Analysis of Google's closing stock pricing data with a simple forecast

```
library(forecast)
library(fpp2)

## Loading required package: ggplot2

## Loading required package: fma

## Loading required package: expsmooth
```

First, I loaded the data and now would like to understand the simple time series I'm working with

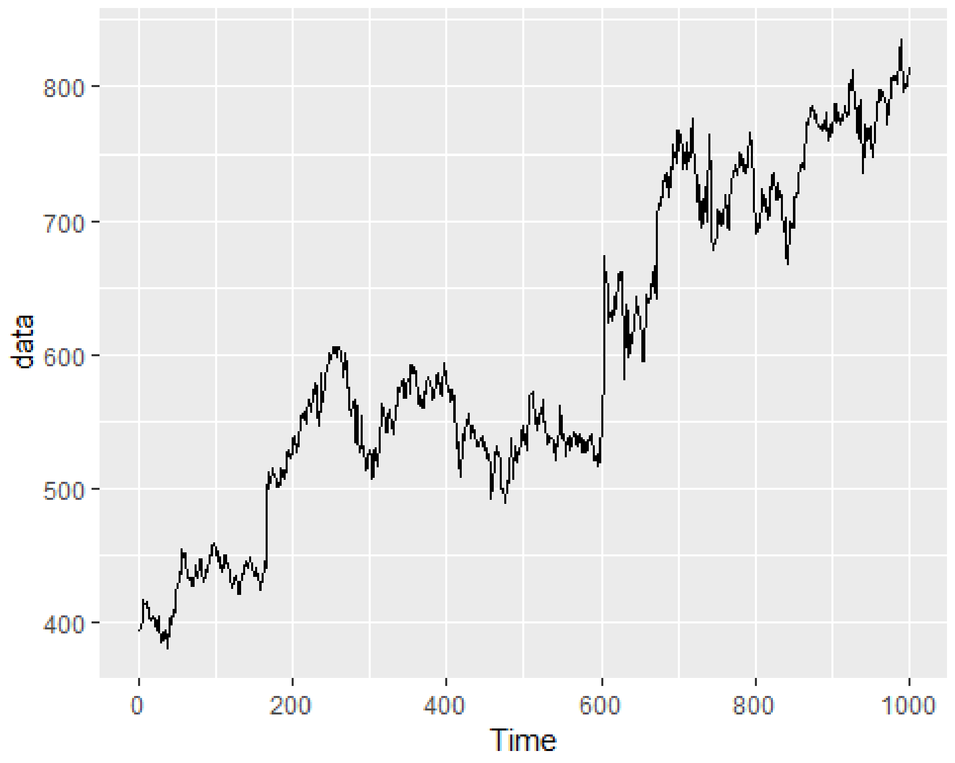
```
data <- fpp2::goog
head(data)

## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1] 392.8300 392.5121 397.3059 398.0113 400.4902 408.0957
```

From it's documentation - Closing stock prices of GOOG from the NASDAQ exchange, for 1000 consecutive trading days between 25 February 2013 and 13 February 2017. Adjusted for splits. goog200 contains the first 200 observations from goog.

R is already reading the data as a Time Series, however, I would have passed the data through a ts() function to shape it correctly if not

Let's plot the timeseries over time

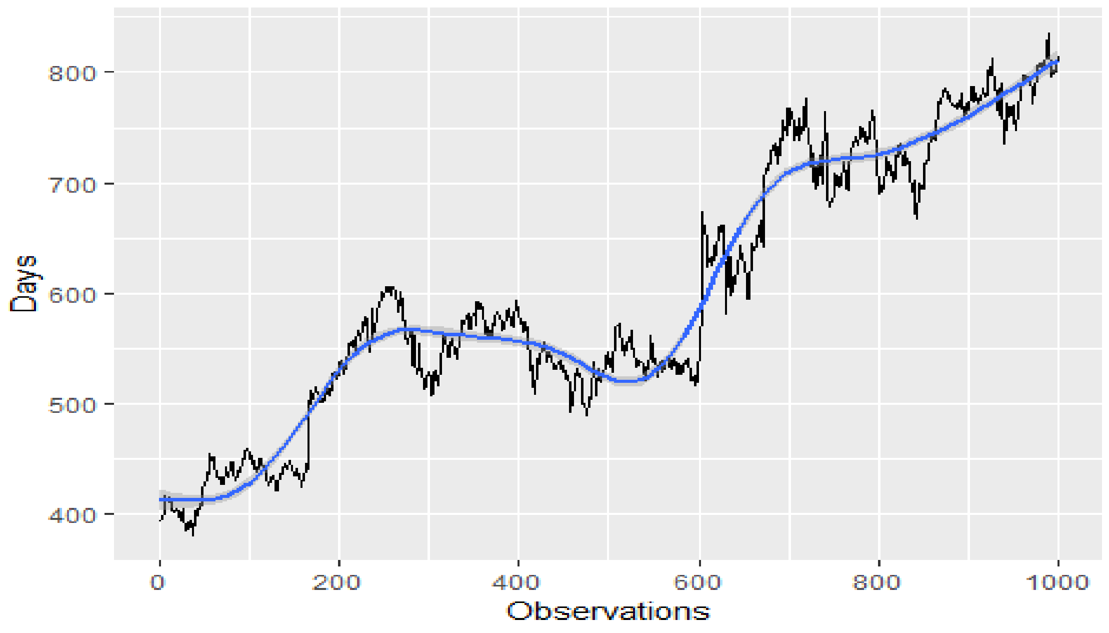
```r
autoplot(data)
```



Above we can see that throughout the time series, we have 1000 consecutive trading days, and the data plots in a linear upward trend

