TimeSeriesAnalysis_TaoJin.R

my_macbook

Wed Oct 24 11:47:11 2018

```
library(knitr)
opts_chunk$set(message = FALSE, warning = FALSE, cache = TRUE, cache.lazy = FALSE)
options(width = 100, dplyr.width = 100)
library(ggplot2)
theme_set(theme_light())


library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(forecast)
library(TTR)
library(tseries)

startDate = as.Date ("2010-06-29")
endDate = as.Date("2018-06-15")


# Get the Telsa Stock information from Yahoo

getSymbols("TSLA",src="yahoo",from = startDate, to = endDate)
```

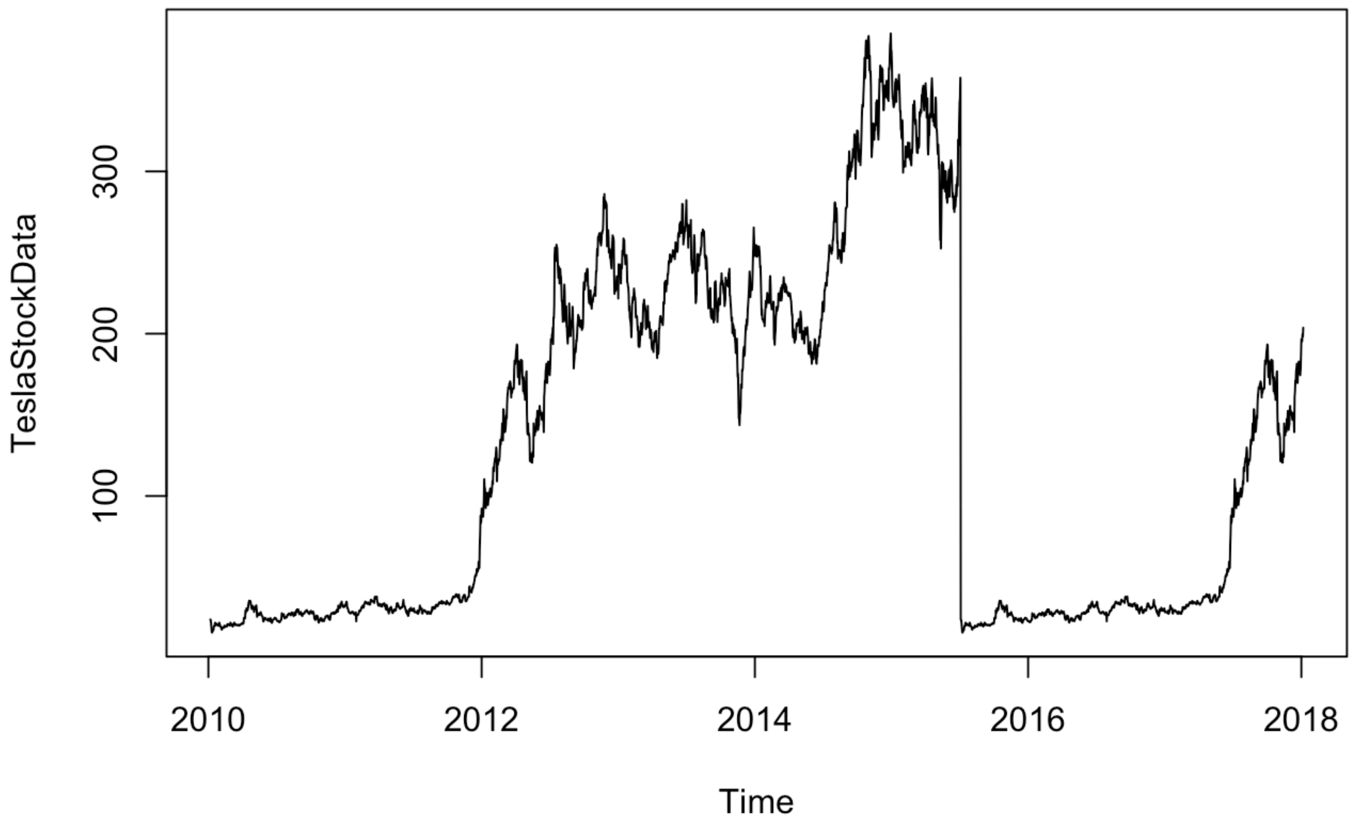
```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

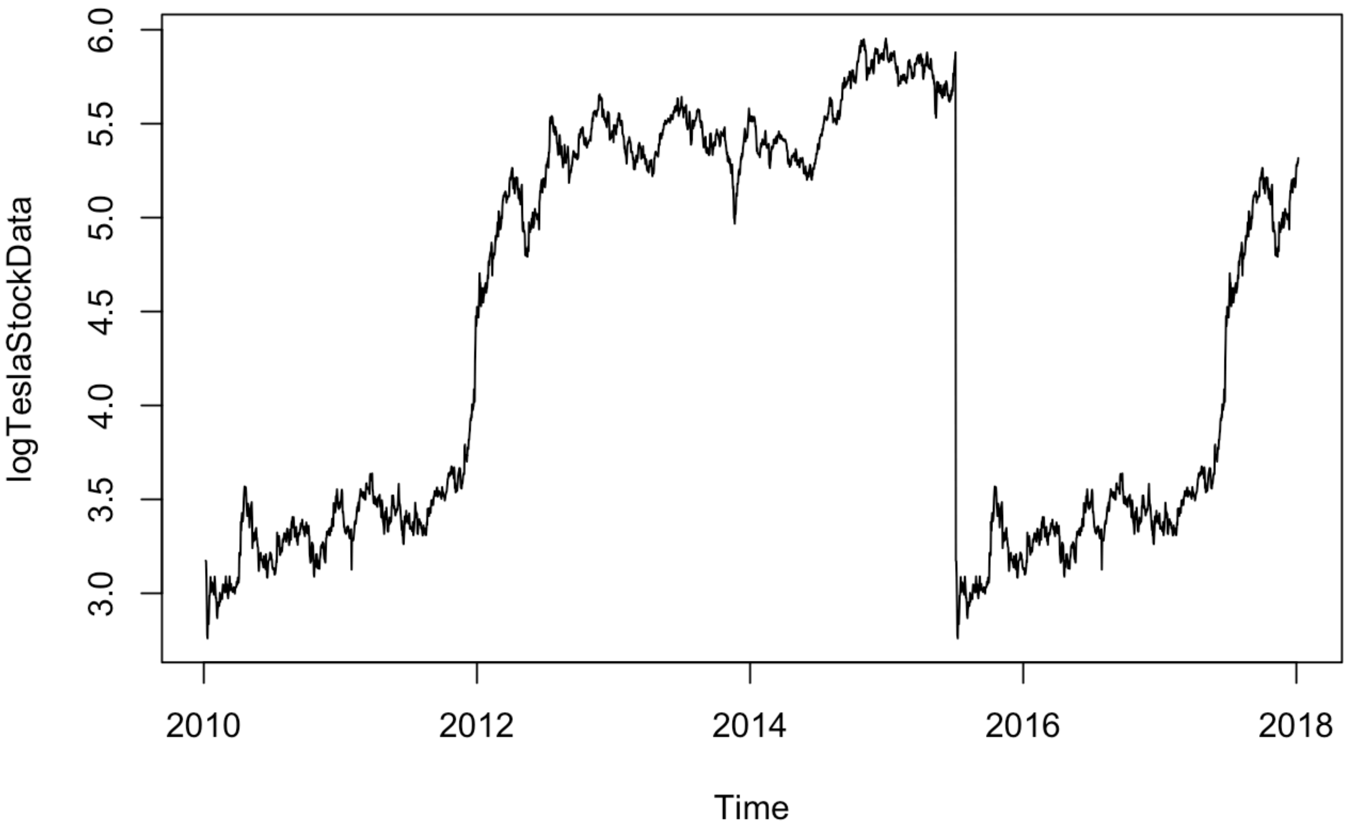
```
## [1] "TSLA"
```

