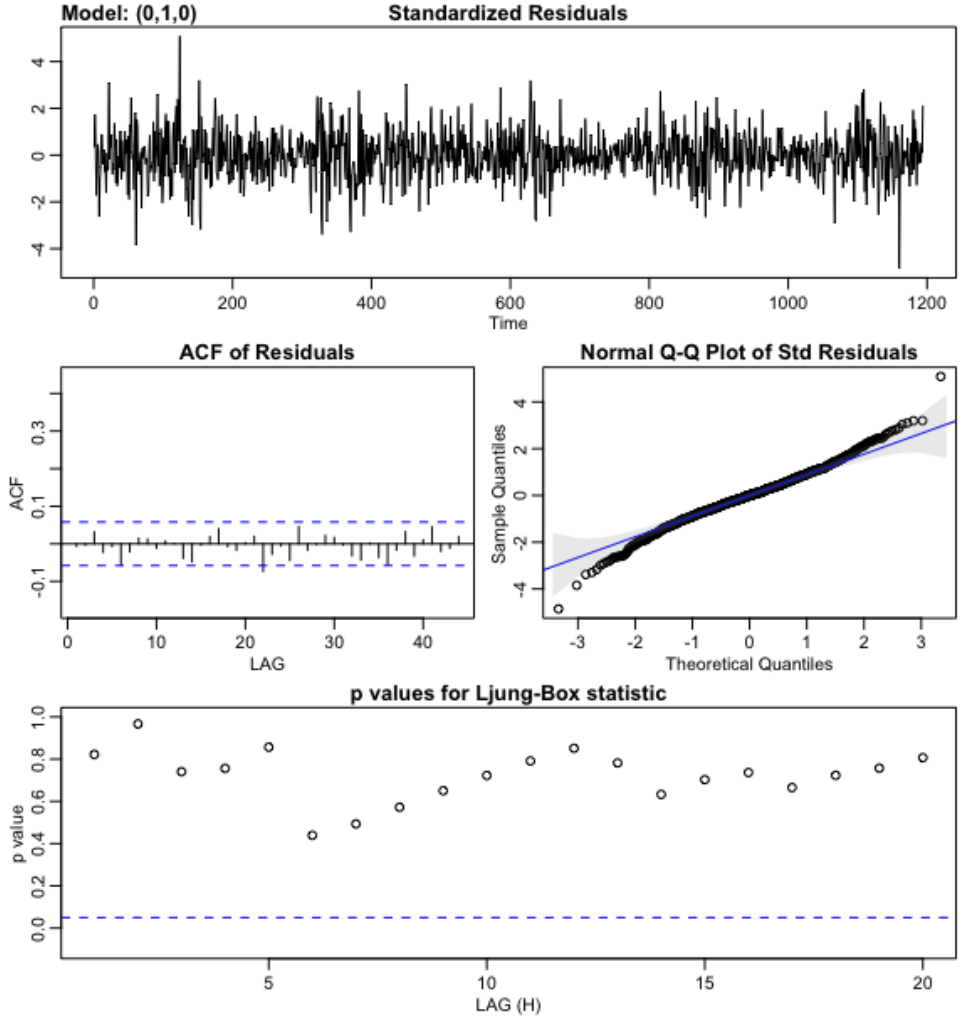

Figure 3: *Group model residual plot (top), ACF of residuals (middle left), Q-Q plot (middle right), and p-value plot for the Ljung-Box statistic (bottom)*

Assessing the residual plot, we notice zero-mean residuals but it is difficult to pin down the nature of the variance and its desired homoskedastic nature. The ACF of the residuals displays white noise-like behavior since we see no significant autocorrelation spikes. While this is a positive result pointing in the direction of homoskedasticity, we can only confirm this behavior for less than the first fifty lags. However, the Q-Q plot can help us with this issue because it tells us if our noise is normally distributed, thereby affirming a constant mean and variance by the normal distribution's parametric nature. Notice that the points in the Q-Q plot follow the straight line in the middle of the graph, but diverge in the extremities. This implies that the residuals exhibit more extreme values than would be expected from a theoretical normal distribution. It then appears that our residuals may be normally distributed with "fatter" tails and a slightly skewed variance.

