Litecoin Cryptocurrency Forecast – Variations on the Autoregressive Moving Average Model: A Time Series Analysis

Leonid Shpaner, Dingyi Duan

Shiley-Marcos School of Engineering, University of San Diego

Abstract

Cryptocurrency and blockchain are often synonymous, but over the last ten years each unique yet non distinct entity has staked a claim into the world of financial technology—a territory riddled with numerical puzzles; blending the art of predicting future results and seasonality with the science of time series projections hinges on a few important notions. Past performance is not indicative of future results, though it is useful in establishing trajectories. Investors and speculators alike can leverage the power of predictive analytics to establish trends on an ever-changing, ever-evolving domain that will remain relevant into the distant future. This paper aims to provide more than a cursory analysis of the behaviors and patterns of Litecoin (LTC) over the last decade, leveraging seasonality of the autoregressive integrated moving average model to forecast a sound and proper price trajectory that will give prospective investors a healthy outlook for future growth.

Keywords: time series analysis, ARIMA, GARCH, Litecoin, cryptocurrency, forecast, volatility, R programming

Table of Contents

Background: LTC Forecast - Variations on the Autoregressive Moving Average Model

2009 was an impactful year. An economic recession was underway, Barack Obama made history as the first African American President to be inaugurated into office, and Bitcoin (the world's first cryptocurrency) launched a new era in financial technology known as blockchain. Since then, the market has witnessed a plethora of rapidly expanding offshoots of this technology, scaled to provide encrypted solutions to managing smart contracts and currency worldwide. Litecoin, one such cryptocurrency was launched as a peer-to-peer smart contract provider (digital currency) in 2011 by a computer scientist named Charlie Lee. To this day, while most people remain skeptical of the benefits of investing in cryptocurrencies at large, one cannot doubt its rapid expansion and integration into the financial markets. Litecoin started trading at roughly $3.00 (USD) per coin and is now listed as one of the top cryptocurrency providers on the market.

While cryptocurrencies are part of a relatively new landscape within the context of financial markets, it is difficult to neglect their efficacy in producing returns on investments, decentralized systems, and secure financial transactions. Every investment carries with it a degree (standard deviation) of risk, be it a stock, mutual fund, or call option. Assessing that risk or volatility need not be relegated to the confines of a client-fiduciary relationship with an investment firm. For example, making informed decisions from a mathematically oriented vantage point can make the difference between calculated arbitrage and gambling. Litecoin was established to be complementary to Bitcoin, where it "can be used for smaller amounts of money and have lower fees" (SFOX, 2015). For prospective investors that are looking to diversify their portfolios in an evolving market where the future knows no limits, forecasting its potential and entertaining the idea or notion that this can create a possible economic boom (in the long run) is at a minimum, a worthwhile endeavor.

Literature Review

Existing and Alternative Methods

Cryptocurrencies exhibit peaks and troughs in the span of their continuously fluctuating financial life cycles. Despite the approach in analyzing this data, the principles of inherent noise, non-stationarity, and volatility hold true to these time series. Whereas from a machine learning perspective principal component analysis helps reduce the number of dimensions in the training set of data, Gidea et al. (2020) impose PCA on clustered data; this is done to illustrate log transformed price projections from a graphical rendering standpoint alone. Omitting statistical assumptions from modeling is commensurate with removing the inherent bias-variance trade-off structure that abounds. This introduces the geometric method of "topological data analysis (TDA)" (Gidea et al., 2020, p. 1), which helps leverage the unsupervised, non-parametric learning methodology of the *k*-means clustering landscape. However, Gidea et al. (2020) concede the necessity of summarizing statistical output following the fitting of generalized autoregressive conditionally heteroskedastic (GARCH) models.

Moreover, it is noted that "statistical properties of such assets show…distinctly non-stationary behavior" (pp. 9-10). This warrants logarithmic transformation of the asset (i.e., Litecoin) in conjunction with differencing of the volatility shocks, which translate to L^1-norms of the persistence landscapes as functions of TDA.

Forecasting Prices with R

Paul & Sadath (2021) make the case for using R versus Python in forecasting cryptocurrency time series for its relative novelty and reliability in producing statistical output. A short yet effective primer is given on blockchain technology—the instrumental force of smart contracts behind a network of distributed and decentralized ledgers used "to trade digital currency or tokens" (Paul & Sadath, 2021, p. 286). Bitcoin effectively instantiates the digital currency market, creating exponential price hikes through latter 2017, causing it to lose 70% of its value by early 2018 (p. 287). Suggestions for usage of deep learning models are made (i.e., bagging and stacking), citing ensemble methods as an “effective methodology to forecast cryptocurrency prices” (p. 287) and Twitter sentiment analysis. However, the predominant focus stays with time series analysis using the Prophet forecasting library in R, which can forecast “time series data based on additive model, in which non-linear trends are fit with yearly, weekly, and daily seasonality” (p. 288).

Moreover, recommendations for using autoregressive independent moving average (ARIMA), GARCH, and neural network autoregression (NNAR) models are made, citing better performance with NNAR with less volatility. However, “in case of extreme volatility ARIMA models show more accurate results” (p. 288). A cursory comparison of Bitcoin (BTC) prices with those of Ethereum (ETH) using Yahoo finance from 2015 until 2019 sets precedence for subsequent time series analyses for other cryptocurrencies to follow suit. Whereas establishing trends inherently necessitates a differencing of at least the first order, such a recommendation is not provided; albeit a log transformation of closing prices is noted wherein a one year out forecast is made.

Forecasting Comparison by Bayesian Time-Varying Volatility Models

Bohte and Rossini (2019) have contributed to a fine analysis of forecasting comparisons of cryptocurrencies using multiple Bayesian time-varying volatility models. Vector Autoregressive (VAR) models are generally used for empirical macroeconomic applications, and in this case, the Bayesian approach contains the stochastic volatility specification which is computationally tractable while possessing advantages in parameter uncertainty, computing of probabilistic statements and estimation with many parameters (Bohte & Rossini, 2019). To gain a better glance on if a more complex model can outperform a simple model on forecasting, a total of three models are used: The standard VAR model, VAR with stochastic volatility, and VAR with GARCH.

After using a series of point and density measurements which focus on 95% confidence intervals and Root Mean Squared Error (RMSE), the BVAR base model has shown a higher volatility in forecasting compared to the BVAR-GARCH model. The BVAR-SV and BVARX-SV models have the highest percentages of all the cryptocurrencies, which suggests that using Stochastic Volatility will not give a good prediction overall using confidence intervals (Bohte & Rossini, 2019). Since the results between the BVAR model and the BVARX model are close to each other, there is not a clear distinction between these two and hence does not help establish a preference for a model of choice.

Half-Life Volatility Measure

Engle and Patton (2001) define half-life as the time required for the volatility to move halfway back towards its unconditional mean. To investigate the half-life volatility measure of some cryptocurrencies, John et al. (2019) propose choosing two GARCH family models (PGARCH (1, 1) and GARCH (1, 1)) with the Student's-t distribution. After fitting the error term of the two GARCH models into various distributions (Gaussian, Student's-t, and Generalized Error), the PGARCH model is selected (John et al., 2019). During the procedure, the following tests are performed with notable results:

- The Jarque-Bera test for normality is statistically significant at the 5% alpha level for the return, meaning the return series is not normally distributed.
- The Ljung-Box Q-statistics for the return and squared return show evidence of autocorrelation in both the return and squared return series since Q (30) and Q2 (30) are significant at the 5% level of significance.
- The Quantile-Quantile plot is employed to confirm that the return is not normally

distributed, which is confirmed by the presence of outliers at the tails since the points do not approximate the straight line.

GARCH(1,1) is proven to show non-stationarity while PGARCH(1,1) shows stationarity. Thus, the PGARCH model is chosen to investigate the half-life volatility measure of the return of Litecoin. The returns of the cryptocurrencies used in the paper exhibit volatility persistence and long memory by observation of the return series. A shock in the returns of Litecoin will take six days for it to mean revert without any further volatility (John et al., 2019). Therefore, information pertaining to the half-life measure and volatility persistence the cryptocurrency market is important for investors to consider.

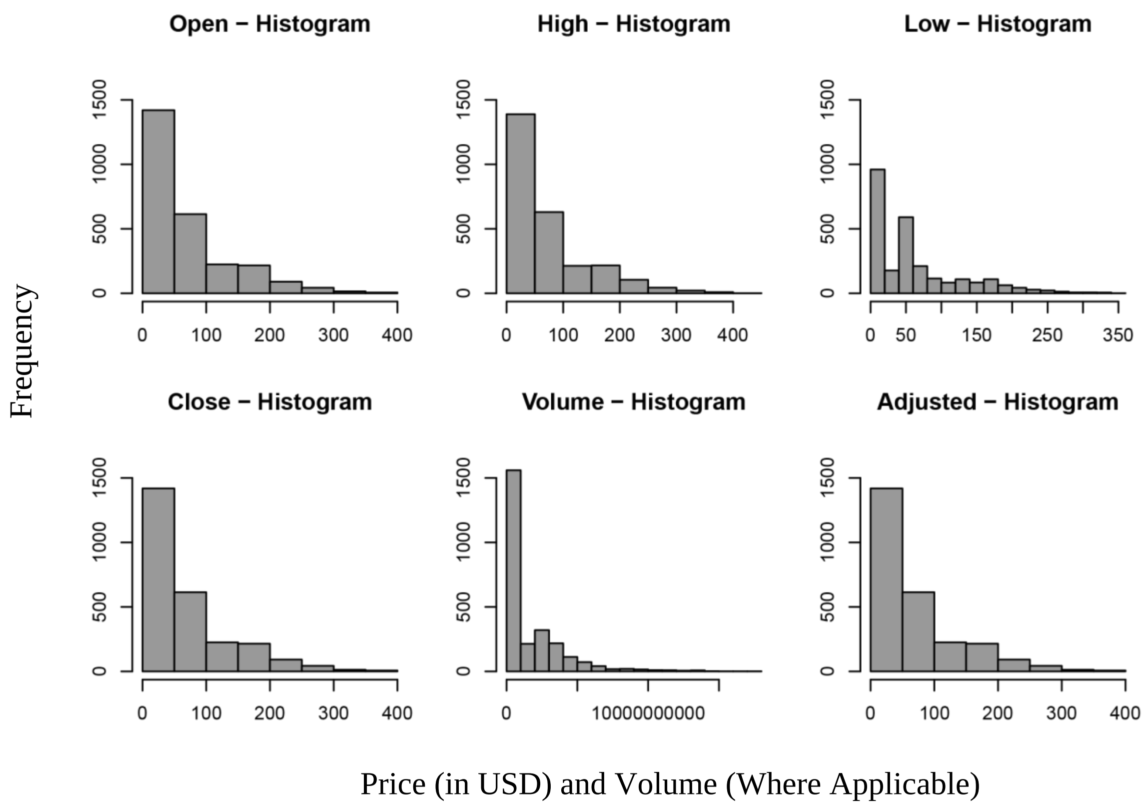
Exploratory Data Analysis (EDA) and Initial Preprocessing Steps

Preprocessing has its own unique procedure within the context of time series analysis; this will be discussed at length in a later section. Nonetheless, the following data cleaning steps are discussed to establish a foundational analytics framework. The quantmod library in R is installed, loaded, and leveraged to extract the Litecoin (LTC-USD) symbol from Yahoo Finance, the source that is connected to this library. The data is presented as a time series object which is subsequently converted into a data frame and assigned to its own unique variable. The dataset contains 2,632 rows, representing the date range of September 17, 2014 through November 30, 2021, and 6 columns (variables), corresponding to open, high, low, close (adjusted prices), and volume.

Data prior to September 17, 2014 is not available for reasons not offered by the provider. OHLC is used to abbreviate open, high, low, close prices in United States Dollars (USD). There are 24 missing values, which are omitted by calling a function that uses complete cases. Incomplete price data should not be imputed (i.e., mean, median, or any other method), for a potential loss in data integrity may result. This lends the dataset to subsequent preprocessing with relative ease. Examination of the OHLC price histograms and boxplots, respectively, reveals non-normal distributions for all variables. Figure 1 uncovers these degenerate, long-tailed distributions.

Figure 1

Litecoin Historical Prices and Volume Distributions (September 17, 2014 – November 30, 2021)



Note. Most prices are between \$0 to \$100 (USD), exhibiting rightly skewed distributions.

The data is therefore pre-processed with a Box-Cox transformation with an estimated λ of 0.2 for prices and 0.1 for volume. The skewness improves considerably where

$$\tilde{\mu}_3 = \frac{\sum_i^N (X_i - \bar{X})^3}{(N-1)\sigma^3}$$

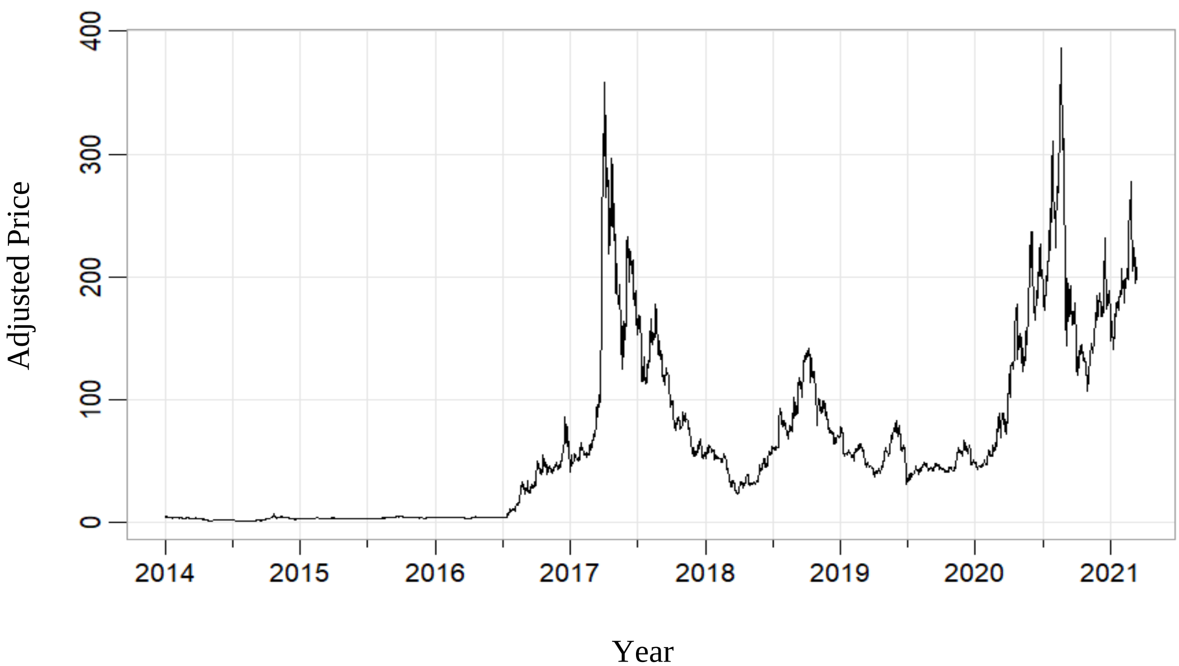
and $-0.383 \leq \tilde{\mu}_3 \leq -0.055$. However, this is strictly an exploratory preprocessing step to show potential improvement in estimating a Gaussian (normal) distribution, thus, not warranting integration into the original data frame; this is done to avoid loss of viability.

In evaluating Litecoin's performance, the close price may be intrinsically of interest, but the "adjusted closing price is considered to be a more technically accurate reflection of the true value" (Bischoff, 2019). Summarizing the data frame yields the following five number summary. Whereas the minimum adjusted price for the last six years is $1.16, the first quartile is $3.88, with a median of $46.32, and a mean of $64.08; the third quartile is $87.12, and the maximum recorded price for this date range is $386.45. From the supplementary correlation matrix used to examine the relationships between all six variables, it is discernible that whereas the OHLC prices exhibit perfect multicollinearity at $r = 1$, their relationship with volume is much less pronounced, where $-0.56 \leq r \leq -0.58$. The moderate correlation of r = 0.57 between the variable of interest (adjusted price) and volume does not lend itself for omission from ensuing analysis, nor does volume itself offer substantial influence on price. It exists to represent the full scope and context of the dataset at large. Granted, it will not be used within the context of this analytical endeavor. Moreover, principal component analysis (PCA) shows that 89.4% of the variance in the data is explained by the first principal component, where "the percentage of the total variance explained by each component" (Kuhn & Johnson, 2016, p. 38), translating to an effective dimension of 1. This is demonstrated numerically in Table 1 (in supplemental materials).

To graphically illustrate the historically adjusted prices, a new time series object in the form of a vector is created for the sole purpose of avoiding the representation of indexed time on the *x*-axis. Indexed time is harder to derive meaning from and defeats the purpose of graphical parsimoniousness. Therefore, it is important to see the impact of volatility visa vie market crashes and the corresponding years that this takes place. Furthermore, this object is placed into a plotting variable called litecoin_plot, with a starting date of 2014 and an annual frequency of 365; this is shown in Figure 2 below.

Figure 2

LTC Adjusted Closing Prices (2014 – 2021)



Note. Several crashes are observed (between 2017 – 2018 and 2020 – 2021).

Spectral Analysis Cyclical Behavior Periodogram Filters

An ensuing spectral analysis to determine the degree of periodicity within the data frame is conducted because "the idea that a time series is composed of periodic components appearing in proportion to their underlying variances is fundamental to spectral analysis" (Shumway & Stoffer, 2019, p. 137). Two dominant peaks (0.001, 0.001) are recorded, translating to cyclical behavior between 675 and 1,350 days. However, the confidence intervals based on the chi-squared distribution for the first (149,449.40 to 21,775,218.90) and second (357,264.80 to 52,054,543.70) period frequencies are too wide to be of use. Additional periodogram analyses will be required (i.e., to measure the effects of tapering), but "the periodogram as an estimator is susceptible to large uncertainties. This happens because the periodogram uses only two pieces of information at each frequency no matter how many observations are available" (p. 153).

