New York Stock Exchange Time Series Analysis

CSC 425 Final Project

Kari Palmier

Table of Contents

I. Introduction

This project studies how time series analysis can be applied to real-world stock data. The dataset chosen is the New York Stock Exchange dataset from Kaggle (https://www.kaggle.com/dgawlik/nyse). It contained daily stock prices from January 2, 2010 through December 30, 2016 for securities in the S&P 500. This dataset contained two different data files with daily price information (one raw and one adjusted for stock splits), as well as fundamental information and security information. I decided to use the data file that had been adjusted for stock splits because the prices would be consistent over time. I focused my work on the analysis of the closing prices of the dataset, since closing prices represent the final result of trading each day. I decided to perform my exploratory analysis on a set of technology stocks that I found interesting, then to use the results of this analysis to chose one of the stocks to model and forecast. The initial set of stocks I looked at was Apple, Cisco, Alphabet Class A, Alphabet Class C, HP Inc., Intel, Microsoft, Nvidia, Oracle, Red Hat, Texas Instruments, Western Digital, Xerox, and Yahoo. I chose these based on my interests in the field of information technology and electronics. The final stock that I decided to model was Apple. I chose this based on the results of the exploratory analysis, as well as due to a personal interest.

II. Exploratory Analysis

The closing price data provided in the dataset was provided daily through the entire period of time (January 4, 2010 through December 30, 2016) for the stocks I chose to initially look at. There were other stocks of interest such as Facebook and HP Enterprises, but these only had a few years worth of data. I started out my analysis looking at the daily granularity of each stock. I performed the kurtosis test, skewness test, Jacque Bera normality test, Ljung Box autocorrelation tests for lags 5, 10, and 25, the zero mean no trend augmented Dickey Fuller stationarity test, the constant mean no trend Dickey Fuller stationarity test, and the constant mean with trend Dickey Fuller stationarity test on the closing prices of all of the stocks individually. I also plotted the closing prices versus time and the closing prices ACF, PACF, and normal probability plots for each stock. Note that these sets of tests and plots were created for all stages in the exploratory analysis (for every stock, every granularity, and every transformation attempted). I found unsurprisingly that the closing prices were not normal, not stationary, and contained a positive time trend (from being able to reject the Jacque Bera test, not being able to reject the augmented Dickey Fuller tests, and from the time plot).

I then performed several transformations on the closing prices for each stock to see which would result in normal and stationary behavior. I performed a log transformation, the difference, and the difference of the log transformation, as well as computed the gross return (Pt / Pt-1) and the simple return ((Pt / Pt-1) -1). After evaluating the test results and plots generated, I found that the difference, the log of the difference, and the simple return were the only transformations that were stationary (from being able to reject the augmented Dickey Fuller tests). Since the difference of the log and the difference transformations were very similar, I decided to only focus on the difference and the simple return transformations. The difference and simple return data were still not normal and had very noisy ACF and PACF responses at the daily granularity.

Next, I decided to see how weekly and monthly granularities impacted the normality of the data. I used the built in R aggregate function to aggregate the daily data to monthly, using the average of the daily stocks of each month as the value to represent each month. I manually aggregated the daily data to weekly in R and used the average of the week of daily stock values to represent each week. I aggregated the data by finding all weekends (the dataset only contained data on non-holiday weekdays). I then used the placement of the weekends to determine the start and stop of each week. I assigned the date of each week to the friday of the week because the average included data through the end of the week. I found that the majority of monthly data for all stocks (both difference and simple return transformed) were normal and stationary (not able to reject the Jacque Bera test and able to reject the augmented Dickey Fuller tests). There were only 84 data points present in the data for each stock at the monthly granularity though. I felt this was too small of a dataset to be comfortable using for back testing since it would have to be separated into training and testing (leaving an even smaller number of training points after the split). Next, I examined the weekly data for all of the stocks. The weekly data contained 587 data points, which was more than enough to allow for a comfortable training and testing split. All of the weekly difference transformed stocks were not normal unfortunately. The Apple, Intel, and Yahoo stocks were the only ones that were normal for the simple return data. The Ljung Box test showed autocorrelation in the first 5 lags for all three of these stocks, but this is expected and desired for modeling (the current time needs to be correlated to past time in order for a model to be generated).

At this point, I chose to proceed with the weekly Apple simple returns as the stock data to model. I chose this instead of Intel or Yahoo simply because I was more interested in it than the others. All three would were shown to be equally acceptable statistically from the exploratory analysis.

Below are the test results for the Apple weekly simple return data. Refer to Appendix I for the test results of the rest of the stocks for the daily, weekly, and monthly granularities for lag of 5, as well as the original Apple daily closing price and weekly simple return plots.

The Jacque Bera, kurtosis, and skewness tests showed normal behavior for all lags for the Apple simple return data. The augmented Dickey Fuller tests show stationarity for 5 and 10 lags. Non-stationary behavior of 25 lags may be due to noise in the dataset.

Number of Lags used in Ljung Box and ADF Tests	Weekly Simple Return JB Test P Value	Weekly Simple Return Kurtosis Test P Value	Weekly Simple Return Skewness Test P Value	Weekly Simple Return Ljung Box Test P Value	Weekly Simple Return No Intercept No Trend DF Test P Value	Weekly Simple Return Constant Intercept, No Trend DF Test P Value	Weekly Simple Return Constant Intercept With Trend DF Test P Value
5 Lags	0.5421	0.2734	0.6999	1.68E-03	0.01	0.01	0.01
10 Lags	0.5421	0.2734	0.6999	4.84E-03	0.01	0.01	0.01
25 Lags	0.5421	0.2734	0.6999	2.38E-03	0.01572	0.06898	0.2343

III. Modeling

I performed an ARCH analysis on the weekly Apple simple return data to determine if a volatility model was appropriate. I performed Ljung Box tests on both the square and abolsute of the Apple simple returns at lags of 5, 10, and 25, as well as plotted the ACF of the both. I found that there was no ARCH effect present (was unable to reject the Ljung Box tests at any lag value and ACF plots quickly dropped to zero after lag 0). Since there was no ARCH effect, a volatility model was not required.

I began the modeling of the Apple simple return data by performing the R auto.Arima. This function recommended an MA(1) moving average model. This matches the assumption from the ACF plot with a lag 1 significance. From here, I decided to evaluate several models including the addition of other stocks as x regressors. For each model, I performed coefficient tests to check for significance, calculated the characteristic polynomial roots, performed the Jacque Bera test on the model residuals to test their normality, performed Ljung Box tests with lags of 5, 10, and 25 on the residuals to test if they are white noise, and performed the augmented Dickey Fuller zero mean no trend, constant mean no trend, and constant mean with trend tests with a lag of 5 on the residuals to test if they are stationary. I also plotted the residuals versus time, the ACF of the residuals, and the normal probability plot of the residuals. I then performed back testing with a training set size of 80% and a testing set of 20%.

I first generated three basic models, MA(1), ARMA(1), and AR(1). These models all had very similar results. They all had similar BIC values of around -1540 and back testing MAPE value of around 1.2. The residual testing showed they all had normal residuals (could not reject the Jacque Bera test) and were all stationary (rejected all Dickey Fuller tests). The Ljung Box tests showed the residuals were white noise for all lags of the MA(1) and ARMA(1) model and were white noise for 5 and 10 lags for the AR(1). The residual normal plot for the MA(1) model had slightly better tail behavior than the ARMA(1) and AR(1) (they all had very small tails but MA(1) was closer to the normal line the longest).

I decided that since the MA(1) model had the best residual behavior, that the MA(1) would be my base model. I then began adding in x regressors. I chose to use the original (non-delayed) simple return data for Cisco, Alphabet Class A, Alphabet Class C, Oracle, Red Hat, Texas Instruments, and Yahoo stocks as x regressors due to their Apple weekly simple return CCF plots (refer to Appendix I for the CCF plots). I started by including all of these stocks into the model. I found that a number of them were not significant from the coefficient test. I kept eliminating the least significant (highest coefficient p value) stock for the model until I reached a model with only significant coefficients. This model had Alphabet Class C, Oracle, and Texas Instruments present. The Jacque Bera test showed the residuals of this model were normal, but the residual normal plot had much larger tails than the MA(1) without x regressors. The BIC was lower than the MA(1) (was -1644 compared to -1541), but this was not a significant difference (both values are very negative). The back-testing MAPE was also lower than the MA(1) (1.07% compared to 1.23%), but again this is not a significant difference. The Ljung Box test of the residuals showed they were white noise for 5 lags but were not white noise for 10 and 25 lags. Since the residuals were worse for this model with x regressors, I decided to see what the different combinations of the three final x regressors would yield. I thought it may be possible that the residuals for one of the combinations would be adequate. I tried Alphabet Class C, Oracle, and Texas Instruments each by themselves, then Alphabet Class C and Oracle, Alphabet Class C and Texas Instruments, and Oracle and Texas Instruments. None of these combinations yielded improved residuals. I decided that the final model to use for was the MA(1) model without x regressors.

Because the MA(1) model was stationary (by definition for a moving average model) and its residuals passed all of the required tests, it was able to be used for forecasting. I performed forecasting with a horizon of 10 on the final MA(1) model, even though only one step could actually be forecasted due to the model order being 1 and it being a moving average model (I did this to see the model converge to the mean in the forecasting plot). I found that the one-step ahead forecast of obtained for one week in the future was 0.00277. This value followed the expected pattern from the observed data. The plot of the observed time series with the forecasted points added does show that after this one-step ahead forecast, the remaining forecasts converge to the model mean of 0.0041.

Below are the results for the final MA(1) model. Refer to Appendix II for the results of all of the models, as well as the residual, back-testing, and forecasting plots for the final MA(1) model.

ARCH Analysis Results:

Number of Lags Used	Squared Weekly Simple Return Ljung Box Test P Value	Absolute of Weekly Simple Return Ljung Box Test P Value
5 Lags	0.3021	0.5932
10 Lags	0.4175	0.1939
25 Lags	0.8155	0.66

MA(1) Model Residual Analysis Test Results:

Model Name	Model Order	Model BIC	Residual JB Normal Test	Residual Ljung Box 5 Lag Test P Val	Residual Ljung Box 10 Lag Test P Val	Residual Ljung Box 25 Lag Test P Val	Residual No Intercept, No Trend DF Test P Value	Residual Constant Intercept, No Trend DF Test P Value	Residual Constant Intercept With Trend DF Test P Value	Model Back Testing RMSE	Model Back Testing MAPE
Apple Simple Return MA(1)	1	-1541.1	0.8009	0.2332	0.2131	0.1382	0.01	0.01	0.01	0.02457	1.22655

IV. Conclusion

The MA(1) model was found to best capture the behavior of the Apple simple return data because it was stationary, had residuals that were white noise and were the most normal of all the models, and had low BIC and back-testing MAPE values. Because the model fit all the criterion, it was found adequate to use for forecasting and the one-step ahead forecasted value was 0.00277. This forecasted value showed an increase from the prior value (the last value of the dataset). This means that the forecast shows an increase in the Apple simple returns. Since this is means that investors will make money because the price of stock will increase, causing the return to increase, the recommendation based on this model would be that a person should invest in Apple stock.

The final MA(1) model equation obtained from R is:
Rt = 0.0041 + at – 0.2634*at-1 where Rt is the simple return at time t and at is white noise

Note that the modeling in this analysis was performed using weekly average values of the daily Apple closing prices. This means that any fluctuation seen day to day during each week is not captured. It is possible that this fluctuation could have an effect on the actual future values of the stock. This means that although the model in this analysis predicted the weekly stock properly, the day in which the stock is purchased in the future may cause the price to vary away from the prediction. This model can only be used to predicted average weekly stocks and cannot be applied to the actualy daily granularity of the stock market. Also, this model can only predict one week in the future due to the constraint of the moving average model having an order of 1 (all forecasts past one step ahead are the model mean by definition).

Appendices

I. Supplemental Exploratory Materials

This section contains some of the visualizations and tables used during the exploratoray analysis stage. I have only included plots for the chosen Apple stock data (the original daily closing price data and the weekly simple return data used for modeling). The plots for the rest of the stocks, granularities, and transforms are in the CSC 425 Project R Output.zip file included in the project submission folder in D2L. This zip file also includes text files with the R output for all of the data evaluated. The tables with the test results with the Ljung Box and augmented Dickey Fuller 10 and 20 lag values, as well as a summary of the ACF and PACF plots for all stocks and the CCF plots for stock combinations are in the CSC 425 Project Analysis.xlsx file, also included in the project submission folder in D2L.

1. Exploratory Results Summary

The tables below show the test results for the daily, weekly, monthly and monthly granularity for all of the stocks, with 5 lags used for the Ljung Box and augmented Dickey Fuller tests.

