UPS STOCK ANALYSIS

Authors:
Ruriko Imai (id: 912212313)
Huong Vu (id: 914320410)

Table of Contents:

Abstract:

The aim of this study is to understand the present trend of the United Parcel Service (UPS) stock market price data. The analysis of the UPS data is conducted using time series analysis and the model chosen through careful analysis of the data is used to forecast two years of the UPS stock's closing price. The models is selected by examining ACF and PACF plots, BIC values, and functions such as auto.arima(). The candidate models undergo another test to check for error rates and the best model is chosen by the least amount of error. Spectral Analysis is also used to analyze and forecast the data in order to have some comparable results that will conclude similar or otherwise differing results. The following results can be used to examine and understand the UPS stock market trend to decide whether to invest or not.

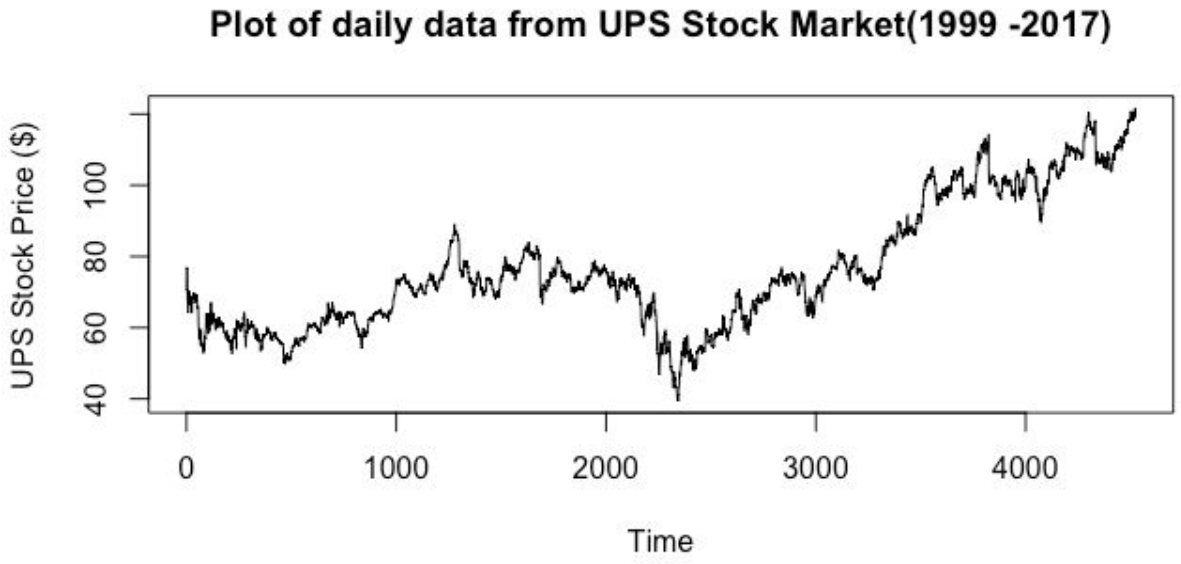
Introduction:

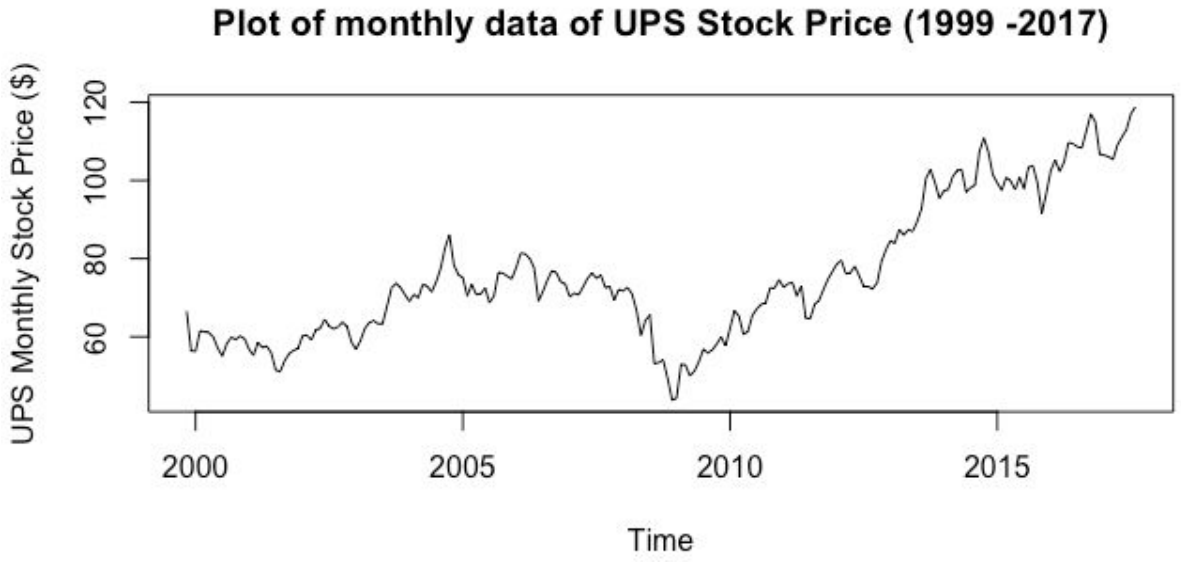
As Christmas is approaching, package delivery companies are getting busy. We expect the stock price of companies like UPS to increase as their services are in high demand during the last few months of the year. Attaining an idea of the future stock price trend of approximately 2 years will be useful in determining if one would consider investing in UPS. Since we have limited insight concerning the package delivery industry, our primary decisions whether to invest or not will be the results from statistically analyzing the UPS stock price. Since UPS stock price is recorded as a sequence of observed data against time, time series analysis will be used. First, we will estimate the deterministic components, trend and seasonality, of the time series. Then, we will look at stochastic component using ARIMA model and spectral analysis. Finally, based on our estimation of the trend, seasonal, and random components, we will make a prediction on the UPS stock price for 2 years.

Data Description:

The data of the UPS Stock Price is found on the Yahoo Finance website where historical stock datas of various businesses are posted. The data ranges over the 18 year period, starting from 11 November 1999 to 31 October 2017. The obtained dataset consists of 4523 days of information and is collected daily. Each row is composed of the date, opening price, highest price, lowest price, closing price, adjusted closing price, and the volume. For the purpose of our analysis, the closing price of the UPS stock market will be used to forecast possible values in the future.

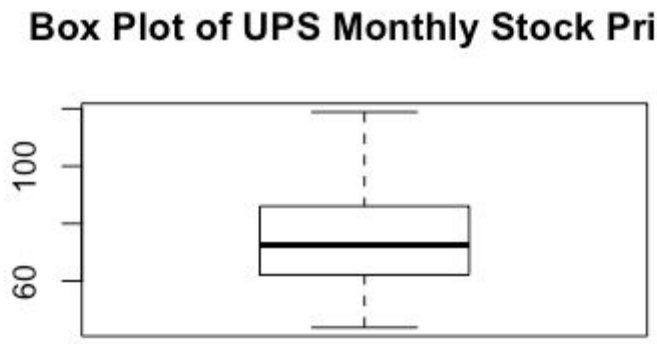
Although data is collected on daily basis, there are missing values for certain days. Due to missing values in various months, we decided to average the monthly prices to prepare the dataset. Plotted below are the raw data and the monthly averaged version of the data.





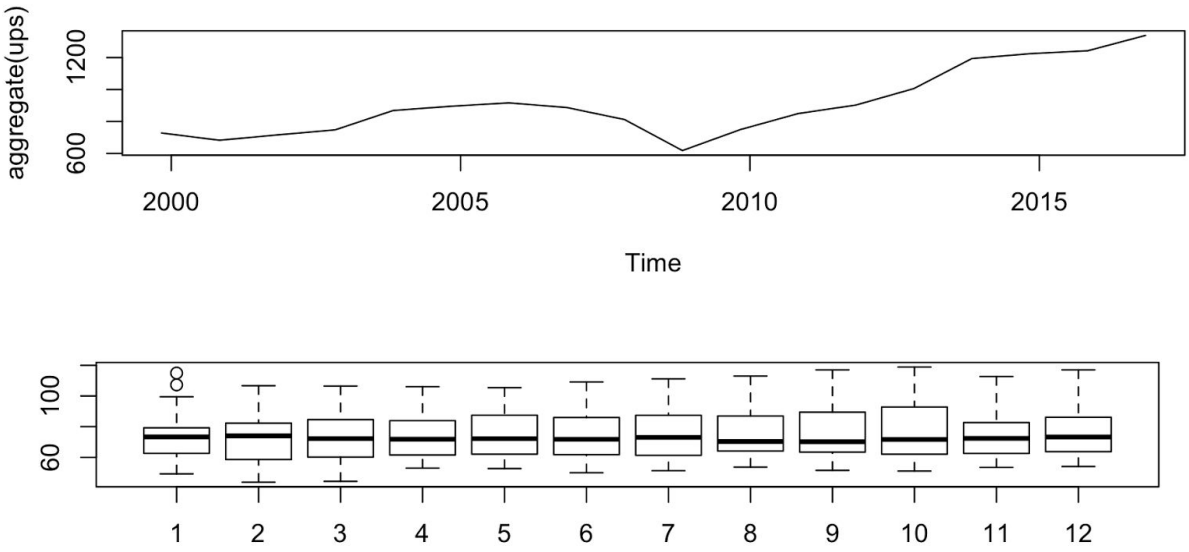
From these plots, we see that the monthly dataset still captures all significant features of the raw data, such as trends and fluctuations. Therefore, averaging data points of every month is an appropriate method to overcome the problem of missing values. The monthly averaged time series has 216 monthly data points.

The time frame from 1999 to 2017 in which the UPS stock price data is collected cover occurrences of two major shocks in the stock market; the 9/11 incident and the global financial crisis in 2008. These two events can generate unusual observations for UPS stock price, however, the box plot of the time series does not indicate any outliers that we need to consider.



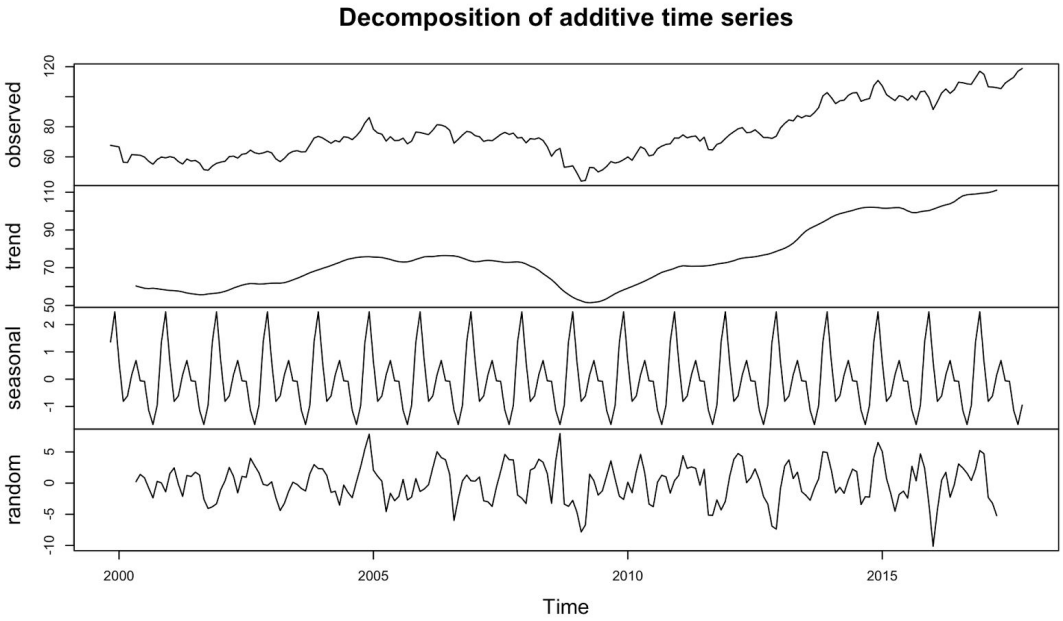
The plot of the monthly averaged data shows a positive trend overall with a presence of seasonal component. We expect to use additive model for UPS Stock Price time series since the range of fluctuations in the time series scatter plot seem to be roughly constant over time.

