Bitcoin Time Series

Group Kappa

3/8/2018

Abstract

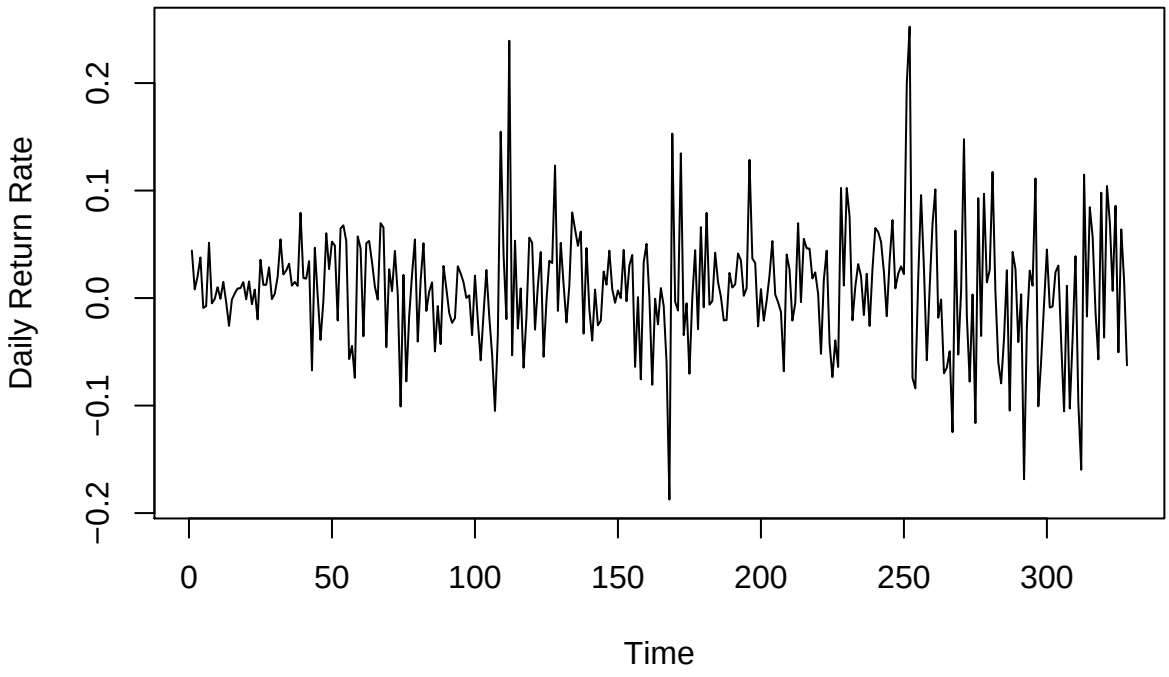
This time series project is based on the price of bitcoins on the market starting from late March of 2017 to the end of February of 2018 and using that data to build a suitable model from where we can create forecasts from. To conduct this project, we downloaded this data from Kaggle, cleaned out unnecessary columns, and created a new column containing the daily return rate of bitcoin prices. Based off of a visual observation of the time series, we proceed to apply relevant transformations like box-cox or differencing before proceeding to decide a model. Based off of the auto.arima function as well as analysis of the PACF, we came up with a list of models before ultimately deciding that white noise was our best choice. We conducted the necessary diagnostic checks like normality of residuals, Box-Ljung, Shapiro Wilks, and so forth to make sure our models were appropriate before going ahead and plotting out the next 100 days of daily return rates of bitcoin.

Introduction

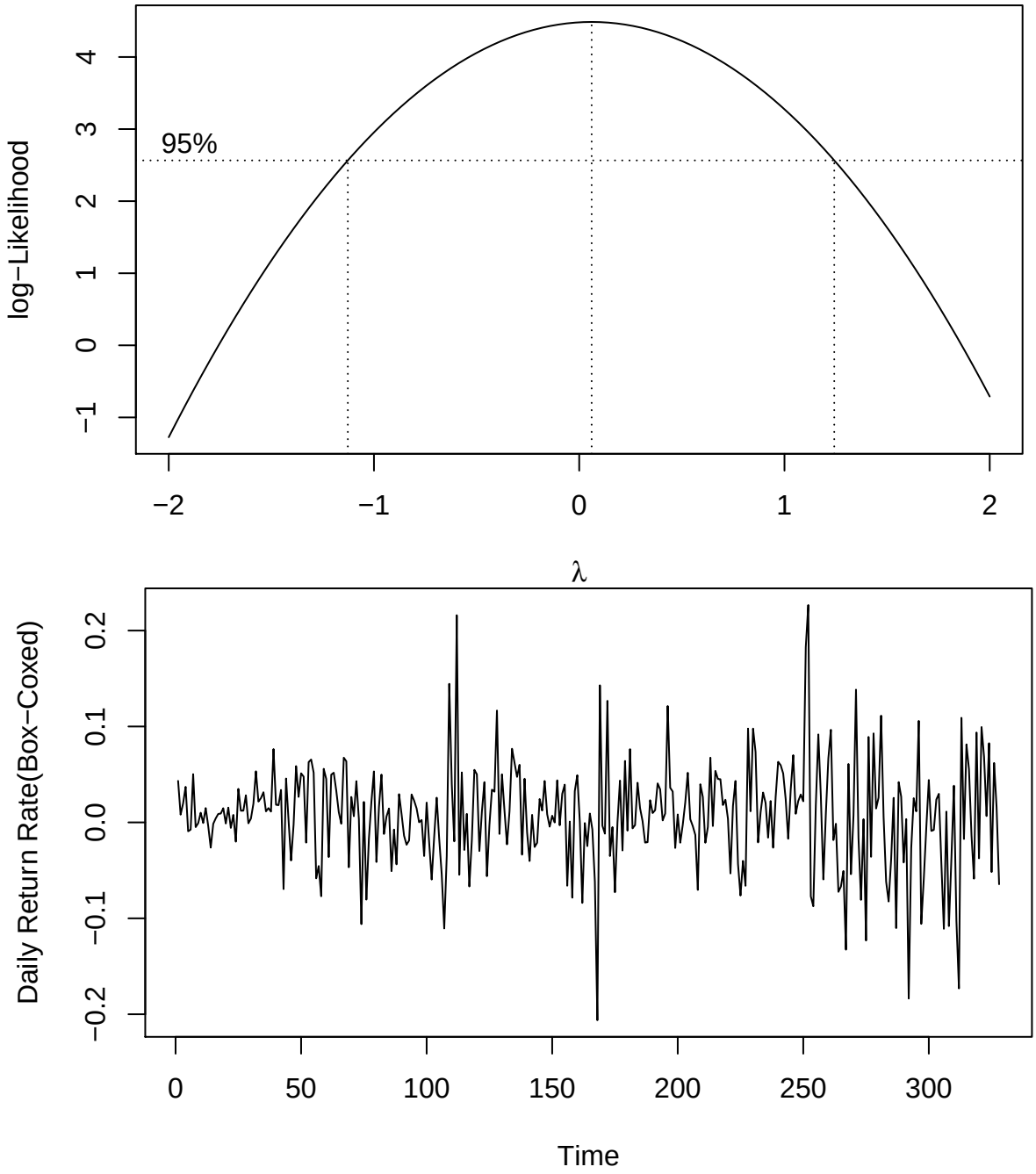
Our main focus for this project was to forecast the daily return of the cryptocurrency Bitcoin. We originally started with a very large dataset including many variables along with most cryptocurrency, not just Bitcoin. We found this dataset to be very interesting considering there has been a sharp rise of interest in cryptocurrency in the past few years. Although analyzing this dataset appeared to be much more difficult than choosing one of the clear cut time series from the links provided, we thought it was worth the challenge considering how fascinating the Bitcoin craze is. Forecasting future daily returns on Bitcoin could be valuable information to any investor. To start, we created a new dataset that filtered out all variables besides "Date" and "Close" which refers to the day and closing price respectively. We also removed all cryptocurrency besides Bitcoin from the dataset since that was our focus as well as cutting off values found before 3/31/2017 since before this date the values were very stagnant. We originally averaged the closing price of every seven days representing a weekly closing price, but realized that we would be losing the volatility of the series and it would smooth the graph out too much. Next, we tried a day-by-day closing price time series but it wasn't informative enough. We finally decided on using the daily return rate for the time series which is calculated by subtracting the previous day's closing price from the current then dividing by the previous day's closing. We also needed to add one to all the return values since positive values are needed for box cox transformations. This proved to be the most informative and we then moved on to transforming the data. Due to the nature of the dataset, it was very difficult to find a concrete model even after transformations. We decided the best model to be a white noise model and forecasted our findings with a 95% confidence interval. Our data set came from https://www.kaggle.com/jessevent/all-crypto-currencies/data. All statistical analysis was done with the software RStudio which is an environment for the coding language R.