Daily Granularity Test Results

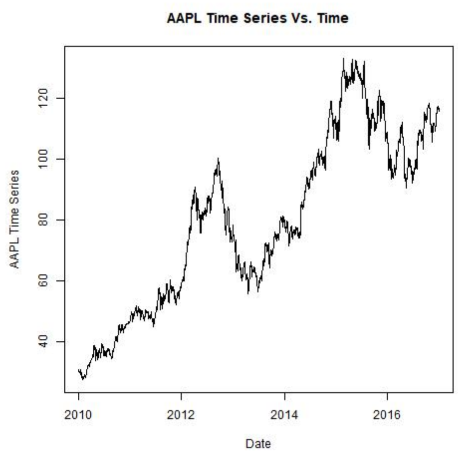
Ticker Symbol	Daily Diff JB Test P Value	Daily Diff Kurtosis Test P Value	Daily Diff Skewness Test P Value	Daily Diff Ljung Box Test P Value	Daily Diff No Interc No Trend DF Test P Value	Daily Diff Const Interc No Trend DF Test P Value	Daily Diff Const Interc W Trend DF Test P Value		Daily Simple Return JB Test P Value	Daily Simple Return Kurtosis Test P Value	Daily Simple Return Skewness Test P Value	Daily Simple Return Ljung Box Test P Value	Daily Simple Return No Interc No Trend DF Test P Value	Daily Simple Return Const Interc No Trend DF Test P Value	Daily Simple Return Const Interc W Trend DF Test P Value
AAPL	2.20E-16	0	1.20E-05	4.00E-03	0.01	0.01	0.01		2.20E-16	0	0.0475	1.26E-04	0.01	0.01	0.01
CSCO	2.20E-16	0	0	0.3012	0.01	0.01	0.01		2.20E-16	0	3.56E-10	0.1081	0.01	0.01	0.01
GOOG	2.20E-16	0	0	2.07E-03	0.01	0.01	0.01		2.20E-16	0	0	2.69E-02	0.01	0.01	0.01
GOOGL	2.20E-16	0	0	4.47E-04	0.01	0.01	0.01		2.20E-16	0	0	3.24E-02	0.01	0.01	0.01
HPQ	2.20E-16	0	0	2.73E-04	0.01	0.01	0.01		2.20E-16	0	0	2.02E-02	0.01	0.01	0.01
INTC	2.20E-16	0	0.058	0.7806	0.01	0.01	0.01		2.20E-16	0	0.428	0.7386	0.01	0.01	0.01
MSFT	2.20E-16	0	0.011	7.89E-02	0.01	0.01	0.01		2.20E-16	0	0.1903	0.2032	0.01	0.01	0.01
NVDA	2.20E-16	0	0	2.20E-16	0.01	0.01	0.01		2.20E-16	0	0	0.03291	0.01	0.01	0.01
ORCL	2.20E-16	0	4.05E-12	5.83E-02	0.01	0.01	0.01		2.20E-16	0	4.99E-14	2.01E-03	0.01	0.01	0.01
RHT	2.20E-16	0	0.144	4.06E-02	0.01	0.01	0.01		2.20E-16	0	0	3.37E-02	0.01	0.01	0.01
TXN	2.20E-16	0	1.18E-14	6.94E-03	0.01	0.01	0.01		2.20E-16	0	3.59E-10	0.000136	0.01	0.01	0.01
WDC	2.20E-16	0	0.314	5.22E-02	0.01	0.01	0.01		2.20E-16	0	1.64E-11	2.08E-02	0.01	0.01	0.01
XRX	2.20E-16	0	0	0.00492	0.01	0.01	0.01		2.20E-16	0	0	0.04943	0.01	0.01	0.01
YHOO	2.20E-16	0	1.49E-08	7.93E-04	0.01	0.01	0.01		2.20E-16	0	0.097	9.68E-03	0.01	0.01	0.01

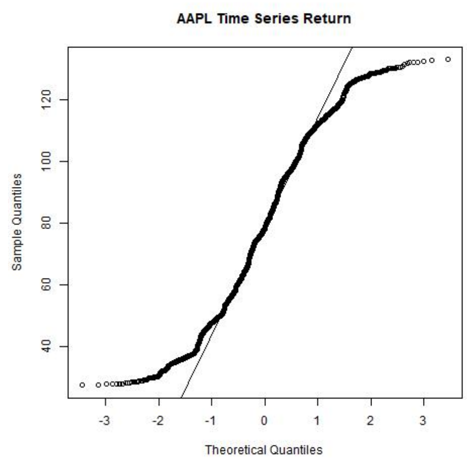
Monthly Granularity Test Results																
Ticker Symbol	Monthly Diff JB Test P Value	Monthly Diff Kurtosis Test P Value	Monthly Diff Skewness Test P Value	Monthly Diff Ljung Box Test P Value	Monthly Diff No Interc No Trend DF Test P Value	Monthly Diff Const Interc No Trend DF Test P Value	Monthly Diff Const Interc W Trend DF Test P Value		Monthly Simple Return JB Test P Value	Monthly Simple Return Kurtosis Test P Value	Monthly Simple Return Skewness Test P Value	Monthly Simple Return Ljung Box Test P Value	Monthly Simple Return No Interc No Trend DF Test P Value	Monthly Simple Return Const Interc No Trend DF Test P Value	Monthly Simple Return Const Interc W Trend DF Test P Value	
AAPL	0.3999	0.929	0.1898	8.77E-01	0.01632	0.08839	0.3109		0.7188	0.6541	0.464	9.45E-01	0.0168	0.06528	0.2093	
CSCO	0.4679	0.7667	0.2304	0.002582	0.01	0.01085	0.03892	***	0.6353	0.861	0.3775	0.007476	0.01	0.01	0.03726	***
GOOG	0.006479	0.0676	0.0157	6.73E-01	0.01	0.01	0.01		0.4767	0.6331	0.3049	6.03E-01	0.01	0.01	0.01	*****
GOOGL	0.000333	0.0091	0.00509	6.16E-01	0.01	0.01	0.01		0.4046	0.5519	0.2713	5.90E-01	0.01	0.01	0.01	*****
HPQ	0.1439	0.8748	0.05544	0.7404	0.01	0.03439	0.09856		0.8658	0.5558	0.7951	8.13E-01	0.01	0.02594	0.085	
INTC	0.6175	0.5057	0.4226	0.5145	0.01	0.01284	0.05399		0.5567	0.3348	0.5119	0.7121	0.01	0.01	0.04322	*****
MSFT	0.03184	0.02162	0.4109	6.11E-02	0.01	0.01	0.01		0.7711	0.5684	0.9727	0.2693	0.01	0.01	0.01	*****
NVDA	2.20E-16	0	0	1.23E-05	0.99	0.99	0.99		2.83E-06	2.49E-05	0.019	0.5221	0.01	0.01	0.01	
ORCL	0.2327	0.7006	0.6888	2.99E-01	0.01	0.01	0.03104	*****	0.2779	0.1553	0.8692	1.88E-01	0.01	0.01	0.01258	*****
RHT	0.2059	0.6719	0.09812	6.21E-03	0.01	0.01	0.01822	***	0.7179	0.704	0.4472	1.46E-02	0.01	0.01	0.01342	***
TXN	0.2346	0.6163	0.12205	9.94E-01	0.01	0.01	0.01	*****	0.8142	0.6758	0.7652	0.9703	0.01	0.01	0.01	*****
WDC	0.004441	0.1159	0.0061	1.77E-01	0.01966	0.1684	0.4127		0.9886	0.8855	0.8823	7.10E-01	0.01	0.03042	0.1059	
XRX	0.8461	0.5533	0.74203	0.767	0.01	0.02413	0.08639		0.7756	0.8868	0.4838	0.5985	0.01	0.02206	0.07899	
YHOO	0.1135	0.08109	0.4269	4.09E-01	0.01	0.01778	0.07469		0.6816	0.6556	0.4236	3.10E-01	0.01	0.02095	0.08599	

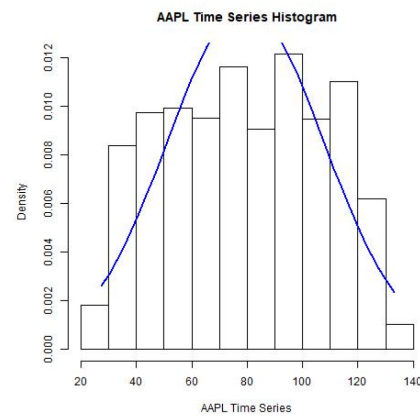
Weekly Granularity Test Results																
Ticker Symbol	Weekly Diff JB Test P Value	Weekly Diff Kurtosis Test P Value	Weekly Diff Skewness Test P Value	Weekly Diff Ljung Box Test P Value	Weekly Diff No Interc No Trend DF Test P Value	Weekly Diff Const Interc No Trend DF Test P Value	Weekly Diff Const Interc W Trend DF Test P Value		Weekly Simple Return JB Test P Value	Weekly Simple Return Kurtosis Test P Value	Weekly Simple Return Skewness Test P Value	Weekly Simple Return Ljung Box Test P Value	Weekly Simple Return No Interc No Trend DF Test P Value	Weekly Simple Return Const Interc No Trend DF Test P Value	Weekly Simple Return Const Interc W Trend DF Test P Value	
AAPL	0.03002	0.2394	0.02001	1.62E-05	0.01	0.01	0.01		0.5421	0.2734	0.6999	1.68E-03	0.01	0.01	0.01	***
CSCO	2.20E-16	0	6.23E-06	2.67E-07	0.01	0.01	0.01		2.20E-16	0	9.54E-05	3.46E-06	0.01	0.01	0.01	
GOOG	2.20E-16	0	1.84E-05	1.74E-03	0.01	0.01	0.01		2.20E-16	0	4.12E-05	1.03E-03	0.01	0.01	0.01	
GOOGL	2.20E-16	0	5.03E-06	2.10E-03	0.01	0.01	0.01		2.20E-16	0	4.62E-05	1.15E-03	0.01	0.01	0.01	.
HPQ	2.20E-16	0	3.95E-09	1.52E-03	0.01	0.01	0.01		3.58E-10	2.16E-10	0.1684	1.06E-04	0.01	0.01	0.01	
INTC	0.001806	0.003402	0.05966	0.001224	0.01	0.01	0.01		0.2549	0.1272	0.6734	0.000633	0.01	0.01	0.01	***
MSFT	2.20E-16	0	0.4434	0.2788	0.01	0.01	0.01		2.20E-16	0	0.1221	0.3827	0.01	0.01	0.01	
NVDA	2.20E-16	0	0	2.20E-16	0.01	0.01	0.01		2.20E-16	0	2.44E-15	3.48E-06	0.01	0.01	0.01	
ORCL	3.23E-09	1.21E-08	0.01927	2.07E-03	0.01	0.01	0.01		2.60E-10	4.73E-10	0.0433	1.48E-03	0.01	0.01	0.01	
RHT	2.20E-16	0	0.6736	1.01E-03	0.01	0.01	0.01		1.69E-07	7.53E-08	0.2676	2.80E-03	0.01	0.01	0.01	
TXN	2.30E-13	8.30E-14	0.3482	0.004742	0.01	0.01	0.01		0.003735	0.00111	0.9793	0.005941	0.01	0.01	0.01	
WDC	2.20E-16	1.20E-14	1.14E-05	4.40E-05	0.01	0.01	0.01		4.21E-06	1.24E-06	0.5546	1.25E-02	0.01	0.01	0.01	
XRX	2.20E-16	5.04E-13	3.36E-09	0.000111	0.01	0.01	0.01		2.03E-08	5.96E-06	0.000184	5.57E-05	0.01	0.01	0.01	
YHOO	1.11E-15	1.11E-15	0.1027	4.15E-05	0.01	0.01	0.01		0.2973	0.2751	0.3015	3.87E-03	0.01	0.01	0.01	***

2. Apple Daily Closing Price Plots

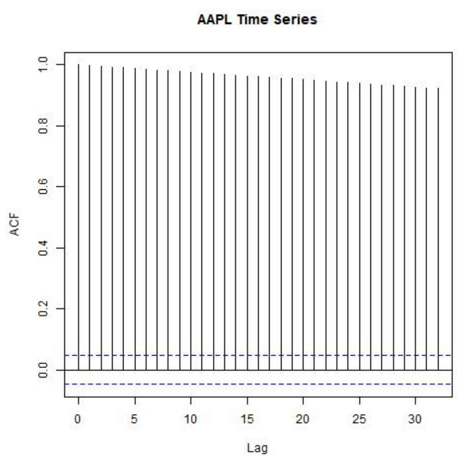
The time plot shows non-stationary behavior with a positive trend. The normal plot shows thick tails. The histogram also shows thick tailed behavior.

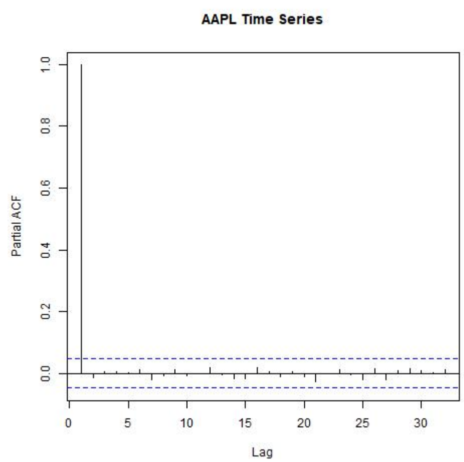






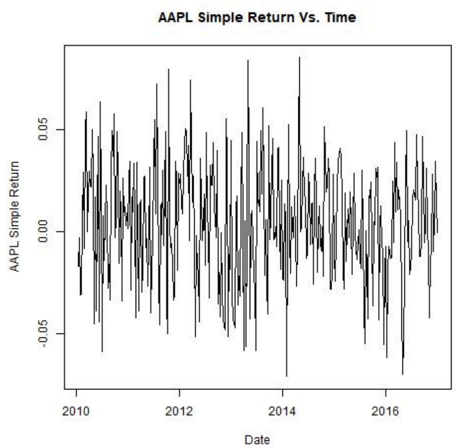
The ACF plot shows a slow decay (non-stationary behavior). The PACF plot shows no significant lags.