Finally, another valuable metric in judging model fit is the Ljung-Box statistic, whose formula is given in equation (1):

$$\hat{Q} = n(n+2)\sum_{i=1}^{k}\frac{\hat{r}_i^2}{n-i} \tag{1}$$

where $\hat{r}_i^2$ denotes the squared sample autocorrelation at the i^{th} lag. It can be shown that for large n ($n = 1193$ here), $\hat{Q}$ follows a chi-square distribution. One rejects a chi-square statistic when it is large and thus in the tail of the chi-square distribution, which generates a small p-value, implying that the probability of observing such a large statistic is small. In our case, we desire the model residuals to exhibit white noise behavior, implying near-zero autocorrelations, a small Ljung-Box statistic, and thus large p-values. This is exactly what we see in the bottom plot above: large p-values at every lag for the Ljung-Box statistic, pointing to residuals with successfully small sample autocorrelations.

3 Model Comparison and Selection

Here we compare our models based on AIC, AICc, BIC, MSE, and RMSE. The first three information criteria are meant to gauge the quality of model fit. The more negative the value, the better, implying a larger log likelihood and hence better fit. MSE measures the model's predictive power while RMSE simply takes the square root of the MSE.

Model Name	Description	AIC	AICc	BIC	MSE	RMSE
Smeet	Log Price + ARIMA(1,1,1)	-6.1388	-6.1388	-6.1218	0.1984	0.4454
Pedram	Log Price + ARIMA(1,1,0)	-6.1405	-6.1405	-6.1277	0.2019	0.4493
Rex	Exponential Smoothing	-5.8617	-5.1512	-7.2510	0.3213	0.5668
Group	Log Price + ARIMA(0,1,0)	-6.1421	-6.1421	-6.1336	0.2019	0.4493

Table 3: Individual models and group model compared by information criteria and squared error metrics

After analyzing the above results, we remain confident that the selected random walk group model provides the best model fit while limiting the number of estimated parameters and thus limiting degrees of freedom.

4 Results

Our simple yet powerful "random walk" ARIMA(0,1,0) model is defined below in equation (2):

$$X_t = \mu + X_{t-1} + W_t \tag{2}$$

We decided to go with this model since empirically it presented superior AIC, AICc, and BIC values, indicating that the model provides an ideal compromise between model efficacy, generating a low out-of-sample MSE, and simplicity, since only a constant parameter need be estimated.

Theoretically, random walks behave similarly to daily fluctuations in the stock market: in short, unpredictably. It is precisely this element of randomness that actually makes random walk models solid stock market predictors, if we believe the efficient market hypothesis, which states that asset prices reflect all available information and it is for this reason that one cannot "beat the market."

4.1 Estimation of model parameters

Parameter	Estimate	(s.e)
μ	-0.0005	(0.0003)

Table 4: The only estimated parameter in our group model is a constant, see equation 2.

4.2 Prediction and Conclusion

As we know, Mediocre Social Network Apps Incorporated has struggled for a few years as people's preferences have been changing. This is evidenced by its generally decreasing stock price. We originally set out to forecast the first ten trading days of October 2019 to assist Mediocre in understanding what the near future entails. After extensive analysis and utilization of an accurate random walk ARIMA(0,1,0) model based on model comparison and diagnostics, we predict the next ten trading days to follow a slightly decreasing linear trend based on Mediocre's historical stock price performance (Figure 4).