Methodology

Further examination of the Litecoin time series includes the autocorrelation function (ACF) and partial autocorrelation function (PACF). ACF "measures the linear predictability of the series at time *t*, say x_t using only the value of x_s" (Shumway & Stoffer, 2019, p. 20). The PACF does the same for a truncated lag length, explaining the partial correlation between the series its own lags. The sample ACF is defined as follows:

$$\rho\, x(h) = \frac{\gamma\, x(h)}{\gamma\, x(0)} = \frac{(X_{t+h} - \bar{X})(X_t - \bar{X})}{\sum(X_t - \bar{X})^2} = \mathrm{Corr}(X_{t+h}, X_t).$$

An initial overview of the data shows that whereas the ACF gradually tapers off, the PACF cuts off after lag 1, thereby relegating the time series to the AR(1) model:

$$(x_t - \mu) = \phi(x_{t-1} - \mu) + \omega_t = \ (x_t - 64.0773) = 0.9960(x_{t-1} - 64.0773) + \omega_t$$

$$x_t = 0.256 + 0.996x_{t-1} + \omega_t$$

Differencing and Stationarity

Establishing non-stationarity in a time series component requires the expression of the mean as a function of time *t* where $E[y_t] = E[\beta_0 + \beta_1 t + \omega_t] = \beta_0 + \beta_1 t$. Time is non-stationary because $t_1 \neq t_2$ and $\mu(t_1) \neq \mu(t_2)$. To mitigate the continuous fluctuations exacerbated by predominant peaks, troughs, and general volatility of cryptocurrency market, the year-over-year trends observed in Litecoin's historically adjusted prices necessitate first order differencing $\nabla y_t = y_t - y_{t-1}$. Stationary is established visa vie the mean and autocovariance functions, respectively.

$$\nabla y_t = y_t - y_{t-1} = (\beta_1)(t - [t-1]) + \omega_t - \omega_{t-1}\,.\ \therefore \beta_1 + \omega_t - \omega_{t-1}$$

The mean function is applied in the following manner:

$$E[\nabla y_t] = E[\beta_1 + \omega_t + w_{t-1}] = \beta_1 + E[\omega_t] - E[\omega_{t-1}] = \beta_1$$

β_1 is independent of time *t* and is thus stationary, which is also implicit using the autocovariance function:

$$\gamma\nabla x_t(t+h,h) = \gamma\nabla x_t(h) = \operatorname{cov}(y_{t+h}, y_t)$$

$$= \operatorname{cov}(\beta_1 + y_{t+h} - y_{t+h-1}, \beta_1 + y_t - y_{t-1}$$

$$= \operatorname{cov}(y_{t+h} - y_{t+h-1}, y_t - y_{t-1})$$

$$= \operatorname{cov}(y_{t+h}, y_t) - \operatorname{cov}(y_{t+h}, y_{t-1}) - \operatorname{cov}(y_{t+h-1}, y_t) + \operatorname{cov}(y_{t+h-1}, y_{t-1})$$

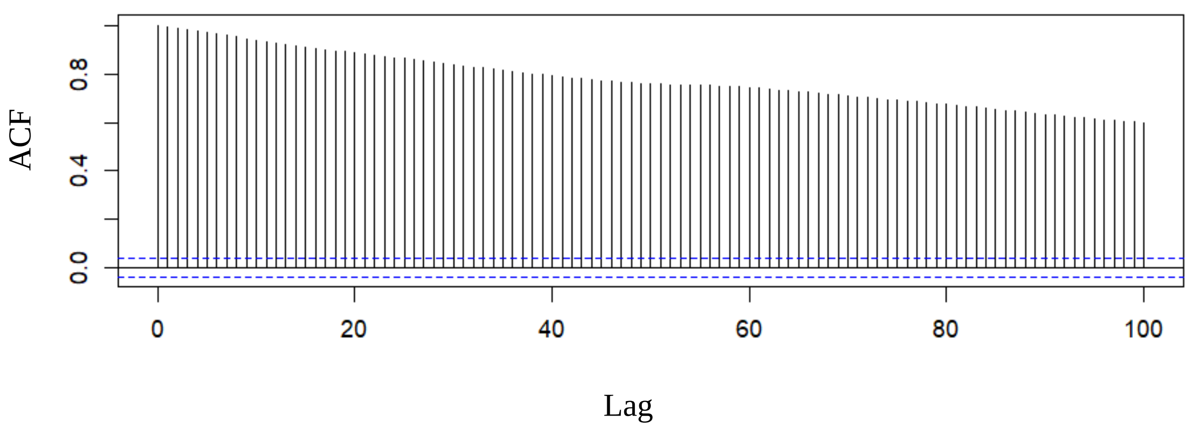
$$= \gamma_y(h) - \gamma_y(h+1) - \gamma_y(h-1) + \gamma_y(h)$$

$$= 2\gamma_y(h) - \gamma_y(h+1) - \gamma_y(h-1)$$

Differencing *h* shows that the autocovariance of the form $\gamma\nabla y_t(h)$ is not dependent on time *t.* Figure 3 of the sample ACF shows a slow dampening which indicates a long memory process. The presence of non-stationarity can be established visa vie trend alone.

Figure 3

Sample Autocorrelation Function (ACF) Plot for Litecoin's Adjusted Price



Note. A lag of 100 is enough to show these effects but is by no means the maximum lag for the series. Shumway & Stoffer (2019) present a basic equation for the expression of a long memory process as a byproduct of differencing fractional values where

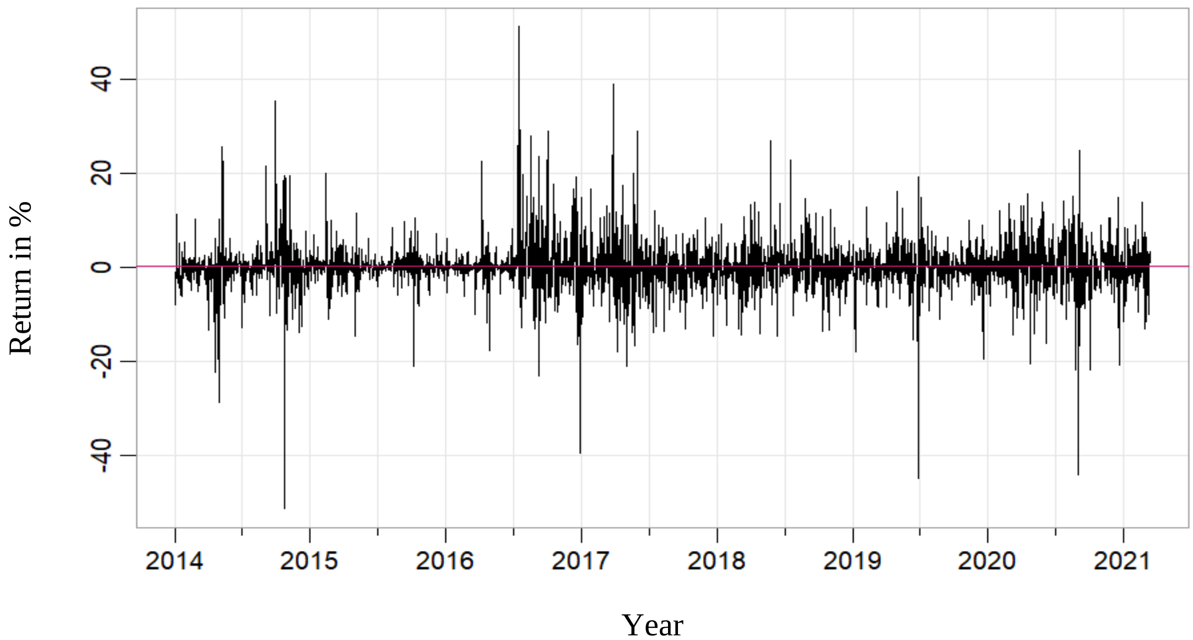
$$(1-B)^d x_t = \omega_t,$$

asserting that "time series data tend to exhibit sample autocorrelations that are not necessarily large (as in the case of *d* = 1), but persist for a long time” (p. 186).

Figure 4 shows the effects of differencing the Litecoin time series; the remaining residuals are relegated to white noise, taking on a constant zero mean.

Figure 4

Litecoin Continuous Compound Return (2014 – 2021)



Note. A continuous compound return is the direct result of differencing. The red line shows the mean annualized return of approximately 14%. This can be likened to the Dow-Jones Industrial Average, differenced data with a mean of zero and stationary property.

While differencing can be effectively performed by a diff() function call on the log of the time series, an alternative method is prescribed whereby the differenced time series is a function of adjusted prices divided by the open prices minus one; the graphical output produced by this function is the same. Differencing the log of adjusted prices produces continuous compound returns over time. This adjusts the overall model to an order and magnitude of MA(1) whereby the ACF cuts off after lag 1. Establishment of a strategical analytics framework is key in forecasting the annualized returns of the Litecoin cryptocurrency. However, prior to commencing the analytics process it is important to establish the null and alternative hypotheses for the mean of returns *r* as follows:

$$H_0: \mu_r = 14\%; H_a: \mu_r > 14\%.$$

ARIMA Models

Combining AR(1) with MA(1) produces an autoregressive integrated moving average model of ARIMA(1,0,1):

$$y_t = c + 0.341 y_{t-1} - 0.351 \varepsilon_{t-1}$$

where $c = 0.1457 \times (1 - 0.3409) = 0.096$ and ε_t is white noise with a standard deviation of 5.725. Subsequently, six ARIMA models are tested for performance visa vie AIC, initializing with ARIMA(0,1,0), a random walk with a zero mean:

$$\nabla y_t = y_t - y_{t-1}$$

$$y_t - y_{t-1} = \varepsilon_t, y_t = y_{t-1} + \varepsilon$$

The Akaike Information Criterion (AIC) score is but one method deployed for model selection where the model with the lowest score is optimum. Table 2 (in the supplementary materials section) shows the corresponding AIC scores commensurate with each respective ARIMA model. From this vantage point alone, ARIMA(3,1,2) can be selected and represented in the following equation:

$$y_t = 0.663 y_{t-1} - 0.008 y_{t-2} - 0.040_{t-3} - 1.664 \varepsilon_{t-1} + 0.664 \varepsilon_{t-2} + \varepsilon_t$$

where $c = 0$ and ε_t is white noise with a standard deviation of 0.058.

At this juncture, operating from a strictly empirical standpoint and structure necessitates the use of an all-encompassing automatic ARIMA model that can determine its own set of unique and optimal parameters. Within the construct of the R environment, the auto.arima() function looks almost identical to that of the standard arima() function, with one exception; the "auto ARIMA takes into account the AIC and BIC values generated…to determine the best combination of parameters" (Singh, 2018). Moreover, "it should be noted that the AIC statistic is designed for preplanned comparisons between models (as opposed to comparisons of many models during automated searches)" (Kuhn & Johnson, 2016, p. 493) and is thus used to select an optimal model (ARIMA(3,1,3)).

GARCH Model

The GARCH model is a response to the volatility shocks of the market, which requires thousands of observations. The generalized form of the GARCH(1,1) model is expressed as follows:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \alpha_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

The forecast for the GARCH(1,1) model originates at time *t* in:

$$\sigma_t^2(1) = \alpha_0 + \alpha_1 \alpha_t^2 + \beta_1 \sigma_t^2$$

and proceeds ℓ steps ahead in the following manner:

$$\sigma_t^2(\ell) = \alpha_0 + (\alpha_1 + \beta_1)\sigma_t^2(\ell - 1), \ell = 2, \cdots$$

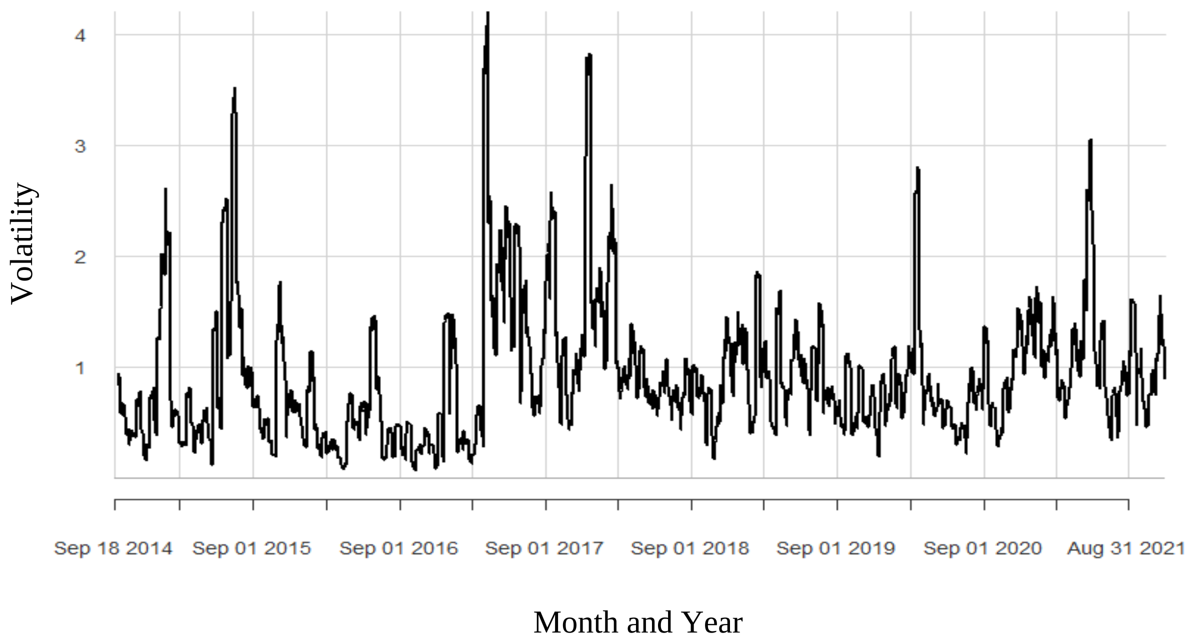
$$\sigma^2 = \alpha_0/(1 - \alpha_1 - \beta_1)$$

$$[\sigma_t^2(\ell) - \sigma^2] = (\alpha_1 + \beta_1)^{\ell-1}[\sigma_t^2(1) - \sigma^2].$$

Thus, as $\ell \to \infty, \sigma_t^2(\ell) \to \sigma^2$, where $\alpha_1 + \beta_1 < 1$. When the volatility forecast approaches infinity, the long-term variance at time *t* approaches infinity in the same manner. This is conditional upon the presence of the persistence of volatility being less than 1. Thus, "the speed of mean reverting to the long-term variance can also be measured by the half-life $\ell = \log(0.5) / \log(\alpha_1 + \beta_1)$" (Tsay, 2013, p. 245). The half-life is the amount of time that it takes for half of the volatility to diminish and revert to the mean—in essence, how long Litecoin's volatility will endure in a post-shock condition prior to reverting to its natural state. Whereas there are 252 trading days in the stock market, cryptocurrencies operate continuously. Therefore, volatility is annualized for *h*-period returns and is defined by $\sigma_{t,h,a} = \sqrt{365/h}\,\sigma_{t,h}$, where *a* refers to annualized volatility and h refers to number of days. Figure 4 represents these volatility shocks which are calculated by the standard deviation of the return over annualized time.

Figure 4

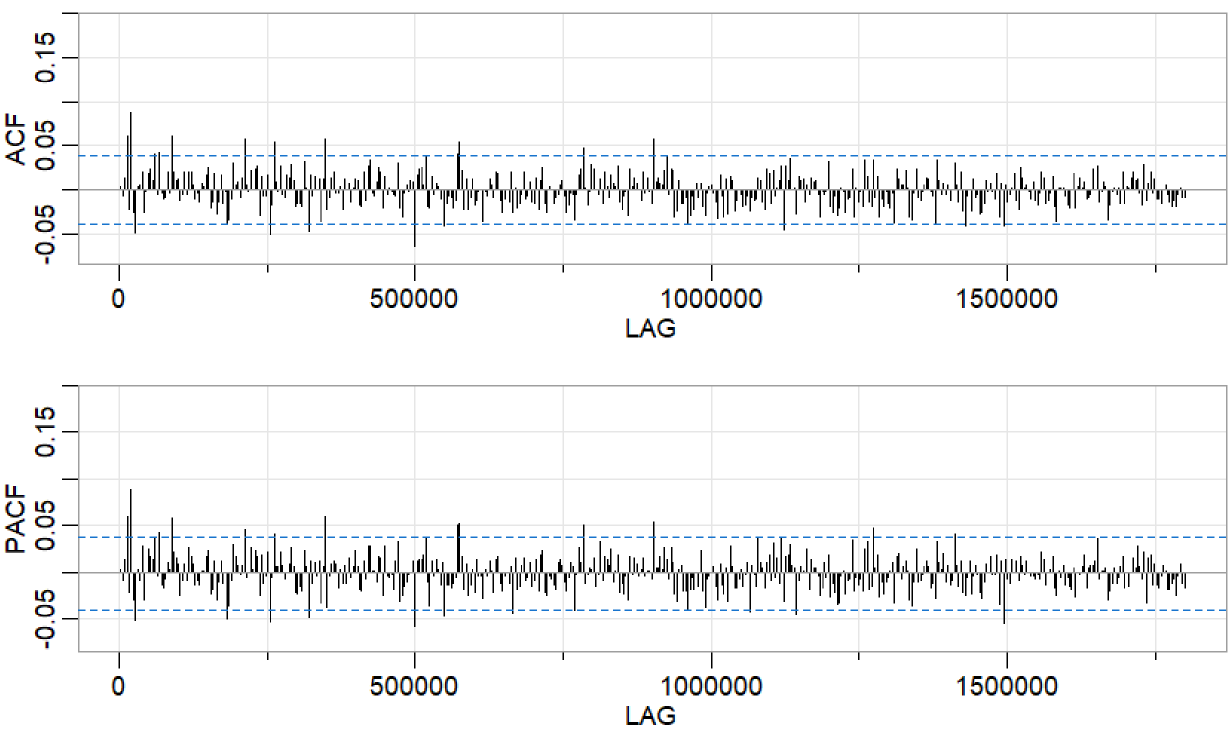
Litecoin – Annualized Volatility



Volatility shocks are observed throughout the series; year 2017 notably shows one of the highest peaks. Analysis of these shocks is conducted with the rugarch package (v1.4-4; Ghalanos, 2020). GARCH is introduced as a response to these stochastic volatility shocks where GARCH(1,1) is the best model fit according to the conditional variable dynamics of the ensuing summary output. However, it is important to note that the mean estimate of 0.000735 with a standard error of 0.000524 bears no statistical significance at a p-value of 0.160717 where $\alpha = 0.05$. The AIC and BIC are relatively low, expressing values of -3.506 and -3.488, respectively. More importantly, the weighted Ljung-Box test on standardized residuals shows corresponding lags with statistically significant p-values. Furthermore, the Adjusted Pearson Goodness-of-Fit “calculates the chi-squared goodness of fit test, which compares the empirical distribution of the standardized residuals with the theoretical ones from the chosen density" (Ghalanos, 2020). Herein, "the null hypothesis is that the conditional error term follows a normal distribution" (Tsafack, 2021). Each respective value presented is statistically significant, thereby rejecting a normal distribution (the null hypothesis) and corroborating the originally presented skewed Student's t-distribution. Moreover, additional evidence for GARCH-like behavior is provided since the ACF and PACF in Figure 5 below are both tapering off incrementally.