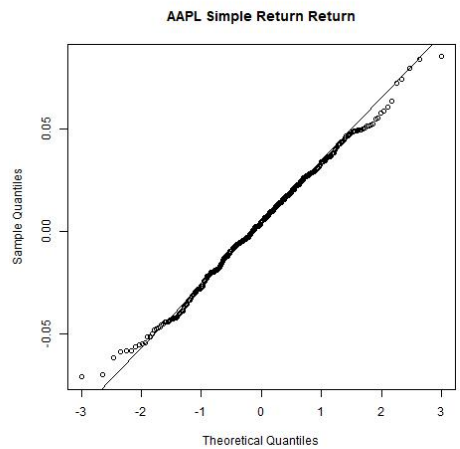


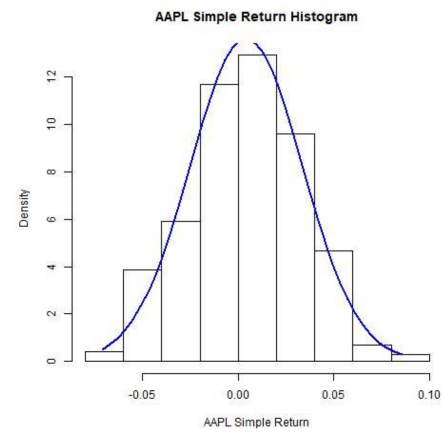


3. Apple Weekly Simple Return Plots

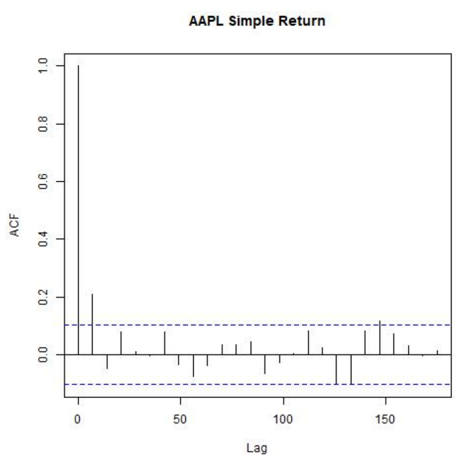
The time plot shows stationary behavior with zero mean and constant variance. The normal plot shows mostly normal behavior with only slight tails. The histogram also shows normal behavior (no skewness or tails).

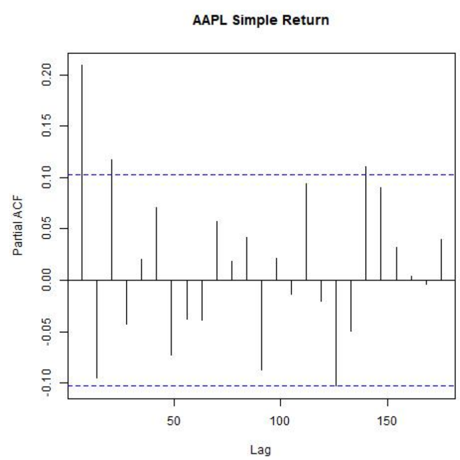






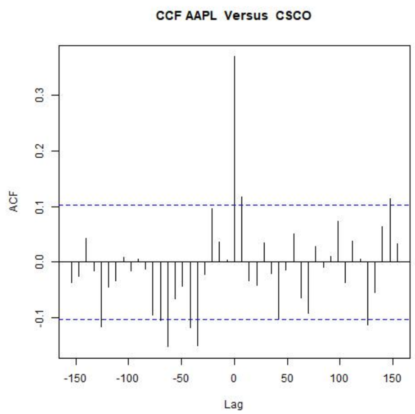
The ACF plot shows one significant lag at lag 1. The PACF plot shows several lags exceeding the confidence limits, but since the y-axis range is very low, these are most likely due to noise.

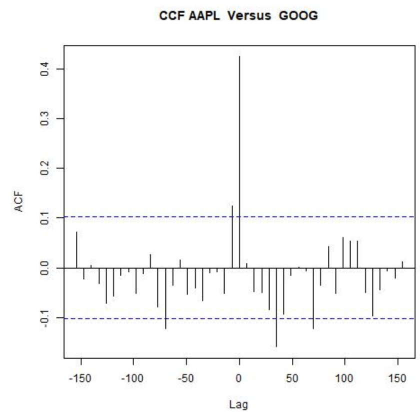


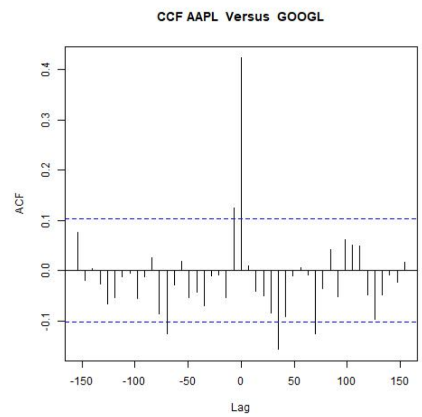


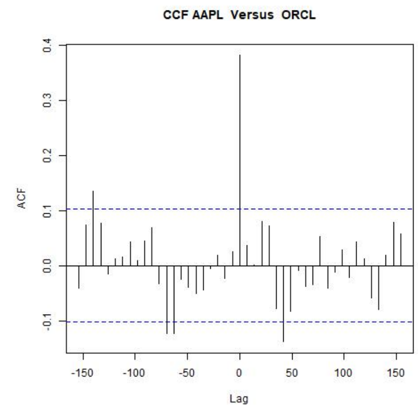
4. Apple Weekly Simple Return CCF Plots

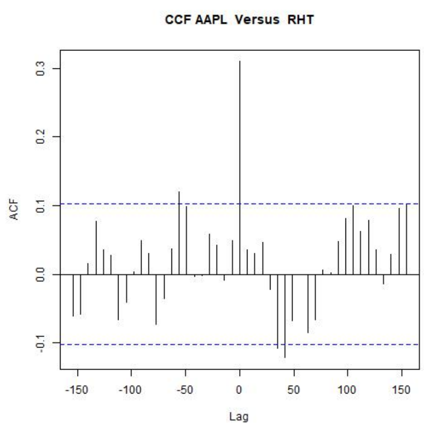
The plots below show the CCF between Apple (AAPL) and Cisco (CSCO), Alphabet Class A (GOOGL), Alphabet Class C (GOOG), Oracle (ORCL), Red Hat (RHT), Texas Instruments (TXN), and Yahoo (YHOO).

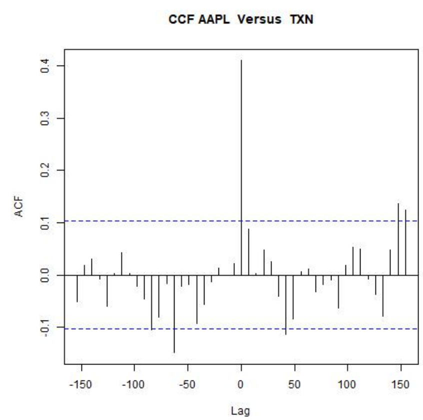


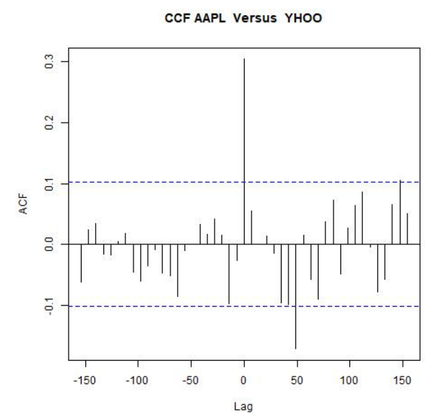










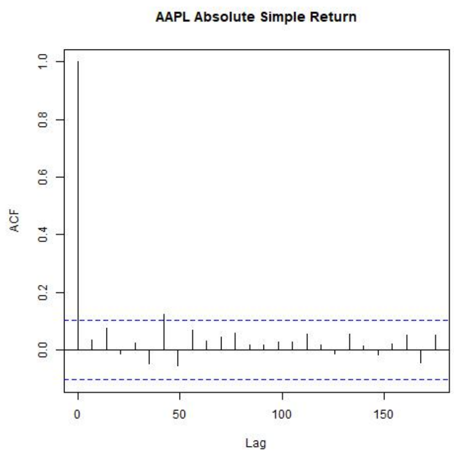


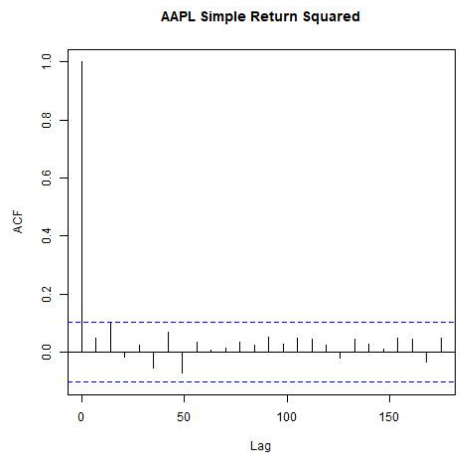
II. Supplemental Modeling Materials

This section contains all of the visualization and tables for the modeling and prediction. I have only included plots for the chosen MA(1) model. The plots for the rest of the models are in the CSC 425 Project Analysis.zip file included in the project submission folder in D2L. This zip file also includes text files with the R output for all of the models evaluated. The tables with the test for all models are in the CSC 425 Project Analysis.xlsx file, also included in the project submission folder in D2L

1. Apple Simple Return ARCH Plots

The ACF of both the absolute and squared Apple simple returns do not show any autocorrelations, therefore no ARCH effect.





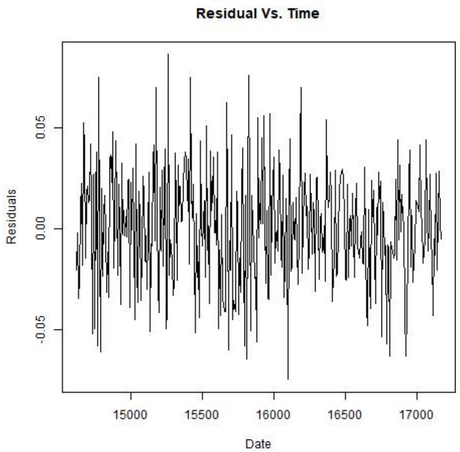
2. Modeling Results Summary

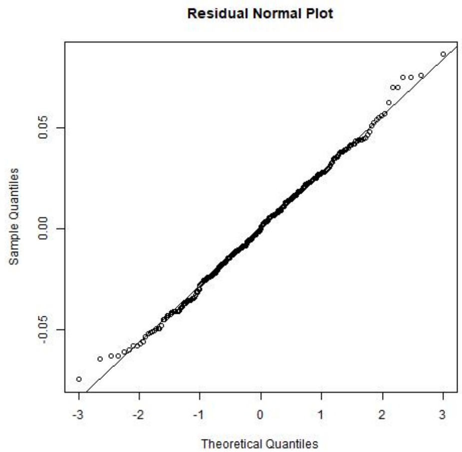
The table below contains the summaries for all models attempted (test results, back testing results, etc). A lag value of 5 was used for all of the augmented Dickey Fuller tests.