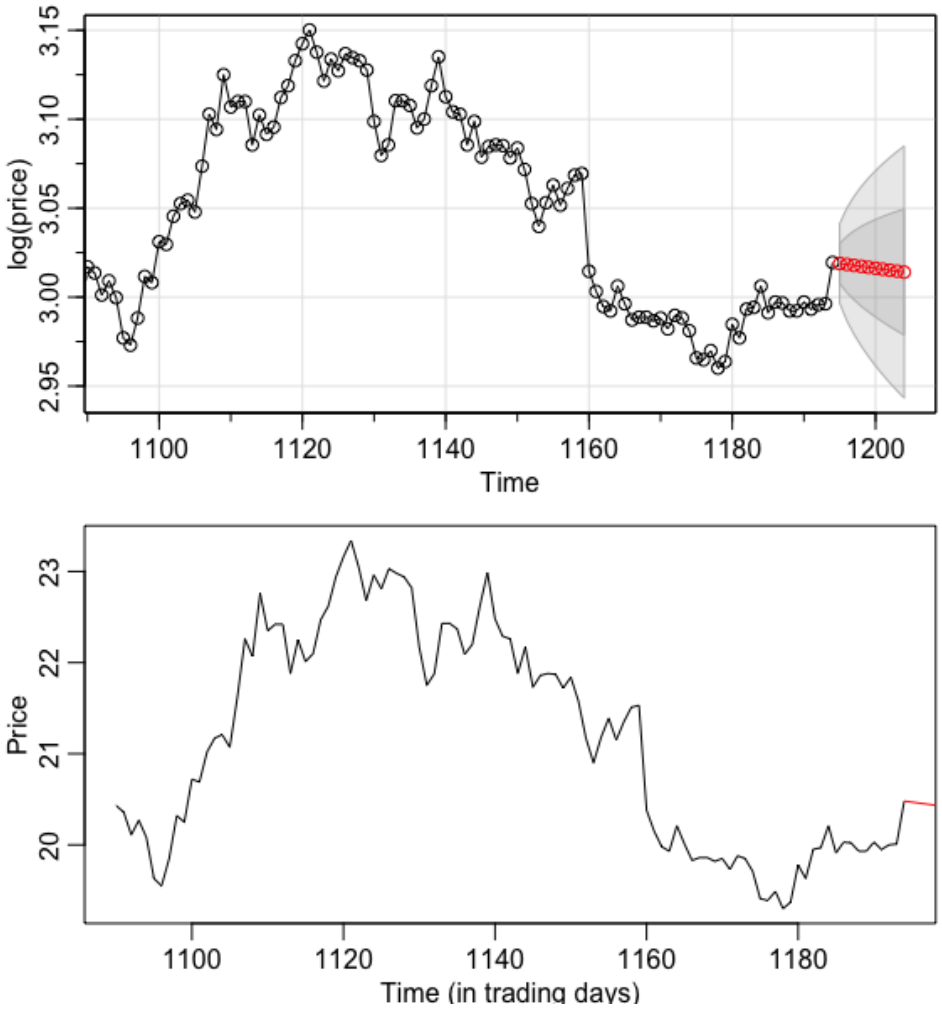

Figure 4: *Log stock price forecast with 68% and 95% prediction intervals shaded in dark gray and gray respectively (top), and raw stock price forecast (bottom)*

While we understand that this was not the optimistic news that Mediocre would've hoped for, it is important to take note of the dark gray and gray areas of the top plot in Figure 4, which represent 68% and 95% prediction intervals respectively. These areas indicate that we can be 68% and 95% certain that the observed values for the next ten trading days will fall within the confines of the dark gray and gray areas respectively. This points to the inherent breadth of uncertainty in forecasting, especially stock price forecasting.

We believe it is best to conclude with the idea of the efficient market hypothesis, which states that asset prices reflect all available information and thus it is impossible for one to "beat the market." We believe that this idea reassures the suitability of the model we've chosen and serves as an important reminder that while we may try our best to predict what is to come tomorrow, we cannot be certain as to what will happen. *What is certain, however, is that Mediocre could definitely use a name change.*