```
plot(TSLA$TSLA.Close)
```

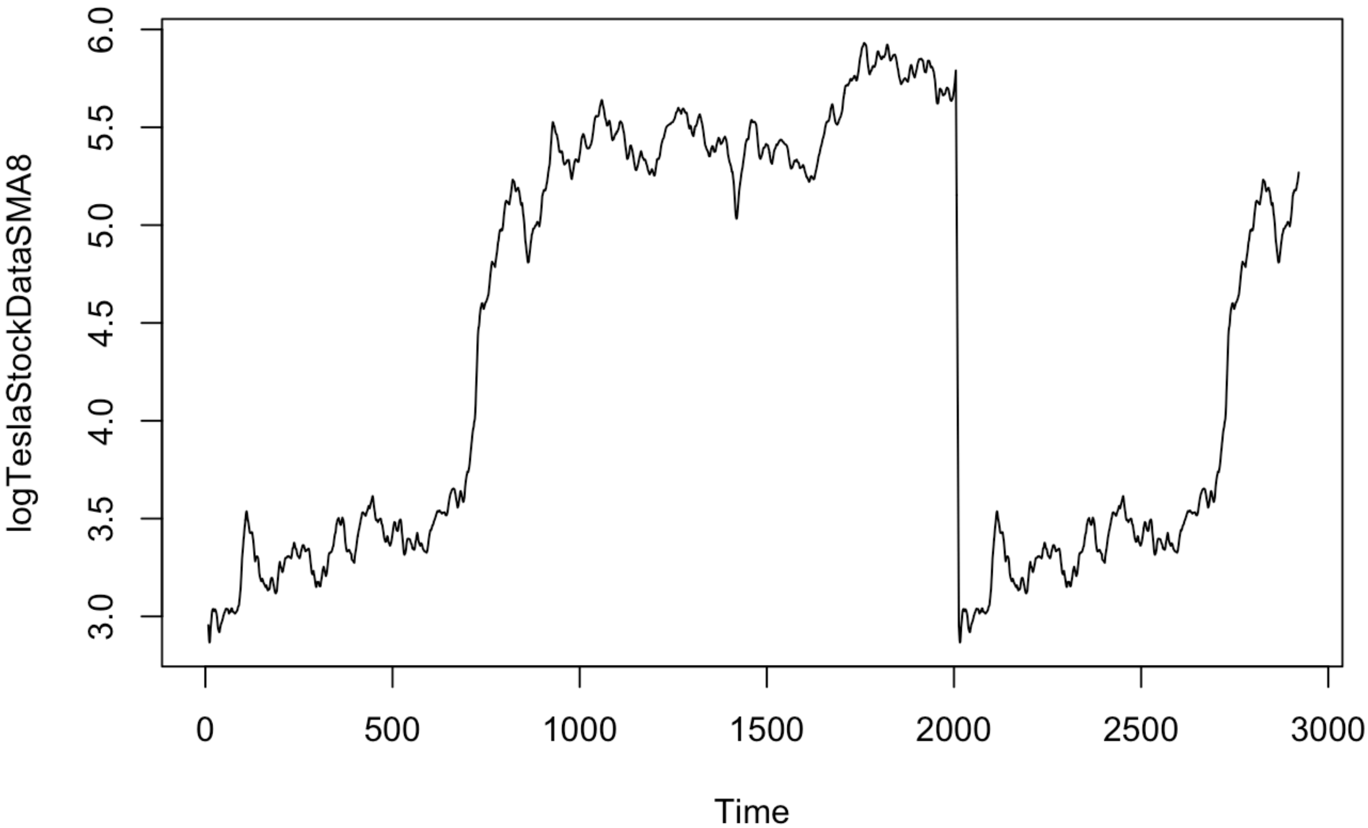

```
TeslaStockData <-ts(TSLA[,4], start = c(2010,6,29), end = c(2018,6,15), frequency = 3
65)


plot.ts(TeslaStockData)
```



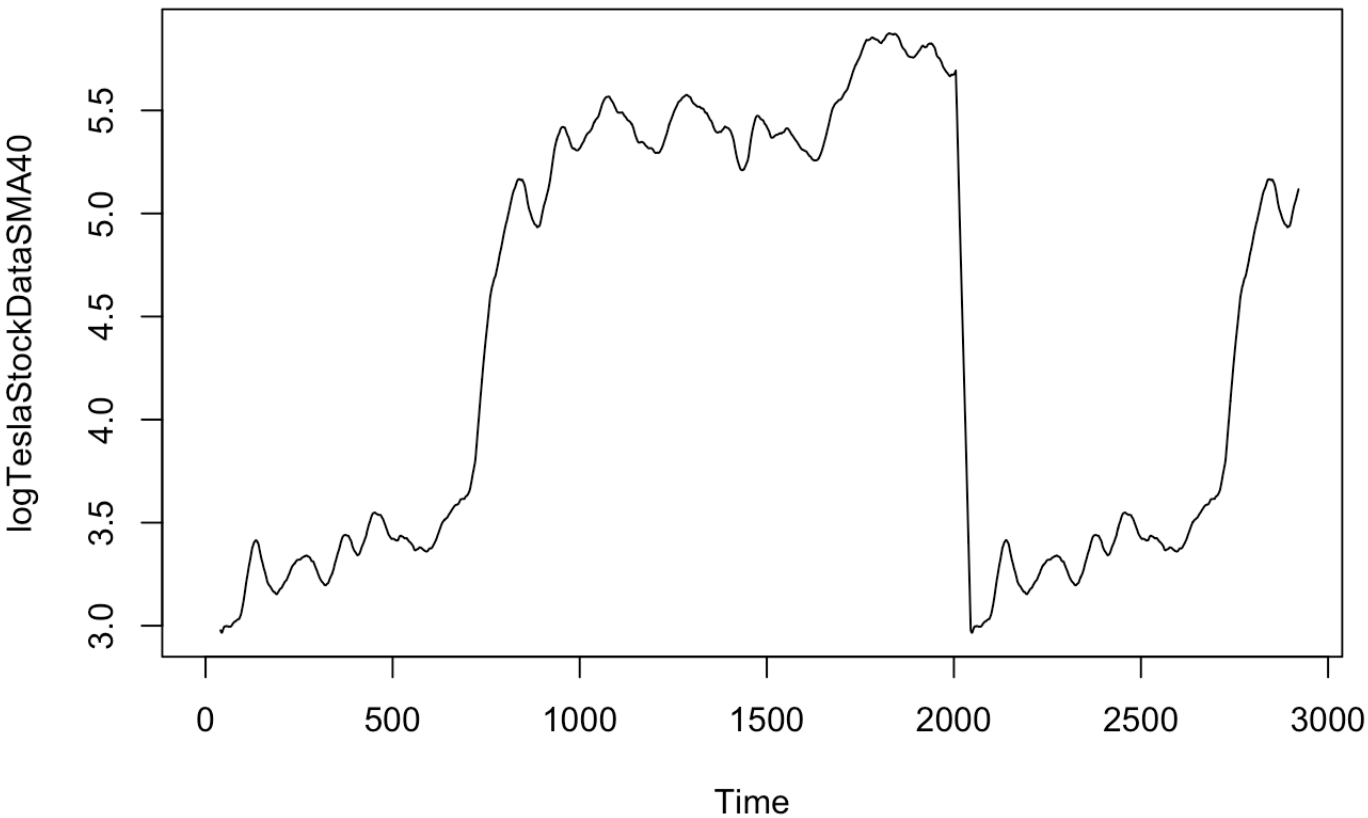

```
#We need to transform the time series in order to get a transformed time series that
can be described using an additive
#model. transform the time series by calculating the natural log of the original data
:

logTeslaStockData <- log(TeslaStockData)

plot.ts(logTeslaStockData)
```



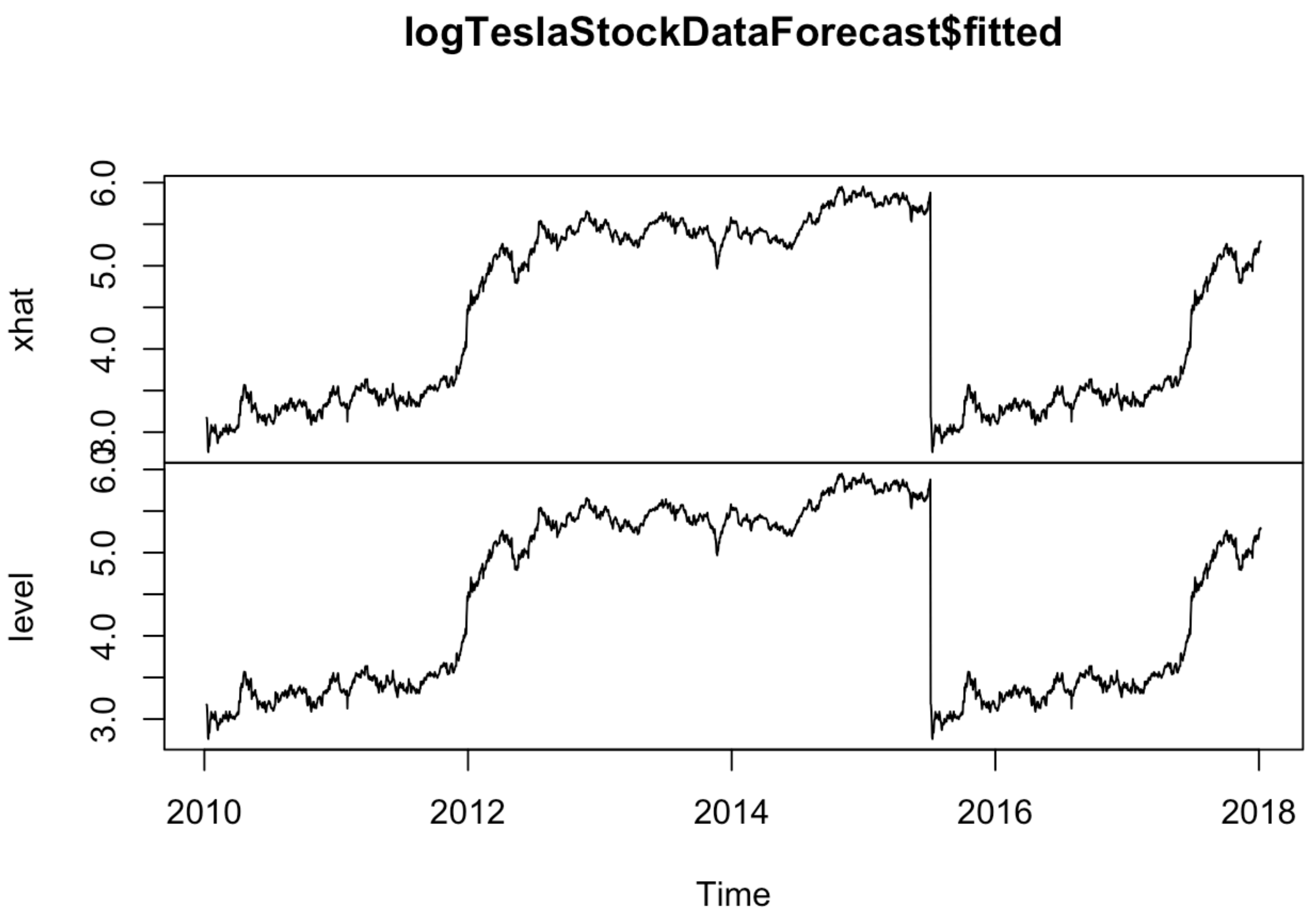

```
#Decompose Time Series
#Decomposing a time series means separating it into its constituent components, which
are usually a trend component
#and an irregular component, and if it is a seasonal time series, a seasonal componen
t.
# use the “SMA()” function to smooth time series data.