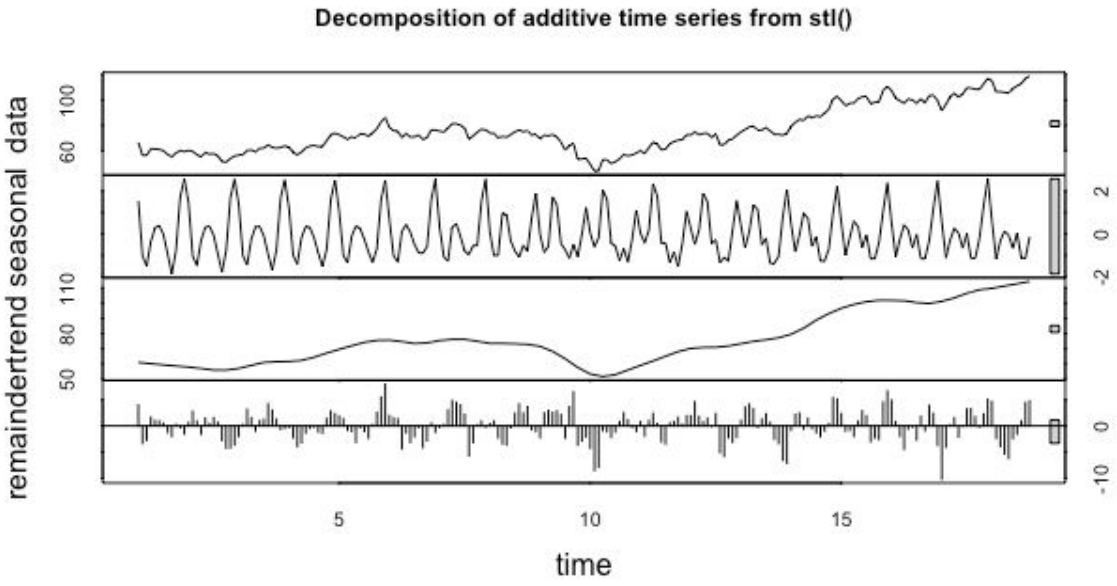
Plot of annual series and seasonal boxplot:



Here, the top plot shows an increasing trend in the annual series and the bottom boxplot represents the seasonal effects. The medians of the seasonal effects are consistent across 12 months that may indicate the absence of the seasonal component. However, we can also observe that the range of quartiles vary monthly, indicating that some months do have a wider range than others. Next, we will take a closer look at each component of the data.

Below, we use decompose() and stl() functions to separate UPS Stock Price time series into trend, seasonal, and random components.



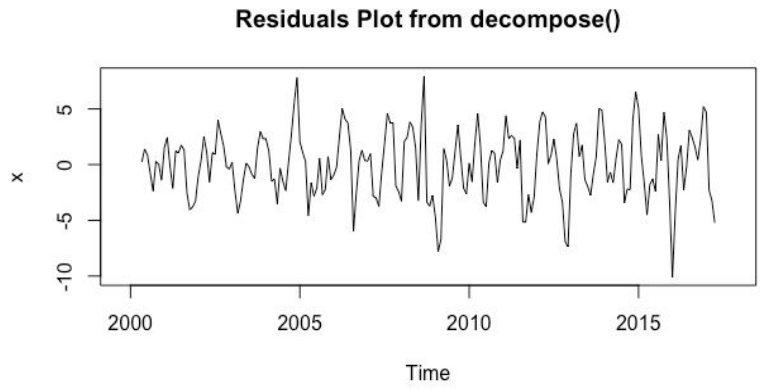


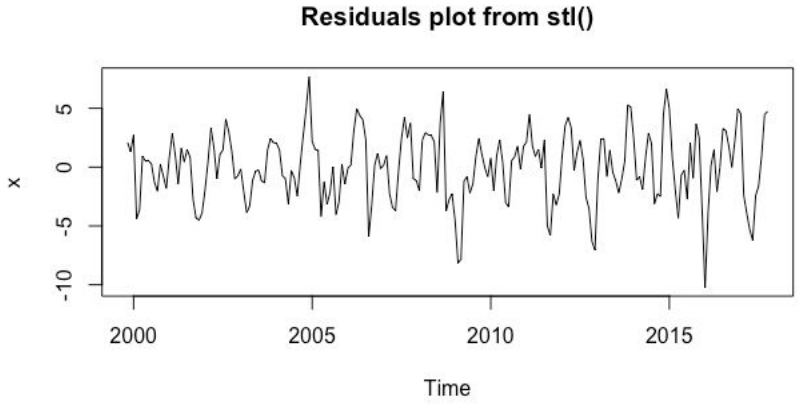
The partitioned plots from decompose() and stl() functions produce similar plots for the trend component. However, the stl() function produces varying patterns for different periods of the seasonal component, while the decompose() function only uses one pattern to estimate the seasonal component.

Both plots shows that the y-axis of the seasonal components are bounded by a small range, approximately (-2, 2) in comparison to the original data which ranges around (40, 120). We interpret this as an indicator for small to no seasonal component in UPS stock price. This result is consistent with what we obtained above from the boxplot of seasonal components across 12 months.

The random components from both decompose() and stl() functions look stationary; however, there might be some periodic patterns present in the random time series. Although we observed a relatively small seasonal patterns the presence of periodic patterns in the random time series raised a question. This observation requires further exploration of the time series using complex statistical analysis and by investigating the delivery service industry for any seasonal patterns or critical events.

The plots below are of random components in the times series using decompose() and stl() functions, respectively.



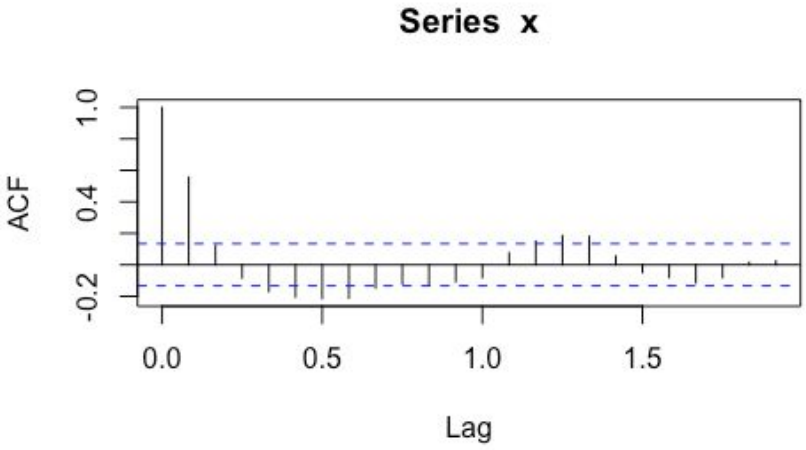


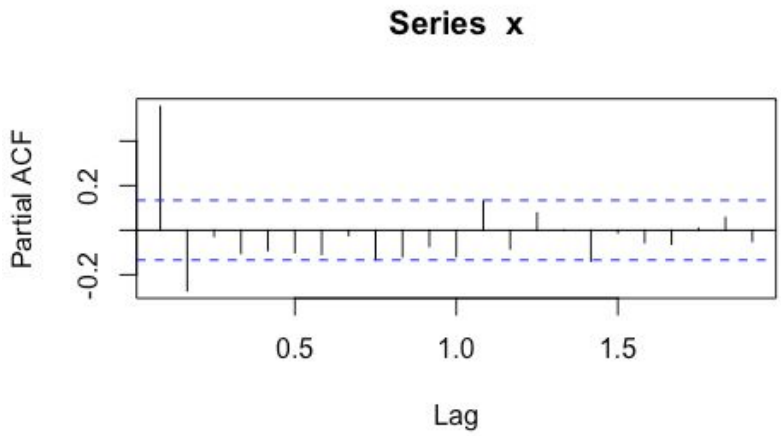
Data Analysis:

- **Trend and Seasonality Estimation**

The decompose() function uses 2-sided moving average method to estimate the trend which indicates the absence of the trend estimated values at the end of the series. The absence of the trend estimated values also leads to missing values for the residual series. On the other hand, the stl() function uses locally weighted regression technique known as loess. Hence, the function produces a full length estimated trend component and residuals time series. Therefore, we will use the stl() function to detrend and deseasonalize the UPS monthly stock price.

The residual plot of the time series after detrending and deseasonalizing looks stationery. However, the residual plot does not look completely random. In other words, we can still fit in some model to obtain a final residual time series that would represent white noise.





- **Fitting model to random time series**

We estimate models for the residuals using three methods: ACF and PACF plots, auto.arima() function and choosing model based on BICs.

1) Examining ACF and PACF plots: The ACF plot of the residuals exhibits an exponentially decaying behavior. At the same time, the PACF plot has significant values at lag 1 and 2. Therefore, we predict the suitable model for the residuals to be ARMA(2,0,0). Another possible ARIMA model from looking at ACF and PACF plots is ARIMA (0,0,1) since the ACF plot has significant value at lag 1.

2) Using auto.arima(): The auto.arima() function returns the model ARIMA(2,0,2)(0,0,1). This seems reasonable since the Seasonal ARIMA(0,0,1) component in the model can describe the periodic patterns that we observe in the residual plot.

3) Choosing model based BICs: We compare ARIMA(p,0,q) models for p and q less than or equal to 6 based on the BICs. The model that gives the lowest BIC value is ARIMA(2,0,2) with BIC value of 987.2177.

A problem when trying to find a model to fit in random time series is that we do not know how to model the periodic pattern mentioned above in the residual with ARIMA model . The only method is to use auto.arima() function to find a suitable seasonal ARIMA model for the random time series, which gives us a seasonal ARIMA model (2,0,2)(0,0,1).

After examining all possible models for the random time series, we conclude with three candidate models: ARIMA(0,0,1), ARIMA(2,0,2) and ARIMA(2,0,2)(0,0,1). Since our goal is to make predictions on future UPS stock price, we will choose our best model based on the Root Mean Squared Error (RMSE) and Mean Average Error (MAE) values.

- **Choosing the best model**

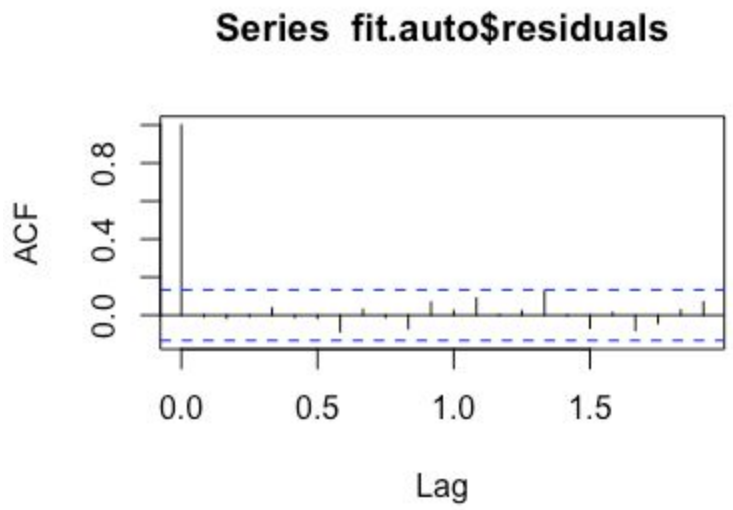
To calculate the RMSE and MAE values, an assumption is made that there are only 200 observed values out of 216. The model fitted over the 216 data points will then be used to make predictions on the last 16 data points. Finally, we will compare our predicted values with the actual values to calculate RMSE and MAE values for each model.