Analysis

Exploratory Data Analysis



Data Transformations

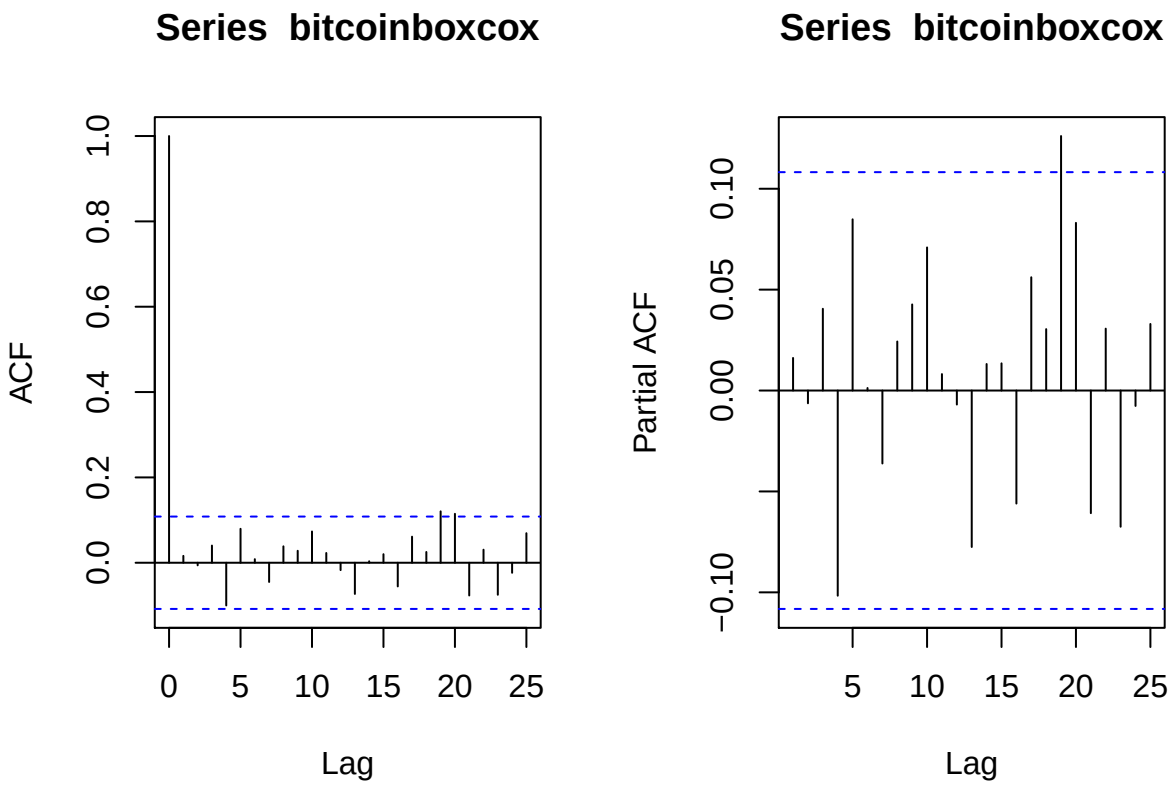


We utilize the box-cox transformation on our data in order to ensure that our data points follow a normal distribution. This enables us to preemptively fulfill one of our assumptions as well as helping to transform the model into a stationary series before we go ahead applying a model onto our data. The box-cox procedure works by identifying an optimal transformation variable labeled lambda that corresponds to the maximum log likelihood of the series. Since our lambda value was neither 0 or 0.5, we recognized that the ideal transformation in this case was neither the square root nor the log of the data.

Differentiation is a way to get rid of the trend of the time series. By definition, we need to differentiate the time series until its variance become smaller and no longer increases which convert the time series to a stationary one. We decided not to differentiate the time series because even if we only differentiate once,

its variance increased. We can know our time series is already stationary by the increasing variance when differencing. So we decided not to difference for our models. Additionally, the lack of seasonality seen in our time series plot, acf, and pacf was another solid indicator that seasonality was not a factor in our data. Note however that this observed trend can also be simply attributed to the fact that our data set did not span a long enough period of time for us to pick up such information.

ACF and PACF Analysis



ACF stands for Autocovariance Function, and PACF stands for Partial Autocovariance Function. As we see from the ACF, there's an obvious cut off in the graph, which indicates that the time series could be a MA model. Since the the ACF cuts off at lag 0, the time series turns out to be a white noise model. For the PACF, we found that the pacf values are really small for most of the lags. However, at lag 19, it seems significant as it surpasses the significance boundaries. From this we determined another potential model of AR(19).

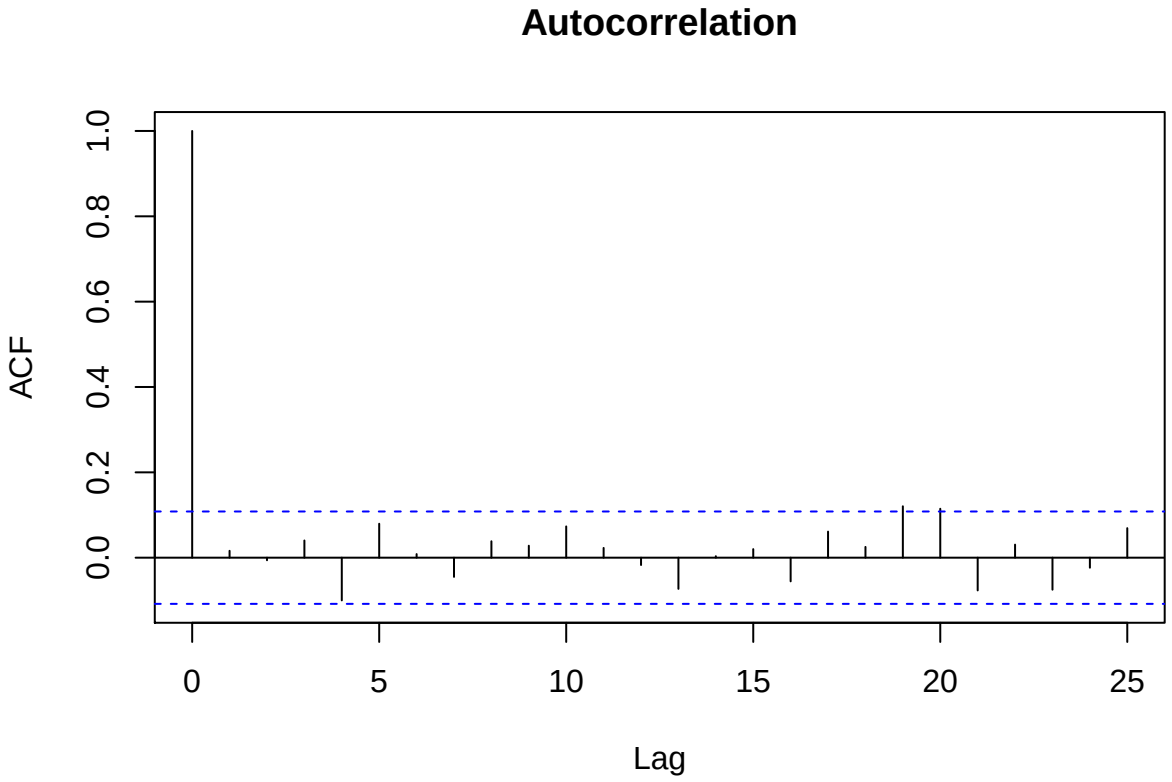
We use both of the models in the following steps in order to find out which is the best one.