logTeslaStockDataSMA8 <- SMA(logTeslaStockData, n=8)
plot.ts(logTeslaStockDataSMA8)
```



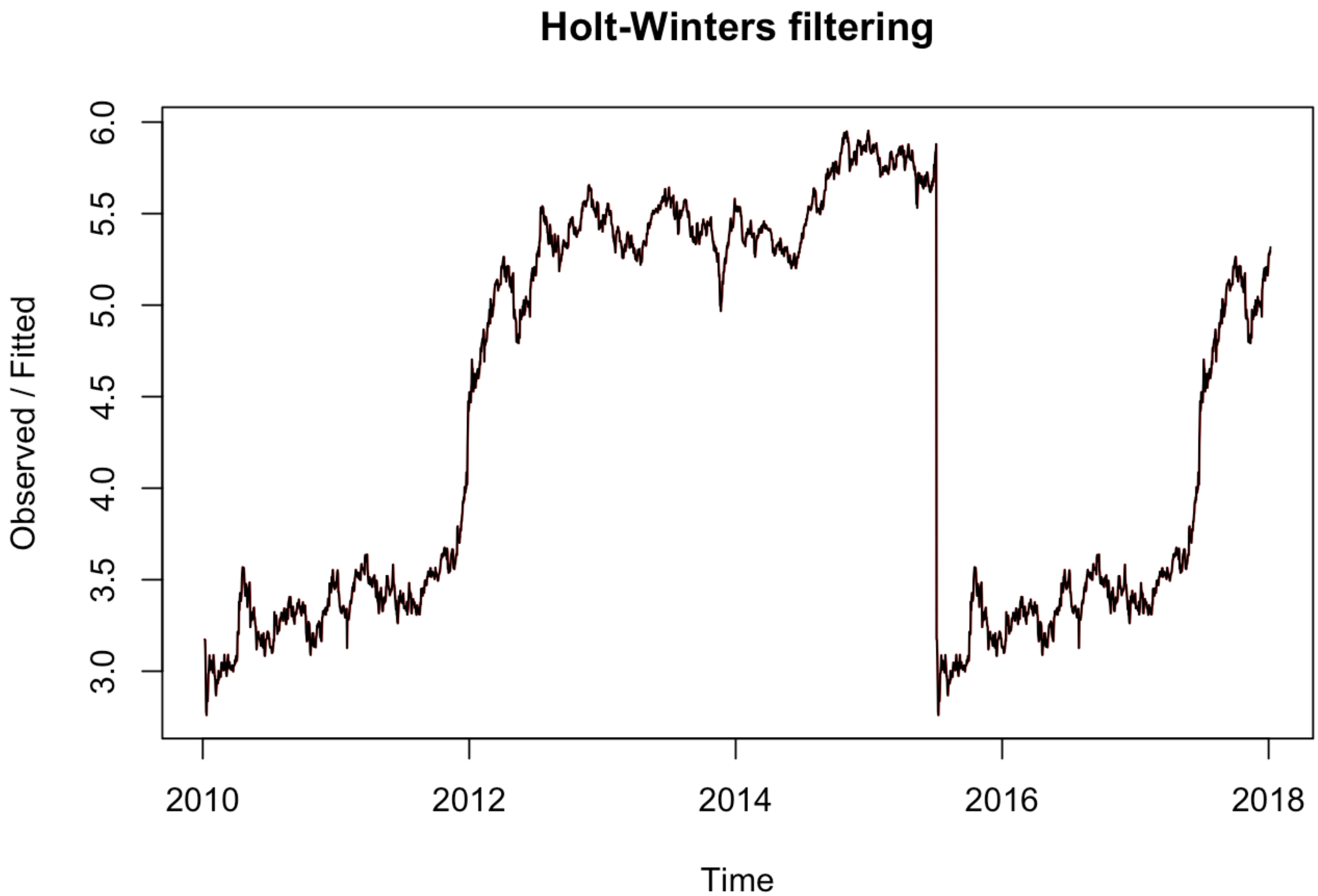

```
logTeslaStockDataSMA40 <- SMA(logTeslaStockData, n=40)
plot.ts(logTeslaStockDataSMA40)
```




```
#Forecasts using Holt's Exponential Smoothing
#Smoothing is controlled by two parameters, alpha, for the estimate of the level at t
he current time point, and beta for
#the estimate of the slope b of the trend component at the current time point.with si
mple exponential smoothing, the paramters alpha
#and beta have values between 0 and 1, and values that are close to 0 mean that littl
e weight is placed on the most
#recent observations when making forecasts of future values.


logTeslaStockDataForecast <- HoltWinters(logTeslaStockData,beta =FALSE, gamma=FALSE)
plot(logTeslaStockDataForecast$fitted)
```



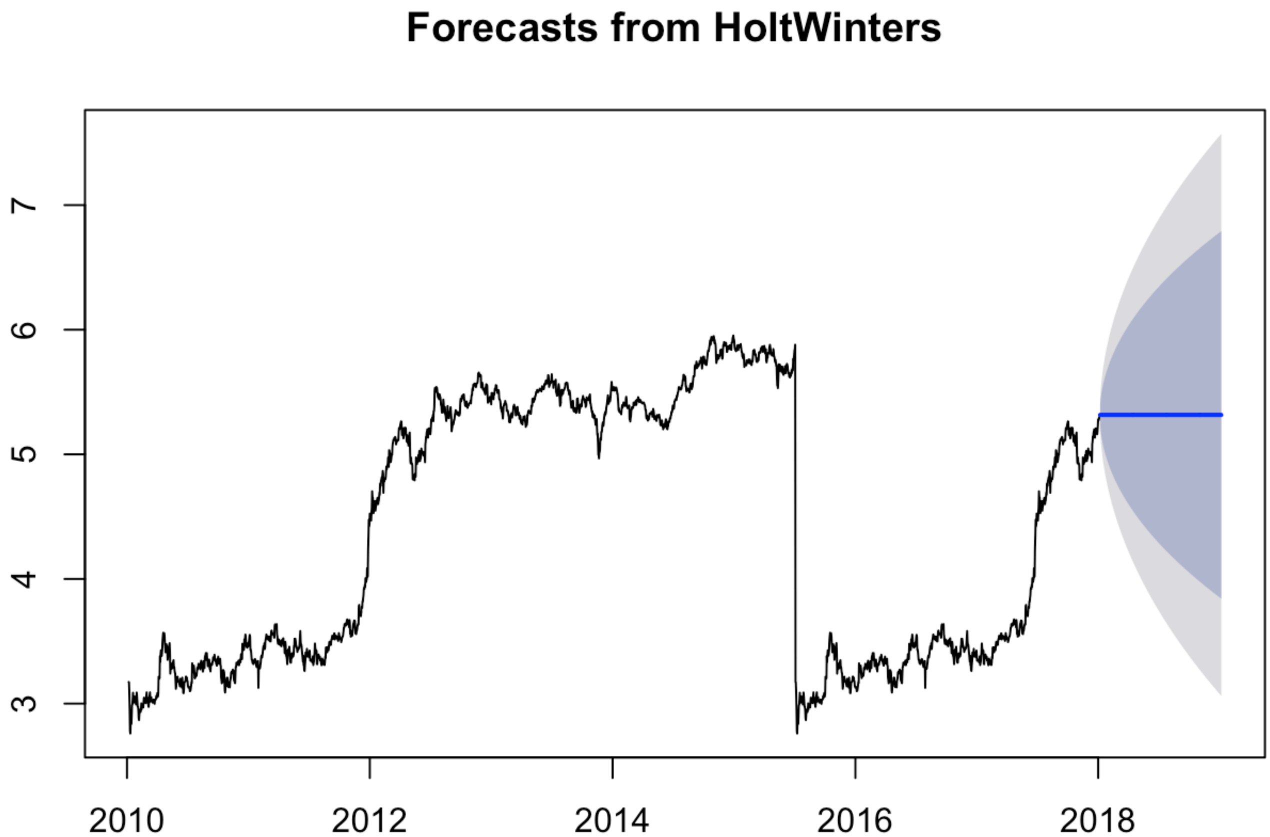

```
plot(logTeslaStockDataForecast)
```



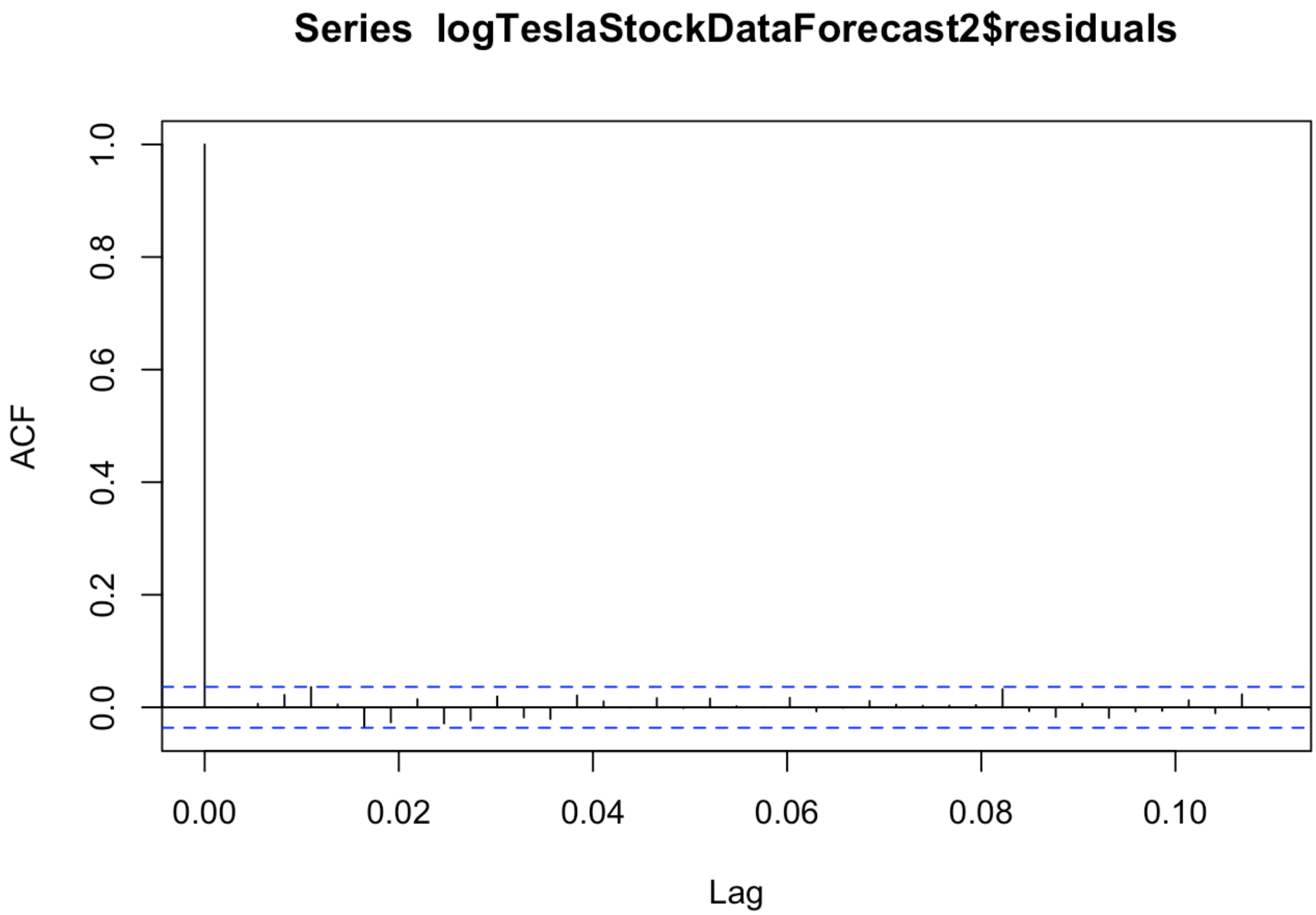