Figure 5

Litecoin Annualized Volatility - ACF and PACF



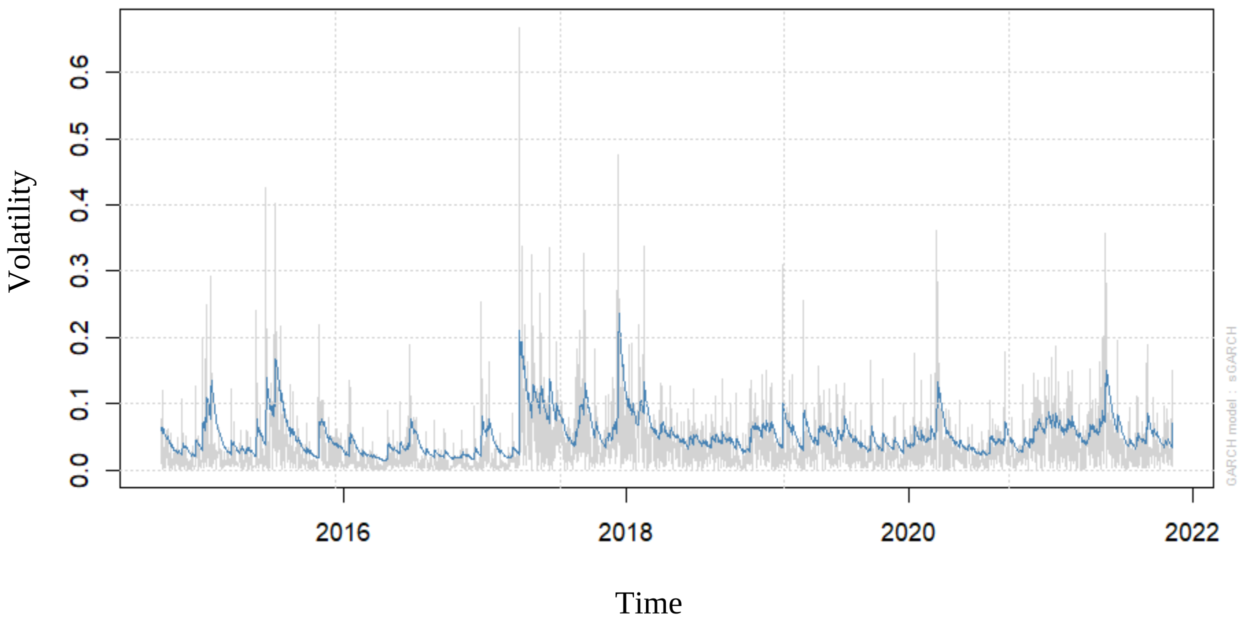
Note. The ACF and PACF is shown at a maximum lag of 500 to highlight the gradual tailing off effect. However, the code corresponding to these plots in the Appendix is not set to a maximum lag, as it "defaults to $\sqrt{n} + 10$ unless $n < 60$" (Stoffer & Poison, 2021).

Summarized Results

Fitting the GARCH(1,1) model to Litecoin's return data uncovers some interesting findings. For example, whereas the presence of stochastic volatility has been established through the lens of annualized returns, its persistence is highlighted visa vie Figure 6 where the conditional standard deviation is plotted versus returns. Two prominent shocks are observed in early 2018 and 2020.

Figure 6

Conditional Standard Deviation (vs |Returns|)



Note. The volatility trajectory is annualized through the end of 2021, expressing a standard deviation on return of approximately 0.058.

Revisiting the ARIMA(3,1,3) model is an important step before exploring predictive modeling. In so doing, additional findings are presented that are consistent with selection of an optimal model. The AIC score (-7446) is among the lowest of all the ARIMA models presented shown thus far. Table 2 shows that ARIMA(3,1,2) is slightly higher (-7453) by 7 units (supplementary materials). Moreover, the summary output table shows that all components of the moving average model are statistically significant where $0 \leq p \leq 0.002$. ARIMA(3,1,3) is represented in the following equation:

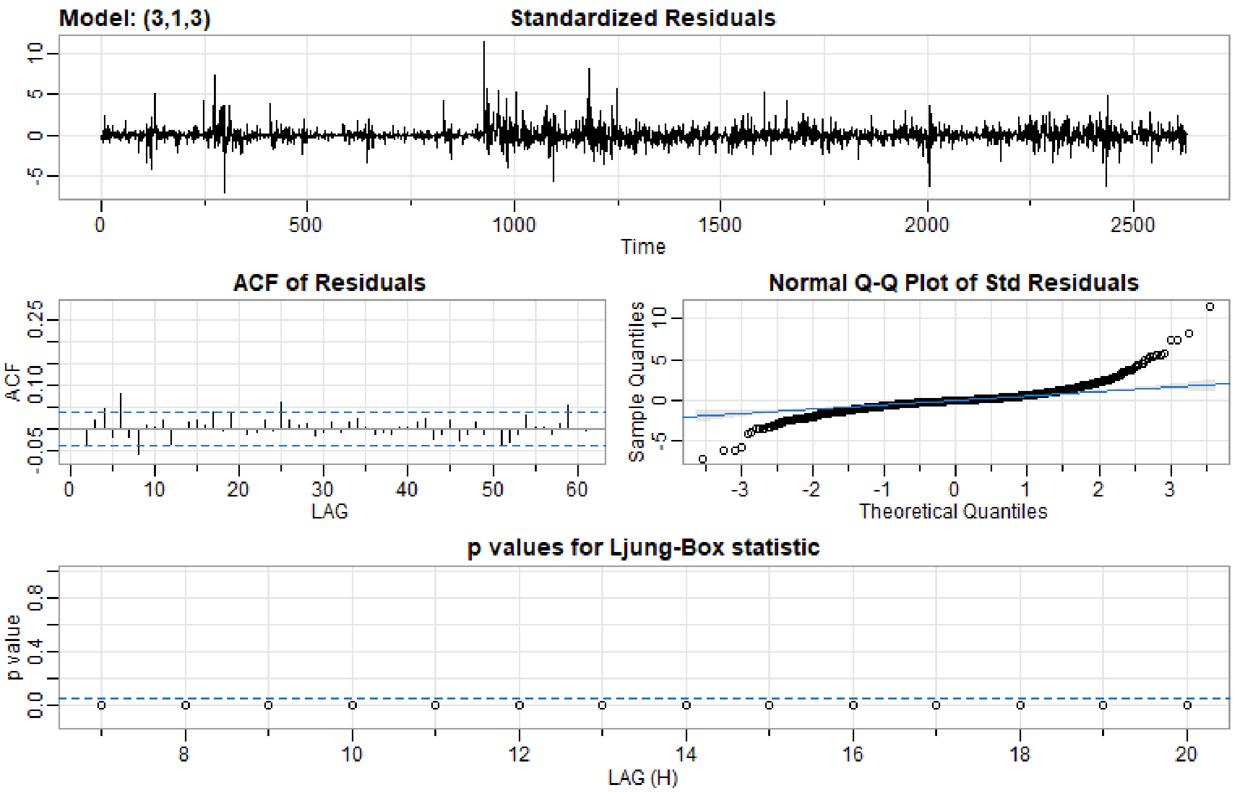
$$y_t = -0.181y_{t-1} + 0.567y_{t-2} - 0.002y_{t-3} - 0.816\varepsilon_{t-1} - 0.722\varepsilon_{t-2} + 0.540\varepsilon_{t-3} + \varepsilon_t$$

where $c = 0$ and ε_t is white noise with a standard deviation of 0.058.

Figure 7 shows the ensuing graphical output for this model which contains the diagnostics for the standardized residual, ACF, Q-Q and Ljung-Box statistic.

Figure 7

ARIMA(3,1,3) Diagnostics

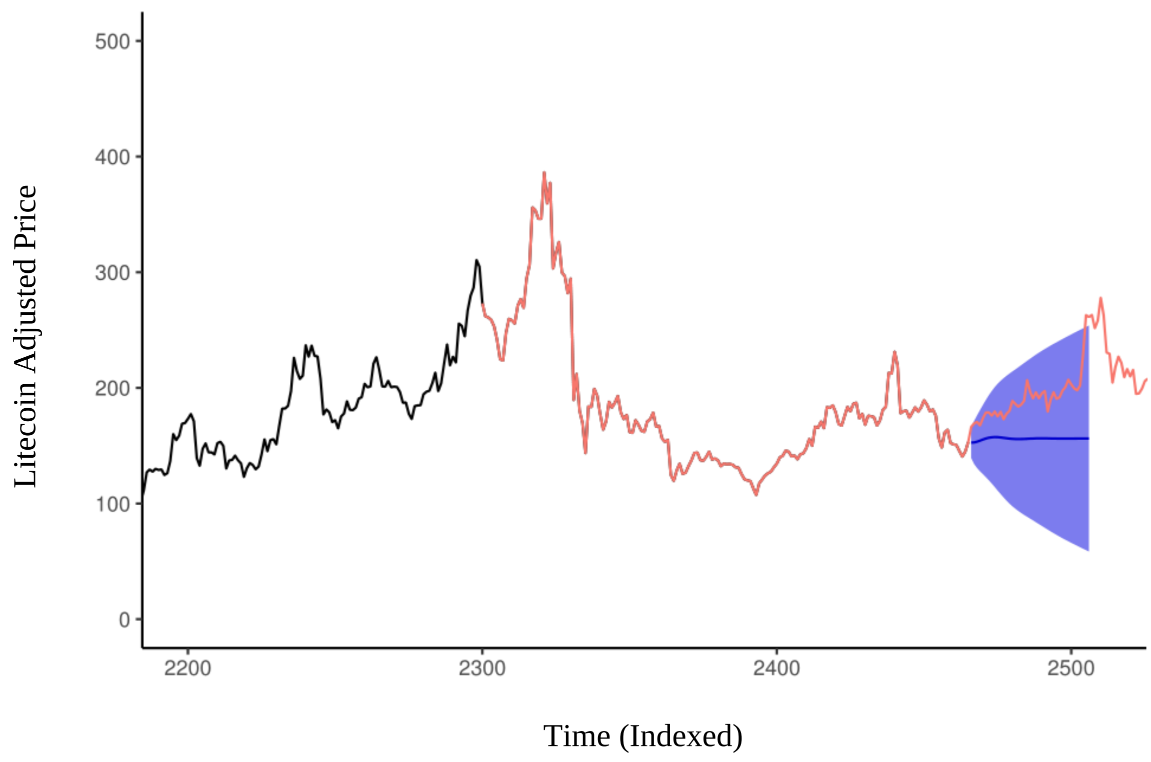


Note. The standardized residuals report trend-less and white noise-like behavior. The ACF of the residuals exhibits a sharp decline after lag 1, corroborating its MA(1) behavior. Moreover, the Normal Q-Q plot of the standardized residuals operates off the assumption of normality, albeit with a reasonable number of outliers at both tails. Lastly, the *p*-values shown in the Ljung-Box statistic plot are below the threshold of 0. Overall, the model has a good fit.

The modeling phase is complete, and an ensuing forecast of 41 days from November 30, 2021 is produced with a 95% confidence interval as shown in Figure 8 below. The time horizon (*x*-axis) is indexed on a numerical scale ranging from 2200-2500, where the trajectory is illustrated using a red color.

Figure 8

Forecasts from ARIMA(3,1,3) Model



Note. Forecast prediction: Once Litecoin surpasses an adjusted closing price of $400, it will drop below $200 and continue to fluctuate normally (as expected), reaching a maximum value of $253.00 within the next 41 days.

Limitations

Predictions are made within a 95% confidence level based on a chi-squared distribution. Moreover, whereas the premise behind building GARCH(1,1) is to strengthen the case for volatility as a function of time, it is not used to make ℓ step ahead forecasts in this paper. Subsequent explorations along these lines will only stand to strengthen pre-existing time series analyses of similar magnitudes. Furthermore, additional data visa vie larger sample size will be required to perform more robust, meaningful analyses (ARIMA included). Moreover, other variants of the GARCH model can be considered, like the GJR-GARCH model “developed in 1993 by Glosten, Jagannathan and Runkle” (Tsafack, 2021). One final note that is important to make is that the null hypothesis regarding the mean of the return warrants subsequent testing to establish whether it is possible to reject or fail to reject the premise that will remain at 14%.

Conclusion

Litecoin, one of the highly traded cryptocurrencies on the market possesses non-stationarity and high volatility, but substantial technological and financial potential. Its high skewness on all price (and volume) attributes aside, its periodical peaks can cycle every 675 and 1,350 days, which adds sentiment to a probable economic gain (thrust) in the near future. Differencing the time series is required, and from the tailing off of the ACF and PACF it is reasonable to use the ARIMA model on the cryptocurrency’s data to perform the forecast from the final model selection of ARIMA(3,1,3). In the next 41 days from November 30, 2021, the adjusted price should be fluctuating between $400 and $200 and then continue with its expected volatile movement.

This paper commences with the behaviors, characteristics, and historical movements of a highly rated cryptocurrency. It culminates with a forecast on its adjusted return, which is a direct byproduct of quantitative analysis. While past performance is not indicative of future results, a calculated effort is made so that investors can make informed decisions backed by facts and figures alone.

References

Bischoff, B. & Cockerham, R. (2019, March 31). Adjusted Closing Price vs. Closing Price. *Zacks.* https://finance.zacks.com/adjusted-closing-price-vs-closing-price-9991.html

Bohte, R., & Rossini, L. (2019). Comparing the forecasting of cryptocurrencies by Bayesian Time-varying volatility models. *Journal of Risk and Financial Management, 12*(3), 150. https://doi.org/10.3390/jrfm12030150

Engle, R. F., & Patton, A. J. (2001). What Good Is a Volatility Model? *Quantitative Finance, 1*(2), 237–245. https://doi.org/10.1088/1469-7688/1/2/305

Ghalanos, A. (2020). rugarch: Univariate GARCH models.

Gidea, M., Goldsmith, D., Katz, Y., Roldan, & Shmalo, Y. (2020). Topological Recognition of Critical Transitions in Time Series of Cryptocurrencies. *Physica A: Statistical Mechanics and its Applications, 548,* 1-22. https://doi.org/10.1016/j.physa.2019.123843

John, A., Logubayom, A. I., & Nero, R. (2019). Half-Life Volatility Measure of the Returns of Some Cryptocurrencies. *Journal of Financial Risk Management, 8,* 15-28. https://doi.org/10.4236/jfrm.2019.81002

Kuhn, M., & Johnson, K. (2016). *Applied Predictive Modeling.* Springer. https://doi.org/10.1007/978-1-4614-6849-3

Paul, L.R., Sadath, L. (2021). Big Data- Forecasting Digital Currency with R. *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 286-291. https://doi.org.sandiego.idm.oclc.org/10.1109/Confluence51648.2021.9377153

SFOX. (2018, October 9). Satoshi Lite in 2018: An Interview with Charlie Lee. *SFOX.* https://www.sfox.com/blog/satoshi-lite-in-2018-an-interview-with-charlie-lee/

Shumway, R., & Stoffer, D. (2019). *Time Series: A Data Analysis Approach Using R.* Chapman and Hall/CRC.

Singh, A. (2018, August 30). Build High Performance Time Series Models using Auto ARIMA in Python and R. *Analytics Vidhya.* https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/

Stoffer, D., Poison, N. (2021, October 20). Package ‘astsa’. *Github*. https://github.com/nickpoison/astsa/blob/master/astsa.pdf

Tsafack, I. (2021, January 25). GARCH models with R programming : a practical example with TESLA stock. *Idriss Tsafack.* https://www.idrisstsafack.com/post/garch-models-with-r-programming-a-practical-example-with-tesla-stock

Tsay, R.S. (2013). *An Introduction to Analysis of Financial Data with R.* John Wiley & Sons, Inc.