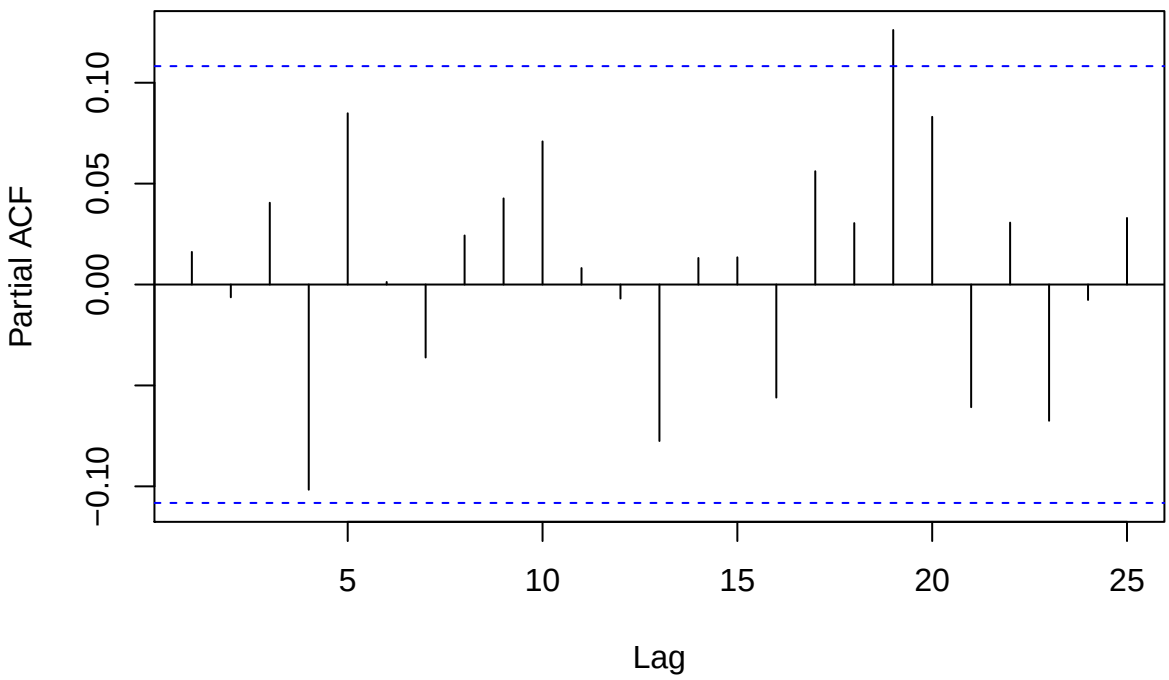
We had to choose from the two models, AR(19) and our White Noise model. Usually when using the AR models we tend not to have such high kth order autoregressions due to the many different variables/lags that these high-order models have. When choosing the right model we always want to follow parsimony and choose the one that provides us the simplest path to receiving the correct results needed in order to find the significance of our data. One method that we used to determine which model to choose was using the AIC(c) values. When using the AIC, you want to choose the smallest value. For our white noise model we have: -972.96 and for our AR(19) model the value is -954.36. Thus using AIC guidelines, the white noise model is the better choice to use.

Model Diagnostics

White Noise Model (Model 1)

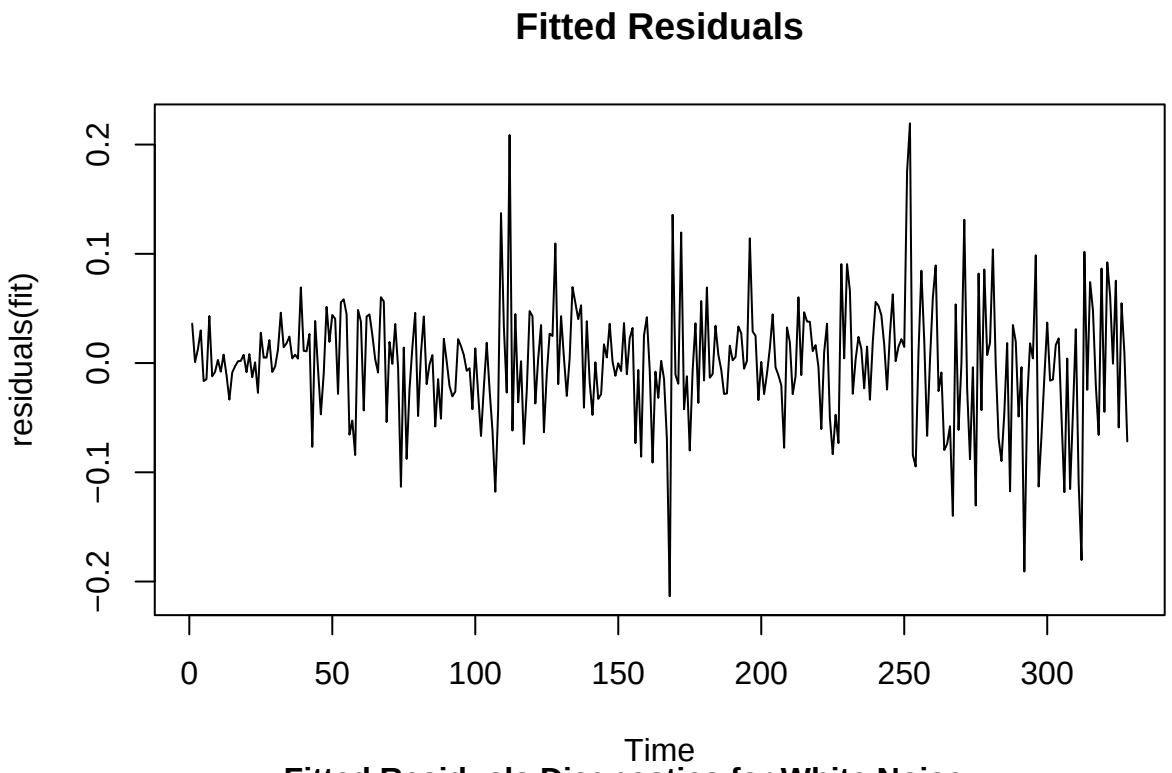
```
##
## Call:
## arima(x = bitcoinboxcox, order = c(0, 0, 0), method = "ML")
##
## Coefficients:
##       intercept
##          0.0072
## s.e.     0.0030
##
## sigma^2 estimated as 0.002978:  log likelihood = 488.48,  aic = -972.96
```