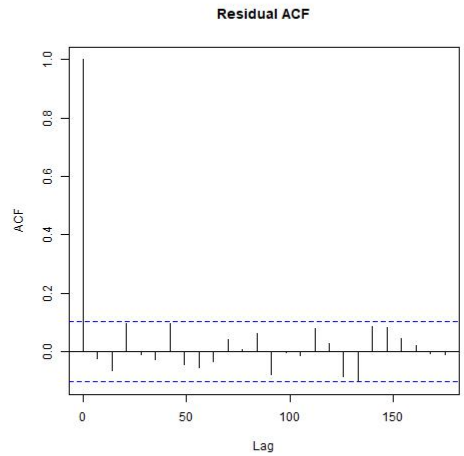
Model Name	Model Order	All Coefs Sig?	Model BIC	Residual JB Normal Test	Residual Ljung Box 5 Lag Test P Val	Residual Ljung Box 10 Lag Test P Val	Residual Ljung Box 25 Lag Test P Val	Residual No Interc No Trend DF Test P Value	Residual Const Interc No Trend DF Test P Value	Residual Const Interc W Trend DF Test P Value	Model Back Testing MAPE	Residual QQ Plot Tail Behavior	X Regressors
Apple_SimpleRtn_All_ARIMA_0_0_1_X_Regs	1	No	-1624.05	0.1802	0.2268	0.00038	0.00175	0.01	0.01	0.01	1.10053	Thick	CSCO, GOOG, GOOGL, ORCL, RHT, TXN, YHOO
Apple_SimpleRtn_ARIMA_0_0_1	1	Yes	-1541.1	0.8009	0.2332	0.2131	0.1382	0.01	0.01	0.01	1.22655	None	None
Apple_SimpleRtn_ARIMA_1_0_0	1	Yes	-1537.3	0.8532	0.06947	0.0827	0.04285	0.01	0.01	0.01	1.124128	None	None
Apple_SimpleRtn_ARIMA_1_0_1	2	Yes	-1538.4	0.7367	0.6798	0.3961	0.1991	0.01	0.01	0.01	1.205954	None	None
Apple_SimpleRtn_No_GOOGL_ARIMA_0_0_1_X_Regs	1	No	-1630	0.1805	0.2265	0.00038	0.00175	0.01	0.01	0.01	1.093953	Thick	CSCO, GOOG, ORCL, RHT, TXN, YHOO
Apple_SimpleRtn_No_GOOGL_RHT_ARIMA_0_0_1_X_Regs	1	No	-1635.8	0.1803	0.2315	0.00038	0.00158	0.01	0.01	0.01	1.092434	Thick	CSCO, GOOG, ORCL, TXN, YHOO
Apple_SimpleRtn_No_GOOGL_RHT_YHOO_ARIMA_0_0_1_X_Regs	1	No	-1641.2	0.1304	0.2472	0.00032	0.00107	0.01	0.01	0.01	1.056037	Thick	CSCO, GOOG, ORCL, TXN
Apple_SimpleRtn_W_GOOG_ARIMA_0_0_1_X_Regs	1	Yes	-1611.3	0.6467	0.0697	0.00131	0.00157	0.01	0.01	0.01	1.013589	Thick	GOOG
Apple_SimpleRtn_W_GOOG_ORCL_ARIMA_0_0_1_X_Regs	1	Yes	-1635.9	0.3465	0.2372	0.00123	0.00328	0.01	0.01	0.01	1.154316	Thickish	GOOG, ORCL
Apple_SimpleRtn_W_GOOG_ORCL_TXN_ARIMA_0_0_1_X_Regs	1	Yes	-1644.3	0.167	0.271	0.00055	0.00268	0.01	0.01	0.01	1.074363	Thick	GOOG, ORCL, TXN
Apple_SimpleRtn_W_GOOG_TXN_ARIMA_0_0_1_X_Regs	1	Yes	-1638.2	0.1603	0.1435	0.00033	0.00192	0.01	0.01	0.01	0.985778	Thick	GOOG, TXN
Apple_SimpleRtn_W_ORCL_ARIMA_0_0_1_X_Regs	1	Yes	-1603.2	0.5477	0.5438	0.03175	0.04749	0.01	0.01	0.01	1.295805	Thin	ORCL
Apple_SimpleRtn_W_TXN_ARIMA_0_0_1_X_Regs	1	Yes	-1608.8	0.3794	0.2543	0.00438	0.01671	0.01	0.01	0.01	1.098201	Thickish	TXN
Apple_SimpleRtn_W_TXN_ORCL_ARIMA_0_0_1_X_Regs	1	Yes	-1624.3	0.3557	0.5025	0.00522	0.01566	0.01	0.01	0.01	1.158269	Thickish	ORCL, TXN

3. MA(1) Model Residual Plots

The time plot shows the residuals are stationary with zero mean and constant variance. The normal probability plot shows normal behavior with only a few points at the ends that stray form the normal line. The ACF of the residuals show no autocorrelations.

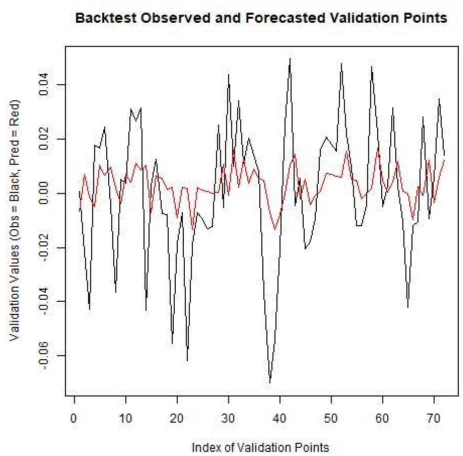


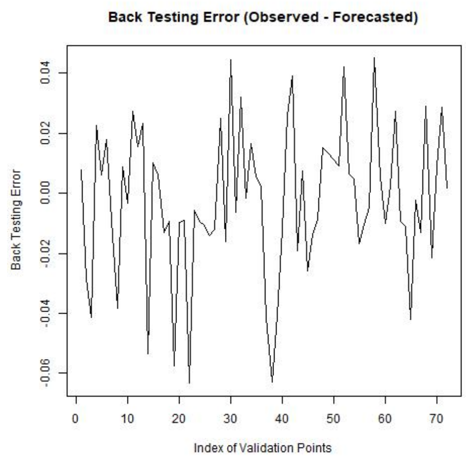




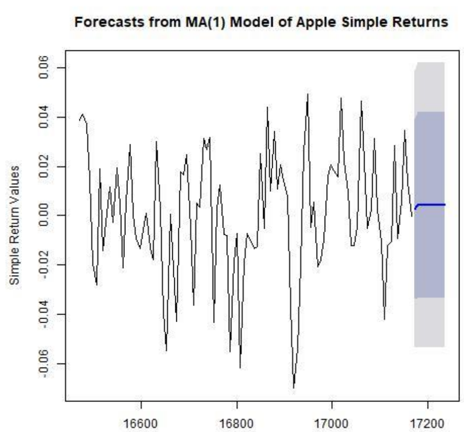
4. MA(1) Model Back Testing and Forecasting Plots

The back-testing plot of observed (black) and predicted points (red) shows that there was less than approximately -0.06 difference between them. The plot of the back-testing error (observed – predicted) confirms that the maximum error is -0.06.





The plot of the 10 predicted horizon points and last 100 observed points shows that the one-step ahead forecast follows the expected pattern and that the rest of the predicted points converge to the mean as expected.



III. Code

This section contains the R code I wrote for exploratory and modeling analysis. I have also included the code in the CSC 425 Project Code.zip file in the project submission folder in D2L.

Note that for back-testing, I used the backtesting.R function provided in lecture 4. I only replaced the forecast.Arima function calls with forecast since I was using a later version of R.