Supplementary Materials

Table 1

Percent Variance and Change by Principal Component

Principal Component	Percent Variance	Percent Change (Delta)
1	89.35	
2	10.46	78.90
3	0.11	10.35
4	0.07	0.03
5	0.01	0.06
6	0.00	0.01

Table 2

ARIMA Models: Log Likelihood and AIC

Model	$Sigma^2$	Log Likelihood	AIC
ARIMA(0,1,0)	0.006817	2825	-5647
ARIMA(1,1,0)	0.005141	3195	-6386
ARIMA(0,1,1)	0.003416	3730	-7456
ARIMA(1,1,1)	0.003416	3730	-7454
ARIMA(2,1,2)	0.003416	3730	-7450
ARIMA(3,1,2)	0.003405	3733	-7453

Appendix

Loading the Necessary Packages (Libraries)

```
# Pack function: install and load more than one  R packages.
# Check to see if packages are installed.
# Install them if they are not,
# Then load them into the R session.

pack <- function(lib){
    new.lib <- lib[!(lib %in%
                     installed.packages()[, "Package"])]
    if (length(new.lib))
        install.packages(new.lib, dependencies = TRUE)
    sapply(lib, require, character.only = TRUE)
}

# usage
packages <- c('astsa', 'xts', 'tidyquant', 'quantmod', 'tidyverse', 'dplyr',
              'pander', 'fpp2', 'broom', 'caret', 'factoextra', 'corrplot',
              'e1071', 'rugarch')
pack(packages)
```

```
##      astsa        xts  tidyquant   quantmod  tidyverse      dplyr     pander
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
##       fpp2      broom      caret factoextra   corrplot      e1071    rugarch
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
```

Preprocessing - Initial Steps

```
# get (read-in) data for the last 10 years
start = as.Date("2011-11-30") # start date
end = as.Date("2021-11-30")   # end date

# data might not be available for entirety of date range
# but a 10 year look back is done to accommodate full size and scope
getSymbols(c("LTC-USD"),
           src = "yahoo",
           from = start,
           to = end)
```

```
## [1] "LTC-USD"
```

Exploratory Data Analysis (EDA)

```
# cast litecoin time series into dataframe
litecoin_df <- data.frame(`LTC-USD`)
colnames(litecoin_df) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")

litecoin.ts <- tq_get("LTC-USD", from = "2011-11-30", to = "2021-11-30") %>%
select(adjusted) %>% # adjusted price (more accurate than close price)
ts(.) # turning it into a time series object
ltc_xts <- as.xts(litecoin_df)
str(litecoin_df); str(litecoin.ts)
```

```
## 'data.frame':    2632 obs. of  6 variables:
##  $ Open    : num  5.09 5.07 4.69 4.33 4.26 ...
##  $ High    : num  5.17 5.07 4.76 4.62 4.3 ...
##  $ Low     : num  4.97 4.58 4.25 4.2 4.15 ...
##  $ Close   : num  5.06 4.69 4.33 4.29 4.25 ...
##  $ Volume  : num  3071840 4569260 3917450 5490660 2931220 ...
##  $ Adjusted: num  5.06 4.69 4.33 4.29 4.25 ...

##  Time-Series [1:2632, 1] from 1 to 2632: 5.06 4.69 4.33 4.29 4.25 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "adjusted"
```

```
cat("Dimensions of dataset:", dim(litecoin_df)) # dimensions of dataset
```

```
## Dimensions of dataset: 2632 6
```

```
cat("There are", sum(is.na(litecoin_df)), 'missing values in the dataset. \n')
```

```
## There are 24 missing values in the dataset.
```

```
# list columns pertaining to missing values in dataframe
list_na <- colnames(litecoin_df)[ apply(litecoin_df, 2, anyNA)]; list_na
```

```
## [1] "Open"     "High"     "Low"      "Close"    "Volume"   "Adjusted"
```

```
# remove missing values
litecoin_df <- litecoin_df[complete.cases(litecoin_df),]
litecoin.ts <- litecoin.ts[complete.cases(litecoin.ts),]
ltc_xts <- ltc_xts[complete.cases(ltc_xts),]
```

```
# Check for missing values after complete cases (removal)
cat("\n There are", sum(is.na(litecoin_df)), 'missing values in the dataset.\n',
    'New dimensions of dataset:', dim(litecoin_df))# dimensions of dataset
```

```
##
##  There are 0 missing values in the dataset.
##  New dimensions of dataset: 2628 6
```

At the time of the analysis, the dataset has 2628 rows and 6 columns of data.

```
# inspect the first and last few rows of data
head(litecoin_df, 8)
```

```
##               Open    High     Low   Close  Volume Adjusted
## 2014-09-17 5.08589 5.17077 4.96595 5.05855 3071840  5.05855
## 2014-09-18 5.06543 5.06543 4.57996 4.68523 4569260  4.68523
## 2014-09-19 4.68729 4.75582 4.25435 4.32777 3917450  4.32777
## 2014-09-20 4.32920 4.61608 4.20219 4.28644 5490660  4.28644
## 2014-09-21 4.26307 4.30013 4.15499 4.24592 2931220  4.24592
## 2014-09-22 4.24593 4.41688 4.21013 4.24235 1855960  4.24235
## 2014-09-23 4.23999 4.88135 4.18887 4.74657 4661670  4.74657
## 2014-09-24 4.74420 4.74512 4.62769 4.66679 2662290  4.66679
```

```
tail(litecoin_df, 8)
```

```
##                Open     High      Low    Close     Volume Adjusted
## 2021-11-23 209.3134 218.4258 205.9170 216.3890 1944651936 216.3890
## 2021-11-24 216.3625 217.8411 206.5169 209.8066 1884041986 209.8066
## 2021-11-25 212.2335 229.6452 210.9241 215.6350 1953190727 215.6350
## 2021-11-26 222.9104 224.8620 190.8467 194.8746 2690646017 194.8746
## 2021-11-27 195.5579 200.8455 191.1983 195.1744 1406618152 195.1744
## 2021-11-28 195.4146 199.8712 184.1060 199.3542 1712282909 199.3542
## 2021-11-29 199.5910 209.2916 195.7546 205.8702 1784850980 205.8702
## 2021-11-30 205.7330 218.3396 198.1463 208.0145 2122547294 208.0145
```

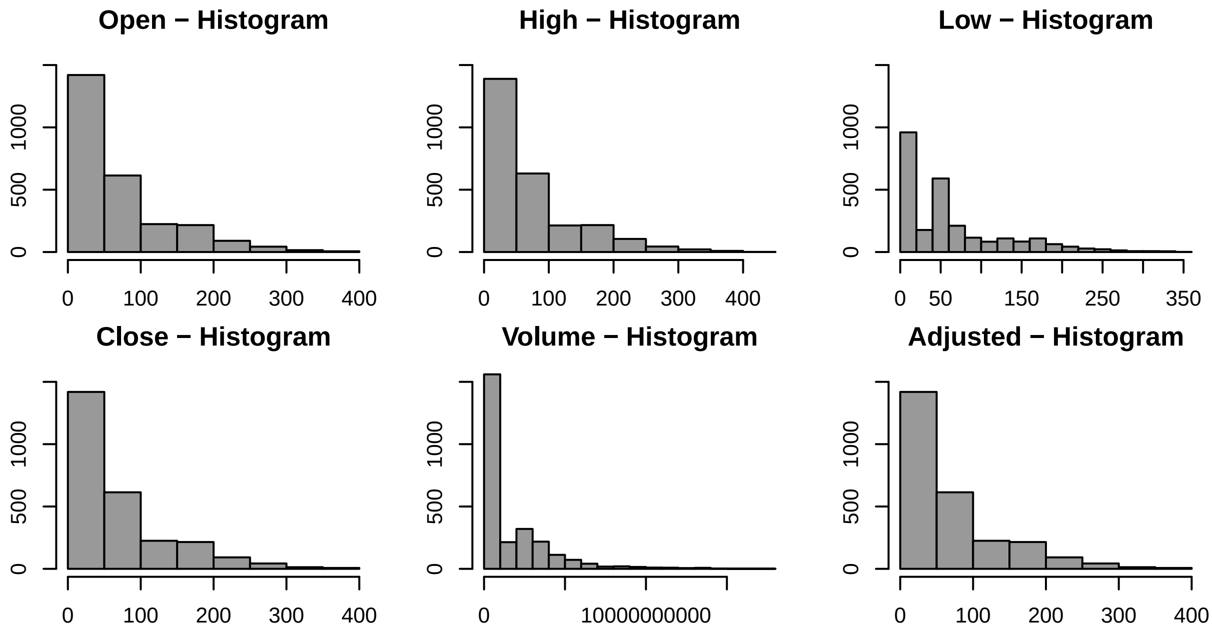
Summary Statistics

```
summary(litecoin_df[,6]) # summary stats of adjusted close prices
```

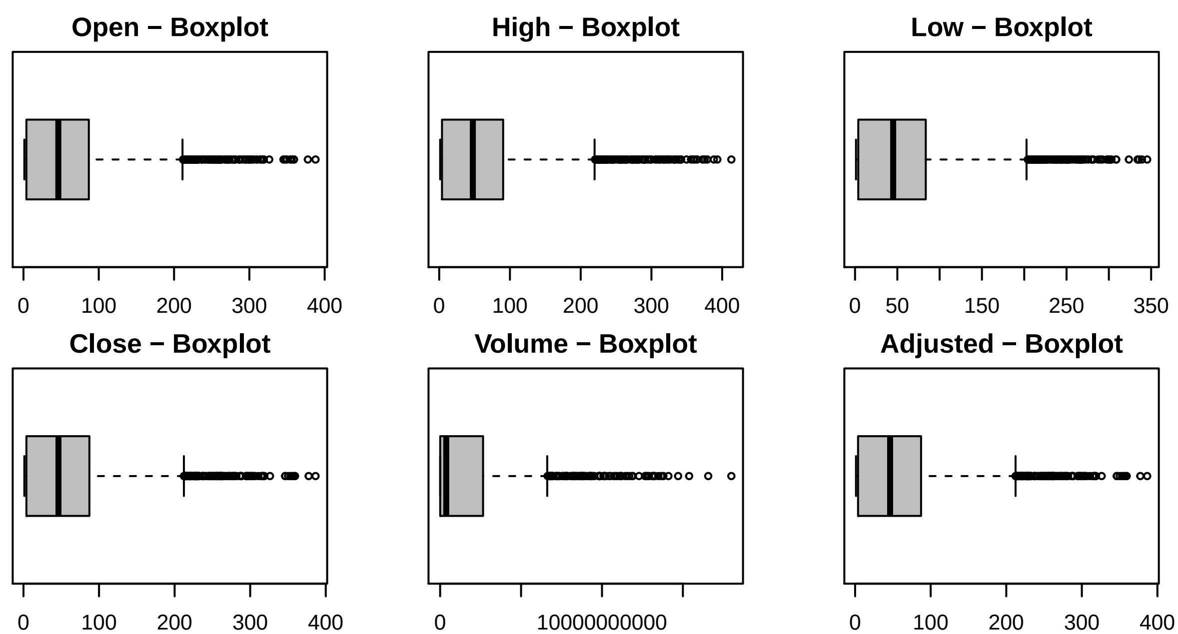
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.157   3.879  46.324  64.075  87.115 386.451
```

Distributions

```
# histogram distributions
par(mfrow = c(2,3), mar = c(2, 2, 2, 2))
options(scipen=999)

for (i in 1:ncol(litecoin_df)) {
    hist(litecoin_df[,i],
    xlab = names(litecoin_df[i]), ylim=c(0,1600),
    main = paste(names(litecoin_df[i]), "- Histogram"),
    col="gray60")
  }
```




```r
# boxplot distributions
par(mfrow = c(2, 3),
    mar = c(2, 2, 2, 2))

for (i in 1:ncol(litecoin_df)) {
    boxplot(litecoin_df[,i],
    ylab = names(litecoin_df[i]),
    main = paste(names(litecoin_df[i]), "- Boxplot"), horizontal=TRUE,
    col="gray")
  }
```



The OHLC (open, high, low, close) and adjusted prices exhibit long-tailed distributions with a right skew;

so does the volume.

Since we are interested in evaluating Litecoin's performance as a cryptocurrency, the final (close) price would be intrinsically of interest; but, more importantly, the "adjusted closing price is considered to be a more technically accurate reflection of the true value" (Bischoff, 2019).

```
# test skewness by looking at mean and median relationship
mean_ltc <- round(apply(litecoin_df, 2, mean, na.rm = T),0)
median_ltc <- round(apply(litecoin_df, 2, median, na.rm = T),0)
distribution<- data.frame(mean_ltc, median_ltc)
distribution$Skewness <- ifelse(mean_ltc > 2 + median_ltc, "skewed", "normal")
distribution
```

```
##            mean_ltc median_ltc Skewness
## Open             64         46   skewed
## High             67         48   skewed
## Low              61         45   skewed
## Close            64         46   skewed
## Volume   1593186877  375924496   skewed
## Adjusted         64         46   skewed
```

```
# Check for exact skewness in LTC.Volume
skewValue <- apply(litecoin_df, 2, skewness, na.rm=T)
skewValue
```

```
##     Open     High      Low    Close   Volume Adjusted
## 1.410603 1.453786 1.357228 1.406507 2.299834 1.406507
```

```
# Applying Box-Cox Transformation on skewed variable
trans <- preProcess(data.frame(litecoin_df), method=c("BoxCox"))
trans
```

```
## Created from 2628 samples and 6 variables
##
## Pre-processing:
##   - Box-Cox transformation (6)
##   - ignored (0)
##
## Lambda estimates for Box-Cox transformation:
## 0.2, 0.2, 0.2, 0.2, 0.1, 0.2
```

```
# look at and compare to transformed data
transformed <- predict(trans, data.frame(litecoin_df))
skew_transformed <- apply(transformed, 2, skewness, na.rm=T)
skew_transformed
```

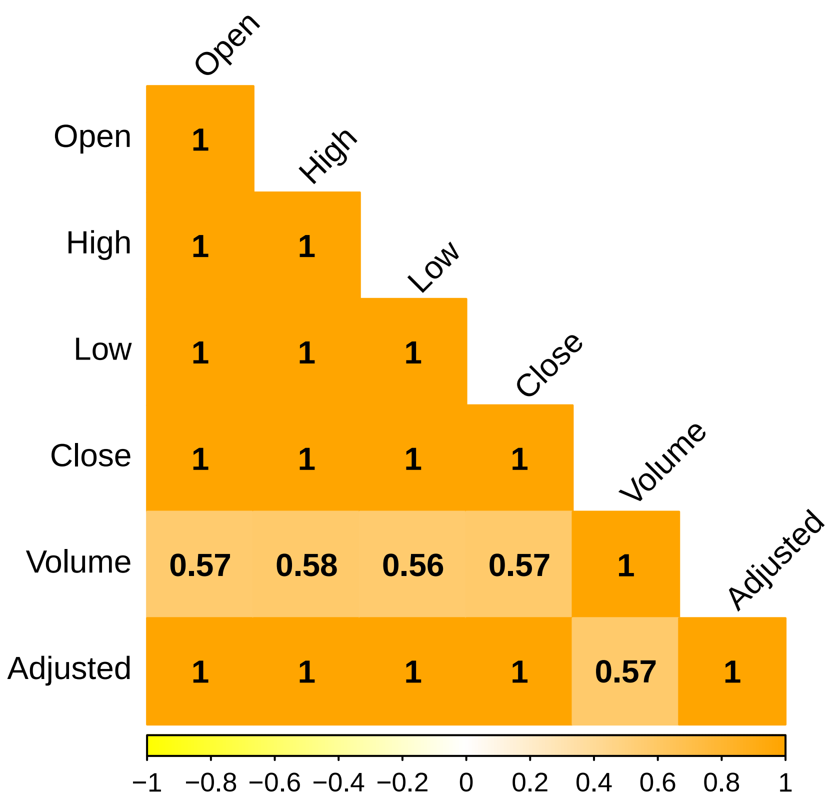
```
##        Open        High         Low       Close      Volume    Adjusted
## -0.06181721 -0.05492312 -0.07210481 -0.06325734 -0.38318043 -0.06325734
```

```
new_skew <- data.frame(skewValue, skew_transformed)
new_skew$Skew_Variance <- ifelse(skewValue < skew_transformed, "More skewed",
                                 "Less skewed")
new_skew
```

```
##          skewValue skew_transformed Skew_Variance
## Open      1.410603      -0.06181721   Less skewed
## High      1.453786      -0.05492312   Less skewed
## Low       1.357228      -0.07210481   Less skewed
## Close     1.406507      -0.06325734   Less skewed
## Volume    2.299834      -0.38318043   Less skewed
## Adjusted  1.406507      -0.06325734   Less skewed
```

Correlation Matrix

```
# assign correlation function call to variable
cor_ltc <- cor(litecoin_df)
# plot the correlation table (matrix)
corrplot(cor_ltc,
         method="color",
         col=colorRampPalette(c("yellow",
                                "white",
                                "orange"))(200),
         addCoef.col = "black",
         tl.col="black", tl.srt=45, type="lower")
```



From the correlation matrix, it can be discerned that whereas the OHLC and adjusted prices exhibit multicollinearity at $r = 1$, their relationships with volume is much less pronounced, where $0.56 \leq r \leq 0.58$.

Principal Component Analysis (PCA)

```
# center, scale the data, and assign to PCA variable
litecoin.pca <- prcomp(litecoin_df, center = TRUE, scale. = TRUE)

# assign to variance explained variable
var_explained <- round(litecoin.pca$sdev^2/sum((litecoin.pca$sdev)^2)*100, 4)
```