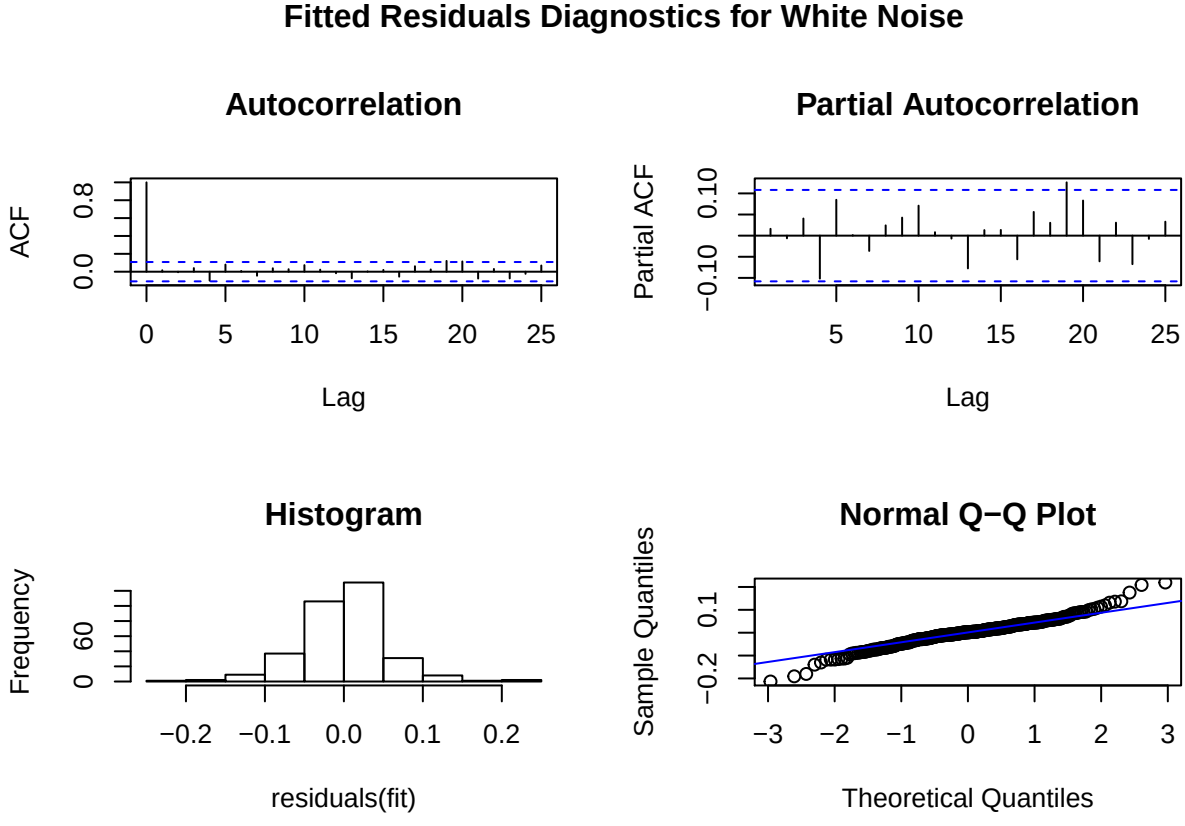





```
##
##  Box-Ljung test
##
## data:  resid(fit)
## X-squared = 0.086284, df = 1, p-value = 0.769
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fit)
## W = 0.96744, p-value = 1.004e-06
```

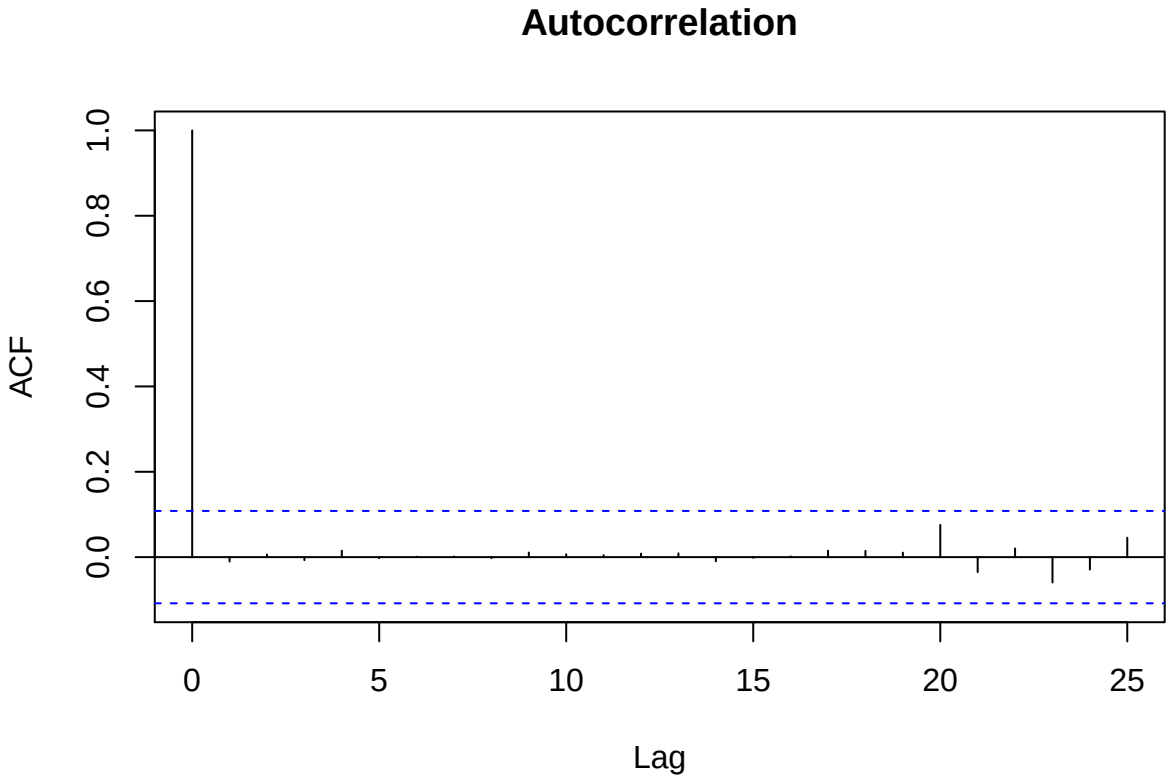

Fitted Residuals
residuals(fit)
0.2
0.1
0.0
−0.1
−0.2
0
50
100
150
200
250
300
Time