1. Exploratory Main Function (CSC425_Project_ExploratoryCode.R)

```
library(zoo)
library(tseries)
library(fBasics)
library(fUnitRoots)

# Get the path of the current based on if RStudio is run or if run from command line
cmdArgs <- commandArgs(trailingOnly = FALSE)

# If run from RStudio, use the rstudio api to get the path of the current R script
mainPath = dirname(rstudioapi::getSourceEditorContext()$path)
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")

source(paste(mainPath, "ExploreTimeData.R", sep=""))
source(paste(mainPath, "CrossCorrTimeData.R", sep=""))

dir.create(file.path(mainPath, "R_Output"), showWarnings = FALSE)
mainAnlysPath = paste(mainPath, "R_Output\\", sep="")

dir.create(file.path(mainAnlysPath, "Exploratory"), showWarnings = FALSE)
mainAnlysPath = paste(mainAnlysPath, "Exploratory\\", sep="")

# Read in NYSE data
nyse_all = read.table('C:\\DePaulCoursework\\Winter 2018 CSC 425\\Project\\Dataset\\prices-split-adjusted.csv', header = T, sep = ',')
nyse_all$date = as.character(nyse_all$date)
nyse_all$symbol = as.character(nyse_all$symbol)

# Stock symbols of interest
stock_syms = c('AAPL', 'GOOGL', 'GOOG', 'CSCO', 'HPQ', 'INTC', 'MSFT', 'NVDA', 'ORCL', 'RHT', 'TXN', 'WDC', 'XRX', 'YHOO')

num_stocks = length(stock_syms)
for (i in 1:num_stocks){

  tmp_symbol = stock_syms[i]
  tmp_ndx = nyse_all$symbol == tmp_symbol
  tmp_data = nyse_all[tmp_ndx,]
  tmp_dates = as.Date(tmp_data$date)

  num_rows = nrow(tmp_data)
  tmp_short_dates = as.Date(tmp_data$date[2:num_rows])


###############################################################################################################################
###
  ######################################################### Daily Data
########################################################

###############################################################################################################################
###
```

```
dir.create(file.path(mainAnlysPath, "Daily"), showWarnings = FALSE)
anlysPath = paste(mainAnlysPath, "Daily\\", sep="")

############################################ Time Series Exploratory Data ##############################################

print(paste(tmp_symbol, 'TS Daily Calculations'))

tmp_data_ts = zoo(tmp_data$close, tmp_dates)

data_desc = "Time Series"
dir.create(file.path(anlysPath, "TS"), showWarnings = FALSE)
ts_path = paste(anlysPath, "TS\\", sep="")

num_lags = floor(length(tmp_data_ts)/3)

ExploreTimeData(timeData = tmp_data_ts, dataSymbol = tmp_symbol, filePath = ts_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(ts_path, "Data"), showWarnings = FALSE)
ts_csv_path = paste(ts_path, "Data\\", sep="")

ts_df = data.frame(tmp_data_ts)
ts_df$date = rownames(ts_df)
rownames(ts_df) <- 1:nrow(ts_df)

full_ts_csv_path = paste(ts_csv_path, tmp_symbol, '_TS_Data.csv', sep = '')
write.csv(ts_df, full_ts_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = ts_path, dataPath = ts_csv_path, dataSuffix = '_TS_Data.csv')
}


######################################## Log Time Series Exploratory Data ##############################################

print(paste(tmp_symbol, 'Log TS Daily Calculations'))

log_data = log(tmp_data$close)
log_data_ts = zoo(log_data, tmp_dates)

data_desc = "Log Time Series"
dir.create(file.path(anlysPath, "Log_TS"), showWarnings = FALSE)
log_path = paste(anlysPath, "Log_TS\\", sep="")

num_lags = floor(length(log_data_ts)/3)

ExploreTimeData(timeData = log_data_ts, dataSymbol = tmp_symbol, filePath = log_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(log_path, "Data"), showWarnings = FALSE)
log_csv_path = paste(log_path, "Data\\", sep="")

log_df = data.frame(log_data_ts)
log_df$date = rownames(log_df)
rownames(log_df) <- 1:nrow(log_df)

full_log_csv_path = paste(log_csv_path, tmp_symbol, '_Log_Data.csv', sep = '')
write.csv(log_df, full_log_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = log_path, dataPath = log_csv_path, dataSuffix = '_Log_Data.csv')
}

######################################## Diff Time Series Exploratory Data #############################################

print(paste(tmp_symbol, 'Diff TS Daily Calculations'))
```

```
diff_data = diff(tmp_data$close)
diff_data_ts = zoo(diff_data, tmp_short_dates)

data_desc = "Diff Time Series"
dir.create(file.path(anlysPath, "Diff_TS"), showWarnings = FALSE)
diff_path = paste(anlysPath, "Diff_TS\\", sep="")

num_lags = floor(length(diff_data_ts)/3)

ExploreTimeData(timeData = diff_data_ts, dataSymbol = tmp_symbol, filePath = diff_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_path, "Data"), showWarnings = FALSE)
diff_csv_path = paste(diff_path, "Data\\", sep="")

diff_df = data.frame(diff_data_ts)
diff_df$date = rownames(diff_df)
rownames(diff_df) <- 1:nrow(diff_df)

full_diff_csv_path = paste(diff_csv_path, tmp_symbol, '_Diff_Data.csv', sep = '')
write.csv(diff_df, full_diff_csv_path)

if (i == num_stocks){
 CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_path, dataPath = diff_csv_path, dataSuffix = '_Diff_Data.csv')
}

##################################### Log Diff Time Series Exploratory Data ##########################################

print(paste(tmp_symbol, 'Diff Log TS Daily Calculations'))

diff_log_data = diff(log(tmp_data$close))
diff_log_data_ts = zoo(diff_log_data, tmp_short_dates)

data_desc = "Diff Log Time Series"
dir.create(file.path(anlysPath, "Diff_Log_TS"), showWarnings = FALSE)
diff_log_path = paste(anlysPath, "Diff_Log_TS\\", sep="")

num_lags = floor(length(diff_log_data_ts)/3)

ExploreTimeData(timeData = diff_log_data_ts, dataSymbol = tmp_symbol, filePath = diff_log_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_log_path, "Data"), showWarnings = FALSE)
log_diff_csv_path = paste(diff_log_path, "Data\\", sep="")

log_diff_df = data.frame(diff_log_data_ts)
log_diff_df$date = rownames(log_diff_df)
rownames(log_diff_df) <- 1:nrow(log_diff_df)

full_log_diff_csv_path = paste(log_diff_csv_path, tmp_symbol, '_LogDiff_Data.csv', sep = '')
write.csv(log_diff_df, full_log_diff_csv_path)

if (i == num_stocks){
 CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_log_path, dataPath = log_diff_csv_path, dataSuffix = '_LogDiff_Data.csv')
}

############################################### Return Data Creation ##################################################

tmp_gross_rtn = rep(1, num_rows)
tmp_simple_rtn = rep(1, num_rows)

for (j in 1:num_rows){

 if (j == 1){
  tmp_gross_rtn[j] = NA
  tmp_simple_rtn[j] = NA
 }
 else{
```

```
    tmp_gross_rtn[j] = tmp_data$close[j] / tmp_data$close[(j-1)]
    tmp_simple_rtn[j] = (tmp_data$close[j] / tmp_data$close[(j-1)]) - 1
  }
}

tmp_data$gross_return = tmp_gross_rtn
tmp_data$simple_return = tmp_simple_rtn

gross_rtn = tmp_gross_rtn[2:num_rows]
simple_rtn = tmp_simple_rtn[2:num_rows]

tmp_gross_rtn_ts = zoo(gross_rtn, tmp_short_dates)
tmp_simple_rtn_ts = zoo(simple_rtn, tmp_short_dates)

############################################ Gross Return Exploratory Data ###############################################

print(paste(tmp_symbol, 'Gross Return TS Daily Calculations'))

data_desc = "Gross Return"
dir.create(file.path(anlysPath, "Gross_Return"), showWarnings = FALSE)
gross_path = paste(anlysPath, "Gross_Return\\", sep="")

num_lags = floor(length(tmp_gross_rtn_ts)/3)

ExploreTimeData(timeData = tmp_gross_rtn_ts, dataSymbol = tmp_symbol, filePath = gross_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(gross_path, "Data"), showWarnings = FALSE)
gross_csv_path = paste(gross_path, "Data\\", sep="")

gross_df = data.frame(tmp_gross_rtn_ts)
gross_df$date = rownames(gross_df)
rownames(gross_df) <- 1:nrow(gross_df)

full_gross_csv_path = paste(gross_csv_path, tmp_symbol, '_GrossRtn_Data.csv', sep = '')
write.csv(gross_df, full_gross_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = gross_path, dataPath = gross_csv_path, dataSuffix = '_GrossRtn_Data.csv')
}

############################################ Simple Return Exploratory Data ###############################################

print(paste(tmp_symbol, 'Simple Return TS Daily Calculations'))

data_desc = "Simple Return"
dir.create(file.path(anlysPath, "Simple_Return"), showWarnings = FALSE)
simple_path = paste(anlysPath, "Simple_Return\\", sep="")

num_lags = floor(length(tmp_simple_rtn_ts)/3)

ExploreTimeData(timeData = tmp_simple_rtn_ts, dataSymbol = tmp_symbol, filePath = simple_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(simple_path, "Data"), showWarnings = FALSE)
simple_csv_path = paste(simple_path, "Data\\", sep="")

simple_df = data.frame(tmp_simple_rtn_ts)
simple_df$date = rownames(simple_df)
rownames(simple_df) <- 1:nrow(simple_df)

full_simple_csv_path = paste(simple_csv_path, tmp_symbol, '_SimpleRtn_Data.csv', sep = '')
write.csv(simple_df, full_simple_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = simple_path, dataPath = simple_csv_path, dataSuffix = '_SimpleRtn_Data.csv')
}
```

```
###########################################################################################################################
###
 ####################################################### Monthly Data
#########################################################

###########################################################################################################################
###

 dir.create(file.path(mainAnlysPath, "Monthly"), showWarnings = FALSE)
 anlysPath = paste(mainAnlysPath, "Monthly\\", sep="")

 ############################################ Time Series Exploratory Data ##################################################

 print(paste(tmp_symbol, 'TS Monthly Calculations'))

 tmp_data_ts_mon = aggregate(tmp_data_ts, as.yearmon, mean)

 data_desc = "Time Series"
 dir.create(file.path(anlysPath, "TS"), showWarnings = FALSE)
 ts_path = paste(anlysPath, "TS\\", sep="")

 num_lags = floor(length(tmp_data_ts_mon)/3)

 ExploreTimeData(timeData = tmp_data_ts_mon, dataSymbol = tmp_symbol, filePath = ts_path, plotDesc = data_desc, numLags = 5)

 dir.create(file.path(ts_path, "Data"), showWarnings = FALSE)
 ts_csv_path = paste(ts_path, "Data\\", sep="")

 ts_df = data.frame(tmp_data_ts_mon)
 ts_df$date = rownames(ts_df)
 rownames(ts_df) <- 1:nrow(ts_df)

 full_ts_csv_path = paste(ts_csv_path, tmp_symbol, '_TS_Data.csv', sep = '')
 write.csv(ts_df, full_ts_csv_path)

 if (i == num_stocks){
   CrossCorrTimeData(allSymbols = stock_syms, plotPath = ts_path, dataPath = ts_csv_path, dataSuffix = '_TS_Data.csv')
 }


 ######################################### Log Time Series Exploratory Data #################################################

 print(paste(tmp_symbol, 'Log TS Monthly Calculations'))

 log_data_ts_mon = aggregate(log_data_ts, as.yearmon, mean)

 data_desc = "Log Time Series"
 dir.create(file.path(anlysPath, "Log_TS"), showWarnings = FALSE)
 log_path = paste(anlysPath, "Log_TS\\", sep="")

 num_lags = floor(length(log_data_ts_mon)/3)

 ExploreTimeData(timeData = log_data_ts_mon, dataSymbol = tmp_symbol, filePath = log_path, plotDesc = data_desc, numLags = 5)

 dir.create(file.path(log_path, "Data"), showWarnings = FALSE)
 log_csv_path = paste(log_path, "Data\\", sep="")

 log_df = data.frame(log_data_ts_mon)
 log_df$date = rownames(log_df)
 rownames(log_df) <- 1:nrow(log_df)

 full_log_csv_path = paste(log_csv_path, tmp_symbol, '_Log_Data.csv', sep = '')
 write.csv(log_df, full_log_csv_path)
```

```
if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = log_path, dataPath = log_csv_path, dataSuffix = '_Log_Data.csv')
}

######################################## Diff Time Series Exploratory Data ##############################################

print(paste(tmp_symbol, 'Diff TS Monthly Calculations'))

diff_data_ts_mon = aggregate(diff_data_ts, as.yearmon, mean)

data_desc = "Diff Time Series"
dir.create(file.path(anlysPath, "Diff_TS"), showWarnings = FALSE)
diff_path = paste(anlysPath, "Diff_TS\\", sep="")

num_lags = floor(length(diff_data_ts_mon)/3)

ExploreTimeData(timeData = diff_data_ts_mon, dataSymbol = tmp_symbol, filePath = diff_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_path, "Data"), showWarnings = FALSE)
diff_csv_path = paste(diff_path, "Data\\", sep="")

diff_df = data.frame(diff_data_ts_mon)
diff_df$date = rownames(diff_df)
rownames(diff_df) <- 1:nrow(diff_df)

full_diff_csv_path = paste(diff_csv_path, tmp_symbol, '_Diff_Data.csv', sep = '')
write.csv(diff_df, full_diff_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_path, dataPath = diff_csv_path, dataSuffix = '_Diff_Data.csv')
}

######################################## Log Diff Time Series Exploratory Data ##########################################

print(paste(tmp_symbol, 'Diff Log TS Monthly Calculations'))

diff_log_data_mon = aggregate(diff_log_data_ts, as.yearmon, mean)

data_desc = "Diff Log Time Series"
dir.create(file.path(anlysPath, "Diff_Log_TS"), showWarnings = FALSE)
diff_log_path = paste(anlysPath, "Diff_Log_TS\\", sep="")

num_lags = floor(length(diff_log_data_mon)/3)

ExploreTimeData(timeData = diff_log_data_mon, dataSymbol = tmp_symbol, filePath = diff_log_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_log_path, "Data"), showWarnings = FALSE)
log_diff_csv_path = paste(diff_log_path, "Data\\", sep="")

log_diff_df = data.frame(diff_log_data_mon)
log_diff_df$date = rownames(log_diff_df)
rownames(log_diff_df) <- 1:nrow(log_diff_df)

full_log_diff_csv_path = paste(log_diff_csv_path, tmp_symbol, '_LogDiff_Data.csv', sep = '')
write.csv(log_diff_df, full_log_diff_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_log_path, dataPath = log_diff_csv_path, dataSuffix = '_LogDiff_Data.csv')
}

########################################### Gross Return Exploratory Data ###############################################

print(paste(tmp_symbol, 'Gross Return TS Monthly Calculations'))

tmp_gross_rtn_ts_mon = aggregate(tmp_gross_rtn_ts, as.yearmon, mean)
```

```
data_desc = "Gross Return"
dir.create(file.path(anlysPath, "Gross_Return"), showWarnings = FALSE)
gross_path = paste(anlysPath, "Gross_Return\\", sep="")

num_lags = floor(length(tmp_gross_rtn_ts_mon)/3)

ExploreTimeData(timeData = tmp_gross_rtn_ts_mon, dataSymbol = tmp_symbol, filePath = gross_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(gross_path, "Data"), showWarnings = FALSE)
gross_csv_path = paste(gross_path, "Data\\", sep="")

gross_df = data.frame(tmp_gross_rtn_ts_mon)
gross_df$date = rownames(gross_df)
rownames(gross_df) <- 1:nrow(gross_df)

full_gross_csv_path = paste(gross_csv_path, tmp_symbol, '_GrossRtn_Data.csv', sep = '')
write.csv(gross_df, full_gross_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = gross_path, dataPath = gross_csv_path, dataSuffix = '_GrossRtn_Data.csv')
}

########################################## Simple Return Exploratory Data ################################################

print(paste(tmp_symbol, 'Simple Return TS Monthly Calculations'))

tmp_simple_rtn_ts_mon = aggregate(tmp_simple_rtn_ts, as.yearmon, mean)

data_desc = "Simple Return"
dir.create(file.path(anlysPath, "Simple_Return"), showWarnings = FALSE)
simple_path = paste(anlysPath, "Simple_Return\\", sep="")

num_lags = floor(length(tmp_simple_rtn_ts_mon)/3)

ExploreTimeData(timeData = tmp_simple_rtn_ts_mon, dataSymbol = tmp_symbol, filePath = simple_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(simple_path, "Data"), showWarnings = FALSE)
simple_csv_path = paste(simple_path, "Data\\", sep="")

simple_df = data.frame(tmp_simple_rtn_ts_mon)
simple_df$date = rownames(simple_df)
rownames(simple_df) <- 1:nrow(simple_df)

full_simple_csv_path = paste(simple_csv_path, tmp_symbol, '_SimpleRtn_Data.csv', sep = '')
write.csv(simple_df, full_simple_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = simple_path, dataPath = simple_csv_path, dataSuffix = '_SimpleRtn_Data.csv')
}


########################################################################################################################
###
 ###################################################### Weekly Data
#######################################################

########################################################################################################################
###

 ################################################# Aggregate Weekly Data
##################################################

 diff_dates = diff(tmp_dates)

 entry_ndx = 1
 day_cnt = 0
```

```
 week_sum = 0
 week_avg = c()
 wk_num_days = c()
 week_data = c()
 week_dates = c()
 num_pts = length(diff_dates)
 curr_date = c()
 for (x in 1:num_pts){

  week_sum = week_sum + tmp_data$close[x]
  day_cnt = day_cnt + 1

   if (diff_dates[x] > 2){

   week_data[entry_ndx] = week_sum
   wk_num_days[entry_ndx] = day_cnt
   week_avg[entry_ndx] = week_sum / day_cnt

   if (entry_ndx == 1){
    curr_date = as.Date(tmp_data$date[x])
    week_dates[entry_ndx] = as.character(curr_date)
   }else{
    curr_date = curr_date + 7
    week_dates[entry_ndx] = as.character(curr_date)
   }

   entry_ndx = entry_ndx + 1
   week_sum = 0
   day_cnt = 0


  } else if (x == num_pts){

   week_sum = week_sum + tmp_data$close[x + 1]
   day_cnt = day_cnt + 1

   week_data[entry_ndx] = week_sum
#     week_dates[entry_ndx] = tmp_data$date[i + 1]
   wk_num_days[entry_ndx] = day_cnt
   week_avg[entry_ndx] = week_sum / day_cnt

   curr_date = curr_date + 7
   week_dates[entry_ndx] = as.character(curr_date)


  }

 }

 tmp_wk_dates = as.Date(week_dates)

 num_wk_rows = length(week_avg)
 tmp_short_wk_dates = as.Date(week_dates[2:num_wk_rows])


##########################################################################################################################
###

 dir.create(file.path(mainAnlysPath, "Weekly"), showWarnings = FALSE)
 anlysPath = paste(mainAnlysPath, "Weekly\\", sep="")

 ########################################### Time Series Exploratory Data ###############################################

 print(paste(tmp_symbol, 'TS Weekly Calculations'))

 wk_tmp_data_ts = zoo(week_avg, tmp_wk_dates)
```

```
data_desc = "Time Series"
dir.create(file.path(anlysPath, "TS"), showWarnings = FALSE)
ts_path = paste(anlysPath, "TS\\", sep="")

num_lags = floor(length(wk_tmp_data_ts)/3)

ExploreTimeData(timeData = wk_tmp_data_ts, dataSymbol = tmp_symbol, filePath = ts_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(ts_path, "Data"), showWarnings = FALSE)
ts_csv_path = paste(ts_path, "Data\\", sep="")

ts_df = data.frame(wk_tmp_data_ts)
ts_df$date = rownames(ts_df)
rownames(ts_df) <- 1:nrow(ts_df)

full_ts_csv_path = paste(ts_csv_path, tmp_symbol, '_TS_Data.csv', sep = '')
write.csv(ts_df, full_ts_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = ts_path, dataPath = ts_csv_path, dataSuffix = '_TS_Data.csv')
}


######################################## Log Time Series Exploratory Data ################################################

print(paste(tmp_symbol, 'Log TS Weekly Calculations'))

wk_log_data = log(week_avg)
wk_log_data_ts = zoo(wk_log_data, tmp_wk_dates)

data_desc = "Log Time Series"
dir.create(file.path(anlysPath, "Log_TS"), showWarnings = FALSE)
log_path = paste(anlysPath, "Log_TS\\", sep="")

num_lags = floor(length(wk_log_data_ts)/3)

ExploreTimeData(timeData = wk_log_data_ts, dataSymbol = tmp_symbol, filePath = log_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(log_path, "Data"), showWarnings = FALSE)
log_csv_path = paste(log_path, "Data\\", sep="")

log_df = data.frame(wk_log_data_ts)
log_df$date = rownames(log_df)
rownames(log_df) <- 1:nrow(log_df)

full_log_csv_path = paste(log_csv_path, tmp_symbol, '_Log_Data.csv', sep = '')
write.csv(log_df, full_log_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = log_path, dataPath = log_csv_path, dataSuffix = '_Log_Data.csv')
}

######################################## Diff Time Series Exploratory Data ###############################################

print(paste(tmp_symbol, 'Diff TS Weekly Calculations'))

wk_diff_data = diff(week_avg)
wk_diff_data_ts = zoo(wk_diff_data, tmp_short_wk_dates)

data_desc = "Diff Time Series"
dir.create(file.path(anlysPath, "Diff_TS"), showWarnings = FALSE)
diff_path = paste(anlysPath, "Diff_TS\\", sep="")

num_lags = floor(length(wk_diff_data_ts)/3)
```

```
ExploreTimeData(timeData = wk_diff_data_ts, dataSymbol = tmp_symbol, filePath = diff_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_path, "Data"), showWarnings = FALSE)
diff_csv_path = paste(diff_path, "Data\\", sep="")

diff_df = data.frame(wk_diff_data_ts)
diff_df$date = rownames(diff_df)
rownames(diff_df) <- 1:nrow(diff_df)

full_diff_csv_path = paste(diff_csv_path, tmp_symbol, '_Diff_Data.csv', sep = '')
write.csv(diff_df, full_diff_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_path, dataPath = diff_csv_path, dataSuffix = '_Diff_Data.csv')
}

######################################## Log Diff Time Series Exploratory Data ############################################

print(paste(tmp_symbol, 'Diff Log TS Weekly Calculations'))

wk_diff_log_data = diff(log(week_avg))
wk_diff_log_data_ts = zoo(wk_diff_log_data, tmp_short_wk_dates)

data_desc = "Diff Log Time Series"
dir.create(file.path(anlysPath, "Diff_Log_TS"), showWarnings = FALSE)
diff_log_path = paste(anlysPath, "Diff_Log_TS\\", sep="")

num_lags = floor(length(wk_diff_log_data_ts)/3)

ExploreTimeData(timeData = wk_diff_log_data_ts, dataSymbol = tmp_symbol, filePath = diff_log_path, plotDesc = data_desc, numLags = 5)

dir.create(file.path(diff_log_path, "Data"), showWarnings = FALSE)
log_diff_csv_path = paste(diff_log_path, "Data\\", sep="")

log_diff_df = data.frame(wk_diff_log_data_ts)
log_diff_df$date = rownames(log_diff_df)
rownames(log_diff_df) <- 1:nrow(log_diff_df)

full_log_diff_csv_path = paste(log_diff_csv_path, tmp_symbol, '_LogDiff_Data.csv', sep = '')
write.csv(log_diff_df, full_log_diff_csv_path)

if (i == num_stocks){
  CrossCorrTimeData(allSymbols = stock_syms, plotPath = diff_log_path, dataPath = log_diff_csv_path, dataSuffix = '_LogDiff_Data.csv')
}

################################################ Return Data Creation ###################################################

tmp_gross_rtn = rep(1, num_wk_rows)
tmp_simple_rtn = rep(1, num_wk_rows)

for (j in 1:num_rows){

  if (j == 1){
    tmp_gross_rtn[j] = NA
    tmp_simple_rtn[j] = NA
  }
  else{
    tmp_gross_rtn[j] = week_avg[j] / week_avg[(j-1)]
    tmp_simple_rtn[j] = (week_avg[j] / week_avg[(j-1)]) - 1
  }
}

wk_gross_rtn = tmp_gross_rtn[2:num_rows]
wk_simple_rtn = tmp_simple_rtn[2:num_rows]

tmp_wk_gross_rtn_ts = zoo(wk_gross_rtn, tmp_short_wk_dates)
```

```
  tmp_wk_simple_rtn_ts = zoo(wk_simple_rtn, tmp_short_wk_dates)

  ########################################### Gross Return Exploratory Data ##############################################

  print(paste(tmp_symbol, 'Gross Return TS Weekly Calculations'))

  data_desc = "Gross Return"
  dir.create(file.path(anlysPath, "Gross_Return"), showWarnings = FALSE)
  gross_path = paste(anlysPath, "Gross_Return\\", sep="")

  num_lags = floor(length(tmp_wk_gross_rtn_ts)/3)

  ExploreTimeData(timeData = tmp_wk_gross_rtn_ts, dataSymbol = tmp_symbol, filePath = gross_path, plotDesc = data_desc, numLags = 5)

  dir.create(file.path(gross_path, "Data"), showWarnings = FALSE)
  gross_csv_path = paste(gross_path, "Data\\", sep="")

  gross_df = data.frame(tmp_wk_gross_rtn_ts)
  gross_df$date = rownames(gross_df)
  rownames(gross_df) <- 1:nrow(gross_df)

  full_gross_csv_path = paste(gross_csv_path, tmp_symbol, '_GrossRtn_Data.csv', sep = '')
  write.csv(gross_df, full_gross_csv_path)

  if (i == num_stocks){
    CrossCorrTimeData(allSymbols = stock_syms, plotPath = gross_path, dataPath = gross_csv_path, dataSuffix = '_GrossRtn_Data.csv')
  }

  ########################################### Simple Return Exploratory Data ##############################################

  print(paste(tmp_symbol, 'Simple Return TS Weekly Calculations'))

  data_desc = "Simple Return"
  dir.create(file.path(anlysPath, "Simple_Return"), showWarnings = FALSE)
  simple_path = paste(anlysPath, "Simple_Return\\", sep="")

  num_lags = floor(length(tmp_wk_simple_rtn_ts)/3)

  ExploreTimeData(timeData = tmp_wk_simple_rtn_ts, dataSymbol = tmp_symbol, filePath = simple_path, plotDesc = data_desc, numLags = 5)

  dir.create(file.path(simple_path, "Data"), showWarnings = FALSE)
  simple_csv_path = paste(simple_path, "Data\\", sep="")

  simple_df = data.frame(tmp_wk_simple_rtn_ts)
  simple_df$date = rownames(simple_df)
  rownames(simple_df) <- 1:nrow(simple_df)

  full_simple_csv_path = paste(simple_csv_path, tmp_symbol, '_SimpleRtn_Data.csv', sep = '')
  write.csv(simple_df, full_simple_csv_path)

  if (i == num_stocks){
    CrossCorrTimeData(allSymbols = stock_syms, plotPath = simple_path, dataPath = simple_csv_path, dataSuffix = '_SimpleRtn_Data.csv')
  }

}
```