```
fviz_eig(litecoin.pca, main="Scree Plot of Six Principal Components",
         xlab="Principal Components",
         ylab = "Percent Variance Explained",
         barcolor = "grey", barfill = "grey",
         linecolor = "blue", addlabels=T,
         ggtheme=theme_classic())
```

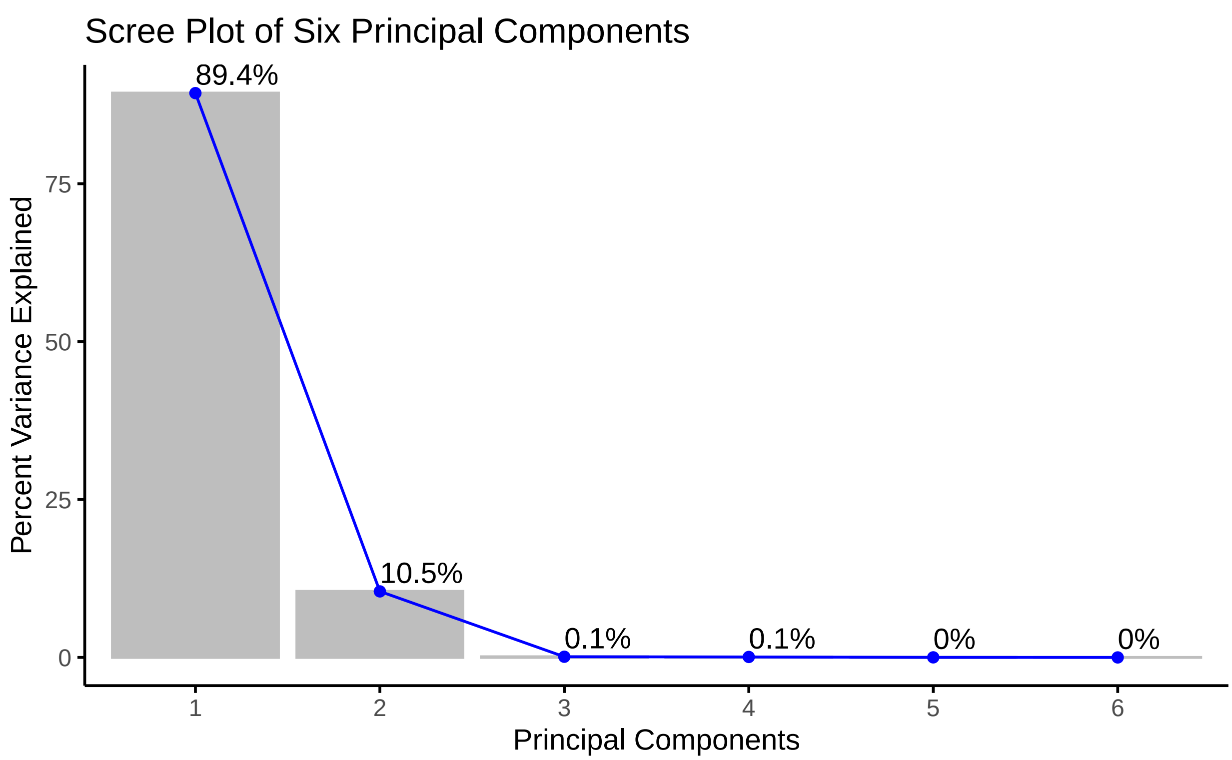
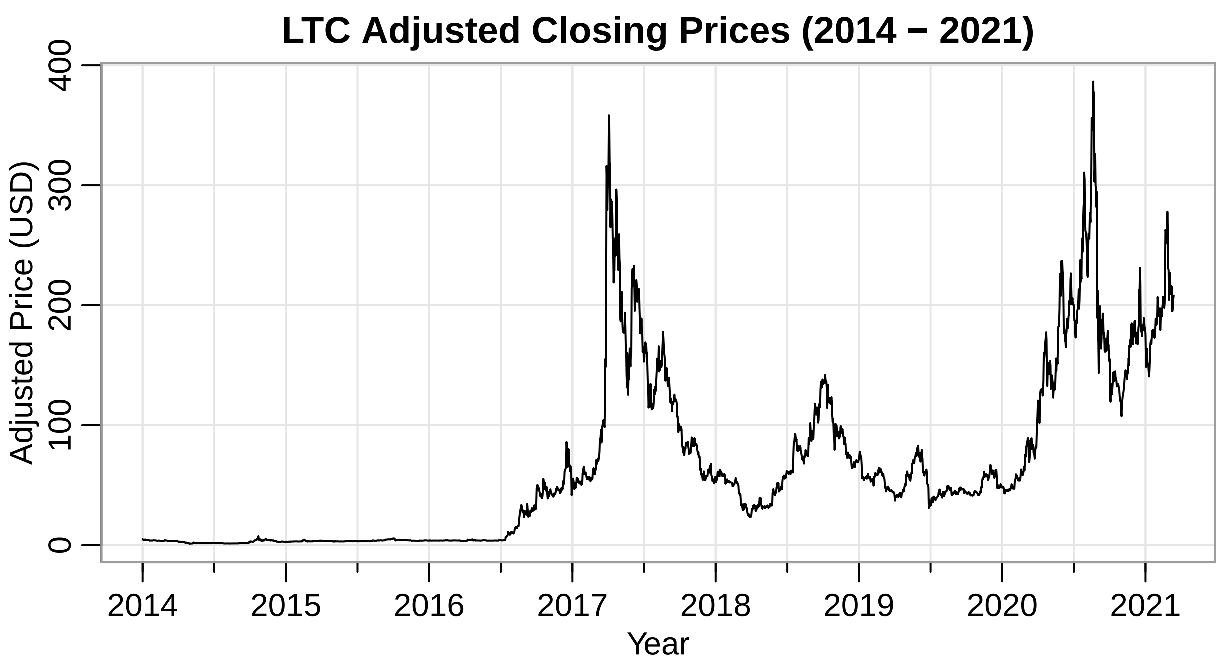


Table 1: Percent Variance and Change by Principal Component

Principal Component	Percent Variance	Percent Change (Delta)
1	89.35	
2	10.46	78.9
3	0.11	10.35
4	0.07	0.03
5	0.01	0.06
6	0	0.01

Approximately 89.35% of the variance in the data is explained by the first principal component; thus, the effective dimension is 1. This is supported by and demonstrated in the scree plot and the ensuing table above. The table itself numerically demonstrates the percent variance that is explained by each respective principal component. The scree plot visually depicts "the percentage of the total variance explained by each component" (Kuhn & Johnson, 2016, p. 38).

```
# create new variable for sole purpose of plotting years on x-axis, not indices
litecoin_plot <- ts(as.vector(litecoin.ts), start=c(2014), frequency = 365)
tsplot(litecoin_plot, main='LTC Adjusted Closing Prices (2014 - 2021)',
       xlab='Year', ylab='Adjusted Price (USD)') # plot the time series
```



The time series shows a clear trend with several predominant peaks and troughs, at approximately 2017 - 2018 and 2020 - 2021, respectively. To mitigate (offset) the trend, differencing will be performed. Furthermore, the autocorrelation function (ACF) and partial autocorrelation function (PACF) are examined. Whereas ACF "measures the linear predictability of the series at time t, say x_t using only the value of x_s" (Shumway & Stoffer, 2019, p. 20), the PACF does the same for a truncated lag length.

Autocorrelation Function (ACF)

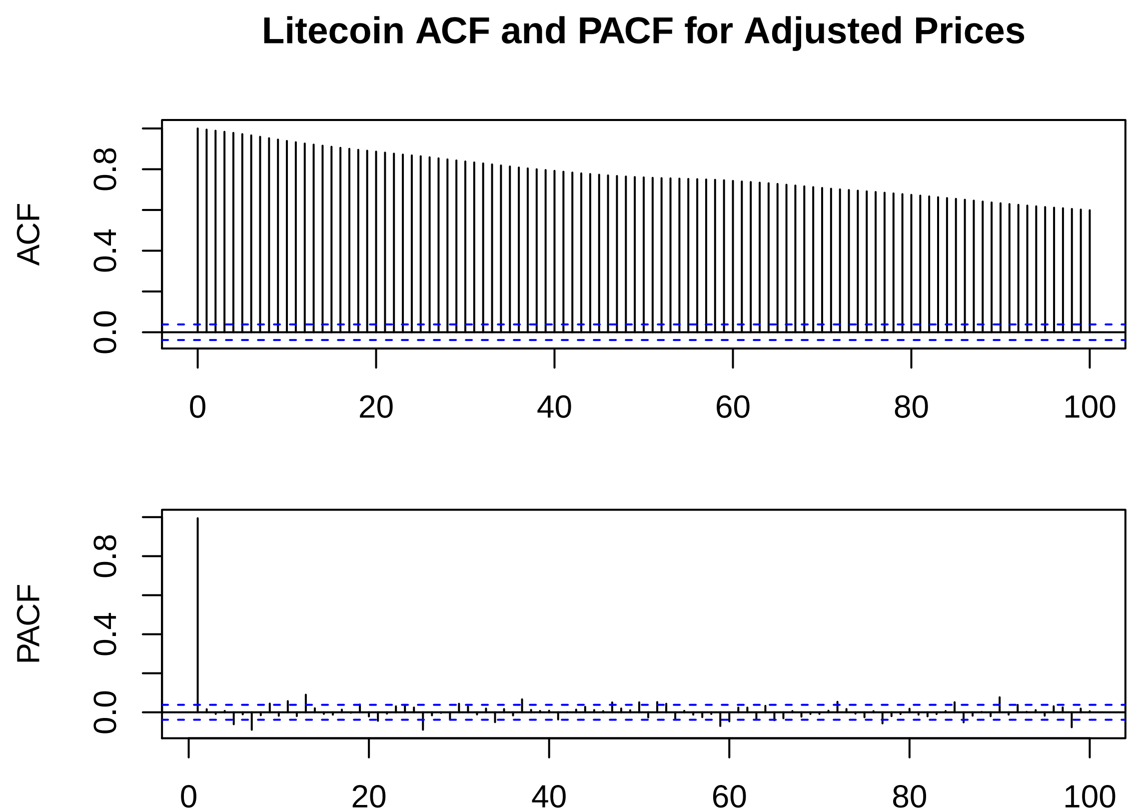
$$\rho(s,t) \quad = \quad \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}$$

where $-1 \leq \rho(s,t) \leq 1$.

For the sample ACF, we have:

$$\begin{aligned}\rho\ x(h) \quad &= \quad \frac{\gamma(x(h)}{\gamma x(0)} = \frac{(X_{t+h} - \bar{X})(X_t - \bar{X})}{\sum(X_t - \bar{X})^2} \\ &= \quad \mathrm{Corr}(X_{t+h,} X_t)\end{aligned}$$

```r
par(mfrow=c(2,1), oma = c(2,2,0,0) + 0.1, mar = c(1,4,3,1) + 0.1)
acf(litecoin_df$Adjusted, lag.max=100, main='Litecoin ACF and PACF for Adjusted Prices')
pacf(litecoin_df$Adjusted, lag.max=100, main='', ylab='PACF')
```



Whereas the ACF gradually tapers off, the PACF cuts off after lag 1, thereby relegating this to an AR(1) model. So we have the following:

```r
arima(litecoin_df$Adjusted, order=c(1, 0, 0))
```

```
##
## Call:
## arima(x = litecoin_df$Adjusted, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9960    64.0773
## s.e.  0.0018    30.5462
##
## sigma^2 estimated as 46.5:  log likelihood = -8776.55,  aic = 17559.09
```

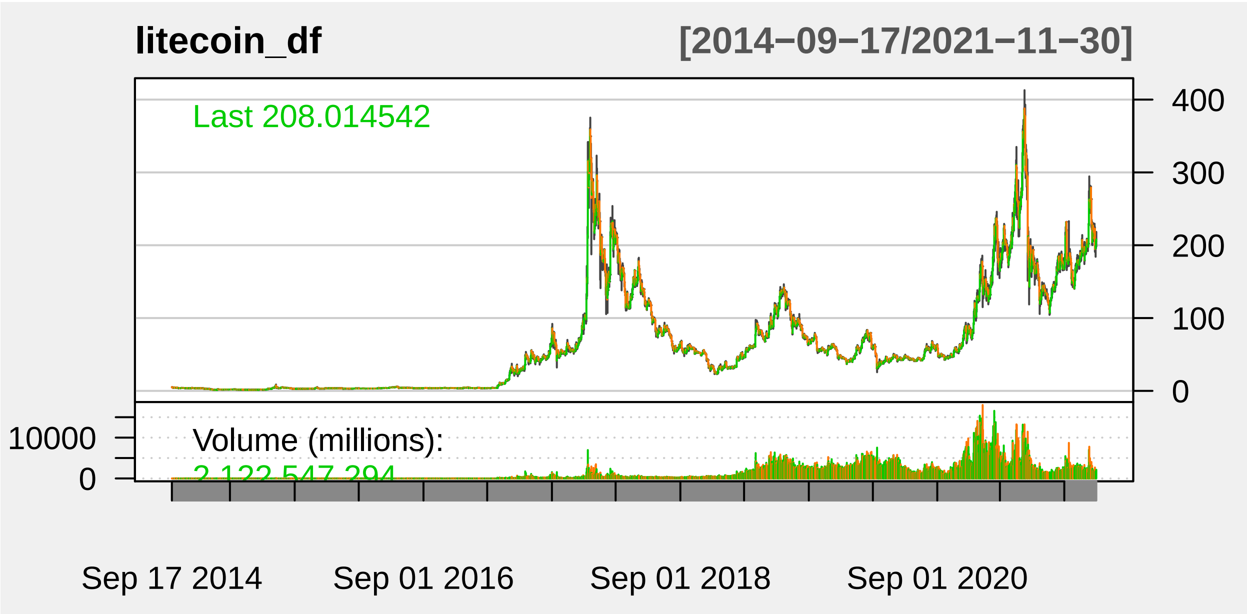
$$
\begin{aligned}
(x_t - \mu) &= \phi_1(x_{t-1} - \mu) + \omega_t. \\
(x_t - 64.0773) &= 0.9960(x_{t-1} - 64.0773) + \omega_t \\
x_t &= 64.0773 - (64.0773 \times 0.9960) + 0.9960x_{t-1} + \omega_t = 0.256 + 0.9960x_{t-1} + \omega_t
\end{aligned}
$$

Smoothing and its Effects

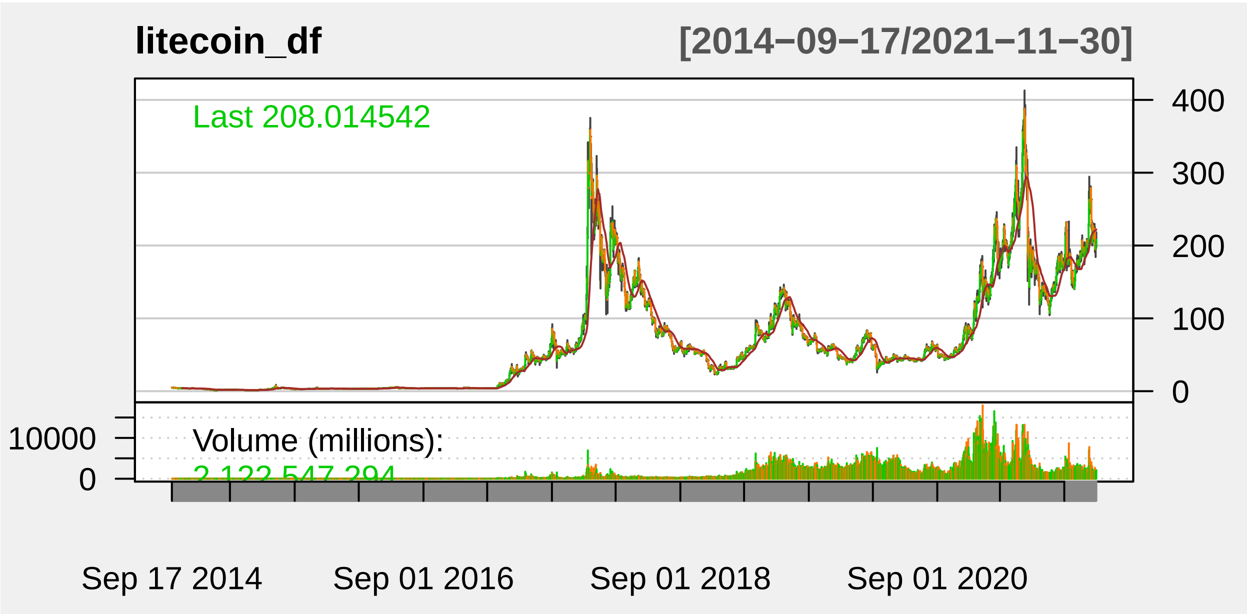
We plot the data for the last six years (November 2014 through November 2021).

Next, we smooth the data by introducing the simple moving average (SMA), and exponential moving average (EMA), respectively, weighting the effects by 30 days (one full month).

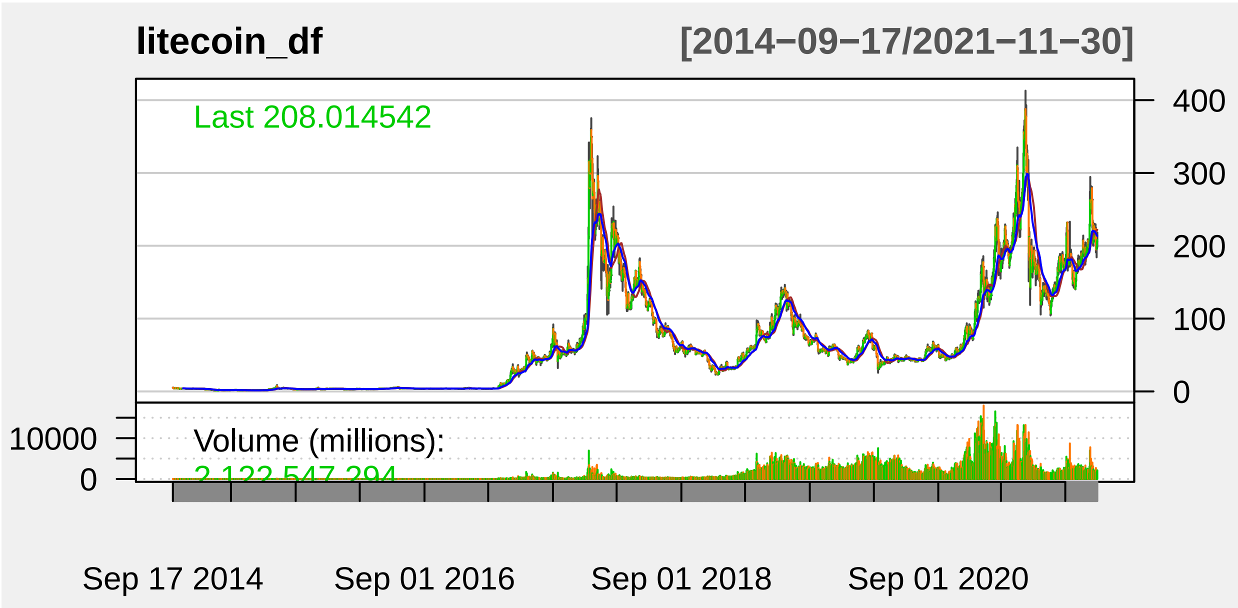
```
chartSeries(litecoin_df, theme = chartTheme("white"))
```