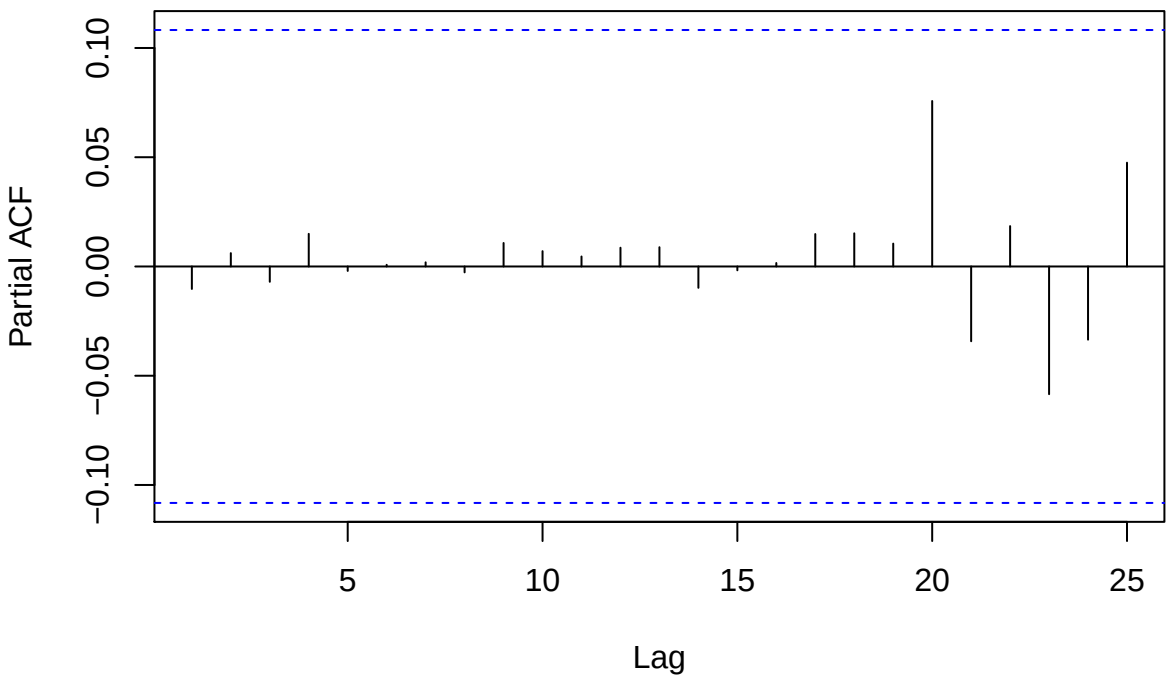

Fitted Residuals Diagnostics for White Noise
Autocorrelation
ACF
0.8
0.0
0
5
10
15
20
25
Lag
Partial Autocorrelation
Partial ACF
0.10
−0.10
5
10
15
20
25
Lag
Histogram
Frequency
60
0
−0.2
−0.1
0.0
0.1
0.2
residuals(fit)
Normal Q−Q Plot
Sample Quantiles
0.1
−0.2
−3
−2
−1
0
1
2
3
Theoretical Quantiles

AR(19) (Model 2)

##

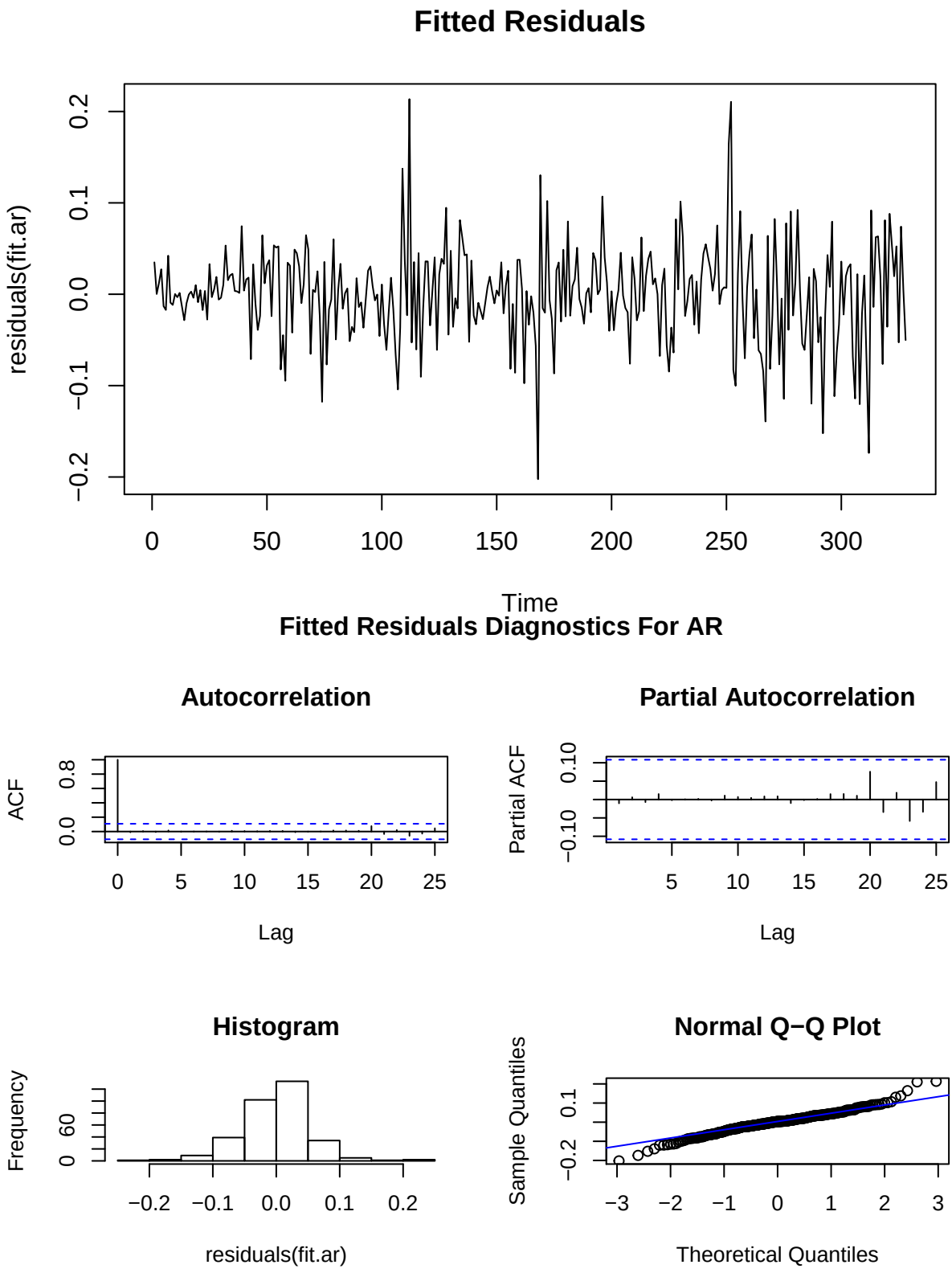
```
## Call:
## arima(x = bitcoinboxcox, order = c(19, 0, 0), method = "ML")
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5     ar6      ar7     ar8
##       0.0243  -0.0116  0.0479  -0.0996  0.0815  0.0188  -0.0404  0.0259
## s.e.  0.0547   0.0547  0.0546   0.0546  0.0548  0.0550   0.0549  0.0552
##          ar9    ar10    ar11     ar12     ar13     ar14    ar15     ar16
##       0.0135  0.0708  0.0096  -0.0104  -0.0689  -0.0004  0.0258  -0.0644
## s.e.  0.0551  0.0550  0.0553   0.0552   0.0551   0.0554  0.0551   0.0552
##         ar17    ar18    ar19  intercept
##       0.0598  0.0220  0.1349     0.0073
## s.e.  0.0563  0.0566  0.0565     0.0038
##
## sigma^2 estimated as 0.002801:  log likelihood = 498.18,  aic = -954.36
```






```
##
##  Box-Ljung test
##
## data:  resid(fit.ar)
## X-squared = 0.035084, df = 1, p-value = 0.8514
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fit.ar)
## W = 0.97412, p-value = 1.271e-05
```