2. Modeling Main Function (CSC425_Project_ModelingCode.R)

```
library(zoo)
library(tseries)
library(fBasics)
library(fUnitRoots)
library(lmtest)
library(forecast)


# Get the path of the current based on if RStudio is run or if run from command line
cmdArgs <- commandArgs(trailingOnly = FALSE)

# If run from RStudio, use the rstudio api to get the path of the current R script
mainPath = dirname(rstudioapi::getSourceEditorContext()$path)
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")

source(paste(mainPath, "backtest.R", sep=""))
source(paste(mainPath, "ModelTimeData.R", sep=""))

dir.create(file.path(mainPath, "R_Output"), showWarnings = FALSE)
mainAnlysPath = paste(mainPath, "R_Output\\", sep="")

dir.create(file.path(mainAnlysPath, "Modeling"), showWarnings = FALSE)
mainAnlysPath = paste(mainAnlysPath, "Modeling\\", sep="")

# Read in Apple weekly simple return data
apple_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\AAPL_SimpleRtn_Data.csv',
                header = T, sep = ',')
apple_data$date = as.character(apple_data$date)
apple_data$date = as.Date(apple_data$date)

apple_ts = zoo(apple_data$tmp_wk_simple_rtn_ts, apple_data$date)

out_file_name = paste(mainAnlysPath, 'Apple_SimpleRtn_AutoArima.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)

# Basic Statistics
print('*** Auto Arima BIC Criterion Results ***')
print(auto.arima(apple_ts, ic = c("bic")))
print("", quote=FALSE)

sink()

close(outFile)
closeAllConnections()

model_bics = c()
model_names = c()

###################################################### MA (1) Model
##########################################################
# Model recommended by auto.arima

ma1_orders = c(0,0,1)
ma1_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), FALSE, NULL, mainAnlysPath, 'Apple_SimpleRtn')
model_bics[1] = ma1_model$bic
model_names[1] = 'Apple_SimpleRtn_MA1'

###################################################### ARMA (1) Model
########################################################
```

```
arma1_orders = c(1,0,1)
arma1_model = ModelTimeData(apple_ts, arma1_orders, 0, c(), FALSE, NULL, mainAnlysPath, 'Apple_SimpleRtn')
model_bics[2] = arma1_model$bic
model_names[2] = 'Apple_SimpleRtn_ARMA1'

################################################### AR (1) Model
#######################################################

ar1_orders = c(1,0,0)
ar1_model = ModelTimeData(apple_ts, ar1_orders, 0, c(), FALSE, NULL, mainAnlysPath, 'Apple_SimpleRtn')
model_bics[3] = ar1_model$bic
model_names[3] = 'Apple_SimpleRtn_AR1'

################################################### X Regressors
#######################################################

csco_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\CSCO_SimpleRtn_Data.csv',
                      header = T, sep = ',')
csco_data$date = as.character(csco_data$date)
csco_data$date = as.Date(csco_data$date)
csco_ts = zoo(csco_data$tmp_wk_simple_rtn_ts, csco_data$date)

goog_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\GOOG_SimpleRtn_Data.csv',
                     header = T, sep = ',')
goog_data$date = as.character(goog_data$date)
goog_data$date = as.Date(goog_data$date)
goog_ts = zoo(goog_data$tmp_wk_simple_rtn_ts, goog_data$date)

googl_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\GOOGL_SimpleRtn_Data.csv',
                     header = T, sep = ',')
googl_data$date = as.character(googl_data$date)
googl_data$date = as.Date(googl_data$date)
googl_ts = zoo(googl_data$tmp_wk_simple_rtn_ts, googl_data$date)

orcl_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\ORCL_SimpleRtn_Data.csv',
                      header = T, sep = ',')
orcl_data$date = as.character(orcl_data$date)
orcl_data$date = as.Date(orcl_data$date)
orcl_ts = zoo(orcl_data$tmp_wk_simple_rtn_ts, orcl_data$date)

rht_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\RHT_SimpleRtn_Data.csv',
                     header = T, sep = ',')
rht_data$date = as.character(rht_data$date)
rht_data$date = as.Date(rht_data$date)
rht_ts = zoo(rht_data$tmp_wk_simple_rtn_ts, rht_data$date)

txn_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\TXN_SimpleRtn_Data.csv',
                    header = T, sep = ',')
txn_data$date = as.character(txn_data$date)
txn_data$date = as.Date(txn_data$date)
txn_ts = zoo(txn_data$tmp_wk_simple_rtn_ts, txn_data$date)

yhoo_data = read.table('C:\\DePaulCoursework\\Winter 2018 CSC
425\\Project\\R_Output\\Exploratory\\Weekly\\Simple_Return\\Data\\YHOO_SimpleRtn_Data.csv',
                    header = T, sep = ',')
yhoo_data$date = as.character(yhoo_data$date)
yhoo_data$date = as.Date(yhoo_data$date)
yhoo_ts = zoo(yhoo_data$tmp_wk_simple_rtn_ts, yhoo_data$date)
```

```
tmp_pair = paste(mainAnlysPath, 'Apple_SimpleRtn_ScatterMatrix.jpg', sep = '')
jpeg(file = tmp_pair)
pairs(~apple_ts + csco_ts + goog_ts + googl_ts + orcl_ts + rht_ts + txn_ts + yhoo_ts)
dev.off()

################################################ MA (1) All X Regressors Model ###############################################

xreg_data = data.frame(csco_ts, goog_ts, googl_ts, orcl_ts, rht_ts, txn_ts, yhoo_ts)

ma1_orders = c(0,0,1)
ma1_all_xregs_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_All')
model_bics[4] = ma1_all_xregs_model$bic
model_names[4] = 'Apple_SimpleRtn_All'

##################################################### MA (1) No GOOGL Model
##################################################

xreg_data = data.frame(csco_ts, goog_ts, orcl_ts, rht_ts, txn_ts, yhoo_ts)

ma1_orders = c(0,0,1)
ma1_no_googl_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_No_GOOGL')
model_bics[5] = ma1_no_googl_model$bic
model_names[5] = 'Apple_SimpleRtn_No_GOOGL'

################################################## MA (1) No GOOGL, RHT Model
################################################

xreg_data = data.frame(csco_ts, goog_ts, orcl_ts, txn_ts, yhoo_ts)

ma1_orders = c(0,0,1)
ma1_no_googl_rht_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath,
'Apple_SimpleRtn_No_GOOGL_RHT')
model_bics[6] = ma1_no_googl_rht_model$bic
model_names[6] = 'Apple_SimpleRtn_No_GOOGL_RHT'

################################################ MA (1) No GOOGL, RHT, YHOO Model
############################################

xreg_data = data.frame(csco_ts, goog_ts, orcl_ts, txn_ts)

ma1_orders = c(0,0,1)
ma1_no_googl_rht_yhoo_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath,
'Apple_SimpleRtn_No_GOOGL_RHT_YHOO')
model_bics[7] = ma1_no_googl_rht_yhoo_model$bic
model_names[7] = 'Apple_SimpleRtn_No_GOOGL_RHT_YHOO'

################################################# MA (1) W GOOG, ORCL, TXN Model
#############################################

xreg_data = data.frame(goog_ts, orcl_ts, txn_ts)

ma1_orders = c(0,0,1)
ma1_w_goog_orcl_txn_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath,
'Apple_SimpleRtn_W_GOOG_ORCL_TXN')
model_bics[8] = ma1_w_goog_orcl_txn_model$bic
model_names[8] = 'Apple_SimpleRtn_W_GOOG_ORCL_TXN'

################################################### MA (1) W GOOG, TXN Model
#################################################

xreg_data = data.frame(goog_ts, txn_ts)

ma1_orders = c(0,0,1)
ma1_w_goog_txn_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_W_GOOG_TXN')
model_bics[9] = ma1_w_goog_txn_model$bic
model_names[9] = 'Apple_SimpleRtn_W_GOOG_TXN'
```

```
################################################ MA (1) W GOOG, ORCL Model
#############################################

xreg_data = data.frame(goog_ts, orcl_ts)

ma1_orders = c(0,0,1)
ma1_w_goog_orcl_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath,
'Apple_SimpleRtn_W_GOOG_ORCL')
model_bics[10] = ma1_w_goog_orcl_model$bic
model_names[10] = 'Apple_SimpleRtn_W_GOOG_ORCL'

################################################ MA (1) W TXN, ORCL Model
##############################################

xreg_data = data.frame(txn_ts, orcl_ts)

ma1_orders = c(0,0,1)
ma1_w_txn_orcl_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_W_TXN_ORCL')
model_bics[11] = ma1_w_txn_orcl_model$bic
model_names[11] = 'Apple_SimpleRtn_W_TXN_ORCL'

################################################ MA (1) W GOOG Model
###################################################

xreg_data = data.frame(goog_ts)

ma1_orders = c(0,0,1)
ma1_w_goog_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_W_GOOG')
model_bics[12] = ma1_w_goog_model$bic
model_names[12] = 'Apple_SimpleRtn_W_GOOG'

################################################ MA (1) W TXN Model
####################################################

xreg_data = data.frame(txn_ts)

ma1_orders = c(0,0,1)
ma1_w_txn_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_W_TXN')
model_bics[13] = ma1_w_txn_model$bic
model_names[13] = 'Apple_SimpleRtn_W_TXN'

################################################ MA (1) W ORCL Model
###################################################

xreg_data = data.frame(orcl_ts)

ma1_orders = c(0,0,1)
ma1_w_orcl_model = ModelTimeData(apple_ts, ma1_orders, 0, c(), TRUE, xreg_data, mainAnlysPath, 'Apple_SimpleRtn_W_ORCL')
model_bics[14] = ma1_w_orcl_model$bic
model_names[14] = 'Apple_SimpleRtn_W_ORCL'

############################################## Find the Best Model BIC #################################################

max_bic = min(model_bics)
min_ndx = model_bics == max_bic

out_file_name = paste(mainAnlysPath, 'Apple_SimpleRtn_Model_BIC_Summary.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)

print('Model With Lowest BIC:')
print(model_names[min_ndx])
print(paste('BIC = ', model_bics[min_ndx]))
print("", quote=FALSE)
```

```
print('All Model Information:')
bic_df = data.frame(model_names, model_bics)
print(bic_df)

sink()

close(outFile)
closeAllConnections()

################################ Forecast 10 Points into future with best (MA(1)) model ####################################

f_ma1=forecast(ma1_model, h=10)

out_file_name = paste(mainAnlysPath, 'Apple_SimpleRtn_MA1_Model_H10_Forecast.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)
print('Forecast of Apple Simple Return MA(1) Model 10 Points Into The Future:')
print(f_ma1)

sink()

close(outFile)
closeAllConnections()

tmp_ma1_bt_scatter = paste(mainAnlysPath, 'Apple_SimpleRtn_MA1_Model_H10_Forecast.jpg', sep = '')
jpeg(file = tmp_ma1_bt_scatter)
y_title = 'Simple Return Values'
plot_title = 'Forecasts from MA(1) Model of Apple Simple Returns'
plot(f_ma1, include=100, ylab = y_title, main = plot_title)
lines(c(f_ma1$fitted, f_ma1$mean), col="blue")

dev.off()

############################### Forecast 10 Points into future with best (ARMA(1)) model ###################################

f_arma1=forecast(arma1_model, h=10)

out_file_name = paste(mainAnlysPath, 'Apple_SimpleRtn_ARMA1_Model_H10_Forecast.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)
print('Forecast of Apple Simple Return ARMA(1) Model 10 Points Into The Future:')
print(f_arma1)

sink()

close(outFile)
closeAllConnections()

tmp_arma1_bt_scatter = paste(mainAnlysPath, 'Apple_SimpleRtn_ARMA1_Model_H10_Forecast.jpg', sep = '')
jpeg(file = tmp_arma1_bt_scatter)
y_title = 'Simple Return Values'
plot_title = 'Forecasts from ARMA(1) Model of Apple Simple Returns'
plot(f_arma1, include=100, ylab = y_title, main = plot_title)
lines(c(f_arma1$fitted, f_arma1$mean), col="blue")

dev.off()

############################## Forecast 10 Points into future with best (AR(1)) model #####################################

f_ar1=forecast(arma1_model, h=10)

out_file_name = paste(mainAnlysPath, 'Apple_SimpleRtn_AR1_Model_H10_Forecast.txt', sep = '')
outFile = file(out_file_name, open="wt")
```

```
sink(file = outFile, append = TRUE)
print('Forecast of Apple Simple Return AR(1) Model 10 Points Into The Future:')
print(f_ar1)

sink()

close(outFile)
closeAllConnections()

tmp_ar1_bt_scatter = paste(mainAnlysPath, 'Apple_SimpleRtn_AR1_Model_H10_Forecast.jpg', sep = '')
jpeg(file = tmp_ar1_bt_scatter)
y_title = 'Simple Return Values'
plot_title = 'Forecasts from AR(1) Model of Apple Simple Returns'
plot(f_ar1, include=100, ylab = y_title, main = plot_title)
lines(c(f_ar1$fitted, f_ar1$mean), col="blue")

dev.off()

################################################ Apple Squared ACF Plot
######################################################

apple_data$sq_wk_simple_rtn = (apple_data$tmp_wk_simple_rtn_ts)^2
apple_sq_ts = zoo(apple_data$sq_wk_simple_rtn, apple_data$date)

tmp_appl_sq_acf = paste(mainAnlysPath, 'AAPL_Squared_ACF.jpg', sep = '')
jpeg(file = tmp_appl_sq_acf)
plot_title = 'AAPL Simple Return Squared'
acf(apple_sq_ts, plot = TRUE, na.action = na.exclude, main = plot_title)
dev.off()

tmp_sq_scatter = paste(mainAnlysPath, 'AAPL_Squared_Scatter.jpg', sep = '')
jpeg(file = tmp_sq_scatter)
x_title = 'Date'
y_title = 'AAPL Simple Returns Squared'
plot_title = 'AAPL Simple Return Squared Vs. Time'
plot(apple_sq_ts, xlab = x_title, ylab = y_title, main = plot_title)
dev.off()

################################################ Apple Absolute ACF Plot
#####################################################

apple_data$abs_wk_simple_rtn = abs(apple_data$tmp_wk_simple_rtn_ts)
apple_abs_ts = zoo(apple_data$abs_wk_simple_rtn, apple_data$date)

tmp_appl_abs_acf = paste(mainAnlysPath, 'AAPL_Abs_ACF.jpg', sep = '')
jpeg(file = tmp_appl_abs_acf)
plot_title = 'AAPL Absolute Simple Return'
acf(apple_abs_ts, plot = TRUE, na.action = na.exclude, main = plot_title)
dev.off()

tmp_abs_scatter = paste(mainAnlysPath, 'AAPL_Abs_Scatter.jpg', sep = '')
jpeg(file = tmp_abs_scatter)
x_title = 'Date'
y_title = 'AAPL Absolute of Simple Returns'
plot_title = 'AAPL Absolute of Simple Return Vs. Time'
plot(apple_sq_ts, xlab = x_title, ylab = y_title, main = plot_title)
dev.off()

############################################## Apple ARCH Ljung Box Test
#####################################################

out_file_name = paste(mainAnlysPath, 'Apple_ARCH_LjungBox.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)
```

```
print('Ljung Box test for Squared Simple Returns for 5 Lags:')
print(Box.test(apple_sq_ts, lag = 5, type = 'Ljung'))
print("", quote=FALSE)

print('Ljung Box test for Absolute of Simple Returns for 5 Lags:')
print(Box.test(apple_abs_ts, lag = 5, type = 'Ljung'))
print("", quote=FALSE)
print("", quote=FALSE)

print('Ljung Box test for Squared Simple Returns for 10 Lags:')
print(Box.test(apple_sq_ts, lag = 10, type = 'Ljung'))
print("", quote=FALSE)

print('Ljung Box test for Absolute of Simple Returns for 10 Lags:')
print(Box.test(apple_abs_ts, lag = 10, type = 'Ljung'))
print("", quote=FALSE)
print("", quote=FALSE)

print('Ljung Box test for Squared Simple Returns for 25 Lags:')
print(Box.test(apple_sq_ts, lag = 25, type = 'Ljung'))
print("", quote=FALSE)

print('Ljung Box test for Absolute of Simple Returns for 25 Lags:')
print(Box.test(apple_abs_ts, lag = 25, type = 'Ljung'))
print("", quote=FALSE)

sink()

close(outFile)
closeAllConnections()
```

