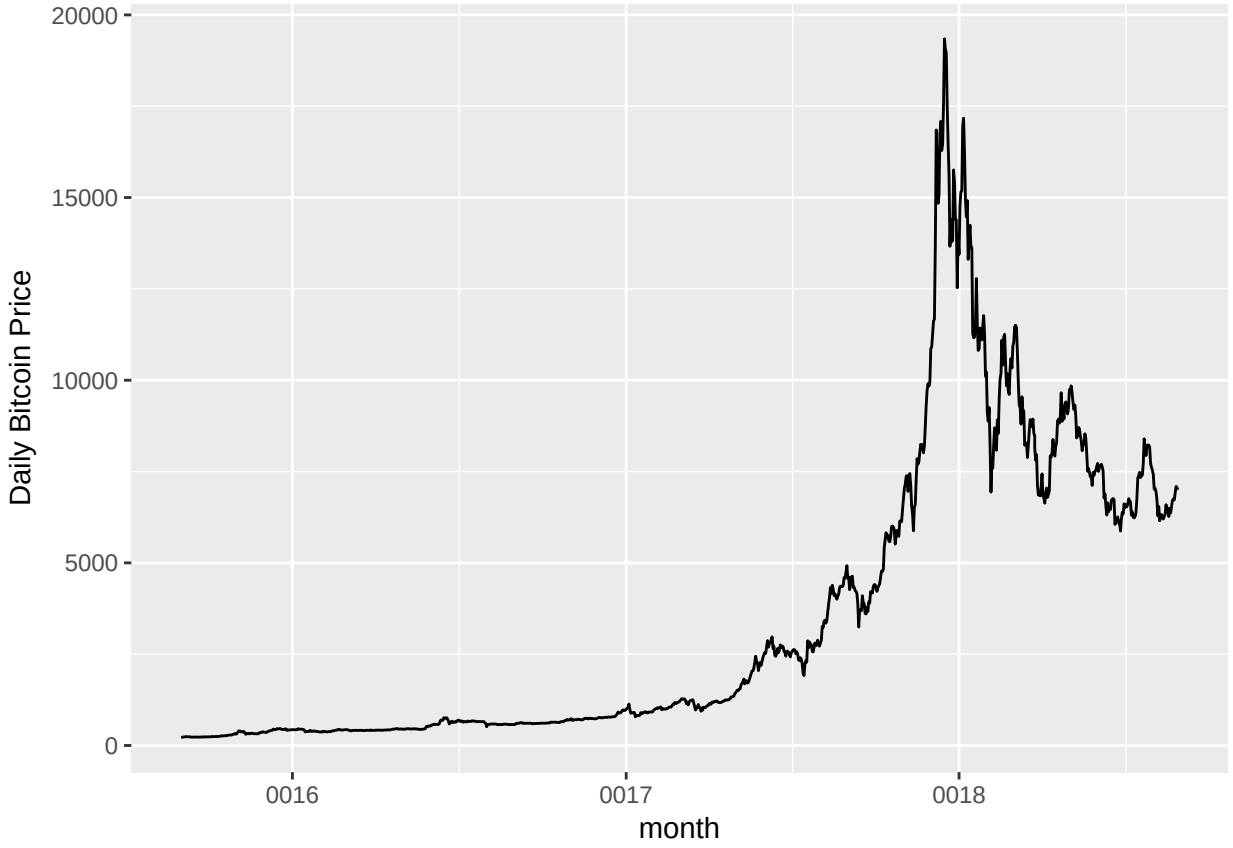
ARIMA Notebook

This is a brief report generated by R Markdown for the Bitcoin ARIMA prediction task.

The first step is to import data, which is sourced from Yahoo Finance.