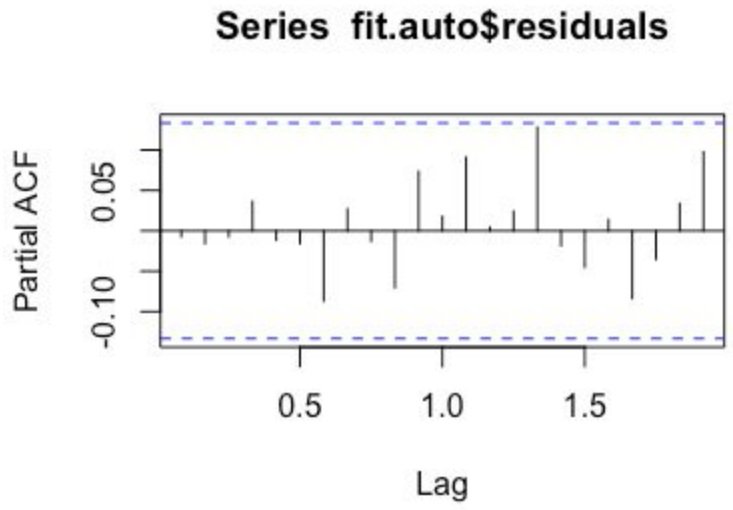
Model	RMSE	MAE
ARIMA (0,0,1)	3.611653	3.196592
ARIMA(2,0,2)	3.418994	2.941269
ARIMA(2,0,2)(0,0,1)	3.44212	2.89255

The ARIMA(2,0,2) has smallest RMSE value and ARIMA(2,0,2)(0,0,1) has smallest MAE value. However, when looking at the difference between RMSE values and MAE values between ARIMA(2,0,2) and ARIMA(2,0,2)(0,0,1), we see that ARIMA(2,0,2)(0,0,1)'s RMSE value is 0.023126 larger than ARIMA(2,0,2), while the MAE value of ARIMA(2,0,2) is 0.048719 larger than ARIMA(2,0,2)(0,0,1). Therefore, we choose ARIMA(2,0,2)(0,0,1) as our best model.

- **Testing model for independency**

Below are the ACF and PACF plots using the ARIMA(2,0,2)(0,0,1) model.

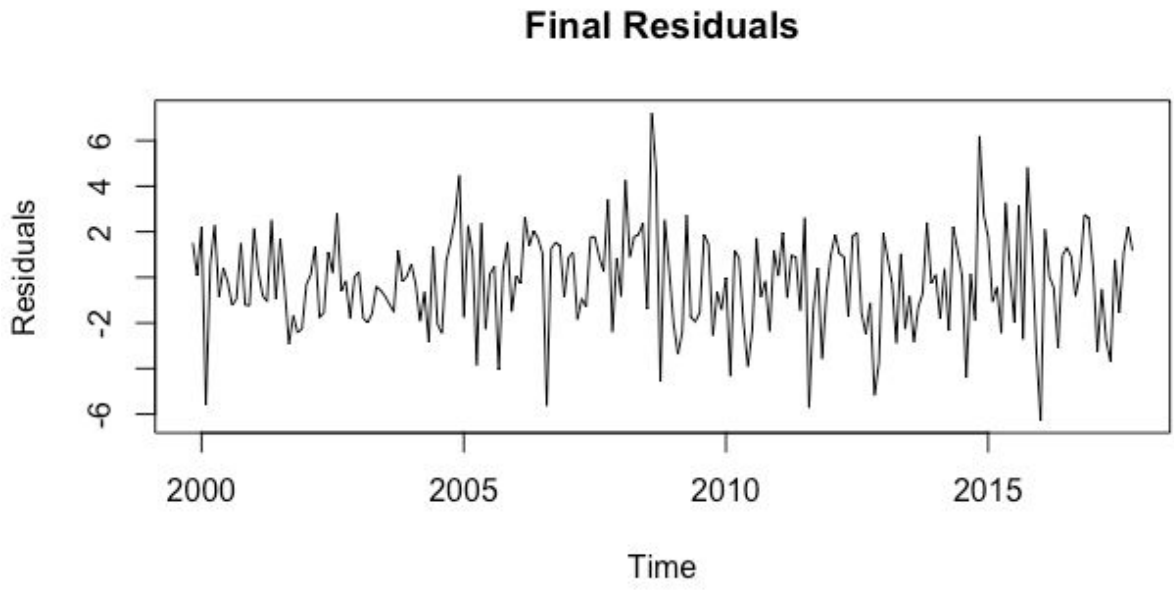




Observe that all lags in both ACF and PACF plots are within the confidence interval lines. We also ran a Ljung Box test on the lags up to 17 of ACF plot to see if any lags should be considered as significant. Since p-values from Ljung Box test is 0.5721 (greater than any alpha), we conclude that all lags are

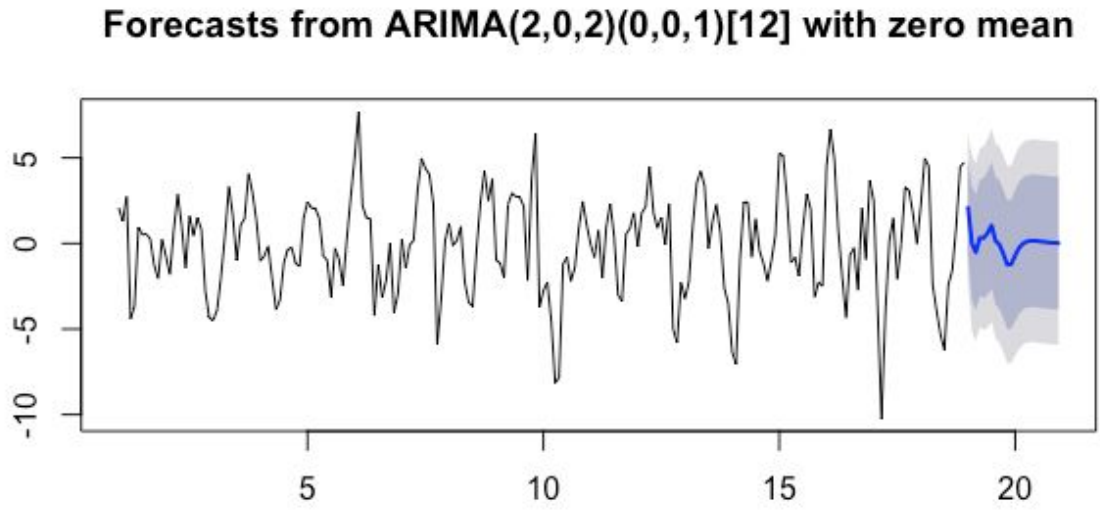
insignificant and there is no dependency among different lags. Hence, the final residual time series is white noise.

Below is the plot of the final residual time series representing white noise.



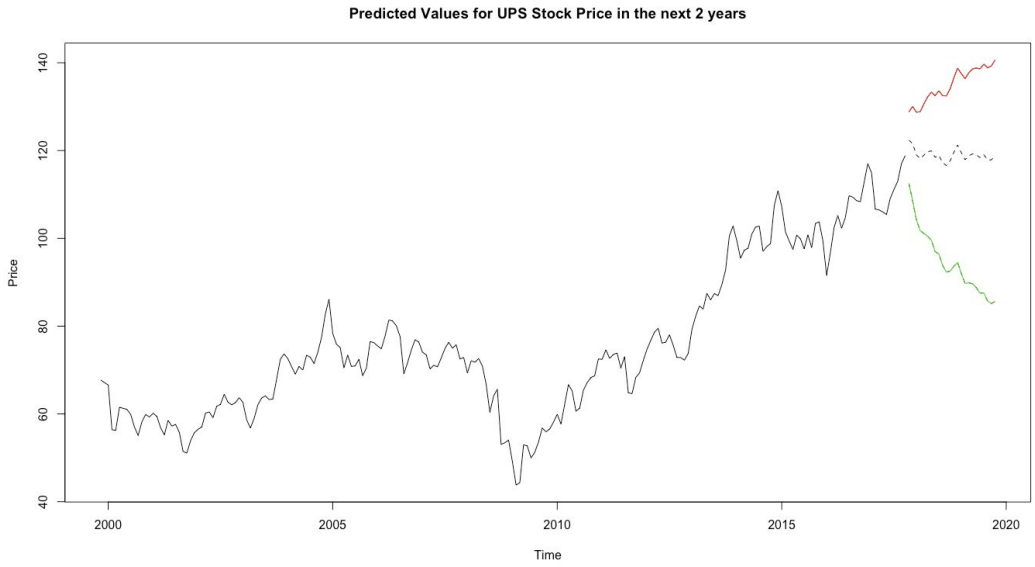
Finally, we want to forecast values of UPS stock price for the next two years. First, we want to forecast the random shock of UPS stock price using selected model ARIMA(2,0,2)(0,0,1).

- **Forecasting for the next 2 years**

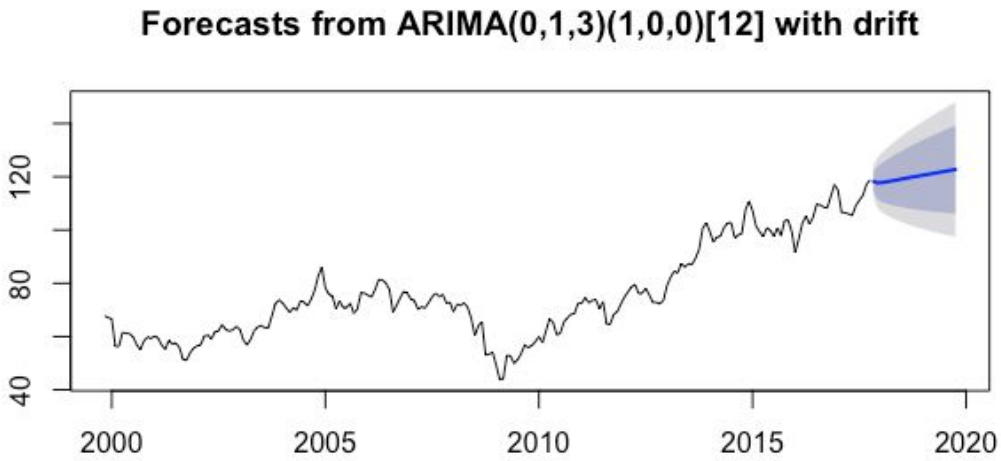


Notice that this forecasting method is only good for short term prediction considering that the predicted values converges to the mean after a certain length.

Next, we will predict the trend and seasonality using forecast() and stl() functions. After getting the estimated values for trend and seasonality for the next 2 years, we add them up with the predicted random component. The forecast() function on stl() function provides prediction interval for the trend and seasonal components. We added the prediction interval of the trend and seasonality with the prediction interval of random component to obtain the prediction interval for the original values, UPS stock closing price. The plot below is the predicted UPS stock price for the next 2 years with 95% prediction interval.



In addition, we also use auto.arima() function on the initial UPS time series to forecast. Below is the plot of predicted values from auto.arima() function.



Observe that the predicted values from auto.arima() function create a smooth, slightly upward trend unlike the forecast plot we obtain from manual method that has fluctuations. This observation from the forecast values of auto.arima() function can be explained due to irregular trend and small seasonal components in the original time series. Auto.arima() produces ARIMA(0,1,3)(1,0,0), which uses first order differencing to detrend and some periodic pattern approximated by seasonal AR (1). Meanwhile, the predicted values from combining random component, seasonality, and trend has fluctuations with small decrease to no change in the trend. Notice that both prediction intervals (manually and automatically) become wider as we go further into the future. This is because of the prediction accumulation which means with further predictions, more estimated values are used to predict the next data point rather than using observed values. Again, due to prediction accumulation, the forecast values are only good in short term.

Since we are working with stock price, single value of prediction will not provide much meaningful interpretation for economists. Instead, a prediction interval of the value will provide a better overall view on how the company will do in the next 2 years. Therefore, to narrow the choices for the predicted values, we take a closer look at the 95% prediction interval from each method. The prediction with a smaller interval will be the best method.