5 Appendix

5.1 Exploration

5.1.1 Smeet

```
df = read.csv("stocks.csv")
dim(df)
tail(df)
time = 1:nrow(df)

# Change price series to returns series
n = nrow(df)
df_ret = ts((df$Price[2:n] - df$Price[1:(n-1)])/df$Price[1:(n-1)])
#names(df_ret) = df$Date[2:n]
par(mfrow = c(2,1))
# 1 unit = 1 business day
plot(time, df$Price, type = 'l', ylab = 'Stock Price', xlab = 'Time')
plot(df_ret, type = 'l', ylab = 'Stock Returns', xlab = 'Time')

par(mfrow = c(2,1))
acf2(as.numeric(df$Price), lag(700))
acf2(as.numeric(df_ret), lag(700))
```

5.1.2 Pedram

```
# import dataset
stocks <- read.csv('stocks.csv',
                   row.names = 1
                   )

# format stock prices as time series object
price <- as.ts(stocks$Price)

# plot time series
plot.ts(price,
        xlab = 'Time (in trading days)',
        ylab = 'Price'
        )

# plot log of time series
plot.ts(log(price),
        xlab = 'Time (in trading days)',
        ylab = 'Log Price'
        )

# plot differenced data
plot.ts(diff(price),
        xlab = 'Time (in trading days)',
        ylab = 'First Difference of Price'
        )

# plot differenced log data
plot.ts(diff(log(price)),
        xlab = 'Time (in trading days)',
        ylab = 'First Difference of Log Price'
        )

# plot ACF of differenced, transformed data
acf(diff(log(price)),
    lag.max = 700,
    main = 'First Difference of Log Price',
    ylab = 'ACF'
    )
```

```
# plot PACF of differenced, transformed data
pacf(diff(log(price)),
     lag.max = 700,
     main = 'First Difference of Log Price',
     ylab = 'PACF'
     )
```

5.1.3 Rex

```
setwd("C:/Users/Rexgb/OneDrive/R Practice/Stat153/Stat153")
data = read.csv("stocks.csv")
plot.ts(data$Price,
        main = "Stock Prices over time",
        xlab = "Time",
        ylab = "Price"
        )
```

5.2 Model fitting and diagnosis

5.2.1 Smeet

```
# try different models
m = 10
model1 = sarima(df_ret, p=1, d=1, q=1)
model2 = sarima(df_ret, p=1, d=0, q=1)
model3 = sarima(df_ret, p=1, d=1, q=0)
model4 = sarima(df_ret, p=1, d=1, q=0, P = 0, Q = 2, S = 1)
model5 = sarima(df_ret, p=1, d=1, q=0, P = 0, Q = 1, S = 1)
model6 = sarima(df_ret, p=1, d=1, q=0, P = 0, Q = 2, S = 0)
model7 = sarima(df_ret, p=1, d=1, q=0, P = 0, Q = 1, S = 0)

# select model 2 for prediction based on AIC, AICc, and BIC
# predict
fcast = sarima.for(df_ret,p=1,d=0,q=1, n.ahead = 10, plot.all = F)
model2 = sarima(df_ret, p=1, d=0, q=1)
```

5.2.2 Pedram

```
# create chosen model
m1 <- sarima(log(price), p = 1, d = 1, q = 0)

# AIC, AICc, BIC
m1$AIC; m1$AICc; m1$BIC

# create predictions vector
p1 <- exp(sarima.for(log(price), n.ahead = 10, 1, 1, 0)$pred)

# create actual values vector
actual <- as.vector(window(price, start = 1090, end  = 1194))

# create time vector
t <- 1090:1194

# plot both actual and predicted values
plot(x = t,
     y = actual,
     type = 'l',
     xlab = 'Time (in trading days)',
     ylab = 'Price'
     )
lines(1194:1204,
      c(tail(actual,1), p1),
      col = 'red',
```

```
      type = 'l'
      )
```