3. Exploratory ExploreTimeData.R Function

```
ExploreTimeData <- function(
  timeData = NULL,
  dataSymbol = NULL,
  filePath = NULL,
  plotDesc = NULL,
  numLags = 25
) {

  ############################################## Histograms #################################################

  dir.create(file.path(filePath, "Histograms"), showWarnings = FALSE)
  hist_path = paste(filePath, "Histograms\\", sep="")

  tmp_hist = paste(hist_path, dataSymbol, '_Hist.jpg', sep = '')
  jpeg(file = tmp_hist)
  x_title = paste(dataSymbol, plotDesc)
  plot_title = paste(dataSymbol, plotDesc, 'Histogram')
  hist(timeData, xlab = x_title, prob = TRUE, main = plot_title)
  # add approximating normal density curve
  xfit = seq(min(timeData), max(timeData), length=40)
  yfit = dnorm(xfit, mean = mean(timeData), sd = sd(timeData))
  lines(xfit, yfit, col = "blue", lwd = 2)
  dev.off()

  ############################################## Scatterplots ###############################################

  dir.create(file.path(filePath, "Scatterplots"), showWarnings = FALSE)
  scatter_path = paste(filePath, "Scatterplots\\", sep="")

  tmp_scatter = paste(scatter_path, dataSymbol, '_Scatter.jpg', sep = '')
  jpeg(file = tmp_scatter)
  x_title = 'Date'
  y_title = paste(dataSymbol, plotDesc)
  plot_title = paste(dataSymbol, plotDesc, 'Vs. Time')
  plot(timeData, xlab = x_title, ylab = y_title, main = plot_title)
  dev.off()

  ##############################################Normal Plots ################################################

  dir.create(file.path(filePath, "QQ"), showWarnings = FALSE)
  qq_path = paste(filePath, "QQ\\", sep="")

  tmp_qq = paste(qq_path, dataSymbol, '_QQ.jpg', sep = '')
  jpeg(file = tmp_qq)
  plot_title = paste(dataSymbol, plotDesc, 'Return')
  qqnorm(timeData, main = plot_title)
  qqline(timeData)
  dev.off()

  ############################################## ACF Plots ##################################################

  dir.create(file.path(filePath, "ACF"), showWarnings = FALSE)
  acf_path = paste(filePath, "ACF\\", sep="")

  tmp_acf = paste(acf_path, dataSymbol, '_ACF.jpg', sep = '')
  jpeg(file = tmp_acf)
  plot_title = paste(dataSymbol, plotDesc)
  acf(timeData, plot = TRUE, na.action = na.exclude, main = plot_title)
  dev.off()

  ############################################## PACF Plots #################################################

  dir.create(file.path(filePath, "PACF"), showWarnings = FALSE)
```

```
pacf_path = paste(filePath, "PACF\\", sep="")

tmp_pacf = paste(pacf_path, dataSymbol, '_PACF.jpg', sep = '')
jpeg(file = tmp_pacf)
plot_title = paste(dataSymbol, plotDesc)

pacf(timeData, plot = TRUE, na.action = na.exclude, main = plot_title)

dev.off()

############################################ Statistics ################################################

out_file_name = paste(filePath, dataSymbol, '_Statistics.txt', sep = '')
outFile = file(out_file_name, open="wt")

sink(file = outFile, append = TRUE)

# Basic Statistics
print('*** Basic Statistics ***')
print(basicStats(timeData))
print("", quote=FALSE)

# Perform Jarque-Bera normality test
print('*** Jarque Bera Normality Test ***')
print(normalTest(timeData, method=c("jb")))
print("", quote=FALSE)

# Fat-tail test
print('*** Kurtosis (Tail) Test ***')
k_stat = kurtosis(timeData) / sqrt(24/length(timeData))
print("Kurtosis Test Statistic:")
print(k_stat)
print("P-value:")
tmp_p = 2*(1 - pnorm(abs(k_stat)))
print(tmp_p)
print("", quote=FALSE)

#Skewness test
print('*** Skewness Test ***')
skew_test = skewness(timeData) / sqrt(6/length(timeData))
print("Skewness Test Statistic:")
print(skew_test)
print("P-value:")
tmp_p = 2* (1 - pnorm(abs(skew_test)))
print(tmp_p)
print("", quote=FALSE)

#Ljung Box test with 25 lags
print('*** Ljung Box Serial Correlation Test ***')
print(Box.test(timeData, lag = numLags, type = 'Ljung'))
print("", quote=FALSE)

#Augmented Dickey Fuller zero mean no trend
print('*** Augmented Dickey Fuller Test - No Intercept (Zero Mean), No Time Trend ***')
print(adfTest(timeData, lag = numLags, type = c('nc')))
print("", quote=FALSE)

#Augmented Dickey Fuller constant mean no trend
print('*** Augmented Dickey Fuller Test - Constant Intercept (Constant Mean), No Time Trend ***')
print(adfTest(timeData, lag = numLags, type = c('c')))
print("", quote=FALSE)

#Augmented Dickey Fuller constant mean no trend
print('*** Augmented Dickey Fuller Test - Constant Intercept (Constant Mean), Time Trend ***')
print(adfTest(timeData, lag = numLags, type = c('ct')))
print("", quote=FALSE)
```

```
  sink()

  close(outFile)
  closeAllConnections()

}
```