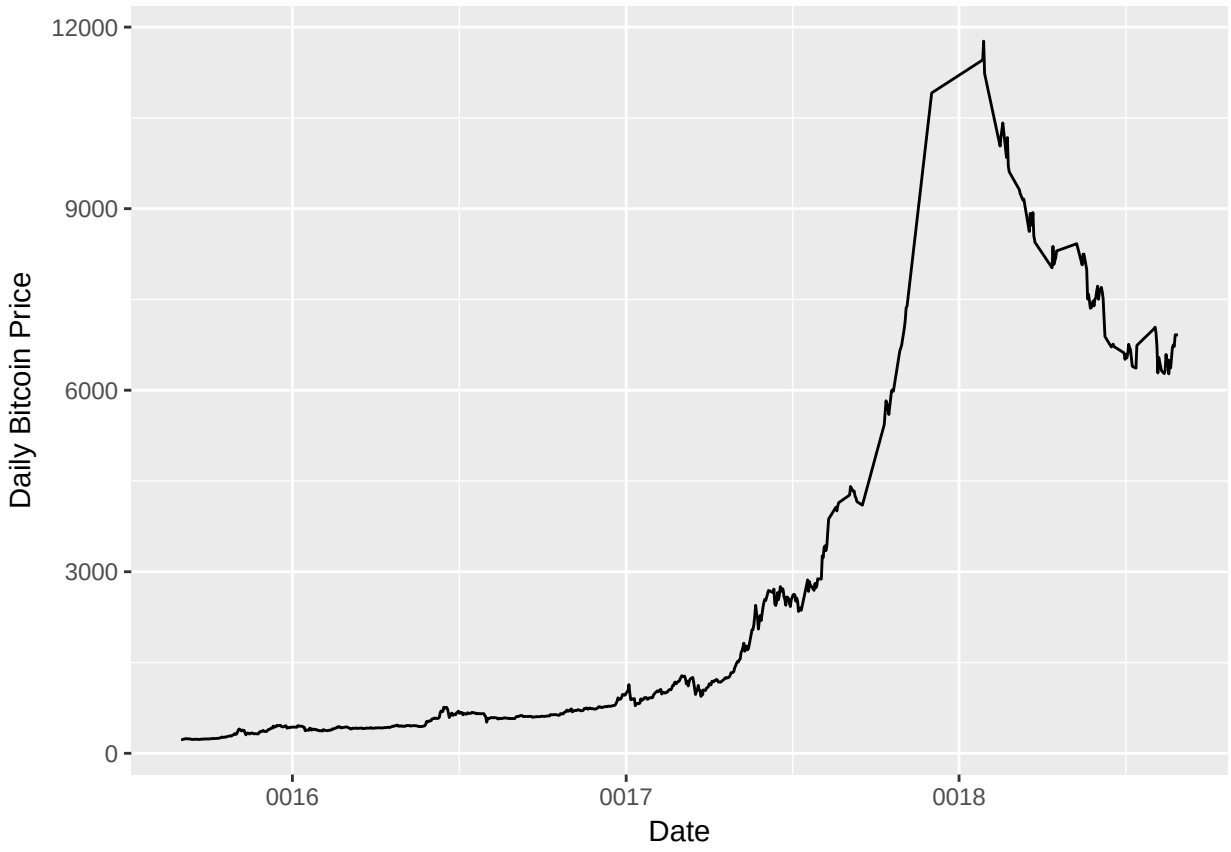
```
library('ggplot2')
library('forecast')
library('tseries')
csv_path='~/Downloads/BTC-USD (1).csv'
daily_data = read.csv(csv_path, header=TRUE, stringsAsFactors=FALSE)
  daily_data$Date = as.Date(daily_data$Date,format="%d/%m/%Y")
  ggplot(daily_data, aes(Date, Close)) + geom_line() + scale_x_date('month')  + ylab("Daily Bitcoin Pri
    xlab("")
```



The second step is to cleanse the data for outliers and missing values.

```
  close_ts = ts(daily_data[, c('Close')])
  daily_data$clean_close = tsclean(close_ts)
  ggplot() +
    geom_line(data = daily_data, aes(x = Date, y = clean_close)) + ylab('Daily Bitcoin Price')
```