```
logTeslaStockDataForecast$SSE
```

```
## [1] 10.64326
```

```
# Forecast the next one year's price
logTeslaStockDataForecast2 <- forecast(logTeslaStockDataForecast, h = 365)
plot(logTeslaStockDataForecast2)
```



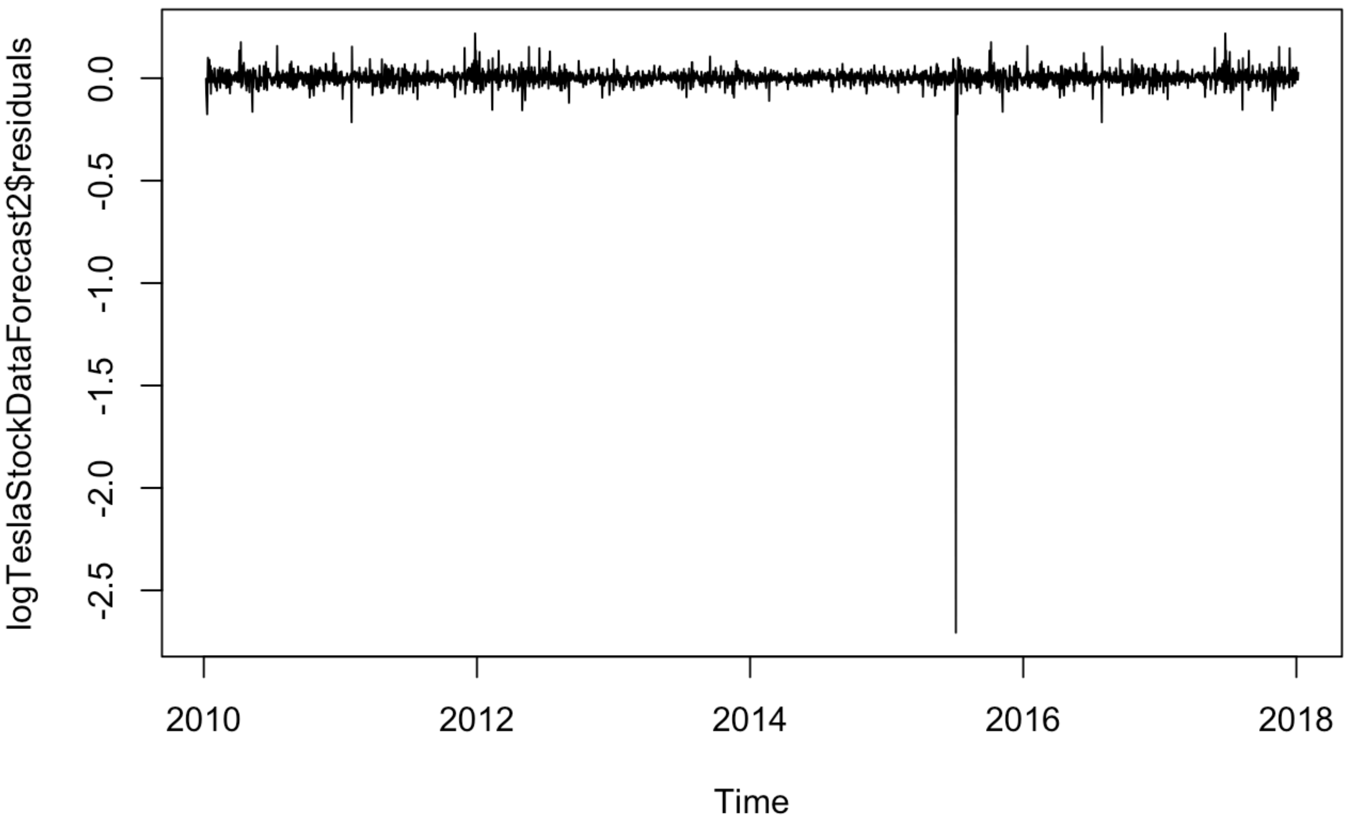

```
# in other words, if there are correlations between forecast errors for
# successive predictions, it is likely that the simple exponential smoothing forecast
s could be improved upon by
# another forecasting technique.

#We can calculate a correlogram of the forecast errors using the "acf()" function in
R. To specify the maximum lag
#that we want to look at, we use the "lag.max" parameter in acf().


plot(acf(logTeslaStockDataForecast2$residuals, lag.max=40, na.action = na.pass))
```



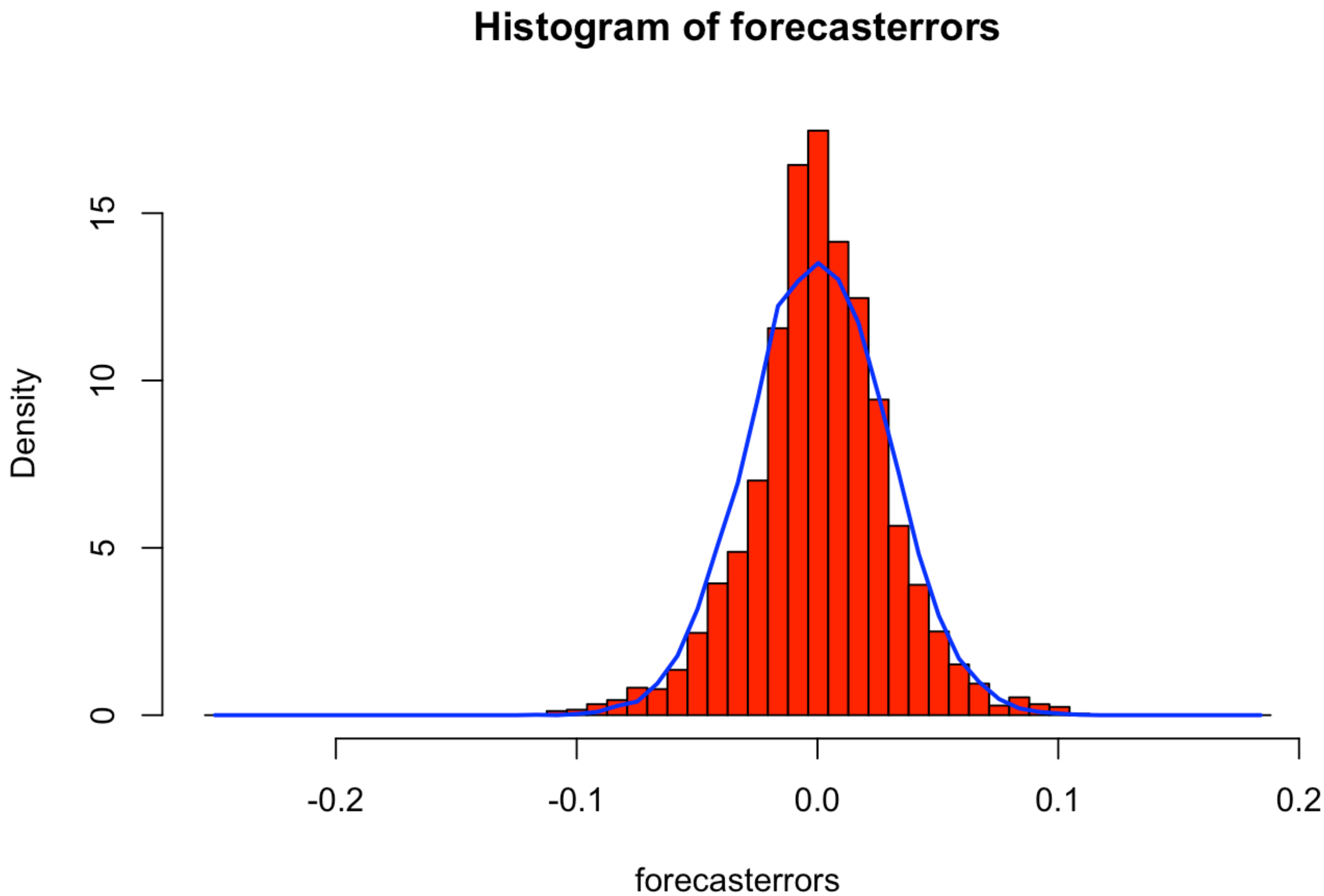