5.2.3 Rex

```
# test the model for different weighting parameter
a_lis = c(.99,.95,.90,.80)
weights_1 = (1-a_lis[1])/a_lis[1] *(a_lis[1]^(1:20))
weights_1 = weights_1/sum(weights_1)
weights_2 = (1-a_lis[2])/a_lis[2] *(a_lis[2]^(1:20))
weights_2 = weights_2/sum(weights_2)
weights_3 = (1-a_lis[3])/a_lis[3] *(a_lis[3]^(1:20))
weights_3 = weights_3/sum(weights_3)
weights_4 = (1-a_lis[4])/a_lis[4] *(a_lis[4]^(1:20))
weights_4 = weights_4/sum(weights_4)

# apply filters to weights
m_1 = filter(data$Price,sides = 1,filter=c(0,weights_1))

plot.ts(data$Price,main = "Param = .99")
lines(m_1,col=2,lwd=3)

m_4 = filter(data$Price, sides = 1, filter= c(0,weights_4))
plot.ts(data$Price, main = "Param = .80")
lines(m_4, col = 2, lwd = 3)
```

5.2.4 Group

```
# create group model
m2 <- sarima(log(price), p = 0, d = 1, q = 0)

# AIC, AICc, BIC
m2$AIC; m2$AICc; m2$BIC

# create predictions vector
p2 <- exp(sarima.for(log(price), n.ahead = 10, 0, 1, 0)$pred)

# plot both actual and predicted values
plot(t, actual, type = 'l', xlab = 'Time (in trading days)', ylab = 'Price')
lines(1194:1204, c(tail(actual,1), p2), col = 'red', type = 'l')
```

5.3 Cross-validation

5.3.1 Smeet

```
sqrsd <- vector()
mse <- vector()
rmse <- vector()
for (d in 900:1183) {
  train <- window(log(price), end = d)
  test <- window(log(price), start = d + 1, end = d + 10)
  fcast <- exp(median(predict(arima(train, order = c(1,1,1)),
                              n.ahead = 10)$pred
                      )
               )
  sqrsd[d-899] <- (fcast - exp(median(test)))^2
  if (d == 1183) {
    mse <- mean(sqrsd)
    rmse <- sqrt(mse)
    print(c(mse,rmse))
  }
}
```

5.3.2 Pedram

```
sqrsd1 <- vector()
mse1 <- vector()
rmse1 <- vector()
for (d in 900:1183) {
  train <- window(log(price), end = d)
  test <- window(log(price), start = d + 1, end = d + 10)
  fcast <- exp(median(predict(arima(train, order = c(1,1,0)),
                              n.ahead = 10)$pred
                      )
               )
  sqrsd1[d-899] <- (fcast - exp(median(test)))^2
  if (d == 1183) {
    mse1 <- mean(sqrsd1)
    rmse1 <- sqrt(mse1)
    print(c(mse1,rmse1))
  }
}
```

5.3.3 Rex

```
# switch from R to Python for predictions/cross validation
predictions = np.linspace(0,0,30)
plot_data = np.arange(0,30,1)
predictions[:20] = m_int
for i in range(20,30):
    predictions[i] = np.sum(predictions[i-20:i]*weights)
print("These␣are␣the␣next␣ten␣data␣points:",predictions[20:30])

# calculate accuracy scores
testing_pred = np.linspace(0,0,len(cv_data))

for i in range(20,len(cv_data)+20):
    #print(len(weights), len(cv_data[i-20:i]))
    testing_pred[i-20] = np.sum(weights*cv_data[i-20:i])

test_data = cv_data[20:]

mse_ave = np.sum((testing_pred-test_data)**2)/79
n = 20
k = 2
sumsq = .3213*n
AIC = 2*k + n*np.log10(sumsq/n)
BIC = n*np.log10(sumsq/n)+k*np.log10(n)
AICC = AIC + (2*k*(k+1))/(n-k-1)
```

5.3.4 Group

```
sqrsd2 <- vector()
mse2 <- vector()
rmse2 <- vector()
for (d in 900:1183) {
  train <- window(log(price), end = d)
  test <- window(log(price), start = d + 1, end = d + 10)
  fcast <- exp(median(predict(arima(train, order = c(0,1,0)),
                              n.ahead = 10)$pred
                      )
               )
  sqrsd2[d-899] <- (fcast - exp(median(test)))^2
  if (d == 1183) {
    mse2 <- mean(sqrsd2)
    rmse2 <- sqrt(mse2)
    print(c(mse2,rmse2))
  }
}
```