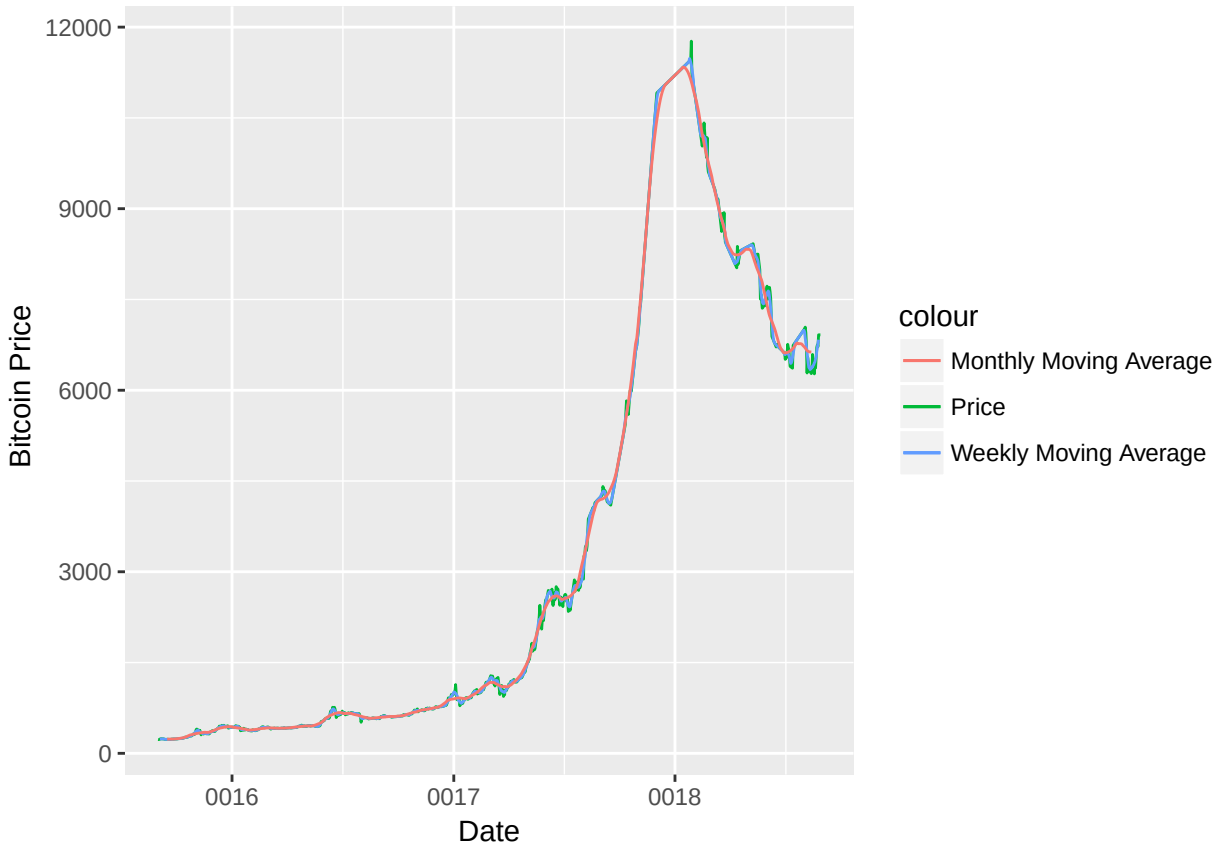
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



Next step is to define moving average.