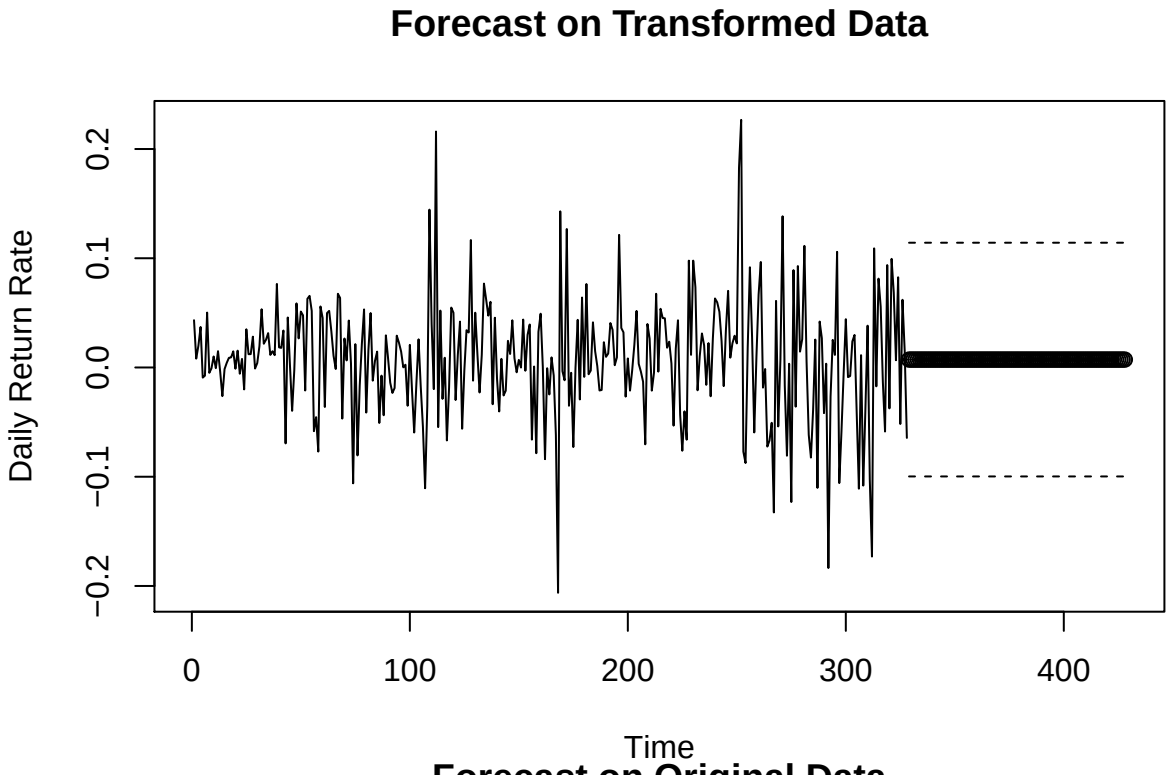


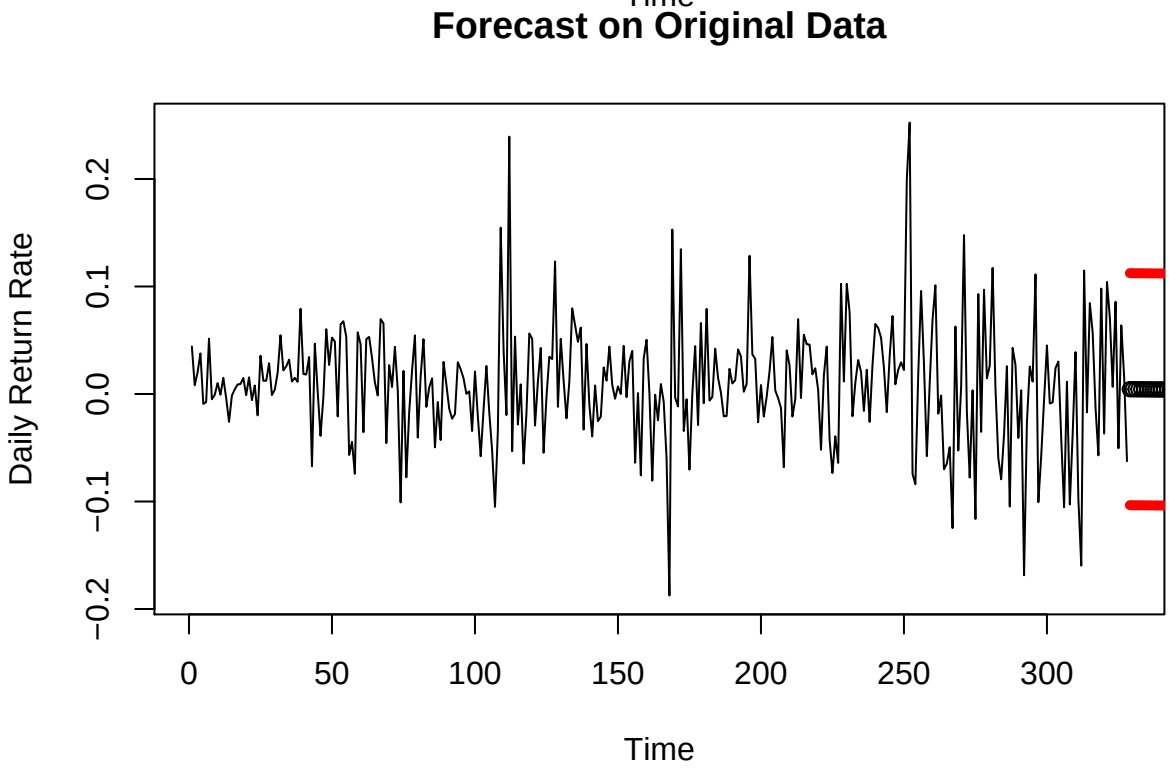
Diagnostics For White Noise Model: Diagnostic checking aims to test the adequacy of the model representing the actual data. A white noise process is one with a mean zero and no correlation between its values at different times. The residuals of the model do not show any distinct pattern or change in variability and therefore can succinctly conclude as a white noise process. Since this a white noise model we could not