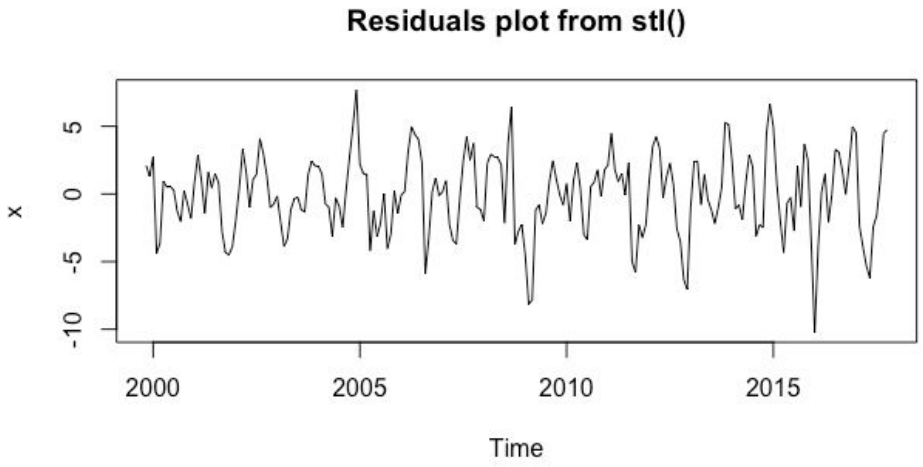
The table below compares the prediction interval between manual method and auto method. It shows that the manual method has wider prediction interval compared to the auto method for all period. Therefore, we will use the predicted values derived from auto.arima() function.

Prediction Int. (Manual)	Prediction Int. (Auto)	If manual <= auto
19.74459562	12.15131931	0
26.07483881	18.37176153	0
29.66301294	21.5935886	0
32.75974979	23.75750722	0
35.67901078	25.74014821	0
38.36395604	27.58063288	0
40.78169701	29.30575696	0
42.95897806	30.93482619	0
44.94563473	32.48229598	0
46.78861459	33.9593232	0

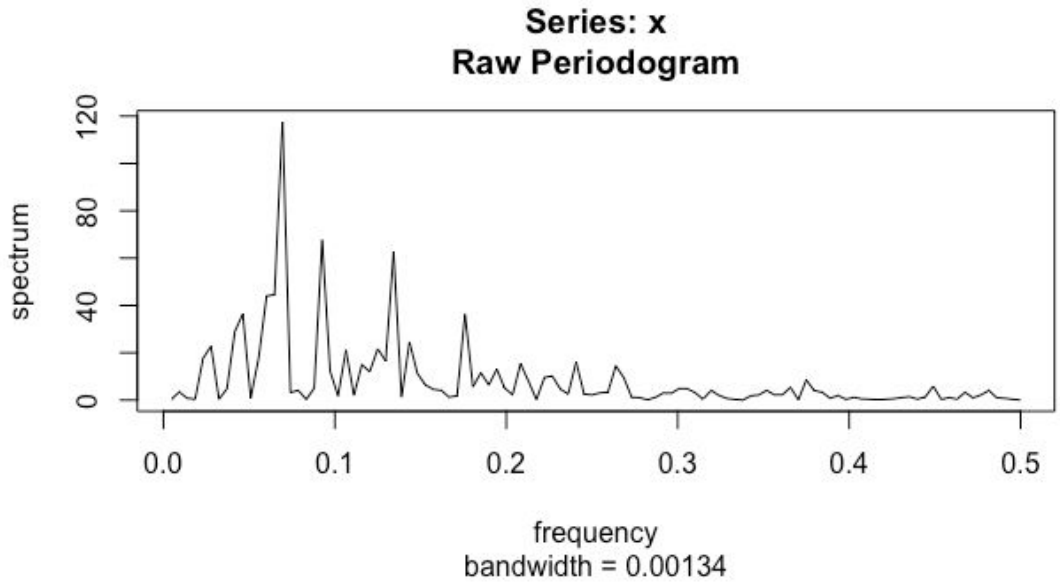
48.52295378	35.37473269	0
50.17202612	36.73564745	0
51.95068681	38.06882046	0
53.54996822	39.35957997	0
55.0168679	40.60539561	0
56.44008764	41.81184652	0
57.82960382	42.98444912	0
59.1805476	44.12590195	0
60.49079296	45.23856297	0
61.76223241	46.32450687	0
62.9985494	47.38557052	0
64.20349744	48.4233895	0
65.38020396	49.43942766	0
66.53110125	50.43500138	0

Spectral Analysis:

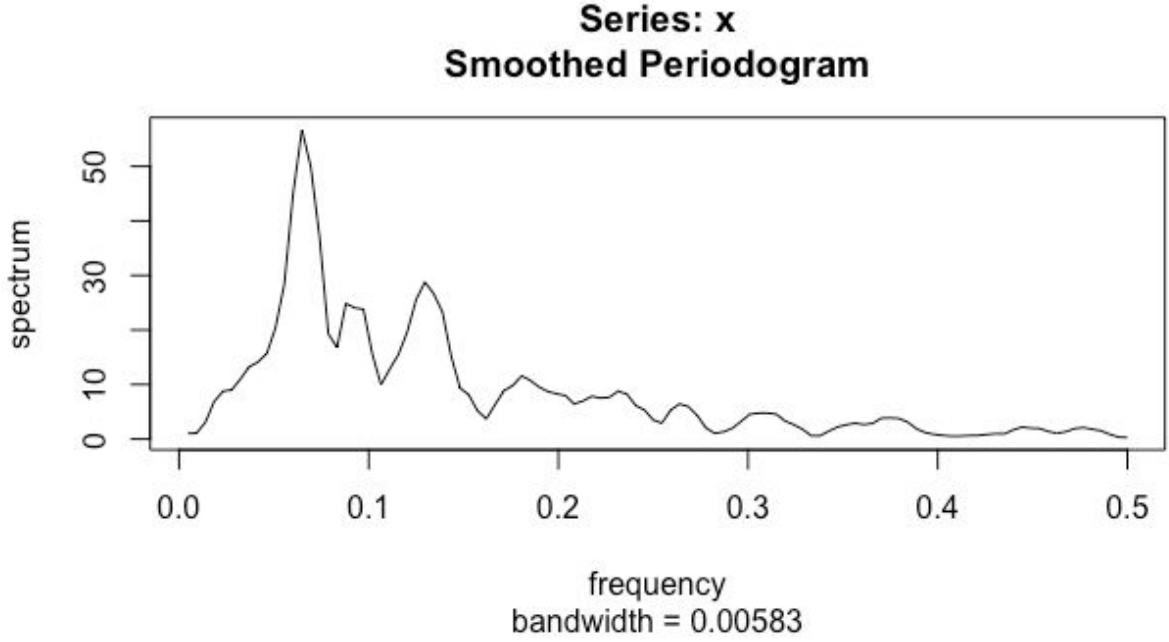
For this section, we will use spectral analysis to find an appropriate model consisting of sine and cosine functions with different frequencies that fits the random component of UPS time series data. We know from previous analysis that time series x is stationary and its autocovariance will converge to 0 fast enough so that its sum is finite. Hence, we have a conversion between autocovariance of the random time series and its frequency. Therefore, beside looking at the random time series as a regression on the past, we can look at the time series as a regression on frequency.



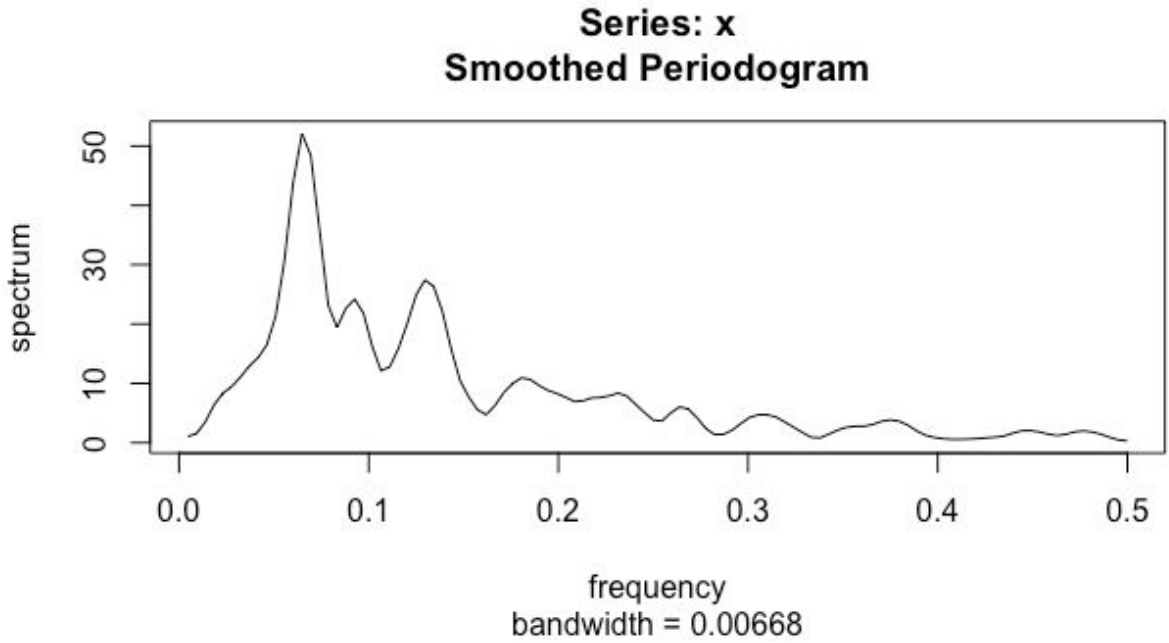
Below is the raw periodogram of x:



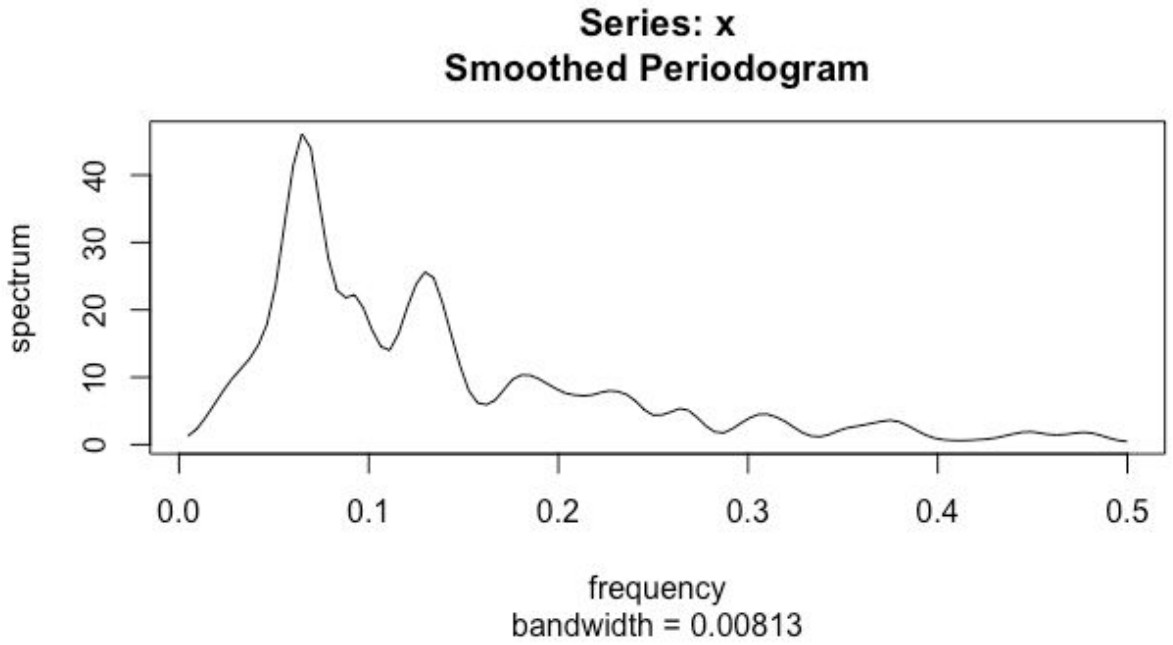
By calculating twice the area under the curve of periodogram, we get an estimation for the variance of random time series as 2.451. Observing the periodogram plot above, there are many spikes in the range from 0 to 0.2 along with the dominant spikes. These spikes indicate the low frequencies are dominant, which means those frequencies have the most contribution in the oscillations of x time series. The three dominant spikes has the spectrum confidence interval of (15.431188, 2248.3697), (20.48319, 2984.462), and (14.99165, 2184.327), left to right, respectively. The corresponding frequencies are 0.06944444, 0.09259259, and 0.1342593. Since the plot has too many spikes and large confidence intervals, we would like to perform some smoothing on spectrum density to remove small variances. First, we smooth spectrum density with span = 4.