```
addSMA(30) # smoothed out moving average by 30 days
```

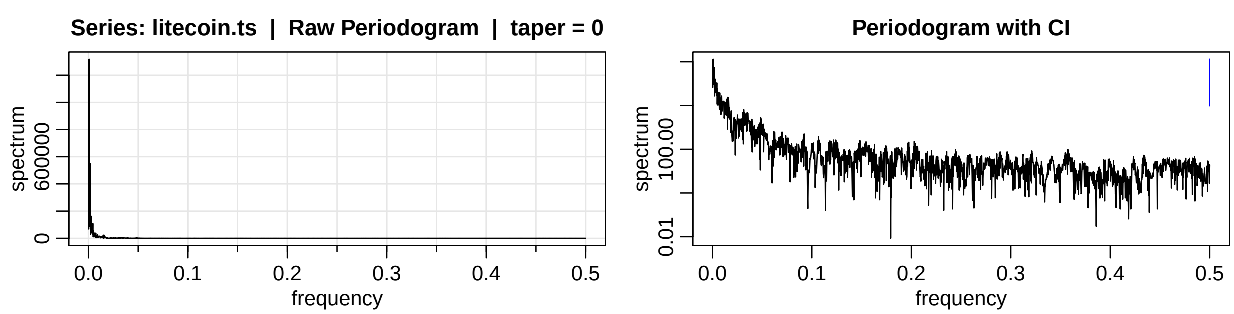


```r
addEMA(30) # exponential moving average by 30 days
```



Spectral Analysis Cyclical Behavior Periodogram Filters

```r
par(mfrow=c(1,2)); ltcfreq <- mvspec(litecoin.ts,taper=0,log="no") # peaks
ltcfreq2 <- spec.pgram(litecoin.ts,taper=0,log="yes",
                       main ='Periodogram with CI') # graph confidence interval
```




```r
# sort the frequencies in descending order and get  top 2
sort_ltcfreq <- sort(ltcfreq$spec, decreasing = TRUE)[c(1,2)]
p1 <- ltcfreq$freq[ltcfreq$spec==sort_ltcfreq[1]]; p1
```

```
## [1] 0.0007407407
```

```r
p2 <- ltcfreq$freq[ltcfreq$spec==sort_ltcfreq[2]]; p2
```

```
## [1] 0.001481481
```

```
cat('Cycles are occuring every', round(1/p1,1), 'days and ', 1/p2, 'days')
```

```
## Cycles are occuring every 1350 days and  675 days
```

```
CI <- function(peak_spec){
 u <- qchisq(0.025,2)
 l <- qchisq(0.975,2)
      c((2*peak_spec)/l,(2*peak_spec)/u)} # confidence intervals of the peaks
CI(sort_ltcfreq[1]) # CI for peak 1
```

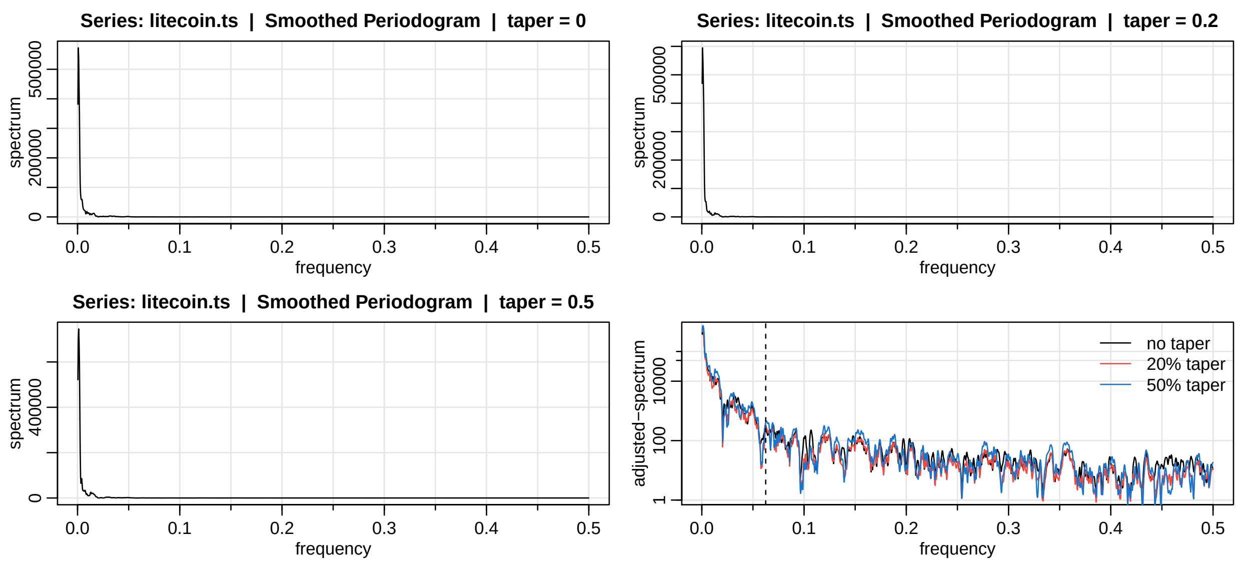
```
## [1]   357264.8 52054543.7
```

```
CI(sort_ltcfreq[2]) # CI for peak 2
```

```
## [1]   149449.4 21775218.9
```

Dominant peak is ≈ 0.0. Each of the generic confidence intervals is too wide to be of much use.

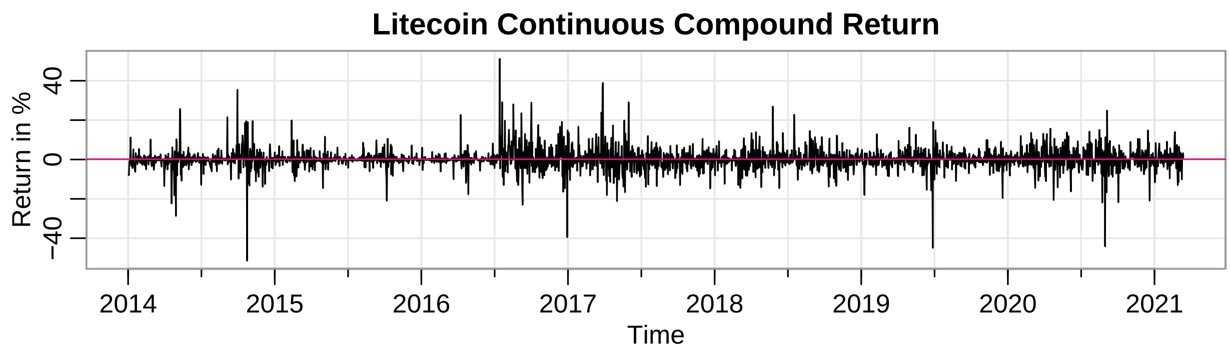
```
# nonparametric spectral estimation + graph the data with different tapering
par(mfrow=c(2,2))
ltcfreq_taper0 = mvspec(litecoin.ts, spans=c(2,2), log="no", taper=0)
ltcfreq_taper2 = mvspec(litecoin.ts, spans=c(2,2), log="no", taper=0.2)
ltcfreq_taper5 = mvspec(litecoin.ts, spans=c(2,2), log="no", taper=0.5)
plot(ltcfreq_taper0$freq, ltcfreq_taper0$spec, log="y", type="l",
     ylab="adjusted-spectrum", xlab="frequency", panel.first=Grid())
lines(ltcfreq_taper2$freq, ltcfreq_taper2$spec, col=2)
lines(ltcfreq_taper5$freq, ltcfreq_taper5$spec, col=4)
abline(v=1/16, lty=2)
legend("topright", legend=c("no taper", "20% taper", "50% taper"), lty=1,
       col=c(1,2,4), bty="n")
```



By comparing the different tapering, we can see that having more tapering can slightly decrease the degrees of freedom and enhances the center of the data relative to the extremities. Thus we choose the smoothing with 50% tapering.

Preprocessing - Differencing

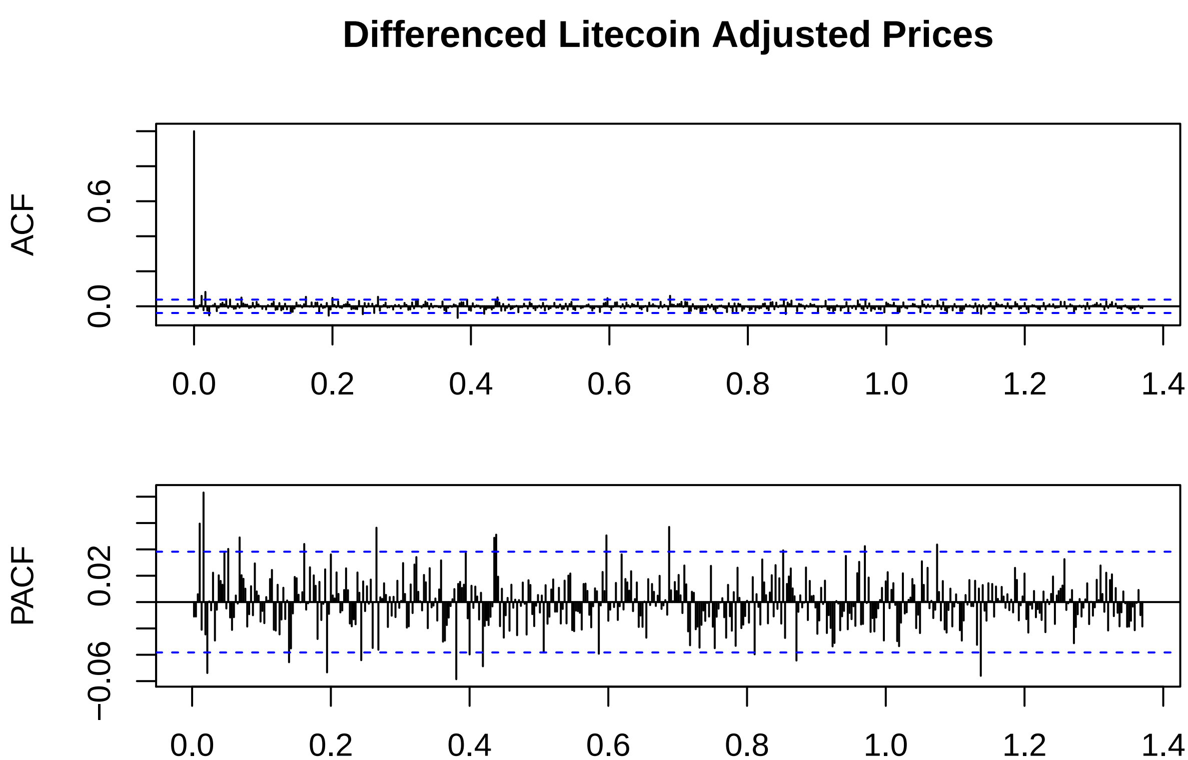
```
diff_ltc_1 <- diff(log(litecoin_plot))*100
tsplot(diff_ltc_1, main='Litecoin Continuous Compound Return',ylab='Return in %')
abline(h=mean(diff_ltc_1),col=6); cat('Mean return:', mean(diff_ltc_1))
```




```
## Mean return: 0.1414742
```

This can be likened to the Dow-Jones Industrial Average (DJIA), which is the differenced data, and shows a mean of zero; this gives it the stationary property.

```
par(mfrow=c(2,1), oma = c(1,1,0,0) + 0.09, mar = c(1,4,3,0.5) + 0.08)
acf(diff_ltc_1, lag.max=500, main = 'Differenced Litecoin Adjusted Prices')
pacf(diff_ltc_1, lag.max=500, main='', ylab='PACF')
```



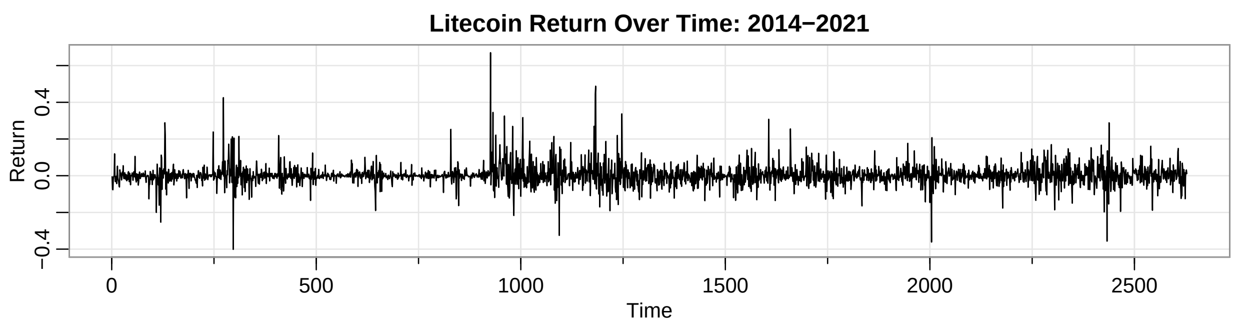
```r
arima(diff_ltc_1, order=c(1,0,1)) # Introductory ARIMA model (1,0,1)
```

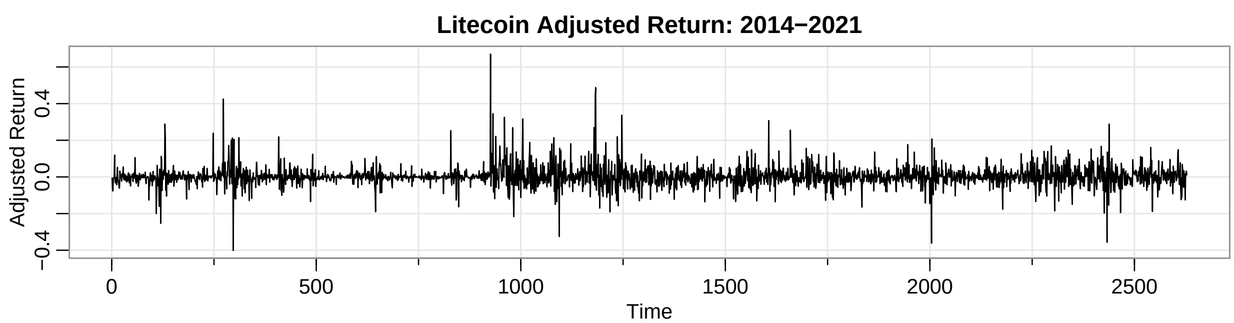
```
##
## Call:
## arima(x = diff_ltc_1, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.3409  -0.3514     0.1457
## s.e.  0.4836   0.4728     0.1099
##
## sigma^2 estimated as 32.77:  log likelihood = -8310.97,  aic = 16629.95
```

$$y_t = c + 0.3409 y_{t-1} - 0.3514 \varepsilon_{t-1}$$

where $c = 0.1457 \times (1 - 0.3409) = 0.096031$ and ε_t is white noise with a standard deviation of $\sqrt{\sigma^2} = \sqrt{32.77} = 5.725$.

```r
par(mfrow=c(2,1))
litecoin_df$Return <- litecoin_df$Close/litecoin_df$Open-1
litecoin_df$Adj_Return <- litecoin_df$Adjusted/litecoin_df$Open-1

# plot return
tsplot(litecoin_df$Return, main='Litecoin Return Over Time: 2014-2021',
       ylab='Return')
# plot adj.return
tsplot(litecoin_df$Adj_Return, main='Litecoin Adjusted Return: 2014-2021',
       ylab='Adjusted Return')
```





ARIMA Models

By differencing the data, we remove the trend, and can use the ARIMA model.

At this stage, we can conclude our exploratory data analysis with a six year historical pricing inquiry. Volatility shocks must be considered.

Cryptocurrency is a relatively new, ever-changing and ever-evolving financial technology. For this reason, we will take more conservative approach by forecasting five years out.

```
# create a few models and compare the AIC scores in a table
arima010 <- arima(litecoin_df$Adj_Return,order=c(0,1,0))
arima110 <- arima(litecoin_df$Adj_Return,order=c(1,1,0))
arima011 <- arima(litecoin_df$Adj_Return,order=c(0,1,1))
arima111 <- arima(litecoin_df$Adj_Return,order=c(1,1,1))
arima212 <- arima(litecoin_df$Adj_Return,order=c(2,1,2))
arima312 <- arima(litecoin_df$Adj_Return,order=c(3,1,2))

# find AIC for each model and assign to variable
sigma_2 <- c(arima010$sigma2, arima110$sigma2, arima011$sigma2, arima111$sigma2,
             arima212$sigma2, arima312$sigma2)

AIC <- c(arima010$aic, arima110$aic, arima011$aic, arima111$aic, arima212$aic,
         arima312$aic)

LOG <- c(arima010$loglik, arima110$loglik, arima011$loglik, arima111$loglik,
         arima212$loglik, arima312$loglik)

rownames <- c('ARIMA(0,1,0)', 'ARIMA(1,1,0)', 'ARIMA(0,1,1)', 'ARIMA(1,1,1)',
              'ARIMA(2,1,2)', 'ARIMA(3,1,2)')

# place the data into a table
tableARIMA <- data.frame(rownames, sigma_2, LOG, AIC)

colnames(tableARIMA) <- c('Model', 'Sigma^2', ' Log Likelihood', 'AIC')
tableARIMA %>% pander(style ='grid',
                      caption='ARIMA Models: Log Likelihood and AIC')
```

Table 2: ARIMA Models: Log Likelihood and AIC

Model	Sigma^2	Log Likelihood	AIC
ARIMA(0,1,0)	0.006817	2825	-5647
ARIMA(1,1,0)	0.005141	3195	-6386
ARIMA(0,1,1)	0.003416	3730	-7456
ARIMA(1,1,1)	0.003416	3730	-7454
ARIMA(2,1,2)	0.003416	3730	-7450
ARIMA(3,1,2)	0.003405	3733	-7453

```
sarima(litecoin_df$Adj_Return, 3,1,2, details = FALSE) # the model with lowest AIC score
```

```
## $fit
##
```

```
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2  constant
##       0.6628  -0.0082  0.0398  -1.6641  0.6641         0
## s.e.  0.1113   0.0234  0.0200   0.1114  0.1115         0
##
## sigma^2 estimated as 0.003405:  log likelihood = 3732.76,  aic = -7451.52
##
## $degrees_of_freedom
## [1] 2621
##
## $ttable
##          Estimate     SE  t.value p.value
## ar1        0.6628 0.1113   5.9542  0.0000
## ar2       -0.0082 0.0234  -0.3504  0.7261
## ar3        0.0398 0.0200   1.9952  0.0461
## ma1       -1.6641 0.1114 -14.9324  0.0000
## ma2        0.6641 0.1115   5.9564  0.0000
## constant   0.0000 0.0000   0.0286  0.9772
##
## $AIC
## [1] -2.836511
##
## $AICc
## [1] -2.836499
##
## $BIC
## [1] -2.82086
```

$$y_t = 0.6628y_{t-1} - 0.0082y_{t-2} + 0.0398_{t-3} - 1.6641\varepsilon_{t-1} + 0.6641\varepsilon_{t-2} + \varepsilon_t$$

where $c = 0$ and ε_t is white noise with a standard deviation of $\sqrt{\sigma^2} = \sqrt{0.003405} = 0.058352$.

Optimal ARIMA Model