```
Box.test(logTeslaStockDataForecast2$residuals, lag=40, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  logTeslaStockDataForecast2$residuals
## X-squared = 31.42, df = 40, p-value = 0.832
```

```
plot.ts(logTeslaStockDataForecast2$residuals)
```


logTeslaStockDataForecast2$residuals
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
2010
2012
2014
2016
2018
Time

```
#To check whether the forecast errors are normally distributed with mean zero, we can
plot a histogram of the forecast
#errors, with an overlaid normal curve that has mean zero and the same standard devia
tion as the distribution
#of forecast errors. To do this, we can define an R function “plotForecastErrors()”,
below:

plotForecastErrors <- function(forecasterrors)
{
  # make a histogram of the forecast errors:
  mybinsize <- IQR(forecasterrors)/4
  mysd <- sd(forecasterrors)
  mymin <- min(forecasterrors) - mysd*5
  mymax <- max(forecasterrors) + mysd*3
  # generate normally distributed data with mean 0 and standard deviation mysd
  mynorm <- rnorm(10000, mean=0, sd=mysd)
  mymin2 <- min(mynorm)
  mymax2 <- max(mynorm)
  if (mymin2 < mymin) { mymin <- mymin2 }
  if (mymax2 > mymax) { mymax <- mymax2 }
  # make a red histogram of the forecast errors, with the normally distributed data o
verlaid:
  mybins <- seq(mymin, mymax, mybinsize)
  hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
  # freq=FALSE ensures the area under the histogram = 1
  # generate normally distributed data with mean 0 and standard deviation mysd
  myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
  # plot the normal curve as a blue line on top of the histogram of forecast errors:
  points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
}

logTeslaStockDataForecast2$residuals <- tsclean(logTeslaStockDataForecast2$residuals)

plotForecastErrors(logTeslaStockDataForecast2$residuals)
```


Histogram of forecasterrors
Density
0
5
10
15
-0.2
-0.1
0.0
0.1
0.2
forecasterrors

```
# it is plausible that the forecast errors are normally distributed with mean zero.Th
e time plot of forecast errors shows
#that the forecast errors have roughly constant variance over time.
#The histogram of forecast errors show that it is plausible that the forecast errors
are normally distributed with mean
#zero and constant variance.


# ARIMA Model
#While exponential smoothing methods do not make any assumptions about correlations b
etween successive values
#of the time series, in some cases you can make a better predictive model by taking c
orrelations in the data into
#account. Autoregressive Integrated Moving Average (ARIMA) models include an explicit
statistical model for the
#irregular component of a time series, that allows for non-zero autocorrelations in t
he irregular component.

#Stationarized the Time Series
adf.test(logTeslaStockData)
```

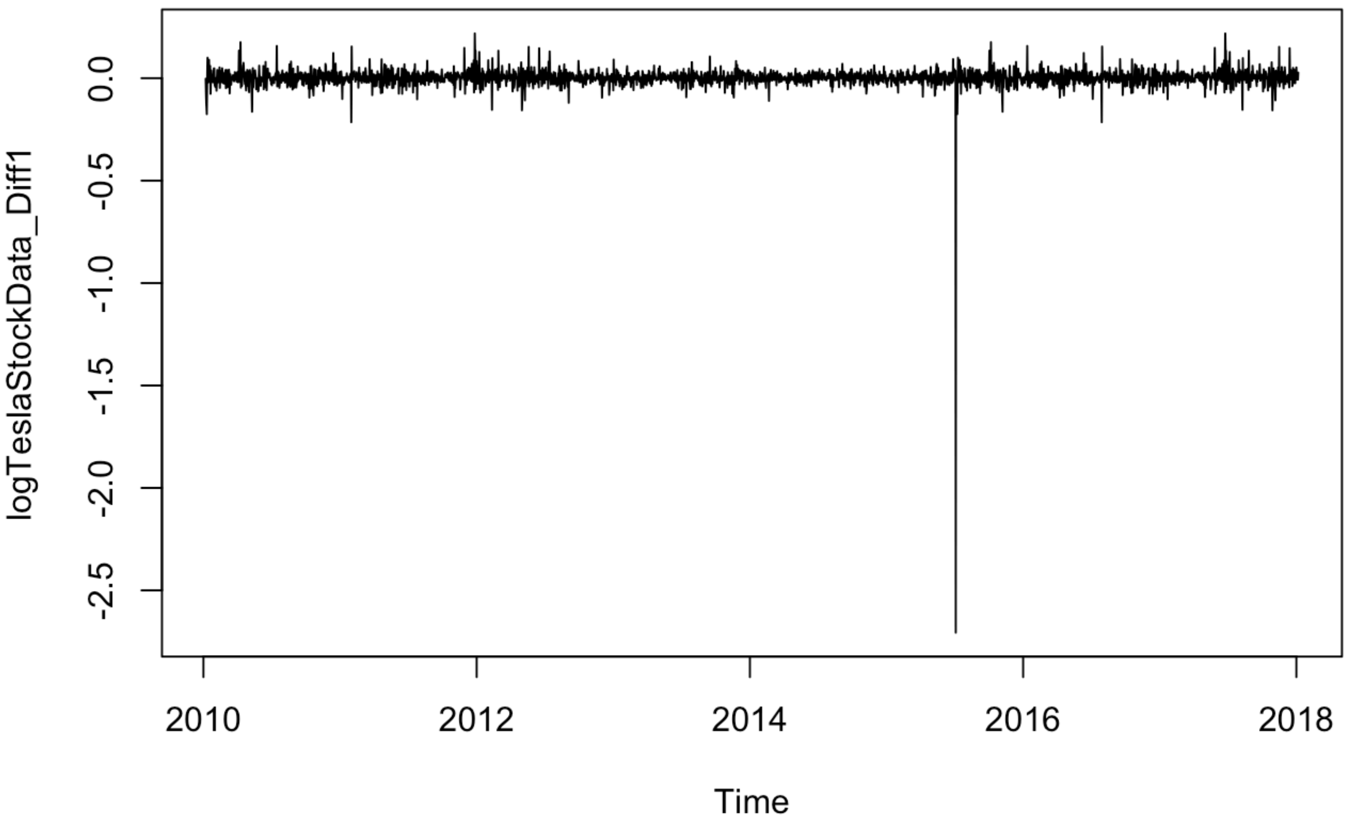
```
##
##  Augmented Dickey-Fuller Test
##
## data:  logTeslaStockData
## Dickey-Fuller = -1.6737, Lag order = 14, p-value = 0.7165
## alternative hypothesis: stationary
```

```
logTeslaStockData_Diff1 <- diff(logTeslaStockData,differences = 1)

adf.test(logTeslaStockData_Diff1)
```

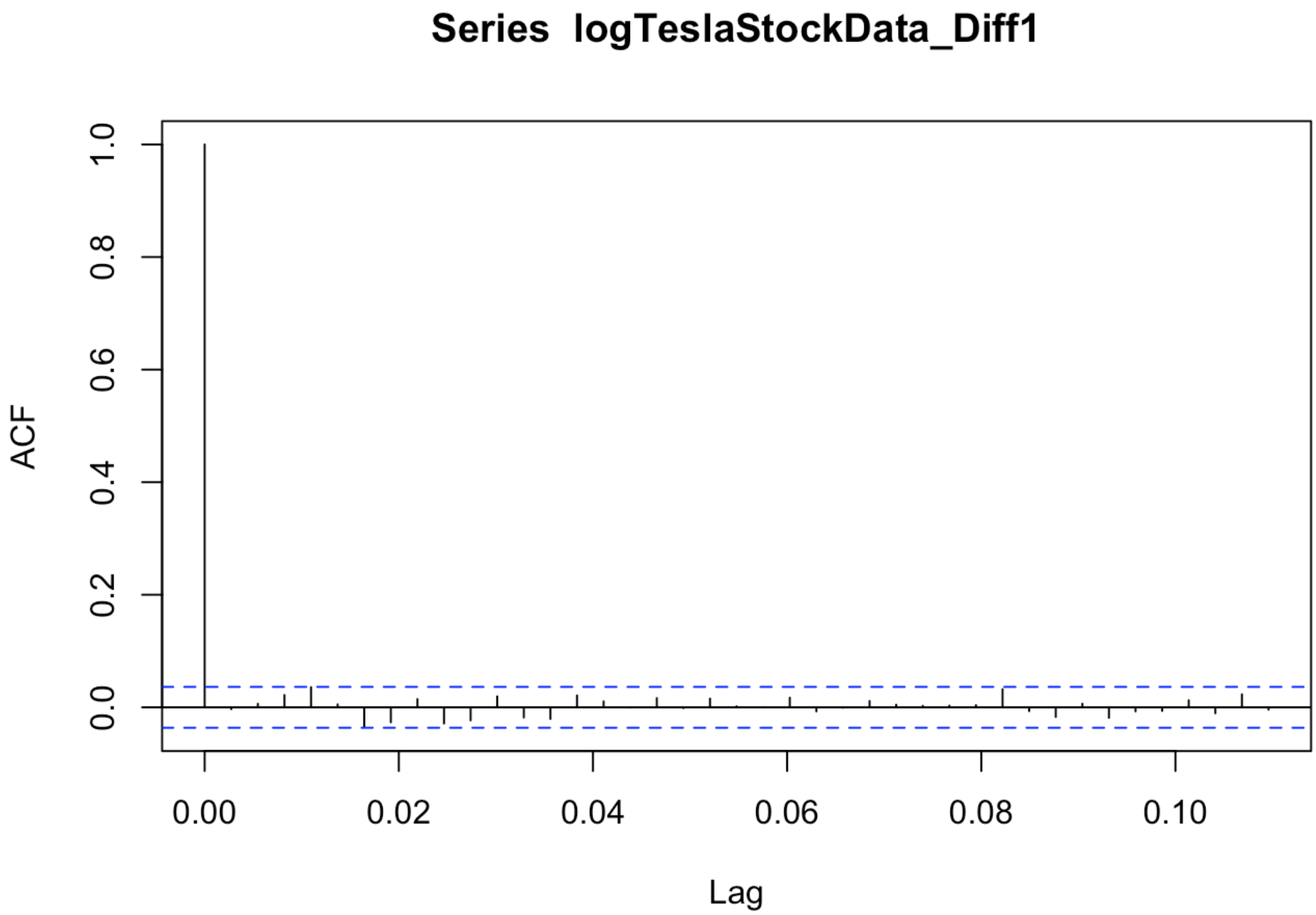
```
## Warning in adf.test(logTeslaStockData_Diff1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  logTeslaStockData_Diff1
## Dickey-Fuller = -14.115, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary
```