This periodogram is smoother and still contains three peaks. The 4th peak is not obvious anymore since its spectrum density has been redistributed to surrounding less dense frequencies through averaging process. The confidence intervals for the three dominant peaks again are, (27.68151, 174.6258), (11.74742, 74.10731), and (14.04613, 88.60848), respectively. The frequencies are 0.06481481, 0.09259259, and 0.1296296. Notice that the confidence intervals for the spectrum density has gotten narrowed significantly. We perform more smoothing with span = c(4,2) to see if we would obtain a better result.

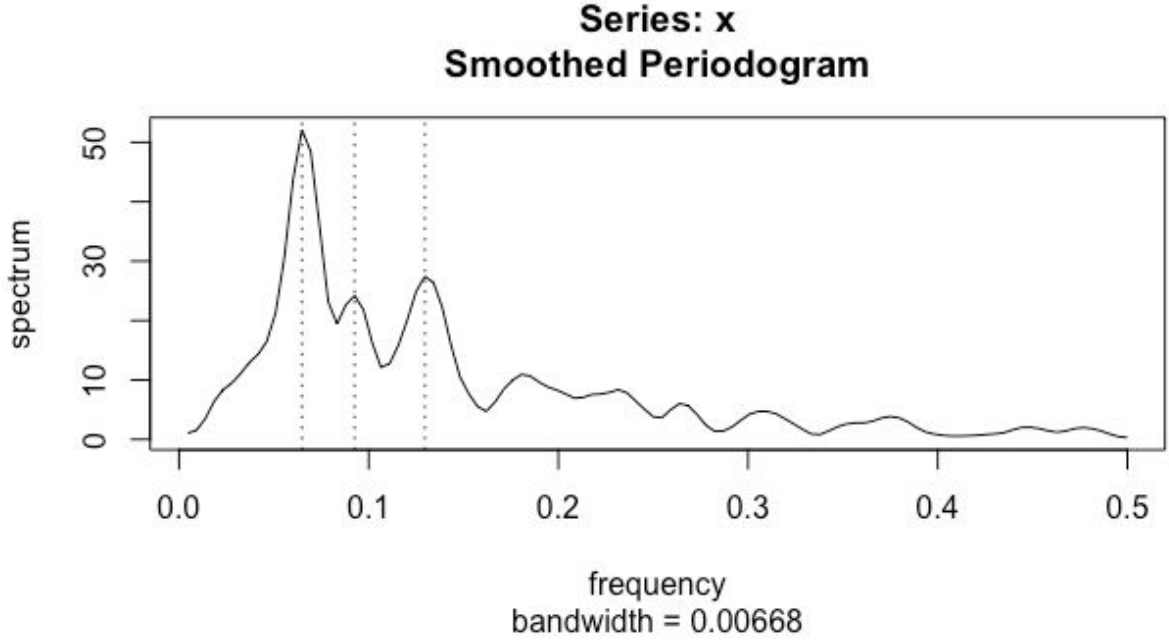


This plot is smooth enough and still remains the first three peaks with the 4th peak is still visible but compared to other peaks, the 4th peak is not that significant. The confidence intervals for the three dominant peaks are, (26.14848, 150.2133), (12.14097, 69.74537), and (13.76763, 79.08989), respectively. The frequencies are 0.06481481, 0.09259259, and 0.1296296. The confidence intervals get smaller, but the decrease is not a lot. To make sure that we have a good enough smoothed periodogram, we smooth the periodogram more with spans = c(4,4).



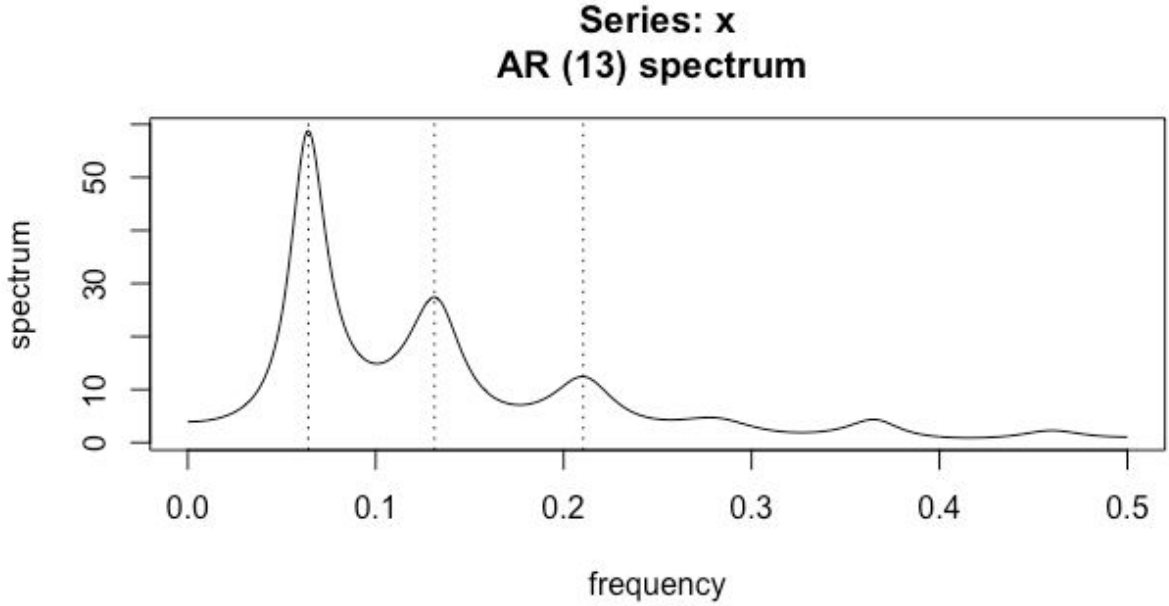
In this plot, the 2nd peak has significantly merged into the 1st peak. Therefore, we would want to stop at spans = c(4,2). The confidence intervals of the two dominant peaks in this plot are (24.2431, 119.724) and (13.46371, 66.49025), respectively. The frequencies are 0.06481481 and 0.1296296. Notice here that the confidence interval shrank significantly as the periodogram smoothed.

After obtaining the periodogram plot, we identify at which frequencies, the peaks are located at since those have the most impact on the time series.



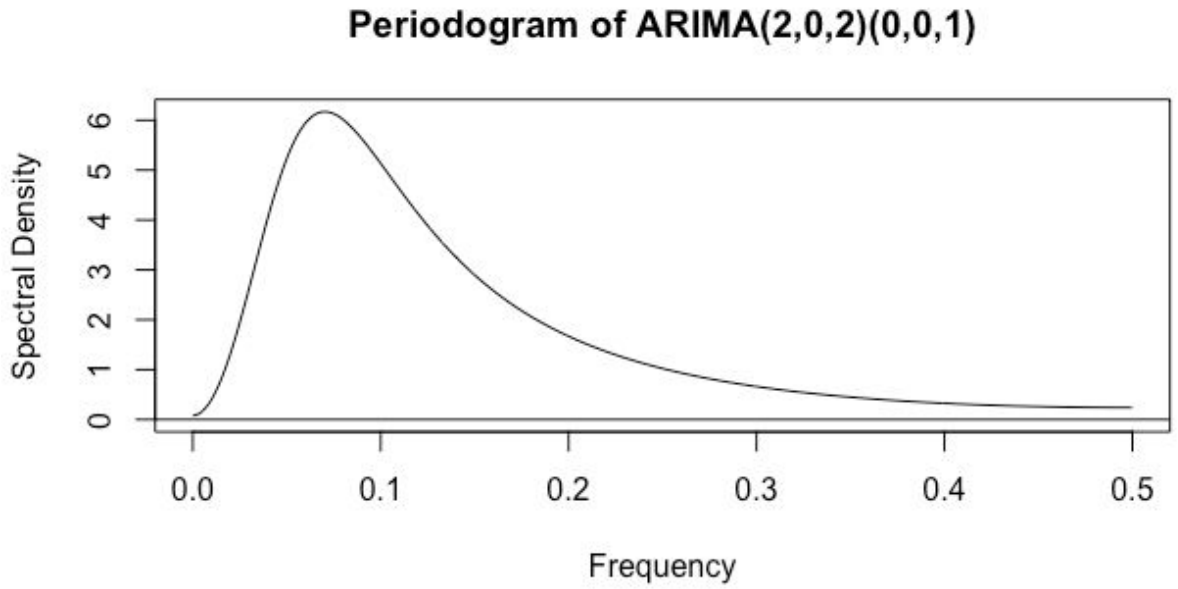
The first peak is at frequency 0.06481481 which means the period associated with this cycle is about 15 months. The 2nd peak is at frequency 0.09259259 and the period associated with this cycle is 10.8/11 months. The 3rd peak is at frequency 0.1296296 and the period associated with this cycle is 7.7/8 months. UPS is a shipping company, so we would expect its stock price would have annual cycles. The 2nd peak with associated period of 11 months is somewhat matched with our expectation.

Beside the smoothing method, we also used parametric estimation of spectral density method in which the ar.spec() function fitted AR(13) into random time series and use periodogram of AR(13) model as periodogram for the random time series.



In this plot, we also have 3 peaks at frequencies 0.06412826 (15.6/16 month period), 0.1312625(7.6/8 month period), and 0.2104208(4.7/5 month period). Compared to the frequencies that we obtain from manually smoothing, the dominant frequencies are shifted to the right.
The periodogram plots of both methods include 16 month period and 8 month period cycles. These periods can be coming from UPS business cycles or they are just purely statistical values.

From the analysis using ARIMA model, we obtain model ARIMA(2,0,2)(0,0,1) for the random time series. We plot the periodogram of the exact ARIMA model found by auto.arima() function.



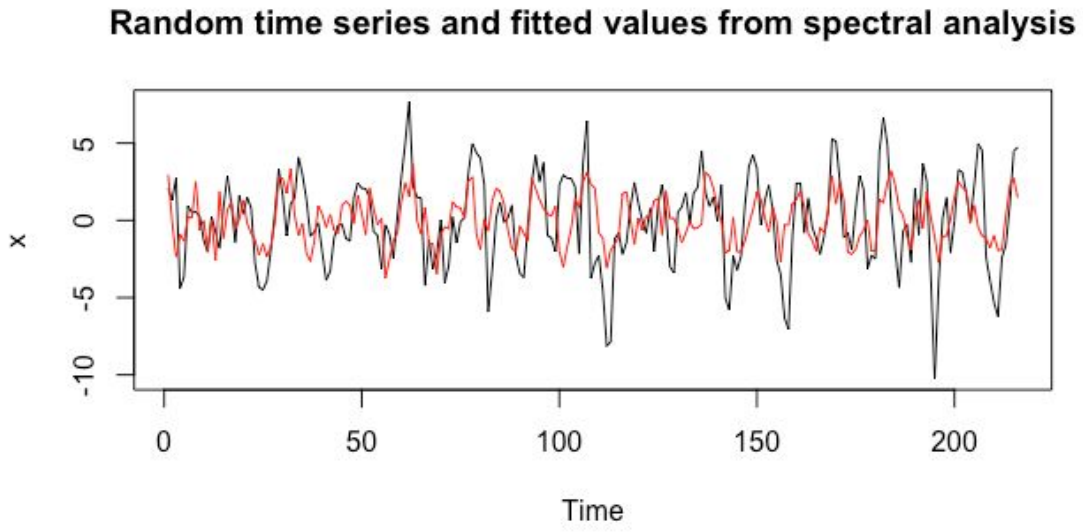
This plot has similar shape as the periodograms we obtain above for our x time series as the spectrum density reaches its peak near 0.7 and decreases as the frequency as increases. The dominant frequencies also range from 0 to 0.2. The similarities in periodograms of ARIMA model and random the time series support our choice of ARIMA model above.