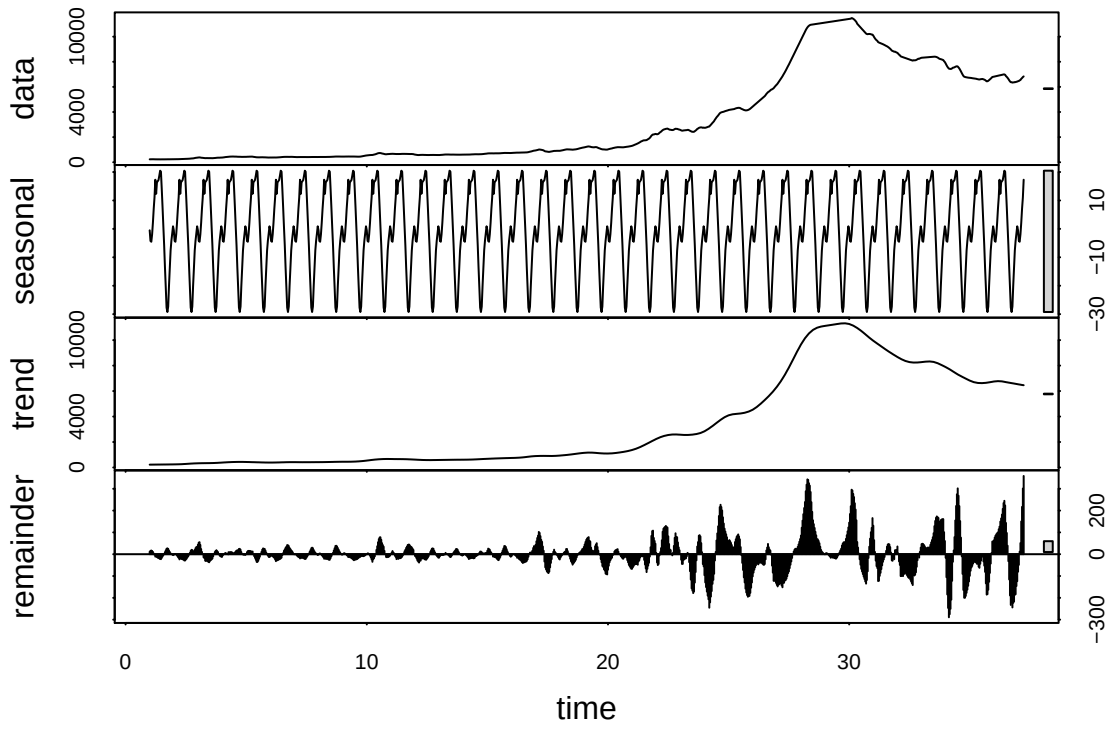
```
daily_data$close_ma = ma(daily_data$clean_close, order=7) # using the clean count with no outliers
daily_data$close_ma30 = ma(daily_data$clean_close, order=30)

ggplot() +
  geom_line(data = daily_data, aes(x = Date, y = clean_close, colour = "Price")) +
  geom_line(data = daily_data, aes(x = Date, y = close_ma,   colour = "Weekly Moving Average"))  +
  geom_line(data = daily_data, aes(x = Date, y = close_ma30, colour = "Monthly Moving Average"))  +
  ylab(' Bitcoin Price')
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Warning: Removed 6 rows containing missing values (geom_path).
## Warning: Removed 30 rows containing missing values (geom_path).
```



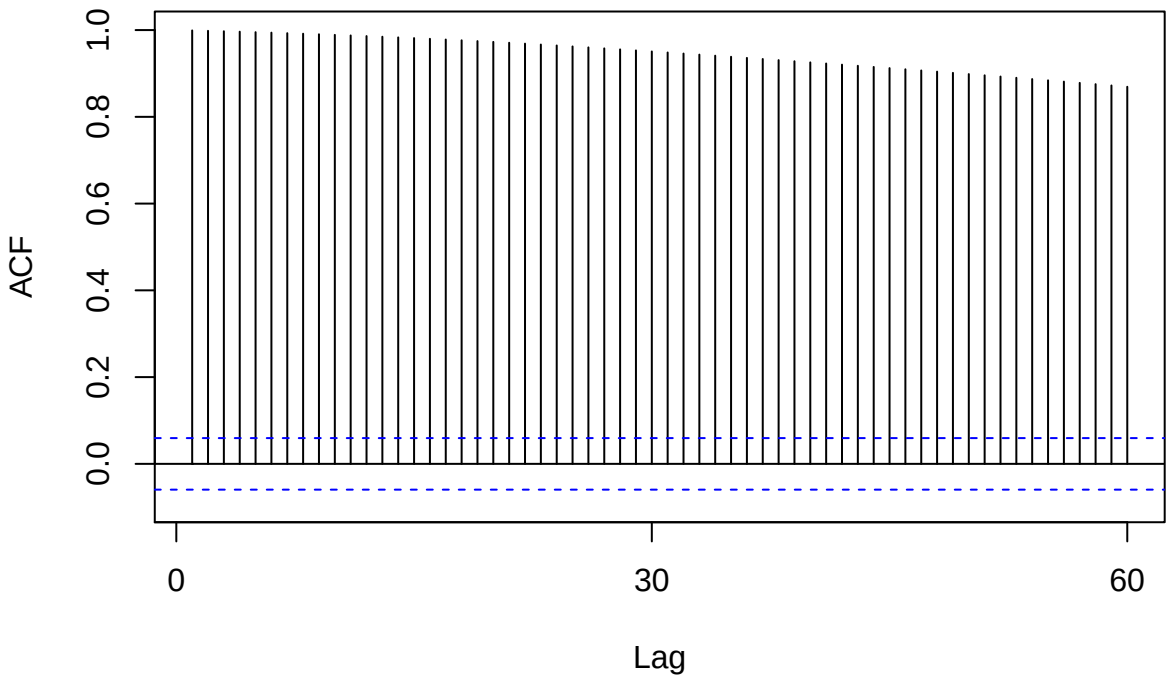
Next step is data decomposition based on TS analysis.

```
close_ma = ts(na.omit(daily_data$close_ma), frequency=30)
decomp = stl(close_ma, s.window="periodic")
deseasonal_cls <- seasadj(decomp)
plot(decomp)
```