```
plot(logTeslaStockData_Diff1)
```



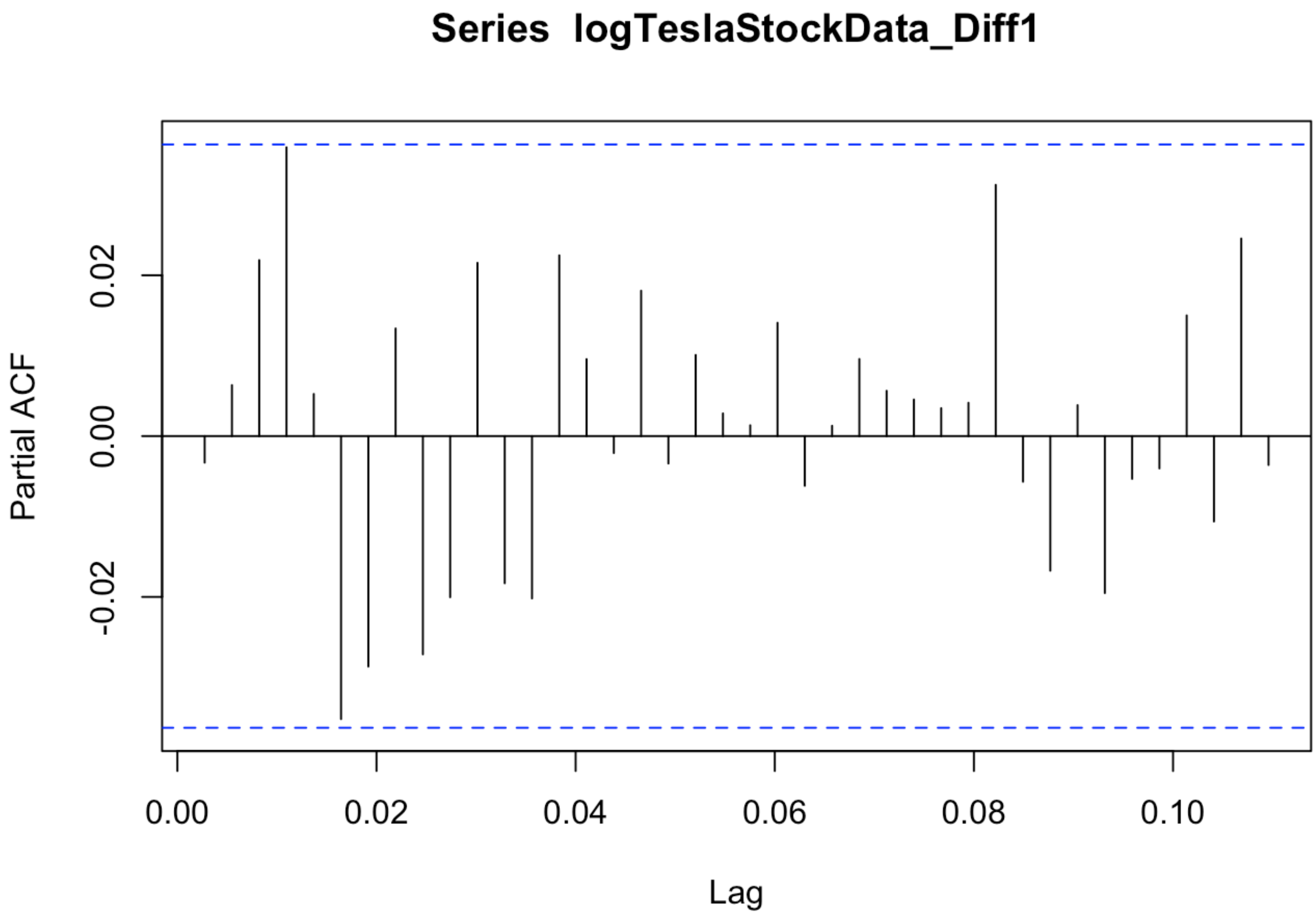

```
acf(logTeslaStockData_Diff1, lag.max = 40)
```




```
acf(logTeslaStockData_Diff1, lag.max = 40, plot = FALSE)
```

```
##
## Autocorrelations of series 'logTeslaStockData_Diff1', by lag
##
## 0.00000 0.00274 0.00548 0.00822 0.01096 0.01370 0.01644 0.01918 0.02192 0.02466 0.
02740 0.03014
##   1.000  -0.003   0.006   0.022   0.036   0.005  -0.034  -0.027   0.015  -0.029  -
0.023   0.020
## 0.03288 0.03562 0.03836 0.04110 0.04384 0.04658 0.04932 0.05205 0.05479 0.05753 0.
06027 0.06301
##  -0.018  -0.021   0.021   0.011  -0.001   0.016  -0.002   0.015   0.002   0.000
0.017  -0.007
## 0.06575 0.06849 0.07123 0.07397 0.07671 0.07945 0.08219 0.08493 0.08767 0.09041 0.
09315 0.09589
##  -0.001   0.011   0.005   0.003   0.003   0.004   0.032  -0.007  -0.017   0.006  -
0.019  -0.007
## 0.09863 0.10137 0.10411 0.10685 0.10959
##  -0.006   0.013  -0.011   0.023  -0.004
```

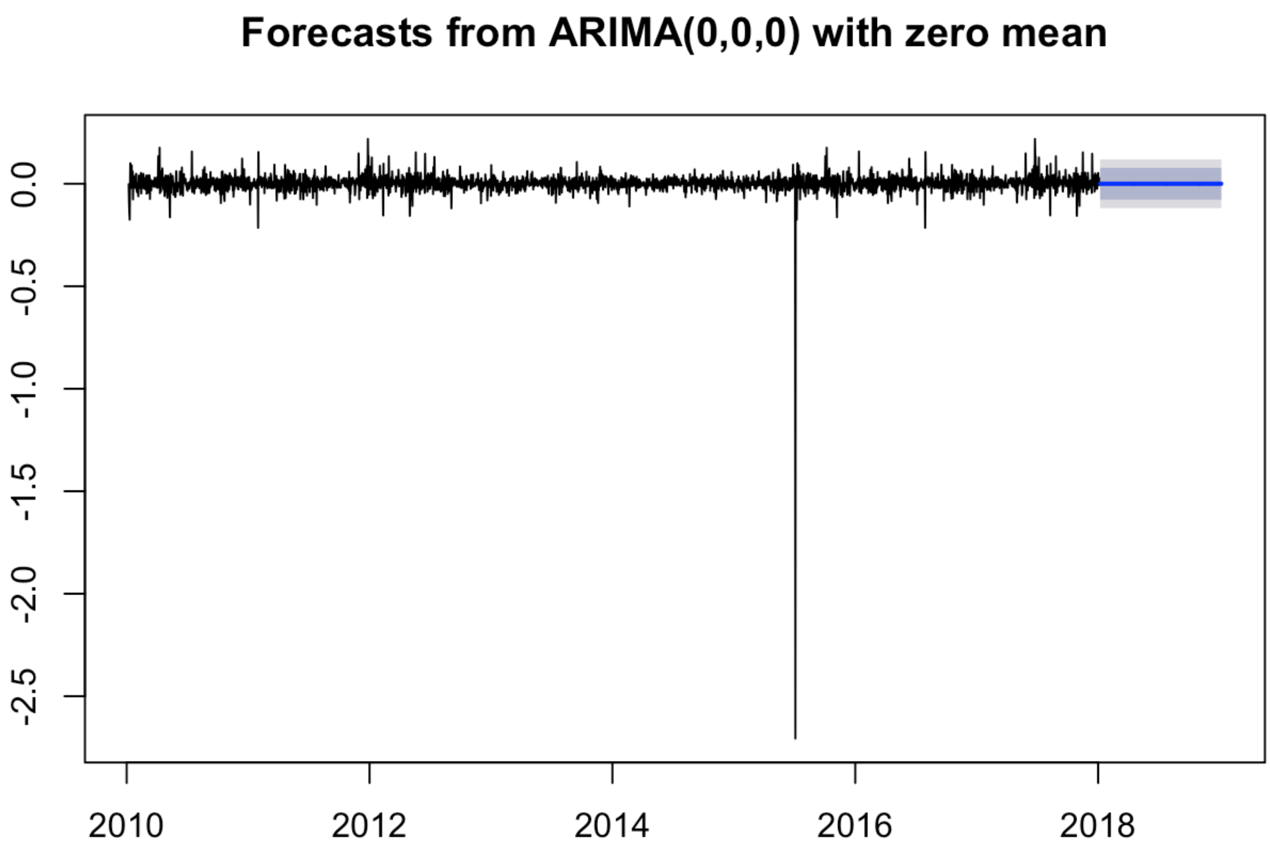
```
pacf(logTeslaStockData_Diff1, lag.max = 40)
```



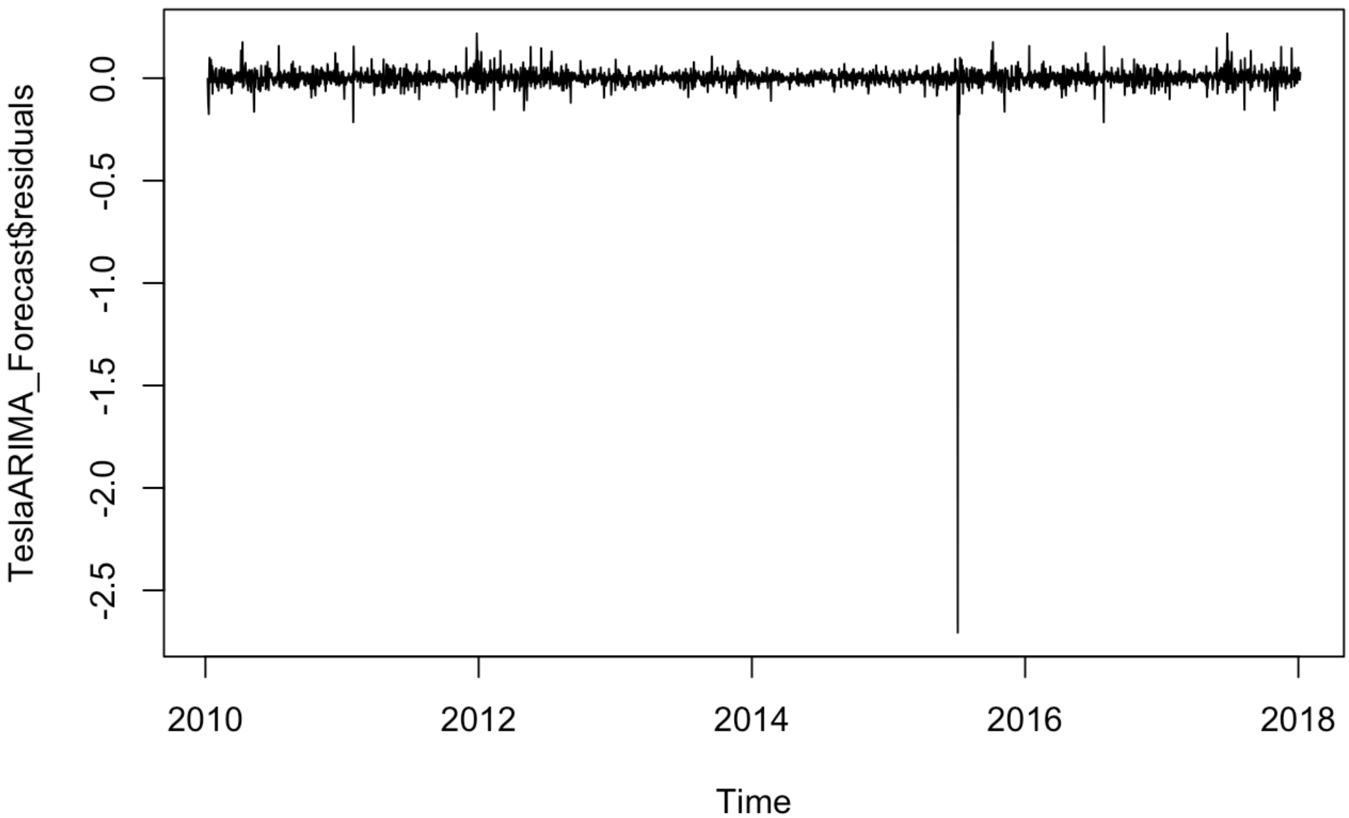