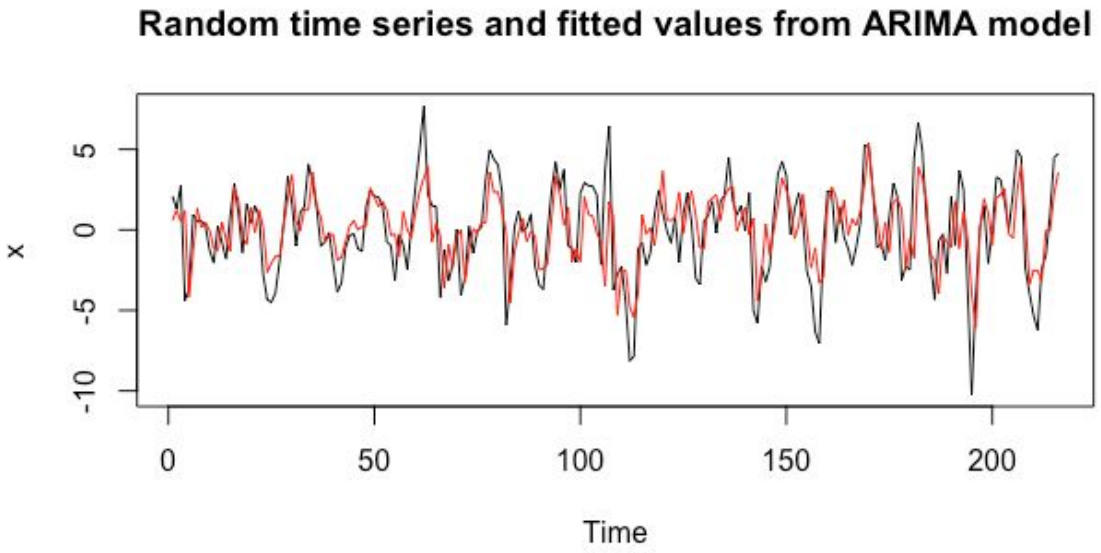
From the periodogram, we pick the frequencies that the peaks are located at and use them to calculate the A's and B's coefficients, which tell us how much the frequencies contributed to the fluctuations of the x time series. We pick 5 frequencies in total ranging from the highest peak to the lower ones from periodogram of smoothing method and periodogram of parametric method. After having the frequencies and the coefficients, we reconstruct x time series with the sine and cosine functions involving 3, 4 and 5 peaks. To decide which combination is the best choice, we compare the fitted values produced from the constructed time series with the observed data to calculate MRSE and MAE values. The table below presents the MRSE and MAE values from using different models.

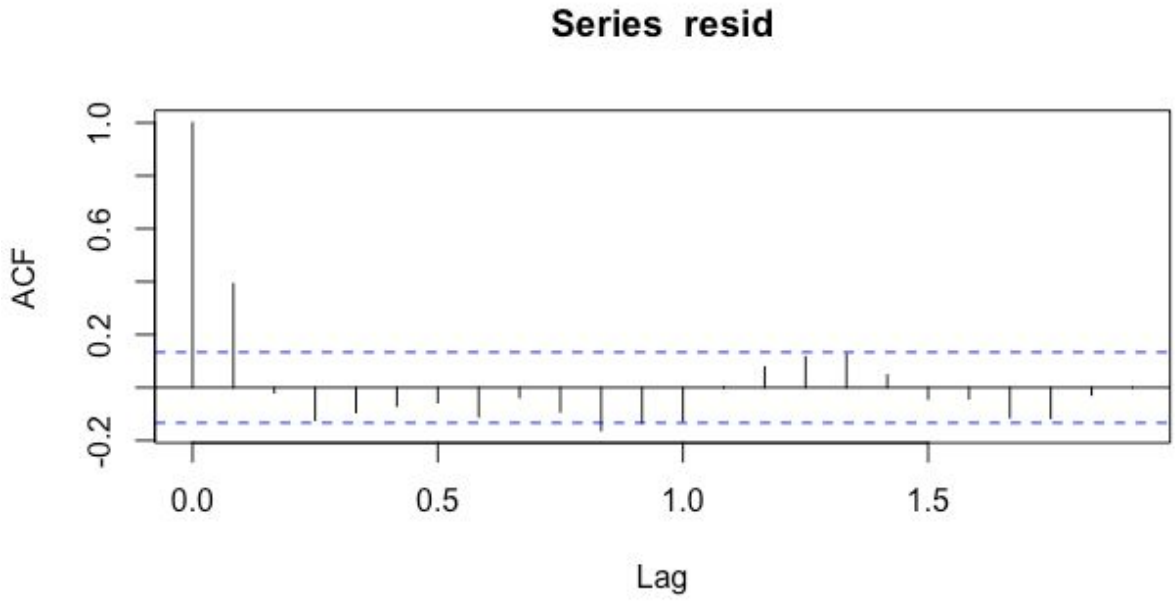
Model			MRSE	MAE
Spectral Analysis	Smoothing	3 peaks	2.793481559	2.256170488

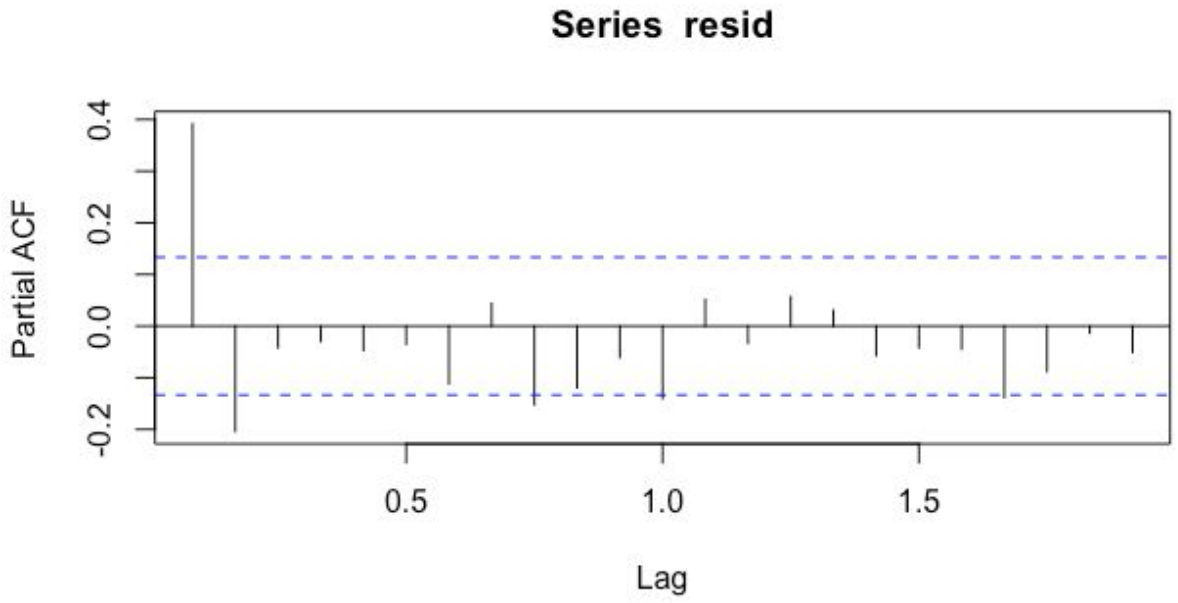
		4 peaks	2.696112594	2.167818254
		5 peaks	2.785064793	2.225534798
	Parametrics	3 peaks	2.978555349	2.331252873
		4 peaks	2.892009438	2.377553646
		5 peaks	2.893315509	2.366205447
ARIMA	ARIMA(2,0,2)(0,0,1)		2.165326	1.717004

As it is shown in the table, among spectral analysis models, the model which uses 4 dominant frequencies from periodogram of smoothing method has the lowest values for RMSE and MAE. However, compared with the ARIMA(2,0,2)(0,0,1) model, those values are still high. Therefore, we conclude that the ARIMA model is the best model to fit in x time series. As the plots of time series and fitted values shown below, the spectral analysis fitted values does follow the period of the observed data. However, the amplitudes of the fitted values' fluctuations are not as big as the observed datas'. Meanwhile, the ARIMA fitted values closely follow the observed data.




Random time series and fitted values from ARIMA model
x
5
0
-5
-10
0
50
100
150
200
Time

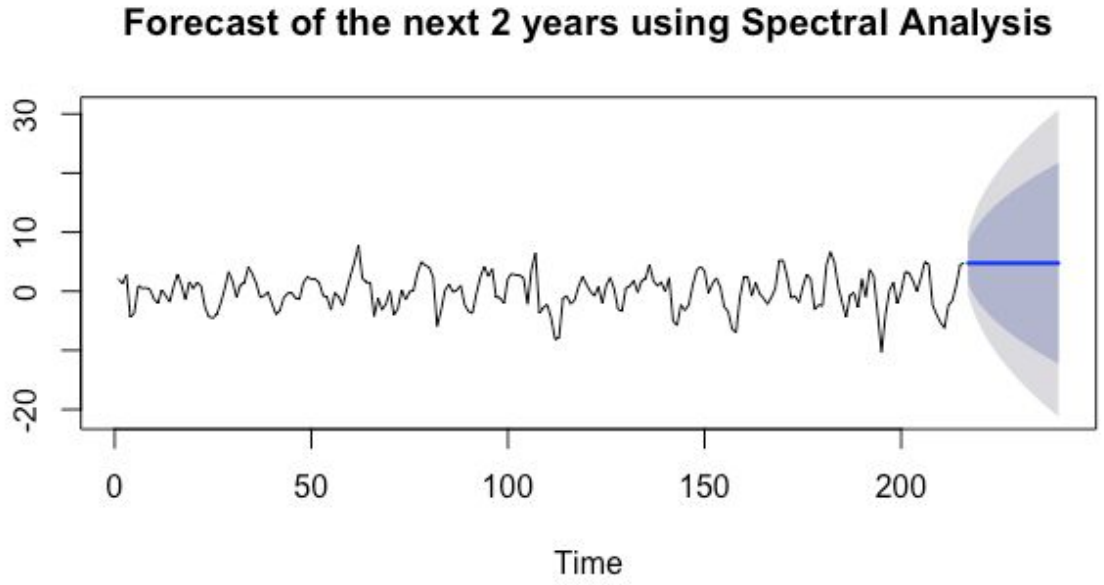

Series resid
ACF
1.0
0.6
0.2
-0.2
0.0
0.5
1.0
1.5
Lag



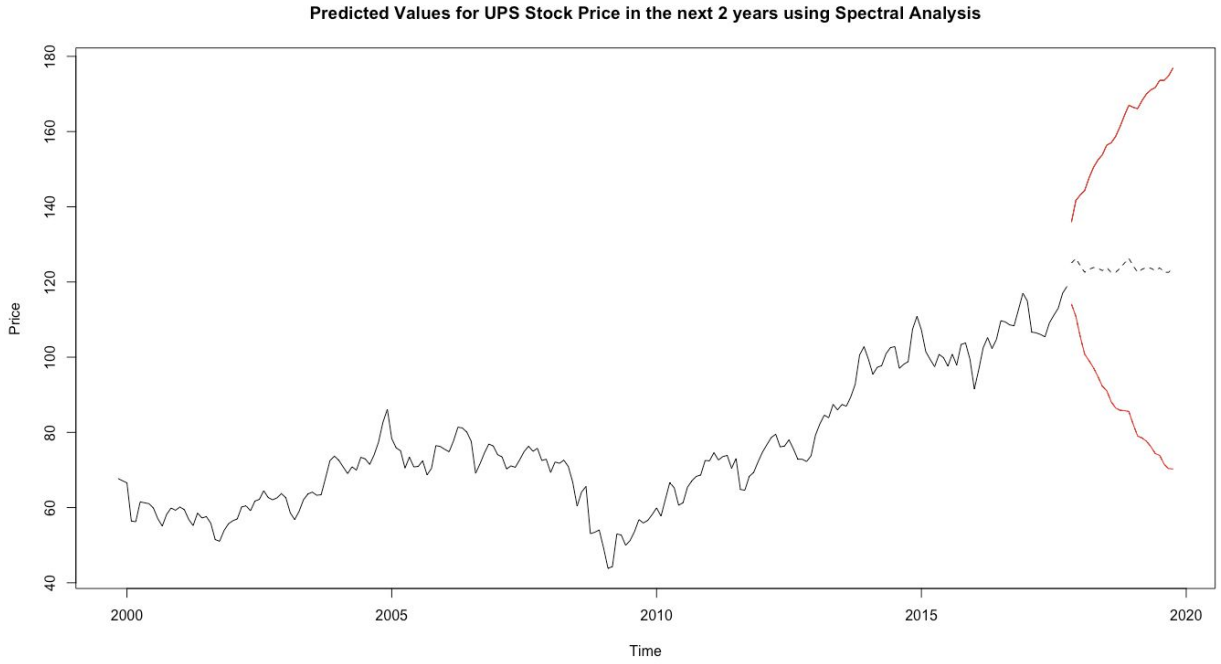
The ACF and PACF plots also show that there is still dependency in the residual time series after we remove the fitted spectral values. This indicates the inefficiency of the spectral model that we fit in x time series.