```
ltc.arima_opt <- tq_get("LTC-USD", from ="2015-01-01", to = "2021-09-30") %>%
select(adjusted) %>% # adjusted price (more accurate than close price)
ts(.) # turning it into a time series object
crypto_model <- auto.arima(ltc.arima_opt); crypto_model # Optimal ARIMA model
```

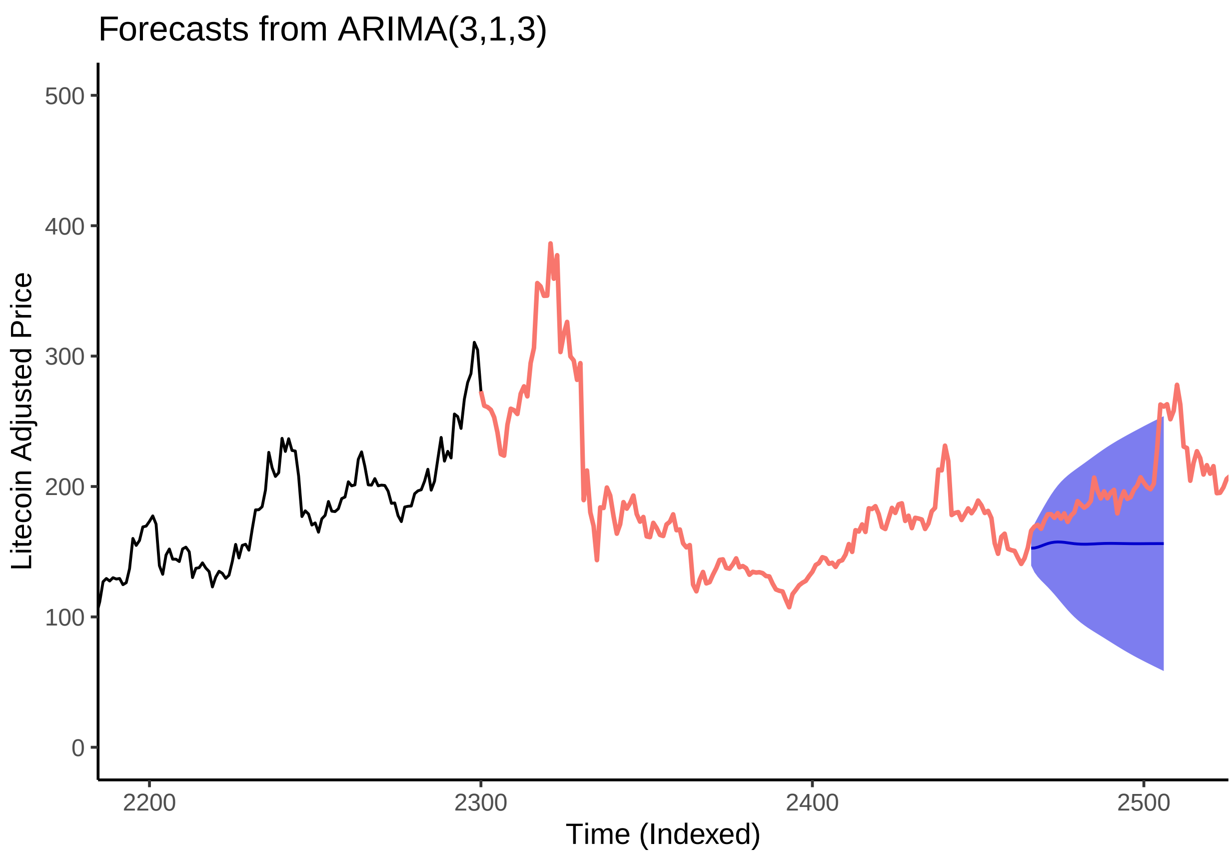
```
## Series: ltc.arima_opt
## ARIMA(3,1,3)
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3
```

```
##        0.6688  0.7767  -0.7404  -0.6954  -0.7546  0.7909
## s.e.   0.5612  0.3431   0.1273   0.5243   0.3831  0.1779
##
## sigma^2 estimated as 46.41:  log likelihood=-8208.97
## AIC=16431.93   AICc=16431.98   BIC=16472.6
```

```r
# forecast the next 41 closing prices, with a 95% CI
ltc_forecast <- forecast(crypto_model, 41, level = c(.95)); ltc_forecast
```

```
##      Point Forecast     Lo 95    Hi 95
## 2466       152.6847 139.33254 166.0369
## 2467       152.8352 134.20225 171.4681
## 2468       153.4791 130.72922 176.2291
## 2469       154.3697 127.92189 180.8175
## 2470       155.3541 125.38883 185.3193
## 2471       156.2272 122.81387 189.6405
## 2472       156.9163 120.12849 193.7040
## 2473       157.3264 117.25122 197.4016
## 2474       157.4894 114.26737 200.7113
## 2475       157.4067 111.20322 203.6101
## 2476       157.1743 108.18792 206.1606
## 2477       156.8340 105.26026 208.4077
## 2478       156.4871 102.52114 210.4531
## 2479       156.1629  99.97227 212.3535
## 2480       155.9287  97.66111 214.1962
## 2481       155.7770  95.54545 216.0086
## 2482       155.7337  93.62908 217.8384
## 2483       155.7605  91.84472 219.6762
## 2484       155.8570  90.17816 221.5358
## 2485       155.9744  88.56290 223.3858
## 2486       156.1080  86.99076 225.2253
## 2487       156.2171  85.41387 227.0203
## 2488       156.3070  83.84153 228.7724
## 2489       156.3528  82.24802 230.4576
## 2490       156.3725  80.65770 232.0873
## 2491       156.3547  79.05945 233.6500
## 2492       156.3242  77.48299 235.1654
## 2493       156.2754  75.92120 236.6296
## 2494       156.2322  74.40000 238.0644
## 2495       156.1880  72.90904 239.4670
## 2496       156.1611  71.46629 240.8559
## 2497       156.1407  70.05638 242.2250
## 2498       156.1389  68.69022 243.5875
## 2499       156.1418  67.34965 244.9339
## 2500       156.1574  66.04200 246.2727
## 2501       156.1714  64.74976 247.5931
## 2502       156.1908  63.48003 248.9015
## 2503       156.2031  62.21833 250.1879
## 2504       156.2160  60.97316 251.4587
## 2505       156.2198  59.73354 252.7060
## 2506       156.2232  58.50920 253.9372
```

```r
# actual prices used for plot below
actual_price <- tq_get("LTC-USD", from = "2015-01-01", to = "2021-11-30") %>%
  select(adjusted) %>% ts(.)
# Plotting forecasted prices against the actual prices
autoplot(ltc_forecast, xlab='Time (Indexed)',ylab=('Litecoin Adjusted Price')) +
  autolayer(window(actual_price, start = 2300), size=0.8) +
  theme_classic() +
  theme(legend.position = "") +
  ylim(0, 500)+
  coord_cartesian(xlim = c(2200,2510))
```



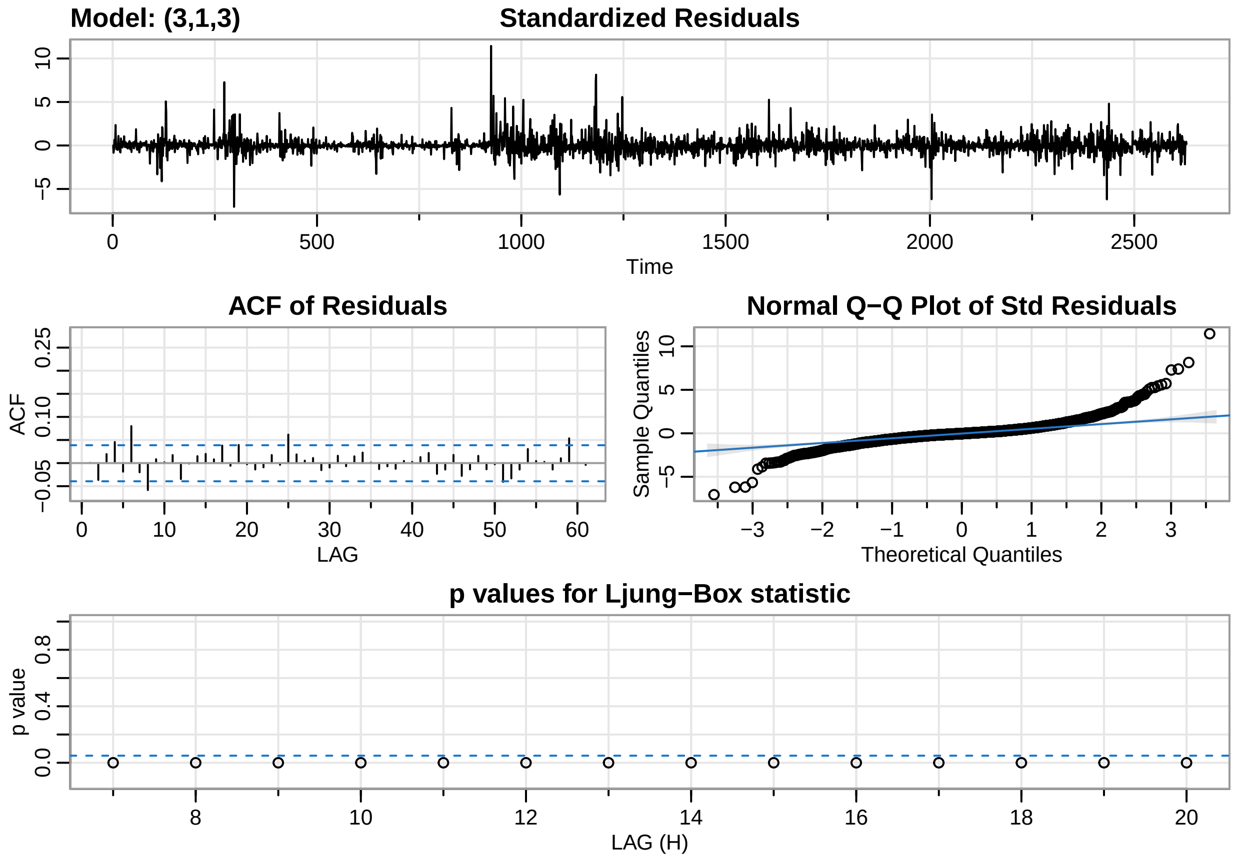
Diagnostics for Optimal ARIMA Model

```r
sarima(litecoin_df$Adj_Return, 3,1,3)
```

```
## initial  value -2.493863
## iter   2 value -2.715988
## iter   3 value -2.764973
## iter   4 value -2.808328
## iter   5 value -2.819179
## iter   6 value -2.820811
## iter   7 value -2.823961
## iter   8 value -2.830462
```

```
## iter   9 value -2.834061
## iter  10 value -2.836192
## iter  11 value -2.837421
## iter  12 value -2.837597
## iter  13 value -2.837726
## iter  14 value -2.837759
## iter  15 value -2.837770
## iter  16 value -2.837810
## iter  17 value -2.837810
## iter  18 value -2.837811
## iter  19 value -2.837844
## iter  20 value -2.837855
## iter  21 value -2.837953
## iter  22 value -2.838398
## iter  23 value -2.838680
## iter  24 value -2.838877
## iter  25 value -2.839033
## iter  26 value -2.839360
## iter  27 value -2.839747
## iter  28 value -2.840184
## iter  29 value -2.840496
## iter  29 value -2.840496
## iter  30 value -2.840544
## iter  30 value -2.840544
## iter  31 value -2.840552
## iter  31 value -2.840552
## iter  31 value -2.840552
## final  value -2.840552
## converged
## initial  value -2.838511
## iter   2 value -2.838563
## iter   3 value -2.838716
## iter   4 value -2.838742
## iter   5 value -2.838783
## iter   6 value -2.838788
## iter   7 value -2.838790
## iter   8 value -2.838792
## iter   9 value -2.838795
## iter  10 value -2.838799
## iter  11 value -2.838800
## iter  12 value -2.838801
## iter  13 value -2.838801
## iter  14 value -2.838802
## iter  15 value -2.838804
## iter  16 value -2.838809
## iter  17 value -2.838809
## iter  18 value -2.838811
## iter  19 value -2.838818
## iter  20 value -2.838818
## iter  21 value -2.838819
## iter  22 value -2.838819
## iter  23 value -2.838824
## iter  24 value -2.838833
## iter  25 value -2.838850
```

```
## iter  26 value -2.838878
## iter  27 value -2.838972
## iter  28 value -2.839057
## iter  29 value -2.839191
## iter  30 value -2.839218
## iter  31 value -2.839286
## iter  32 value -2.839286
## iter  33 value -2.839286
## iter  34 value -2.839291
## iter  35 value -2.839292
## iter  36 value -2.839292
## iter  37 value -2.839294
## iter  38 value -2.839297
## iter  39 value -2.839305
## iter  40 value -2.839314
## iter  41 value -2.839324
## iter  42 value -2.839329
## iter  43 value -2.839329
## iter  43 value -2.839329
## iter  43 value -2.839329
## final  value -2.839329
## converged
```




```
## $fit
##
```

```
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3      ma1      ma2     ma3  constant
##       -0.1813  0.5661  -0.0023  -0.8161  -0.7215  0.5399         0
## s.e.   0.2028  0.1715   0.0203   0.2014   0.1493  0.1709         0
##
## sigma^2 estimated as 0.003412:  log likelihood = 3731.37,  aic = -7446.73
##
## $degrees_of_freedom
## [1] 2620
##
## $ttable
##          Estimate     SE t.value p.value
## ar1       -0.1813 0.2028 -0.8941  0.3713
## ar2        0.5661 0.1715  3.3019  0.0010
## ar3       -0.0023 0.0203 -0.1111  0.9116
## ma1       -0.8161 0.2014 -4.0531  0.0001
## ma2       -0.7215 0.1493 -4.8315  0.0000
## ma3        0.5399 0.1709  3.1597  0.0016
## constant   0.0000 0.0000  0.0777  0.9381
##
## $AIC
## [1] -2.834691
##
## $AICc
## [1] -2.834675
##
## $BIC
## [1] -2.816804
```

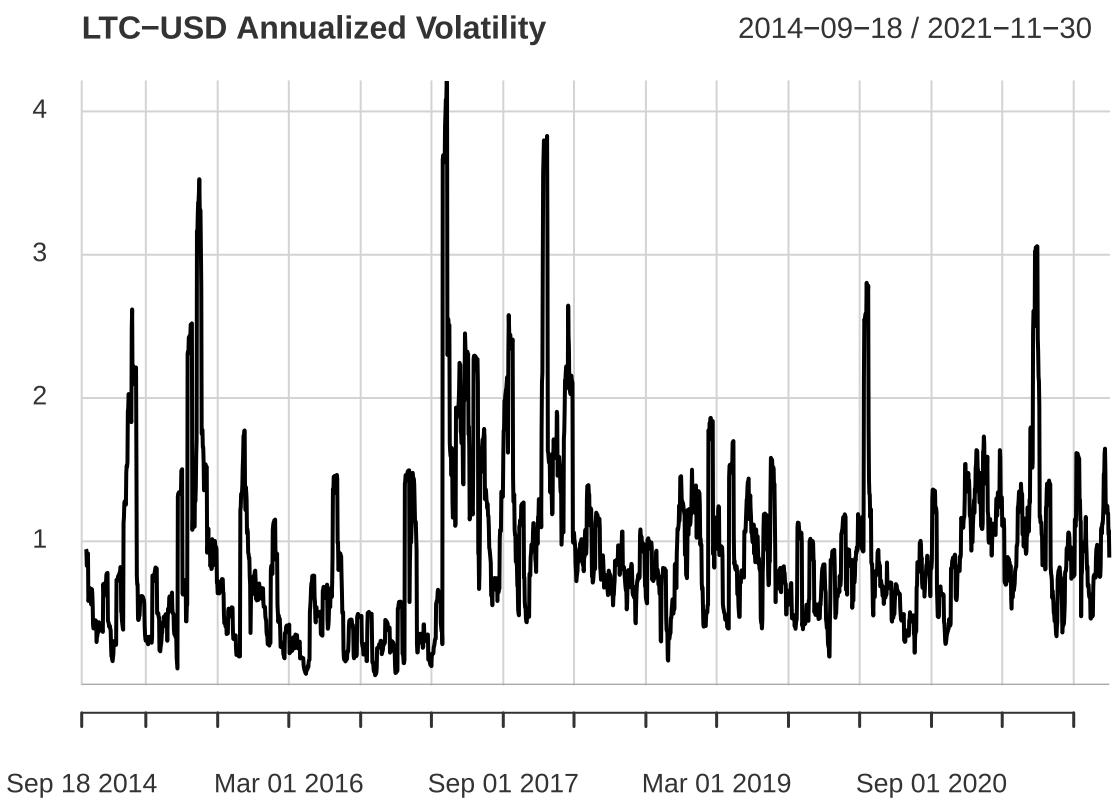
$$y_t \quad = \quad -0.1813y_{t-1} + 0.5661y_{t-2} - -0.0023y_{t-3} - 0.8161\varepsilon_{t-1} - 0.7215\varepsilon_{t-2} + 0.5399\varepsilon_{t-3} + \varepsilon_t$$

where $c = 0$ and ε_t is white noise with a standard deviation of $\sqrt{\sigma^2} = \sqrt{0.003412} = 0.05841233$.

- Standard Residuals: trend-less and white noise-like.
- ACF of Residuals: cuts off after lag 1 indicating its MA behavior.
- Normal Q-Q Plot of Std Residuals: assumption of normality is reasonable w/ some outliers at the tails.
- The p-values for Ljung-Box statistic: all p-values are under 0.0, indicating Q-Statistic is insignificant which means our model may fit really nicely.

Calculate Annualized Volatility