```
pacf(logTeslaStockData_Diff1, lag.max = 40, plot = FALSE)
```

```
##
## Partial autocorrelations of series 'logTeslaStockData_Diff1', by lag
##
## 0.00274 0.00548 0.00822 0.01096 0.01370 0.01644 0.01918 0.02192 0.02466 0.02740 0.
03014 0.03288
##  -0.003   0.006   0.022   0.036   0.005  -0.035  -0.029   0.013  -0.027  -0.020
0.022  -0.018
## 0.03562 0.03836 0.04110 0.04384 0.04658 0.04932 0.05205 0.05479 0.05753 0.06027 0.
06301 0.06575
##  -0.020   0.022   0.010  -0.002   0.018  -0.003   0.010   0.003   0.001   0.014  -
0.006   0.001
## 0.06849 0.07123 0.07397 0.07671 0.07945 0.08219 0.08493 0.08767 0.09041 0.09315 0.
09589 0.09863
##   0.010   0.006   0.005   0.003   0.004   0.031  -0.006  -0.017   0.004  -0.020  -
0.005  -0.004
## 0.10137 0.10411 0.10685 0.10959
##   0.015  -0.011   0.025  -0.004
```

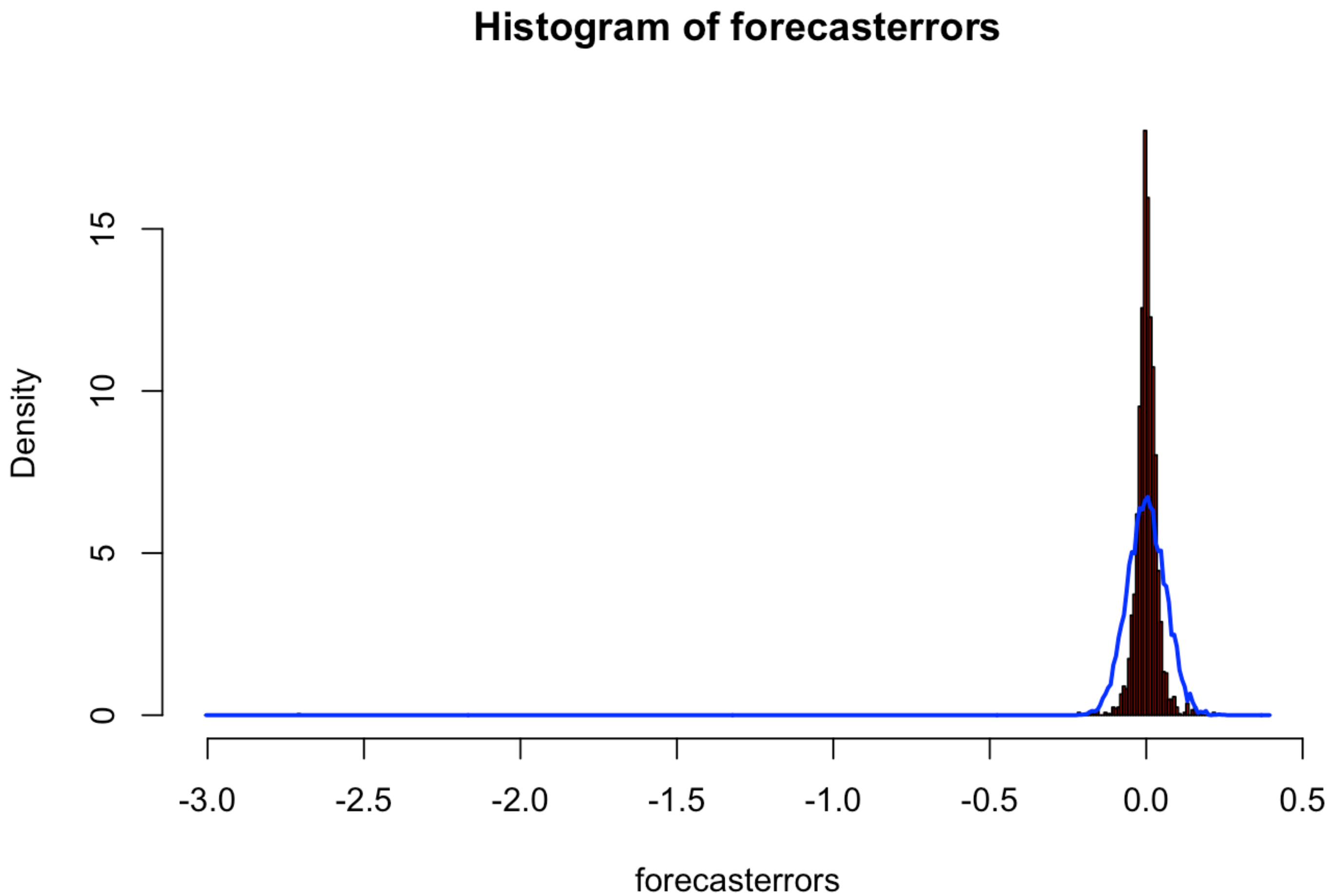
```
TeslaARIMA <- auto.arima(logTeslaStockData_Diff1)

TeslaARIMA_Forecast <-forecast(TeslaARIMA, h = 365)
plot(TeslaARIMA_Forecast)
```



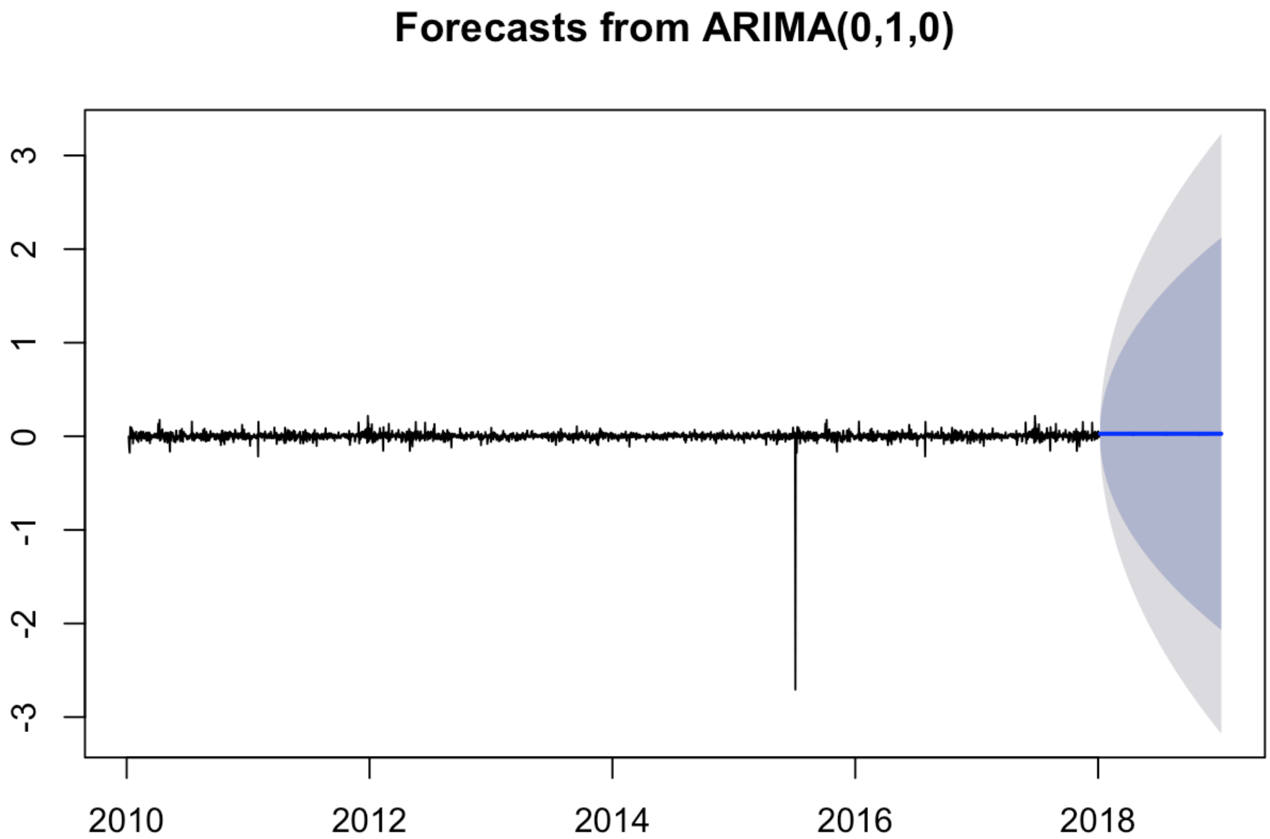