- **Forecasting using spectral analysis**

The following plot shows the predicted values of the random shock using spectral analysis in the next 2 years with the prediction interval.



As a result of inefficient model, in which the residuals are not white noise, when we use the model to predict the random shock for the next 2 years, we have a big range of prediction interval. The prediction interval from spectral analysis is much wider compared to the prediction interval from ARIMA model ((-20,30) compared to (-10, 5)). Moreover, in the ARIMA model, the prediction values fluctuate around zero before they converge to 0, while the spectral analysis predicted values start of from 0 and continue to stay at 0. This may be resulted from the fact that we define our spectral analysis model manually instead of having R choose a model for us.



As a result of having a wide range of prediction interval of random component, the predicted values of UPS stock price have a wide range too. The exact predicted values should not differ a lot from what we obtain above with our choice of ARIMA model since we use the same trend and seasonal estimation.

Discussion:

The analysis of the UPS Stock Price data starts with critical observations of seasonal, trend, and random effects. The decomposition of the trend allows us to understand if we can make plausible predictions by determining if the time series data is stationary or not. The stationarity of the data sets up the basic assumptions for forecasting the trends to come.

Encountered problems:

1) The raw dataset that we obtained consists of daily closing prices of UPS's stock from 1999 to 2017. There are days that the closing prices were not recorded. Our solution is to take the average of each month and conduct analysis on monthly time series.

2) During the time period from 1999 to 2017, there are two major events that would affect UPS's stock price, which are 9/11 and global financial crisis in 2008. As a result, we observe two big troughs in the raw time series plot. One proposed solution to deal with this problem is to segment the time series into different intervals and perform analysis on each interval. Then the final model will result in a combination of all the models. However, when we construct a boxplot of the monthly time series, there are no outliers observed in the plot. Therefore, we decided to analyze the time series as a whole.

3) The decompose() function uses 2-sided moving average method to estimate the trend, hence, result in missing estimated values at the end for the trend. Since we need the full length series of estimated values to predict future trends, we need to find another method to estimate the trend. From the time series plot, we could segment the time series into 2 intervals: from November, 1999 to September, 2008, and from October 2009 to October 2017, and use two polynomials to estimate the trend. However, the stl() function gives us full length series of estimated values for trend, seasonal and random components. For comparison, stl() and decompose() functions provides similar results.

4) We suspect that there is some seasonality leaked into the random time series from the observations of periodic patterns in the random time series. However, we do not have tools or knowledge on modeling the seasonal ARIMA. Therefore, we can either use auto.arima() function gives us the best approximation model for the random time series that includes seasonal ARIMA or use spectral analysis to analyze the fluctuations of the time series.

5) Since there are many possible models that we could use to approximate the random time series such as ARIMA(0,0,1), ARIMA(2,0,2) and ARIMA(2,0,2)(0,0,1), we need to decide how to choose a best model among those three. There are different criterias to choose a best fit model such as AIC or BIC values, parameters minimization, or prediction errors. Based on the purpose of this analysis which is to make predictions on future values of UPS stock prices, we decide to choose MRSE and MAE values as our criteria to choose best fit model.

6) This is a follow up result from problem 4 and 5. Trade offs from choosing MRSE and MAE values as our criteria are the complexity of the model which is ARIMA(2,0,2)(0,0,1). This model, first of all, has more parameters than other two models. It will get more complicated if we want to do further analysis and might be beyond our current ability when dealing with seasonal ARIMA.

7) The obtained time series of UPS stock prices has unusual presence of trend and seasonal components. It makes the process of choosing appropriate model for the time series become complicated. There are many different criterias to choose a model such as AIC, BIC, number of parameters, MRSE and MAE. Even within MRSE and MAE values, we still did not get an agreed result from the comparison. As we have presented above, even though after considering three different models ARIMA(2,0,2), ARIMA(0,0,1) and ARIMA(2,0,2)(0,0,1), we still did not get the best forecast values for UPS stock price.

8) To produce the prediction interval for UPS stock price, we need prediction intervals of trend, seasonality and random estimations. When we use the forecast() function on stl(), we obtain an overall estimations, which we are not sure if the random component is included or not. If we do use forecast() on stl() to get the estimations for trend and seasonality, our prediction interval for UPS stock price is basically coming from the random component. There is a high probability that this prediction interval would be smaller than what it should be. Therefore, we decide to include the prediction interval provided from the forecast() function used with the stl() function. This might produce a wider interval. However, we think wider interval is safer for our investors.

9) After obtaining the raw periodogram for random time series, we smoothed the periodogram. The evaluation of the smoothing process is done mostly by eyeballing the plot. There is no concrete criteria that we can base on in order to decide when to stop smoothing. Hence, there might be a chance that we smooth the periodogram too much that some important frequencies are not included. This would lead to the inefficiency of our construction of random time series using spectral analysis.

10) We do not have enough knowledge in spectral analysis as well as UPS stock cycles in order to decide which frequencies from periodogram we should take in consideration. Spectral analysis is mainly for understanding the periods of a data. However, we have limited options to discover those periods. Our only option is to choose those frequencies that have peaked spectrum density in consideration to construct our time series. This again would lead to the inefficiency of our spectral analysis modelling. Another limitation that we have in spectral analysis is that we do not know enough R commands that would produces an alternative spectral analysis model like auto.arima() so that we can compare with our model.

11) We notice from the spectral analysis that the random time series has cycles in 8, 12 and 16 months. Hence, it would be nice if we also set up our data as every 4 month and fit ARIMA model to the time series. However, if we setup the time series quarterly, we would have smaller number of observations. Smaller number of observations would leave room for more variances and we would not want so.

Conclusions:

From the spectral analysis, we know that UPS random shocks will vary around 0 within a spreading range of 2.457. Those random shocks in UPS stock price will come in as 8 month, 12 month and 16 month cycles. Hence, investors should expect noticeable changes in UPS stock price every 4 months. As shown in the forecasted plots from both ARIMA model and spectral, UPS stock price will have a small increase with no fluctuations. We expect UPS stock price will go from $118.4 in October, 2017 to $122.7 in October 2019. The price will vary within a range of $97.5 to $147.9. The forecasted trend implies that investors will likely benefit from purchasing UPS stocks for the next two years. However, the profits will not be large as the stock price increases slowly over the two years. Investors need to keep in mind that as we go further into the future, the actual values for UPS stock price will vary further from our predicted values. Moreover, this is just a prediction from one perspective looking at the UPS stock price. There are

many other factors that we suggest investors also take in consideration. For example, the fact that big warehouse Amazon has developed their own logistics department, meaning Amazon uses its own shipping service instead of delivering through a third shipping company. Aside from progresses made by Amazon's shipping services, the idea of using drones to deliver packages has also emerged. In fact Amazon has already delivered packages using drones, being the first company to do so. New changes are surfacing as new ideas and technology develop. Investors who are interested in the supply chain industry would need to consider those changes. However, we also need to keep in mind that UPS is a reputable postal service in the United States that survived economic shocks to this day. The company has strong roots and connections domestically as well as globally so the better choice is to use this analysis as a reference and research other opportunities that UPS is investing to determine if it is coincide with your interests as an investor.

References:

A Little Book of R For Time Series." *a-Little-Book-of-r-for-Time-Series.pdf*, Parasite Genomics Group, Wellcome Trust Sanger Institute, Cambridge, U.K., 15 June 2017, www.bing.com/cr?IG=D37D5FB6034A4CB381F25710DF580D0B&CID=2B6FAE05E3756E223104A541E2736F33&rd=1&h=DrC_uyr1KnxXAH7HRu0p1S69N6J7Tovsr2rT4vKGI-4&v=1&r=https%3a%2f%2fmedia.readthedocs.org%2fpdf%2fa-little-book-of-r-for-time-series%2flatest%2fa-little-book-of-r-for-time-series.pdf&p=DevEx,5067.1.

Taieb, Souhaib Ben, et al. "8.9 Seasonal ARIMA Models." *8.9 Seasonal ARIMA Models | OTexts*, OTexts, www.otexts.org/fpp/8/9.

"United Parcel Service." *Wikipedia*, Wikimedia Foundation, 16 Nov. 2017, en.wikipedia.org/wiki/United_Parcel_Service.

"2016 Annual Report." *UPS 2016 Annual Report*, UPS, 2016, nasdaqomx.mobular.net/nasdaqomx/7/3521/5025/.

"12.1: Estimating the Spectral Density." *12.1: Estimating the Spectral Density | STAT 510*, Penn State Eberly College of Science, 2017, onlinecourses.science.psu.edu/stat510/node/80.

Appendix:

Data description

```
ups.raw = read.csv("~/Desktop/Sta137/UPS.csv")

#clean up data
#separating dates by Month and Year
ups.raw$Date = as.Date(ups.raw$Date)
ups.raw$Month = months(ups.raw$Date)
ups.raw$Year = format(ups.raw$Date, format = "%Y")

#re-ordering by time
Ave = aggregate(Close ~ Month + Year, data = ups.raw, mean)
by_months = Ave[order(match(Ave$Month, month.name)), ]

ups.init = by_months[order(by_months$Year),]
ups = ts(ups.init[,3], start = c(1999,11), freq = 12)

#plotting UPS raw data and monthly data
plot(ups, ylab = "UPS Stock Price($)", main = "Plot of monthly data from UPS Stock Market(1999 -
2017")
ts.plot(ups.raw$Close, ylab = "UPS Stock Price($)", main = "Plot of daily data from UPS Stock
Market(1999 - 2017")
boxplot(ups, main = "Box Plot of UPS Monthly Stock Price")


start(ups) #starting year & month
end(ups) #ending year & month
frequency(ups) #12

layout(1:2)
plot(aggregate(ups)) #plot3
boxplot(ups ~ cycle(ups)) #plot4

#decompose() and stl()
ups.stl = stl(ups,s.window = 12)
ups.decom = decompose(ups)
plot(ups.stl,main = "Decomposition of additive time series from stl()")
plot(ups.decom)

#plots of residuals from decompose() and stl()
plot(ups.stl$time.series[,3], ylab = "x", main = "Residuals plot from stl()")
plot(ups.decom$random, ylab = "x", main = "Residuals Plot from decompose()")
```

```
#acf and pacf plot of x
x = ups.stl$time.series[,3]
x = ts(x,start = c(1999,11), freq = 12)
acf(x)
pacf(x)
```