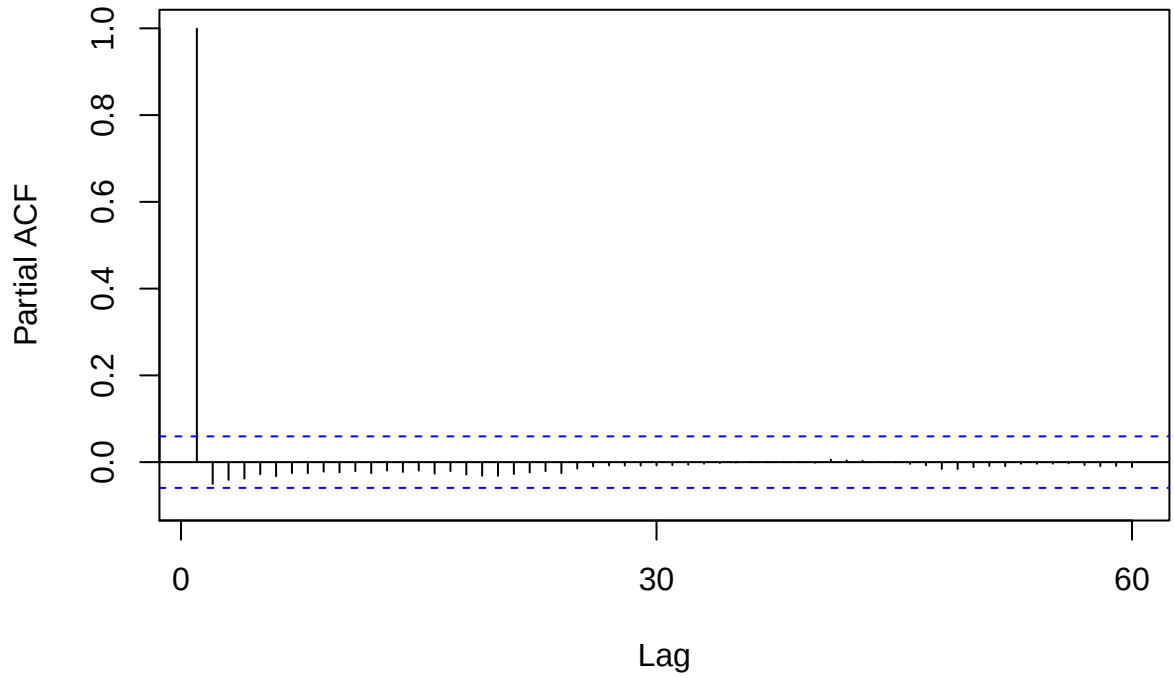


The next step is to make the model stationary, given the time dependency of the mean and variance. The two figures in the following help us to do that.