```
return = CalculateReturns(ltc_xts$Adjusted)
return = return[-1,]
chart.RollingPerformance(R = return, FUN="sd.annualized", scale=365, width=12,
                         main="LTC-USD Annualized Volatility")
```

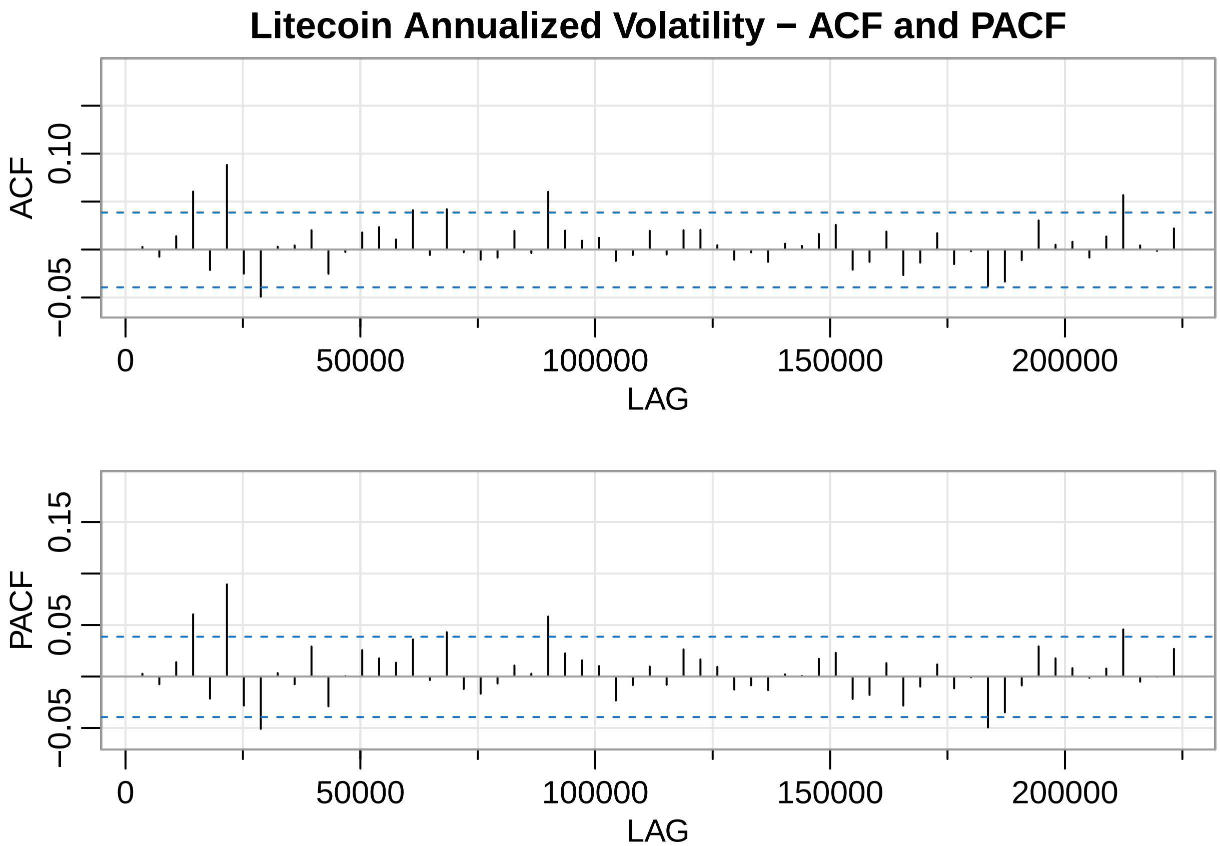



```
volatility <- sd(return)
rolling_window <- sqrt(365)*sd(return["2021"])

rownames <- c('Metric')

table_vol<- data.frame(rownames, volatility, rolling_window)
colnames(table_vol)<-c(' ','Annualized Volatility', 'Rolling Window Volatility')
table_vol %>% pander(style ='grid', caption='Litecoin Volatility of Return')
```

Table 3: Litecoin Volatility of Return

	Annualized Volatility	Rolling Window Volatility
Metric	0.05838	1.189

```r
acf2(return, main='Litecoin Annualized Volatility - ACF and PACF')
```




```
##       [,1]  [,2] [,3] [,4]  [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF     0 -0.01 0.01 0.06 -0.02 0.09 -0.03 -0.05    0  0.00  0.02 -0.03     0
## PACF    0 -0.01 0.01 0.06 -0.02 0.09 -0.03 -0.05    0 -0.01  0.03 -0.03     0
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF    0.02  0.02  0.01  0.04 -0.01  0.04  0.00 -0.01 -0.01  0.02     0  0.06
## PACF   0.03  0.02  0.01  0.04  0.00  0.04 -0.01 -0.02 -0.01  0.01     0  0.06
##       [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF    0.02  0.01  0.01 -0.01 -0.01  0.02 -0.01  0.02  0.02  0.00 -0.01  0.00
## PACF   0.02  0.02  0.01 -0.02 -0.01  0.01 -0.01  0.03  0.02  0.01 -0.01 -0.01
##       [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49]
## ACF   -0.01  0.01     0  0.02  0.03 -0.02 -0.01  0.02 -0.03 -0.01  0.02 -0.02
## PACF  -0.01  0.00     0  0.02  0.02 -0.02 -0.02  0.01 -0.03 -0.01  0.01 -0.01
##       [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61]
## ACF       0 -0.04 -0.03 -0.01  0.03  0.01  0.01 -0.01  0.01  0.06  0.00     0
## PACF      0 -0.05 -0.03 -0.01  0.03  0.02  0.01  0.00  0.01  0.05 -0.01     0
##       [,62]
## ACF    0.02
## PACF   0.03
```

From the graph above, we can see that the annualized volatility is throughout the entire history of existence of LTC, with various magnitudes through different months. This leading to GARCH/conditional volatility.

GARCH Model

```r
model1 = ugarchspec(mean.model = list(armaorder = c(0,0)),
                    variance.model = list(model = "sGARCH", garchorder=c(1,1)),
                    distribution.model='sstd')

################################################
# model fitting
model_fitting=ugarchfit(data = return, spec = model1, out.sample=20)
model_fitting
```

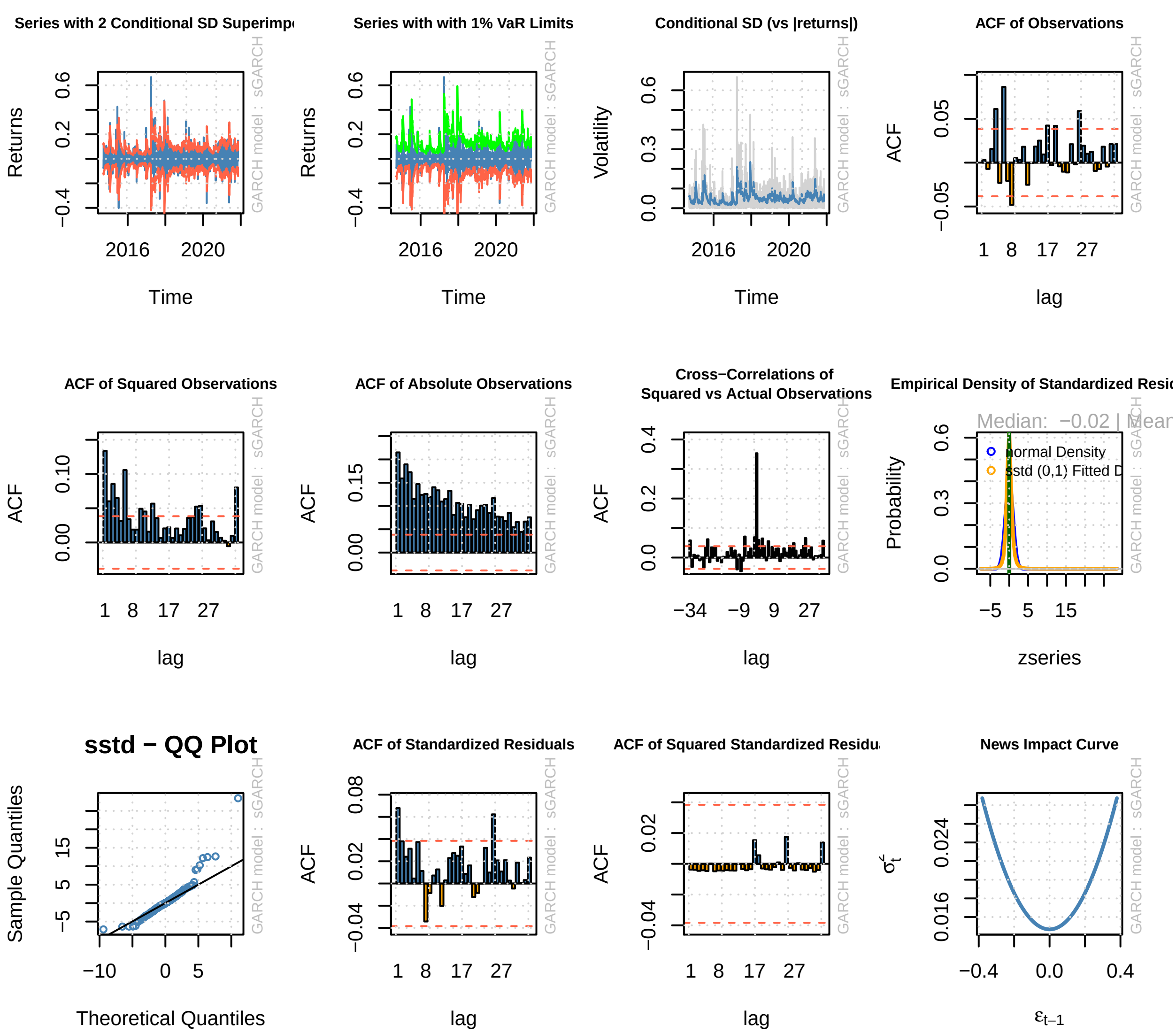
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000735    0.000524   1.4027 0.160717
## ar1     0.418107    0.157935   2.6473 0.008113
## ma1    -0.497210    0.150026  -3.3142 0.000919
## omega   0.000016    0.000007   2.2844 0.022348
## alpha1  0.097471    0.011706   8.3264 0.000000
## beta1   0.901529    0.013195  68.3219 0.000000
## skew    1.067920    0.023746  44.9724 0.000000
## shape   3.018354    0.128460  23.4965 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000735    0.000551   1.3349 0.181904
## ar1     0.418107    0.166875   2.5055 0.012228
## ma1    -0.497210    0.158516  -3.1367 0.001709
## omega   0.000016    0.000013   1.2280 0.219459
## alpha1  0.097471    0.015074   6.4663 0.000000
## beta1   0.901529    0.023431  38.4755 0.000000
## skew    1.067920    0.023843  44.7894 0.000000
## shape   3.018354    0.146389  20.6187 0.000000
##
## LogLikelihood : 4577.366
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.5055
## Bayes        -3.4875
```

```
## Shibata      -3.5055
## Hannan-Quinn -3.4989
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic       p-value
## Lag[1]                       12.05 0.00051805882
## Lag[2*(p+q)+(p+q)-1][5]      17.01 0.00000000000
## Lag[4*(p+q)+(p+q)-1][9]      20.75 0.00000002785
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.03717  0.8471
## Lag[2*(p+q)+(p+q)-1][5]   0.13202  0.9967
## Lag[4*(p+q)+(p+q)-1][9]   0.21580  0.9999
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.05465 0.500 2.000  0.8152
## ARCH Lag[5]   0.12590 1.440 1.667  0.9818
## ARCH Lag[7]   0.16175 2.315 1.543  0.9982
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  19.2766
## Individual Statistics:
## mu     0.11840
## ar1    0.07957
## ma1    0.08344
## omega  1.70341
## alpha1 1.67979
## beta1  2.04314
## skew   0.12408
## shape  6.02957
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.4538 0.1461
## Negative Sign Bias  0.2108 0.8330
## Positive Sign Bias  1.2293 0.2191
## Joint Effect        3.3001 0.3476
##
##
## Adjusted Pearson Goodness-of-Fit Test:
```

```
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     37.41     0.007046
## 2    30     54.32     0.002969
## 3    40     56.28     0.036107
## 4    50     69.24     0.029979
##
##
## Elapsed time : 1.080804
```

```r
################################################

# plot
plot(model_fitting,which="all")
```

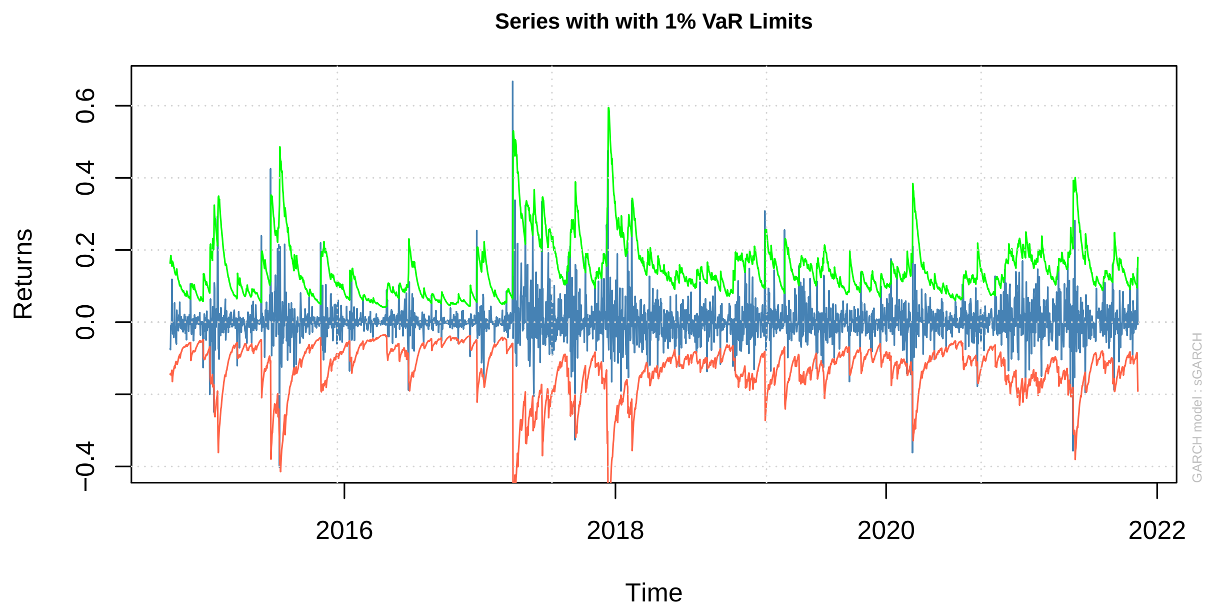
```
##
## please wait...calculating quantiles...
```

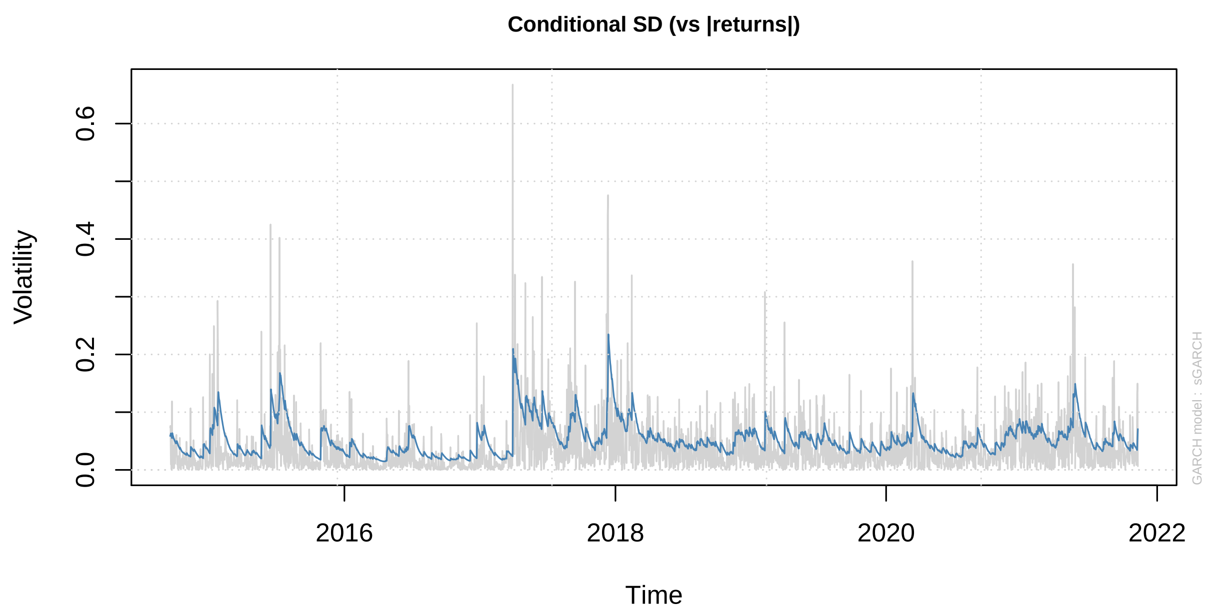


```
par(mfrow=c(2,1),
    oma = c(1.5,0.5,0,0) + 0.10,
    mar = c(5,4,3,2) -0.10)
plot(model_fitting, which=2)
```

```
##
## please wait...calculating quantiles...
```

```
plot(model_fitting, which=3)
```





References

Bischoff, B. & Cockerham, R. (2019, March 31). Adjusted Closing Price vs. Closing Price. *Zacks.* https://finance.zacks.com/adjusted-closing-price-vs-closing-price-9991.html

Kuhn, M., & Johnson, K. (2016). *Applied Predictive Modeling.* Springer. https://doi.org/10.1007/978-1-4614-6849-3

Shumway, R., & Stoffer, D. (2019). *Time Series: A Data Analysis Approach Using R.* Chapman and Hall/CRC.