4. Exploratory CrossCorrTimeData.R Function

```
CrossCorrTimeData <- function(
  allSymbols = NULL,
  plotPath = NULL,
  dataPath = NULL,
  dataSuffix = NULL
) {

  ########################################## Cross Correlation Function Plotting ##########################################

  numVars = length(allSymbols)
  for (i in 1:numVars){

    var = allSymbols[i]
    otherVars = setdiff(allSymbols, var)
    numOtherVars = length(otherVars)

    var_path = paste(dataPath, var, dataSuffix, sep='')
    tmp_var = read.table(var_path, header = T, sep = ',')
    var_data = tmp_var[,2]

    var_dates = tryCatch(as.Date(as.character(tmp_var[,3])), error=function(e) as.Date(as.yearmon(tmp_var[,3])))

    var_ts = zoo(var_data, var_dates)

    if(numOtherVars > 0){
      dir.create(file.path(plotPath, "CCF"), showWarnings = FALSE)
      newPath = paste(plotPath, "CCF\\", sep="")

      for (j in 1:numOtherVars){
        otherVar = otherVars[j]
        otherNdx = match(otherVar, allSymbols)
        fileName = paste(newPath, "CCF_", var, "_Vs_", otherVar, ".jpeg", sep="")

        other_var_path = paste(dataPath, otherVar, dataSuffix, sep='')
        tmp_other_var = read.table(other_var_path, header = T, sep = ',')
        other_var_data = tmp_other_var[,2]

        other_var_dates = tryCatch(as.Date(as.character(tmp_other_var[,3])), error=function(e) other_var_dates =
as.Date(as.yearmon(tmp_other_var[,3])))

        other_var_ts = zoo(other_var_data, other_var_dates)

        plotTitle = paste("CCF", allSymbols[i], " Versus ", allSymbols[otherNdx])

        jpeg(file=fileName)
        ccf(var_ts, other_var_ts, plot = TRUE, na.action = na.pass, main = plotTitle)
        dev.off()

      }
    }
  }
}
```

5. Modleing ModelTimeData.R Function

```
ModelTimeData <- function(
 timeData = NULL,
 ts_orders = NULL,
 periodicity = NULL,
 season_orders = NULL,
 use_regressors = NULL,
 reg_data = NULL,
 filePath = NULL,
 data_desc = NULL
) {

 model_order = sum(ts_orders)
 order_str = paste(ts_orders[1], '_', ts_orders[2], '_', ts_orders[3], sep = '')
 season_str = paste(season_orders[1], '_', season_orders[2], '_', season_orders[3], sep = '')

 if ((periodicity > 0) && (use_regressors == TRUE)){
  ts_model = Arima(timeData, order = ts_orders, seasonal = list(order = season_orders, period = periodicity), xreg = reg_data, method = "ML")
  model_desc = paste(data_desc, '_ARIMA_', order_str, '_Season_', season_str, '_W_X_Regs', sep = '')
 }else if (periodicity > 0){
  ts_model = Arima(timeData, order = ts_orders, seasonal = list(order = season_orders, period = periodicity), method = "ML")
  model_desc = paste(data_desc, '_ARIMA_', order_str, '_Season_', season_str, sep = '')
 }else if (use_regressors == TRUE){
  ts_model = Arima(timeData, order = ts_orders, xreg = reg_data, method = "ML")
  model_desc = paste(data_desc, '_ARIMA_', order_str, '_X_Regs', sep = '')
 }else {
  ts_model = Arima(timeData, order = ts_orders, method = "ML")
  model_desc = paste(data_desc, '_ARIMA_', order_str, sep = '')
 }

 dir.create(file.path(filePath, model_desc), showWarnings = FALSE)
 model_path = paste(filePath, model_desc, "\\", sep="")

 num_coefs = length(ts_model$coef)
 model_coefs = ts_model$coef
 coef_names = names(model_coefs)

 ma_coefs = c()
 ar_coefs = c()
 ma_ndx = 1
 ar_ndx = 1

 for (i in 1:num_coefs){
  if (grepl("ma", coef_names[i])){
   ma_coefs[ma_ndx] = model_coefs[coef_names[i]]
   ma_ndx = ma_ndx + 1
  }else if (grepl("ar", coef_names[i])){
   ar_coefs[ar_ndx] = model_coefs[coef_names[i]]
   ar_ndx = ar_ndx + 1
  }
 }

 if (!is.null(ar_coefs)){
  ar_roots = polyroot(c(1, ar_coefs))
 }

 if (!is.null(ma_coefs)){
  ma_roots = polyroot(c(1, ma_coefs))
 }

 out_file_name = paste(model_path, 'Model_Analysis.txt', sep = '')
 outFile = file(out_file_name, open="wt")

 sink(file = outFile, append = TRUE)
```

```
print ('*** Model Order ***')
print(model_order)
print("", quote=FALSE)

# Model summary and coef test
print('*** Model Summary ***')
print(summary(ts_model))
print("", quote=FALSE)
print("", quote=FALSE)

print('*** Model Coefficient Tests ***')
print(coeftest(ts_model))
print("", quote=FALSE)

print('*** Model Polynomial Roots (> 1 = Stat) ***')
if (!is.null(ar_coefs)){
  print('AR Roots:')
  print(ar_roots)
  print("", quote=FALSE)
}

if (!is.null(ma_coefs)){
  print('MA Roots:')
  print(ma_roots)
  print("", quote=FALSE)
}
print("", quote=FALSE)

# Perform Jarque-Bera normality test
print('*** Residual Jarque Bera Normality Test ***')
print(normalTest(ts_model$resid, method=c("jb")))
print("", quote=FALSE)

# Ljung Box of residuals
print("*** Residual Ljung Box Test With 5 Lags")
print(Box.test(ts_model$resid, lag = 5, type = 'Ljung', fitdf = model_order))
print("", quote=FALSE)

print("*** Residual Ljung Box Test With 10 Lags")
print(Box.test(ts_model$resid, lag = 10, type = 'Ljung', fitdf = model_order))
print("", quote=FALSE)

print("*** Residual Ljung Box Test With 25 Lags")
print(Box.test(ts_model$resid, lag = 25, type = 'Ljung', fitdf = model_order))
print("", quote=FALSE)
print("", quote=FALSE)

#Augmented Dickey Fuller zero mean no trend
print('*** Augmented Dickey Fuller Test - No Intercept (Zero Mean), No Time Trend ***')
print(adfTest(ts_model$resid, lag = 5, type = c('nc'))) #model_order
print("", quote=FALSE)

#Augmented Dickey Fuller constant mean no trend
print('*** Augmented Dickey Fuller Test - Constant Intercept (Constant Mean), No Time Trend ***')
print(adfTest(ts_model$resid, lag = 5, type = c('c')))
print("", quote=FALSE)

#Augmented Dickey Fuller constant mean no trend
print('*** Augmented Dickey Fuller Test - Constant Intercept (Constant Mean), Time Trend ***')
print(adfTest(ts_model$resid, lag = 5, type = c('ct')))
print("", quote=FALSE)

print('*** 80% Training, 20% Validation Backtesting ***')
if (use_regressors == TRUE){
  print(backtest(ts_model, timeData, h = 1, orig = length(timeData)*0.8, xre = reg_data))
```

```
}else{
  print(backtest(ts_model, timeData, h = 1, orig = length(timeData)*0.8))
}

sink()

close(outFile)
closeAllConnections()

############################################ Scatterplots #############################################

tmp_scatter = paste(model_path, 'Residual_Scatter.jpg', sep = '')
jpeg(file = tmp_scatter)
x_title = 'Date'
y_title = 'Residuals'
plot_title = 'Residual Vs. Time'
plot(ts_model$resid, xlab = x_title, ylab = y_title, main = plot_title)
dev.off()

############################################Normal Plots ##############################################

tmp_qq = paste(model_path, 'Residual_QQ.jpg', sep = '')
jpeg(file = tmp_qq)
plot_title = 'Residual Normal Plot'
qqnorm(ts_model$resid, main = plot_title)
qqline(ts_model$resid)
dev.off()

############################################ ACF Plots ################################################

tmp_acf = paste(model_path, 'Residual_ACF.jpg', sep = '')
jpeg(file = tmp_acf)
plot_title = 'Residual ACF'
acf(ts_model$resid, plot = TRUE, na.action = na.exclude, main = plot_title)
dev.off()

########################################## Backtest Plots #############################################

if (use_regressors == TRUE){
  bt_data = backtest(ts_model, timeData, h = 1, orig = length(timeData)*0.8, xre = reg_data)
}else{
  bt_data = backtest(ts_model, timeData, h = 1, orig = length(timeData)*0.8)
}
x = seq(1, length(bt_data$obs), by=1)

tmp_bt_scatter = paste(model_path, 'Backtest_Forecast.jpg', sep = '')
jpeg(file = tmp_bt_scatter)
x_title = 'Index of Validation Points'
y_title = 'Validation Values (Obs = Black, Pred = Red)'
plot_title = 'Backtest Observed and Forecasted Validation Points'
plot(x, bt_data$obs, type='l', col='black', xlab = x_title, ylab = y_title, main = plot_title)
lines(x, bt_data$forecast, type='l', col='red')

dev.off()

tmp_bt2_scatter = paste(model_path, 'Backtest_Error.jpg', sep = '')
jpeg(file = tmp_bt2_scatter)
x_title = 'Index of Validation Points'
y_title = 'Back Testing Error'
plot_title = 'Back Testing Error (Observed - Forecasted)'
plot(x, bt_data$error, type='l', col='black', xlab = x_title, ylab = y_title, main = plot_title)

dev.off()


return(ts_model)
```

```
}
```