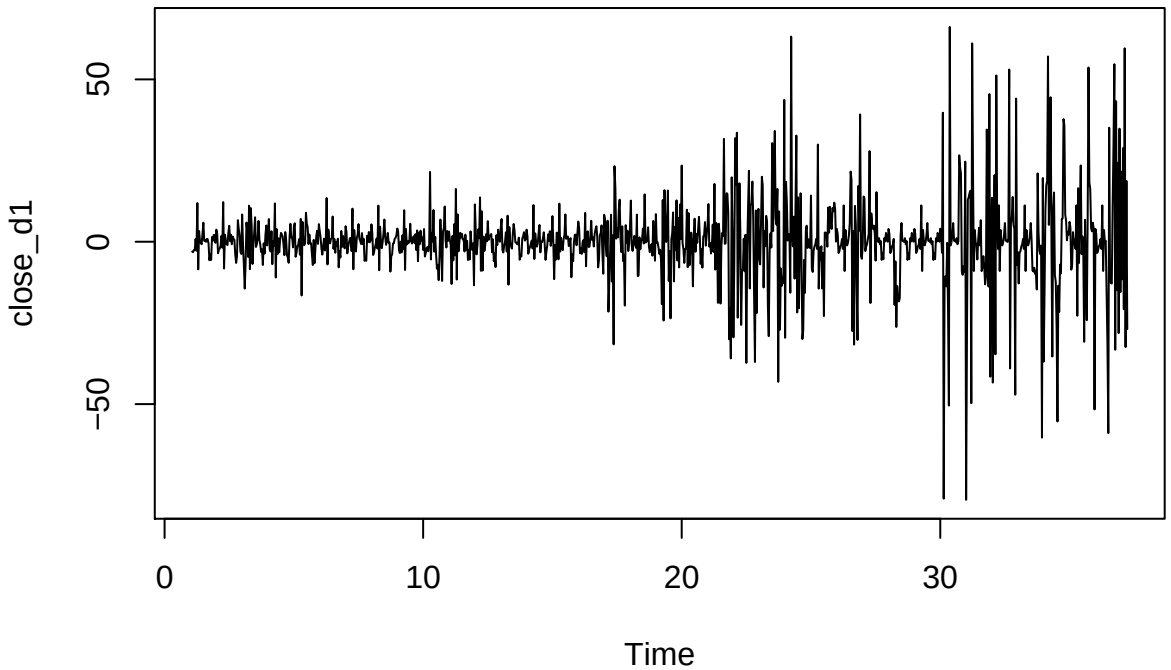
```
Acf(close_ma, main='')
```




```
  Pacf(close_ma, main='')
```




```
close_d1 = diff(deseasonal_cls, differences = 2)
  plot(close_d1)
```



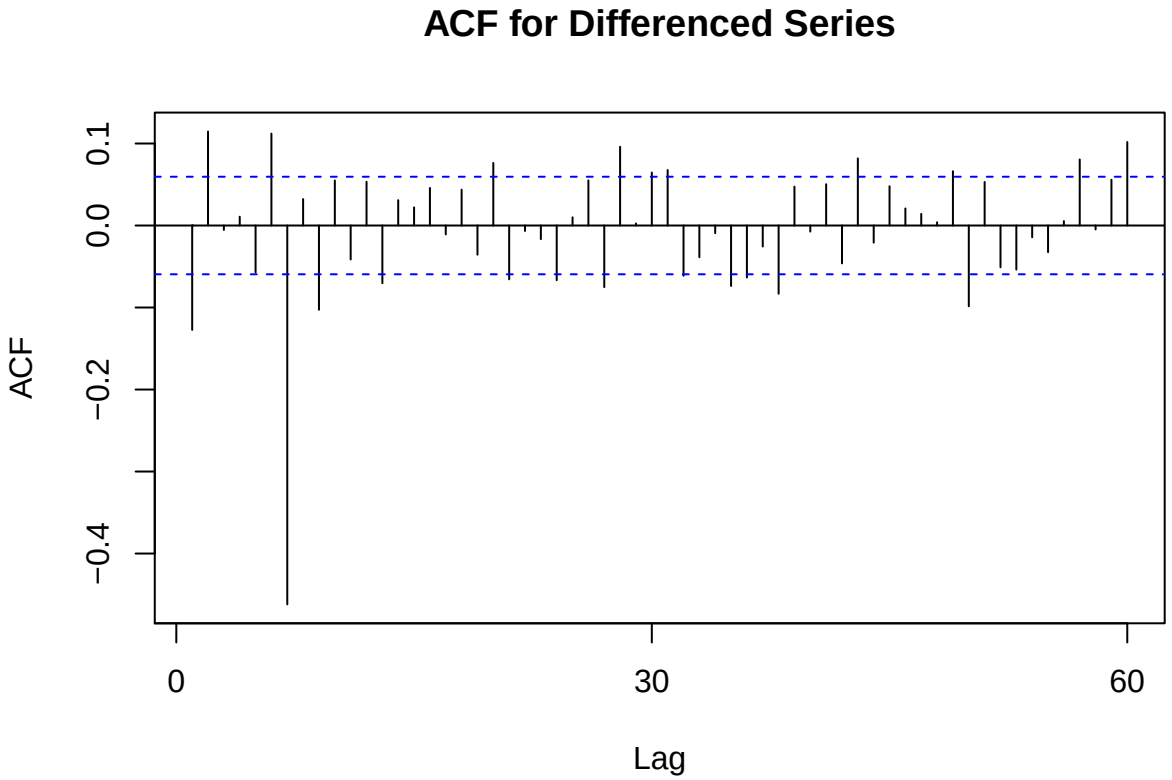

```
  adf.test(close_d1, alternative = "stationary")
```

```
## Warning in adf.test(close_d1, alternative = "stationary"): p-value smaller
## than printed p-value
```