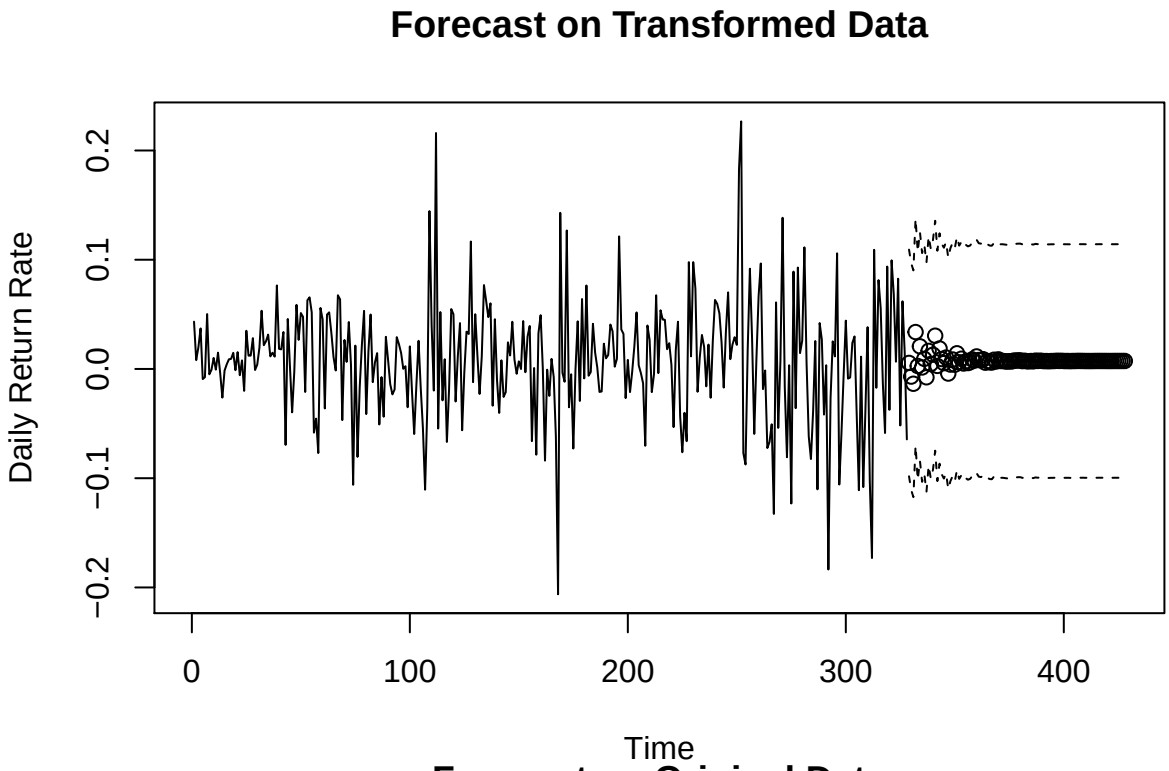
properly use an AIC(c). When looking at our model, we wanted to ensure the validity of our work, thus we used different tests to check for normality and fit. We used the Shapiro-Wilk Test and the Box- Ljung Test. With the Shapiro-Wilk Test we found that the p-value to be 1.004e-06 which is incredibly small. Since the p-value is far less than 0.05 we can conclude that the data does not pass the normality test. However when we ran the Box-Ljung test, we attained a p-value of 0.769 which is greater than the 0.05 needed to reject the null hypothesis and accept the idea that our data points were not correlated to each other. We also used a histogram that we can see fits the normal distribution form. When we analyze the quantile-quantile (QQ) plot, the majority of the data fits within the normality of straight line. We do notice that each side of the plot has a heavy-tail which means there are likely to be residuals that are much larger (or much smaller) than we would expect from a normal distribution. Since both sides are tailed off like this, it helps assume normality since most of the data tends to fit the linear trend.

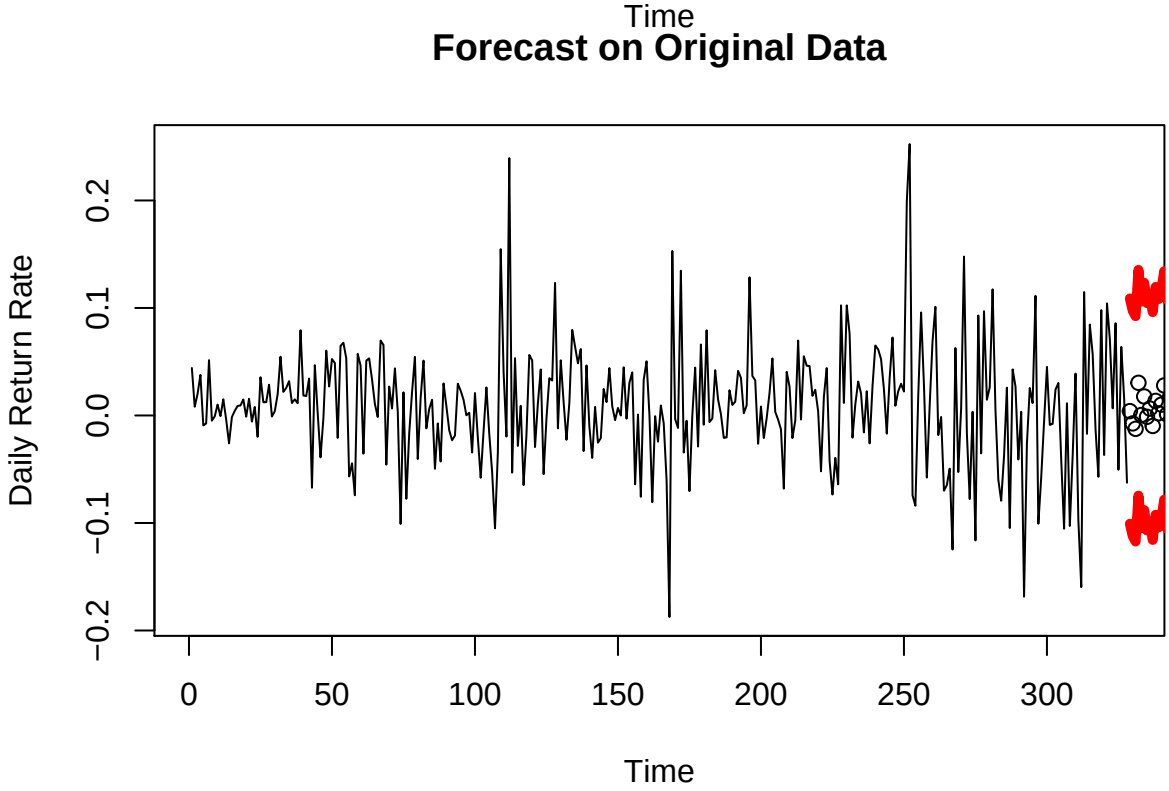
Diagnostics For AR(19) Model: For the most part the residuals do not show any surprising pattern. When observing cryptocurrencies, it is imperative to take into consideration that extreme volatility that these currencies have. The peaks and valleys seen in the residuals are nothing uncommon that we have seen in Bitcoin. We really do not see any trend, seasonality, or change in variance. When looking at our model, we wanted to ensure the validity of our work, thus we used different tests to check for normality and fit. We used the Shapiro-Wilk Test and the Box- Ljung Test. For the Shapiro- Wilk test we got 1.271e-05. Just like our previous model, this p-value is extremely small and does not pass as a check that we needed for normality. When we used the Box-Ljung test, we were able to obtain a p-value that is 0.8514 which is greater than the 0.05 needed to reject the null hypothesis and accept the alternative hypothesis that our data is not correlated with each other. We also used a histogram that we can see fits the normal distribution form. When we analyze the quantile-quantile (QQ) plot, the majority of the data fits within the normality of the linear line. We do notice that each side of the plot has a heavy-tail which means there are likely to be residuals that are much larger (or much smaller) than we would expect from a normal distribution. Since both sides are tailed off like this, it helps to assume normality since most of the data tends to fit the linear trend.

Forecasting









Now that we have successfully identified our ideal model for the time series, we can proceed to forecasting data points. In this report we provide the prediction plots for both of our selected models. We choose to forecast the daily return rate for the next 100 days (roughly 3 months). However, from our our model selection, we have determined that white noise was the best choice in predicting our model. Obviously, this is not ideal as our forecasts essentially tell us that the future daily return of bitcoin is completely random. There is no predicted trend, and daily return continues to center around 0.