```
plot(TeslaARIMA_Forecast$residuals)
```



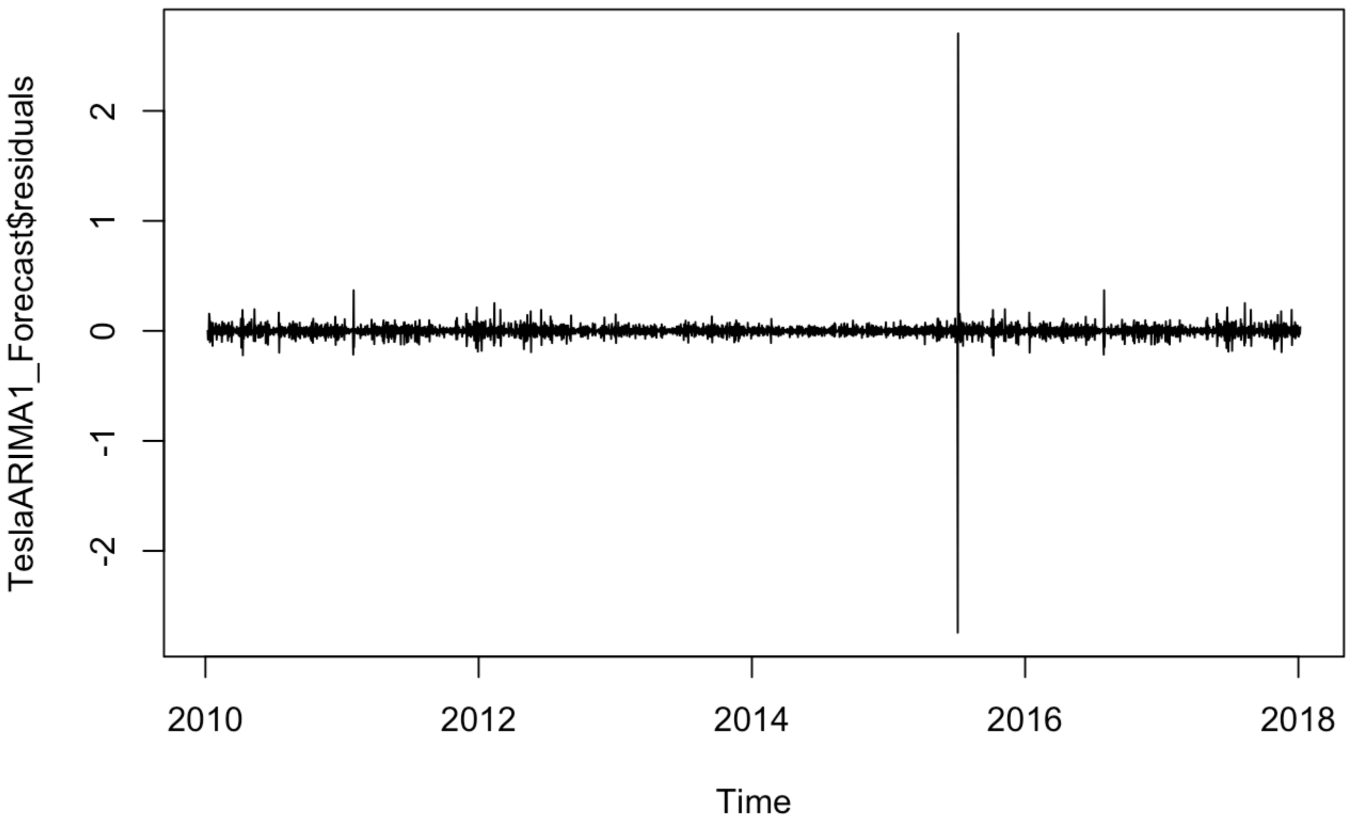

```
plotForecastErrors(TeslaARIMA_Forecast$residuals)
```



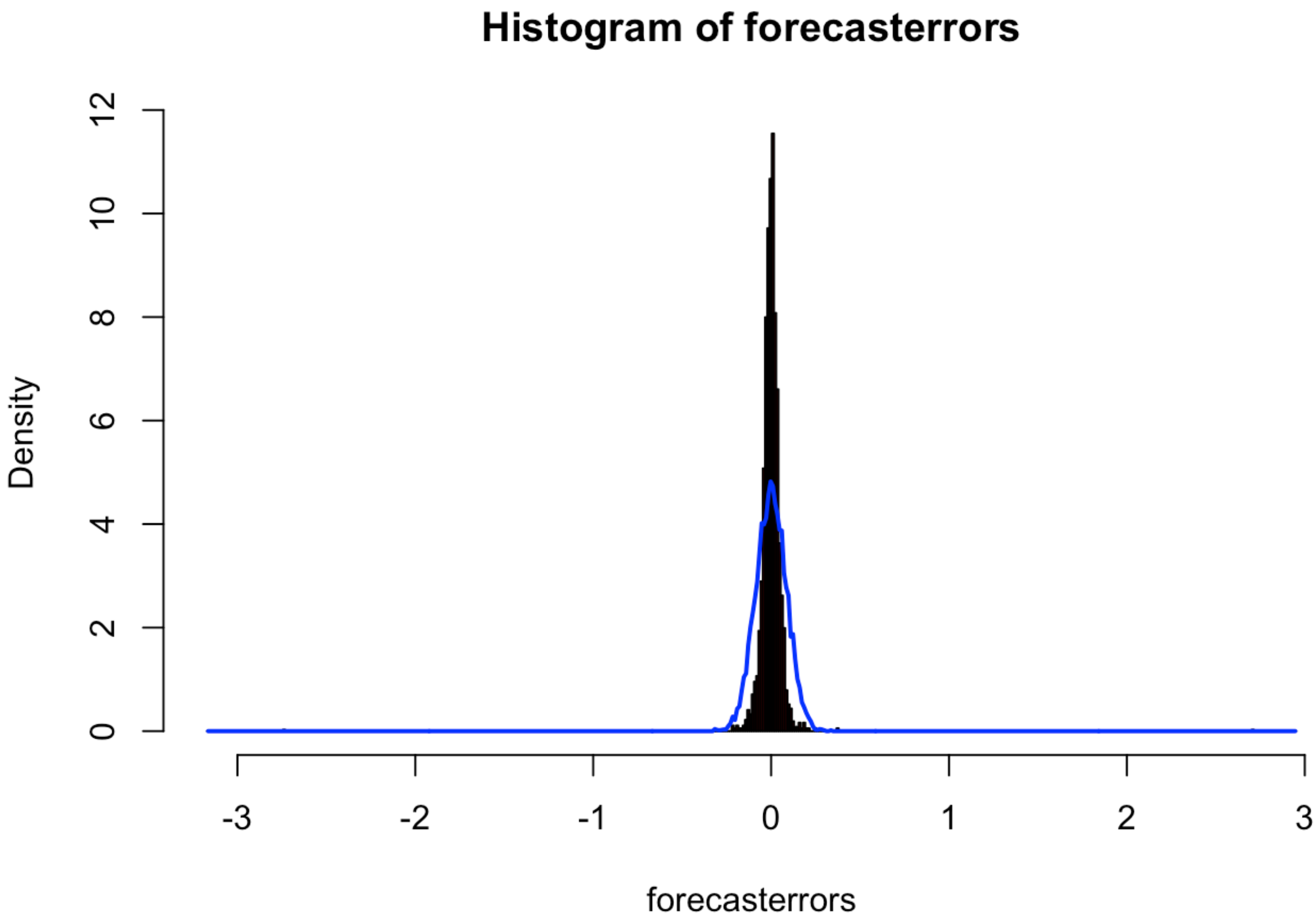

```
# White noise problem, choose the ARIMA (0,1,0) instead of auto.arima

TeslaARIMA1 <- arima(logTeslaStockData_Diff1, order = c(0,1,0))
TeslaARIMA1_Forecast <-forecast(TeslaARIMA1, h = 365)

plot(TeslaARIMA1_Forecast)
```




```
plot(TeslaARIMA1_Forecast$residuals)
```




```
plotForecastErrors(TeslaARIMA1_Forecast$residuals)
```




```
library(sarima)
```

```
## Loading required package: stats4
```

```
library(stats4)
# whiteNoiseTest(TeslaARIMA1_Forecast,h0= iid)
```