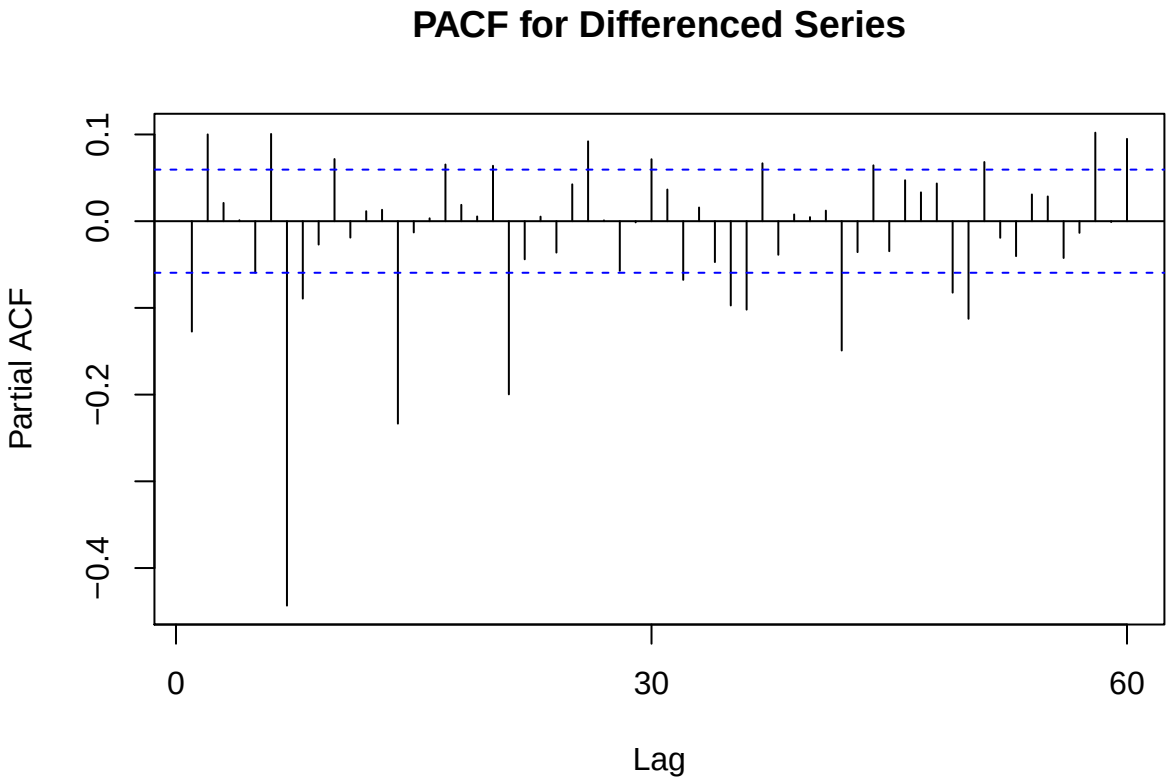
```
##
##  Augmented Dickey-Fuller Test
##
## data:  close_d1
```

```
## Dickey-Fuller = -12.603, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
Acf(close_d1, main='ACF for Differenced Series')
```