Conclusion

From the results of this project, we achieve two tasks: To build a proper time series model for our data, and then use that model to start creating accurate forecasts for the daily return rate of bitcoins. We performed transformations to make our data stationary, follow a normal distribution, as well as to make sure variance remained constant throughout the time series. We followed that up with applying models of white noise and AR(19), backing up our assumptions with diagnostic checks such as the Shapiro Wilks Test, observing residuals, and the Box-Ljung Tests, and then choosing the most ideal model based off the information extracted. Using this sound backing, we could proceed to create a forecast on bitcoin daily returns with a 95% confidence interval. The only result in question is really how valid our model and forecasts were. While we followed the theory behind ARIMA modeling for time series, it seems that our data set was more or less the problem. Since bitcoins have only existed for 5 years, and have only started to grow at a significant rate in the last 2, we were forced to rely on a daily time series as any other metric did not provide enough data points for meaningful analysis. From some research online, we realized that daily time series do not necessarily follow traditional ARIMA modeling rules. In fact, the forecast function in the R forecast library used a model that wasn't even ARIMA! Regardless, our obtained result is still insightful as it does tell us that the market for bitcoin is very volatile and difficult to predict. Even when looking at the time series plot for daily return, we see high positive return and low negative return, an indicator for large swings in pricing. This can be reaffirmed by taking a look at the recent bitcoin prices as well. Starting from the great crash of bitcoins earlier this year, the price of bitcoin has since been in flux, a far cry from its earlier days where it seemed as if it could only keep increasing in price.

References

*https://www.kaggle.com/jessevent/all-crypto-currencies/data

*RStudio

*PSTAT 174 Winter 2018 Lecture Slides and Labs

Appendix

```
library(tidyverse)
library(ggplot2)
library(MASS)
library(qpcR)
library(forecast)
#For macs download https://www.xquartz.org/
library(tseries)

##Setup

#Reading in data
data <- read.csv("crypto.csv", header=TRUE)

#Choosing only Bitcoin
newdata <- subset(data, symbol=="BTC")

#Choosing the two variables
myvars <- c("date","close")
crypto <- newdata[myvars]
```

```r
#Changing class of "date"
crypto$date <- as.Date(crypto$date)

#Cutting off dates before 4/1/17
bitcoin <- subset(crypto, date > "2017-03-29")

#Creating the return variable
r.bitcoin = (bitcoin[2:nrow(bitcoin),2] / bitcoin[1:(nrow(bitcoin)-1),2]-1) + 1

#Creating the return TS
mydata <- bitcoin[-c(1),]
r.bitcoinTS = data.frame(mydata[1],r.bitcoin)

#BoxCox Transform
time <- 1:length(r.bitcoin)
fit <- lm(r.bitcoin ~ time)
boxcoxtransform <- boxcox(r.bitcoin ~ time, plotit = T)
lamb <- boxcoxtransform$x[which(boxcoxtransform$y == max(boxcoxtransform$y))]
bitcoinboxcox <- (1/lamb)*(r.bitcoin^lamb-1)
ts.plot(bitcoinboxcox)

#BoxCox Transform for Dataset
time <- 1:length(r.bitcoinTS[,2])
fit <- lm(r.bitcoinTS[,2] ~ time)
boxcoxtransform <- boxcox(r.bitcoinTS[,2] ~ time, plotit = T)
lamb <- boxcoxtransform$x[which(boxcoxtransform$y == max(boxcoxtransform$y))]
bitcoinboxcoxTS <- (1/lamb)*(r.bitcoinTS[,2]^lamb-1)
ts.plot(bitcoinboxcoxTS)

#PLotting ACF/PACF
op <- par(mfrow=c(2,1))
acf(bitcoinboxcox)
pacf(bitcoinboxcox)
par(op)

##Diagnostics

#MA(1) Model
fit = arima(bitcoinboxcox, order=c(0,0,0), method="ML")

#Testing for independence of residuals
Box.test(resid(fit), type="Ljung")

#Test for normality of residuals
shapiro.test(residuals(fit))

#Plotting Residuals of Fit
ts.plot(residuals(fit),main = "Fitted Residuals")


par(mfrow=c(1,2),oma=c(0,0,2,0))
# Plot diagnostics of residuals
op <- par(mfrow=c(2,2))
```

```r
# acf
acf(residuals(fit),main = "Autocorrelation")
# pacf
pacf(residuals(fit),main = "Partial Autocorrelation")
# Histogram
hist(residuals(fit),main = "Histogram")
# q-q plot
qqnorm(residuals(fit))
qqline(residuals(fit),col ="blue")
# Add overall title
title("Fitted Residuals Diagnostics", outer=TRUE)
par(op)

##Forecasting on bitcoinboxcox dataset
mypred = predict(fit, n.ahead=10)
ts.plot(bitcoinboxcox, xlim=c(0,339))
points(x = 329:338, y = mypred$pred)
lines(329:338,mypred$pred+1.96*mypred$se,lty=2)
lines(329:338,mypred$pred-1.96*mypred$se,lty=2)


##Forecasting on bitcoinboxcox dataset with 100
mypred = predict(fit, n.ahead=100)
ts.plot(bitcoinboxcox, xlim=c(0,429))
points(x = 329:428, y = mypred$pred)
lines(329:428,mypred$pred+1.96*mypred$se,lty=2)
lines(329:428,mypred$pred-1.96*mypred$se,lty=2)

#Forecasting on original
unboxcoxed<-((bitcoinboxcox*lamb)+1)^(1/lamb)
fitma1unboxcoxed<-arima(unboxcoxed,order=c(0,0,0),method="ML",xreg=1:length(unboxcoxed))
predtrans1<-predict(fitma1unboxcoxed,n.ahead=100,newxreg=(length(unboxcoxed)+1)
                    :(length(unboxcoxed)+100))
ltrans1<-predtrans1$pred-1.96*predtrans1$se
ltrans1

utrans1<-predtrans1$pred+1.96*predtrans1$se
utrans1

ts.plot(r.bitcoin)
points(x = 329:428,predtrans1$pred,pch=1)
lines(329:428,ltrans1,col="red",lwd=5)
lines(329:428,utrans1,col="red",lwd=5)


#AR(18) Model
fit.ar = arima(bitcoinboxcox, order=c(19,0,0), method="ML")

par(mfrow=c(1,2),oma=c(0,0,2,0))
# Plot diagnostics of residuals
op <- par(mfrow=c(2,2))
# acf
acf(residuals(fit.ar),main = "Autocorrelation")
```

```r
# pacf
pacf(residuals(fit.ar),main = "Partial Autocorrelation")
# Histogram
hist(residuals(fit.ar),main = "Histogram")
# q-q plot
qqnorm(residuals(fit.ar))
qqline(residuals(fit.ar),col ="blue")
# Add overall title
title("Fitted Residuals Diagnostics For AR", outer=TRUE)
par(op)

#Testing for independence of residuals
Box.test(resid(fit.ar), type="Ljung")

#Test for normality of residuals
shapiro.test(residuals(fit.ar))

##Forecasting AR Model on bitcoinboxcox dataset with 100
mypred = predict(fit.ar, n.ahead=100)
ts.plot(bitcoinboxcox, xlim=c(0,429))
points(x = 329:428, y = mypred$pred)
lines(329:428,mypred$pred+1.96*mypred$se,lty=2)
lines(329:428,mypred$pred-1.96*mypred$se,lty=2)

#Forecasting on original for AR
unboxcoxed<-((bitcoinboxcox*lamb)+1)^(1/lamb)
fitma1unboxcoxed<-arima(unboxcoxed,order=c(18,0,0),method="ML",xreg=1:length(unboxcoxed))
predtrans1<-predict(fitma1unboxcoxed,n.ahead=100,newxreg=(length(unboxcoxed)+1)
                    :(length(unboxcoxed)+100))
ltrans1<-predtrans1$pred-1.96*predtrans1$se
ltrans1

utrans1<-predtrans1$pred+1.96*predtrans1$se
utrans1

ts.plot(r.bitcoin)
points(x = 329:428,predtrans1$pred,pch=1)
lines(329:428,ltrans1,col="red",lwd=5)
lines(329:428,utrans1,col="red",lwd=5)
```