Data Analysis

```
#model by auto.arima
fit.auto = auto.arima(x, stepwise = FALSE, approximation = FALSE)
fit.auto
#calculate the RMSE and MAE
rmse = function(error)
{
  sqrt(mean(error^2))
}
mae = function(error)
{
  mean(abs(error))
}

#testing the ARIMA(2,0,2)(0,0,1) by forecasting the last 16 observations
actual = x[201:216]
y = x[1:200]
forecast.auto = forecast(y,model = fit.auto, h = 16)
error.auto = actual - forecast.auto
rmse(error.auto)
mae(error.auto)

#choosing models based on BICs
BICs = matrix(0, 6,6)
for (i in 0:5){
  for (j in 0:5){
    fit.BIC = arima(x, order=c(i, 0, j))
    BICs[i+1,j+1] = BIC(fit.BIC)
  }
}

#testing model from BIC, ARIMA(2,0,0) by forecasting the last 16 observations
fit.BIC = arima(x, order = c(2,0,2))
forecast.BIC = forecast(y,model = fit.BIC, h = 16)
error.BIC = actual - forecast.BIC$mean
```

```
rmse(error.BIC)
mae(error.BIC)

#testing model ARIMA(0,0,1) by forecasting the last 16 observations
fit.ma1 = arima(y, order = c(0,0,1))
forecast.ma1 = forecast(y,model = fit.ma1, h = 16)
error.ma1 = actual - forecast.ma1$mean
rmse(error.ma1)
mae(error.ma1)

#acf and pacf of chosen model ARIMA(2,0,2)(0,0,1)
acf(fit.auto$residuals)
pacf(fit.auto$residuals)
plot(resid

#Ljung box test H0 = the model does not exhibit lack of fit
#Ljung test give p-value > 0.05, so accept H0
Box.test(residuals(fit.auto), lag = 17, fitdf = 5, type = "Ljung")


-Arima Model (ARIMA(p,d,q))
ups.differenced = diff(ups, differences = 1)
layout(1:2)
plot.ts(ups)
plot.ts(ups.differenced) #looks like WN

acf(ups.differenced, lag.max=20) #plot a correlogram
acf(ups.differenced, lag.max=20, plot=FALSE) #get the autocorrelation values

pacf(ups.differenced, lag.max=20) #plot a partial correlogram
pacf(ups.differenced, lag.max=20, plot=FALSE) #get the partial autocorrelation values

#forecasting UPS stock in 2 years
forecast.stl = forecast(ups.stl, h =24)
forecast.random = forecast(x,model = fit.auto, h = 24)
plot(forecast.random)
forecast.final =ts(ts(forecast.stl$mean) +ts(forecast.random$mean), start = c(2017, 11), freq = 12)
ts.plot(cbind(ups, forecast.final), lty = 1:2, ylab = "Price", main = "Predicted Values for UPS Stock Price
in the next 2 years")

#forecasting using auto.arima
auto = auto.arima(ups)
auto.predict = forecast(ups, model = auto, h = 24)
```

```
plot(auto.predict)

#comparing the 95% confidence interval between manually and auto.arima
Hi95 = forecast.random$upper[,2] + forecast.stl$upper[,2]
Lo95 = forecast.random$upper[,1] + forecast.stl$upper[,1]
confi95 = Hi95 - Lo95
confi95.auto = auto.predict$upper[,2] - auto.predict$upper[,1]
check = rep(0,24)
for (i in 1:24){
  if(confi95 <= confi95.auto) check[i ]= 1
  else check[i] = 0
}

y = data.frame(confi95,confi95.auto, check)
write.table(y, file = 'compare.csv', sep = ",",row.names = F)

#spectral analysis
plot(x, ylab = "UPS Random")
x = ts(x, freq = 1)
#raw periodogram
spec = spectrum(x,taper = 0, log = c("no"))
abline(v = specvalues$freq[15], lty ="dotted")
abline(v = specvalues$freq[20], lty = "dotted")
abline(v = specvalues$freq[29], lty = "dotted")

u = qchisq(.025, 2); l = qchisq(.975, 2)
2*spec$spec[15]/l
2*spec$spec[15]/u
2*spec$spec[20]/l
2*spec$spec[20]/u
2*spec$spec[29]/l
2*spec$spec[29]/u

#smoothing periodogram
spec2 = spec.pgram(x, spans = 4, taper = 0, log = "no")
abline(v = specvalues$freq[14], lty ="dotted")
abline(v = specvalues$freq[20], lty = "dotted")
abline(v = specvalues$freq[28], lty = "dotted")

df = ceiling(spec2$df)
u = qchisq(.025, df); l = qchisq(.975, df)
df*spec2$spec[14]/l
df*spec2$spec[14]/u
```

```
df*spec2$spec[20]/l
df*spec2$spec[20]/u
df*spec2$spec[28]/l
df*spec2$spec[28]/u

specvalues = spec.pgram(x, spans = c(4,2), taper = 0, log = "no")
abline(v = specvalues$freq[14], lty ="dotted")
abline(v = specvalues$freq[20], lty = "dotted")
abline(v = specvalues$freq[28], lty = "dotted")

df = ceiling(specvalues$df)
u = qchisq(.025, df); l = qchisq(.975, df)
df*specvalues$spec[14]/l
df*specvalues$spec[14]/u
df*specvalues$spec[20]/l
df*specvalues$spec[20]/u
df*specvalues$spec[28]/l
df*specvalues$spec[28]/u

spec4 = spec.pgram(x, spans = c(4,4), taper = 0, log = "no")
abline(v = specvalues$freq[14], lty ="dotted")
abline(v = specvalues$freq[28], lty = "dotted")

df = ceiling(spec4$df)
u = qchisq(.025, df); l = qchisq(.975, df)
df*spec4$spec[14]/l
df*spec4$spec[14]/u
df*spec4$spec[28]/l
df*spec4$spec[28]/u

#obtaining peaks' freq from ar method
ar.spec = spec.ar(x, log = "no")
ar.spec
f1 = ar.spec$freq[65]
f2 = ar.spec$freq[132]
f3 = ar.spec$freq[211]
f4 = ar.spec$freq[278]
f5 = ar.spec$freq[365]
abline(v = f1, lty="dotted")
abline(v = f2, lty="dotted")
abline(v = f3, lty="dotted")
abline(v = f4, lty = "dotted")
abline(v = f5, lty = "dotted")
```

```
#generating random time series
#frequency f1
t = 1:216

 #generating sine and cosine from frequencies chosen in parametric
cos1 = array(0,216)
sin1 = array(0,216)
cos2 = array(0,216)
sin2 = array(0,216)
cos3 = array(0,216)
sin3 = array(0,216)
cos4 = array(0,216)
sin4 = array(0,216)
cos5 = array(0,216)
sin5 = array(0,216)
for (i in 1:216){
  cos1[i] = cos(2*pi*i*f1)
  sin1[i] = sin(2*pi*i*f1)
  cos2[i] = cos(2*pi*i*f2)
  sin2[i] = sin(2*pi*i*f2)
  cos3[i] = cos(2*pi*i*f3)
  sin3[i] = sin(2*pi*i*f3)
  cos4[i] = cos(2*pi*i*f4)
  sin4[i] = sin(2*pi*i*f4)
  cos5[i] = cos(2*pi*i*f5)
  sin5[i] = sin(2*pi*i*f5)
}

#generating A's and B's
A1 = 2/216*x%*%cos1
B1 = 2/216*x%*%sin1
A2 = 2/216*x%*%cos2
B2 = 2/216*x%*%sin2
A3 = 2/216*x%*%cos3
B3 = 2/216*x%*%sin3
A4 = 2/216*x%*%cos4
B4 = 2/216*x%*%sin4
A5 = 2/216*x%*%cos5
B5 = 2/216*x%*%sin5

#generating different time series
x1 = A1*cos(2*pi*t*f1)+B1*sin(2*pi*t*f1)
```

```
x2 = A2*cos(2*pi*t*f2)+B2*sin(2*pi*t*f2)
x3 = A3*cos(2*pi*t*f3)+B3*sin(2*pi*t*f3)
x4 = A4*cos(2*pi*t*f4)+B4*sin(2*pi*t*f4)
x5 = A5*cos(2*pi*t*f5)+B5*sin(2*pi*t*f5)

#predicted time series
xtilde1 = x1 + x2 + x3 + rnorm(216)
xtilde2 = x1 + x2 + x3 + x4 + rnorm(216)
xtilde3 = x1 + x2 + x3 +x4 + x5 +rnorm(216)

#MRSE and MAE of different predicted time series
error.tilde1 = x - xtilde1
error.tilde2 = x - xtilde2
error.tilde3 = x - xtilde3
re1 = rmse(error.tilde1)
re2 = rmse(error.tilde2)
re3 = rmse(error.tilde3)
mae1 = mae(error.tilde1)
mae2 = mae(error.tilde2)
mae3 = mae(error.tilde3)

#generating sine and cosine with frequencies chosen from smoothing
cos11 = array(0,216)
sin11 = array(0,216)
cos21 = array(0,216)
sin21 = array(0,216)
cos31 = array(0,216)
sin31 = array(0,216)
cos41 = array(0,216)
sin41 = array(0,216)
cos51 = array(0,216)
sin51 = array(0,216)
for (i in 1:216){
  cos11[i] = cos(2*pi*i*f11)
  sin11[i] = sin(2*pi*i*f11)
  cos21[i] = cos(2*pi*i*f22)
  sin21[i] = sin(2*pi*i*f22)
  cos31[i] = cos(2*pi*i*f33)
  sin31[i] = sin(2*pi*i*f33)
  cos41[i] = cos(2*pi*i*f44)
  sin41[i] = sin(2*pi*i*f44)
  cos51[i] = cos(2*pi*i*f55)
  sin51[i] = sin(2*pi*i*f55)
```

```
}

#generating A's and B's
A11 = 2/216*x%*%cos11
B11 = 2/216*x%*%sin11
A21 = 2/216*x%*%cos21
B21 = 2/216*x%*%sin21
A31 = 2/216*x%*%cos31
B31 = 2/216*x%*%sin31
A41 = 2/216*x%*%cos41
B41 = 2/216*x%*%sin41
A51 = 2/216*x%*%cos51
B51 = 2/216*x%*%sin51

#generating different time series
x11 = A11*cos(2*pi*t*f11)+B11*sin(2*pi*t*f11)
x21 = A21*cos(2*pi*t*f22)+B21*sin(2*pi*t*f22)
x31 = A31*cos(2*pi*t*f33)+B31*sin(2*pi*t*f33)
x41 = A41*cos(2*pi*t*f44)+B41*sin(2*pi*t*f44)
x51 = A51*cos(2*pi*t*f55)+B51*sin(2*pi*t*f55)

#predicted time series
xtilde11 = x11 + x21 + x31 + rnorm(216)
xtilde21 = x11 + x21 + x31 + x41 + rnorm(216)
xtilde31 = x11 + x21 + x31 +x41 + x51 +rnorm(216)

#MRSE of different predicted time series
error.tilde11 = x - xtilde11
error.tilde21 = x - xtilde21
error.tilde31 = x - xtilde31
error.auto = x - as.vector(fit.auto$fitted)

re11 = rmse(error.tilde11)
re21 = rmse(error.tilde21)
re31 = rmse(error.tilde31)
reauto = rmse(error.auto)
mae11 = mae(error.tilde11)
mae21 = mae(error.tilde21)
mae31 = mae(error.tilde31)

#RMSE and MAE for ARIMA(2,0,2)(0,0,1)
reauto = rmse(error.auto)
mae.auto = mae(error.auto)
```

```
#write results into table
Root =  c(re11, re21, re31, re1, re2, re3)
MAman = c(mae11, mae21, mae31, mae1, mae2, mae3)
z = data.frame(Root, MAman)
write.table(z, file = 'error.csv', sep = ',', row.names = F)

#plot fitted values from spectral analys and arima
ts.plot(x, main = "Random time series and fitted values from spectral analysis")
lines(xtilde21, col = "red")
ts.plot(x, main = "Random time series and fitted values from ARIMA model")
fitted.auto = ts(fit.auto$fitted, start =1 , freq =1)
lines(fitted.auto, col =2)

#calculating the variance of x time series
area = s$spec[1]+s$spec[108]
for (i in 2:107){
  area = area + 2*(s$spec[i])
}
area = area*s$bandwidth
area
```