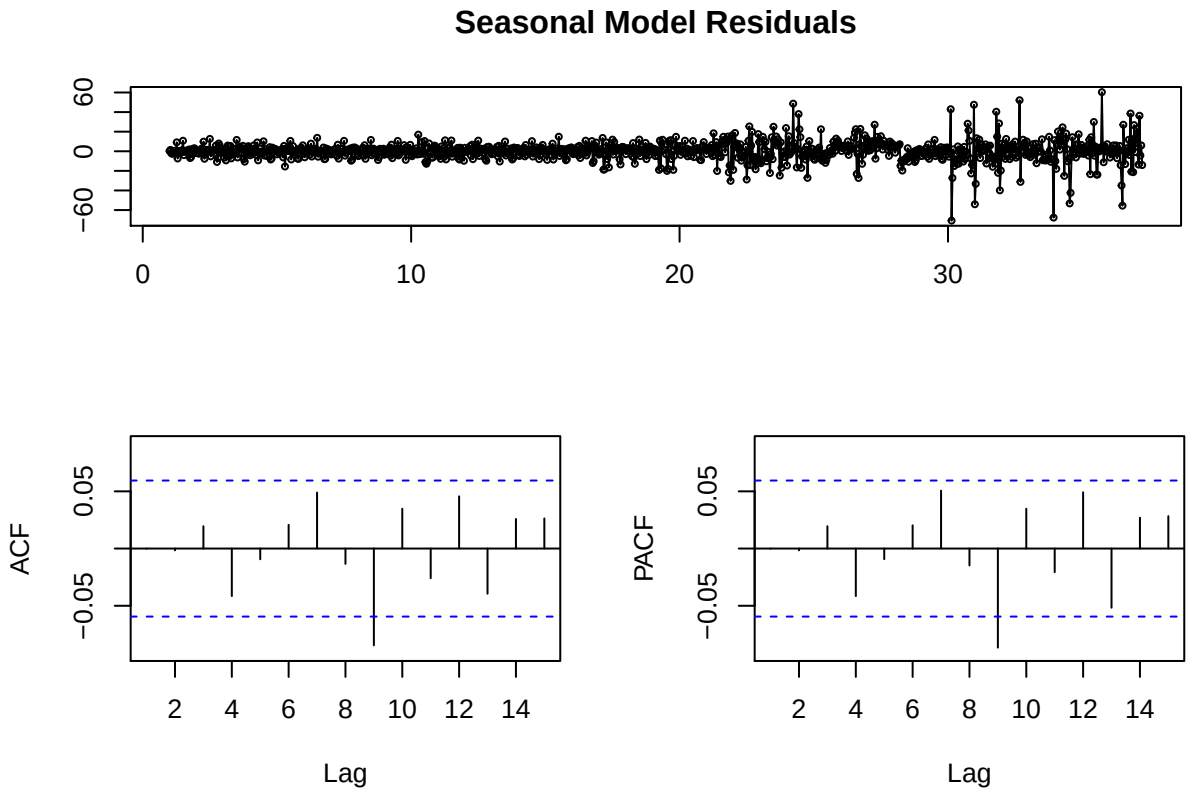



```
Pacf(close_d1, main='PACF for Differenced Series')
```



The final step is to fit the model and generated the forecast. In the following, forecast is generated for the next 90 days to provide some visibility on the graph.

```
  fit2 = arima(deseasonal_cls, order=c(1,1,9))
tsdisplay(residuals(fit2), lag.max=15, main='Seasonal Model Residuals')
```




```
fcast <- forecast(fit2, h=90)
plot(fcast)
```