Let's add a smoothed Trend to the data

```r
autoplot(data) +
  geom_smooth() +
  labs("Google Ticks", y = "Days", x = "Observations")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



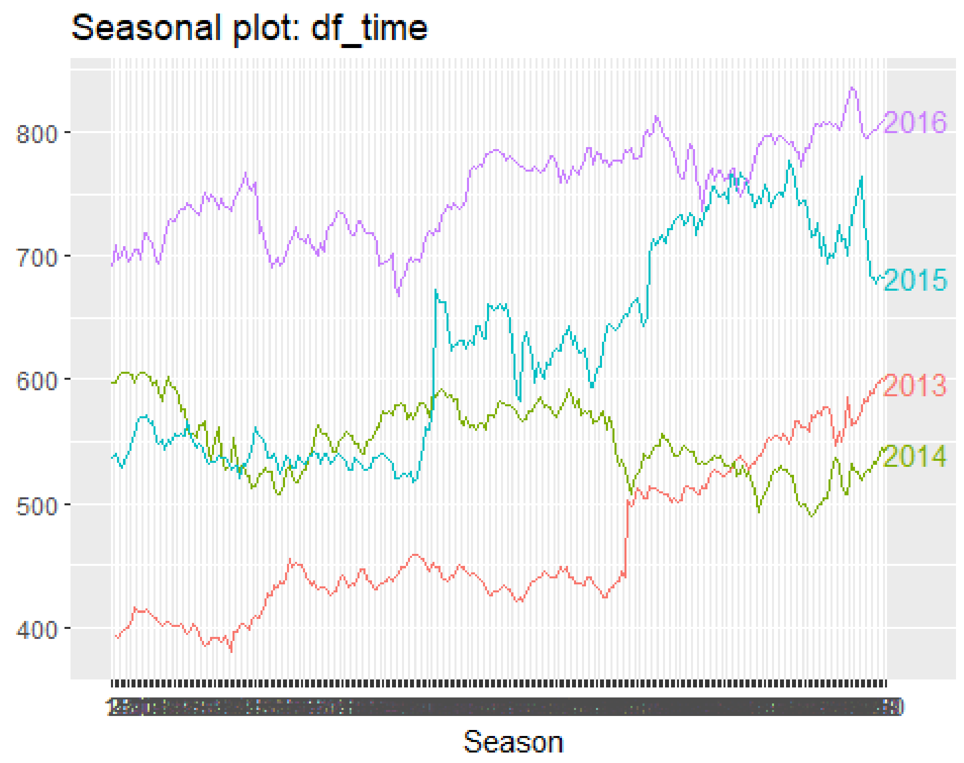
I'm curious what the min, max and other pieces of the data are. We can use the summary function for this.

```
summary(data)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   380.5   524.9   568.9   599.4   716.9   835.7
```

The Trend seems to be a long term increase, however, as it's a trading data there will be volatility over time With a longer outlook we may be able to see some seasonality and cycles occuring as well

Out of curiosity I'd like to check for seasonality using ggplot Below I convert use the ts() function to add dates to our data

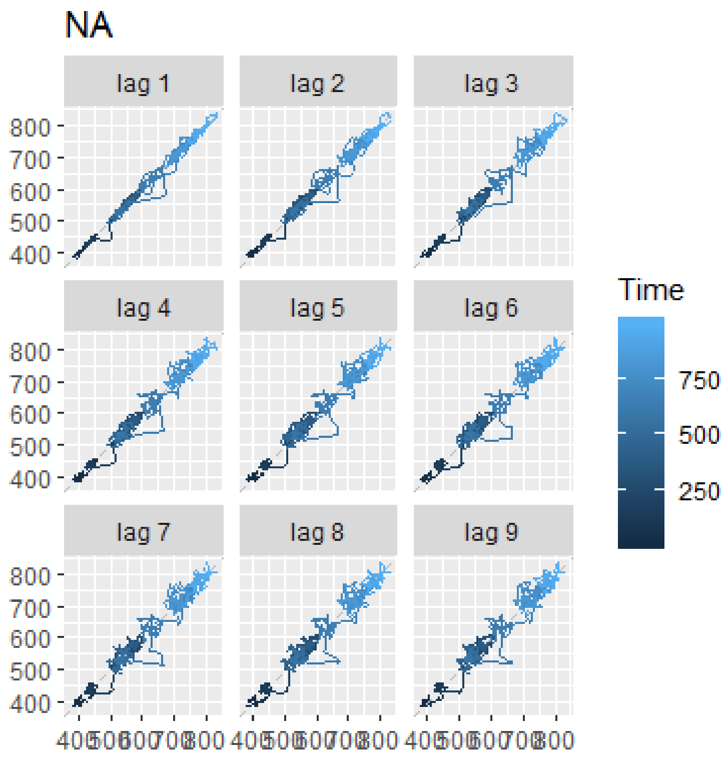
```
#TODO I'd like to figure out a way to have the TS not lose some data while
also staying in the timeframe
df_time <- ts(data, start = c(2013,02,25), end = c(2017,02,13), frequency =
249.5)
ggseasonplot(df_time, year.labels = TRUE)
```



As we can see, there's no clearly defined seasonality and the price movements are based off other variables

Below we plot yt against yt-1. This allows us to plot the time series against itself.

```
gglagplot(data)
```



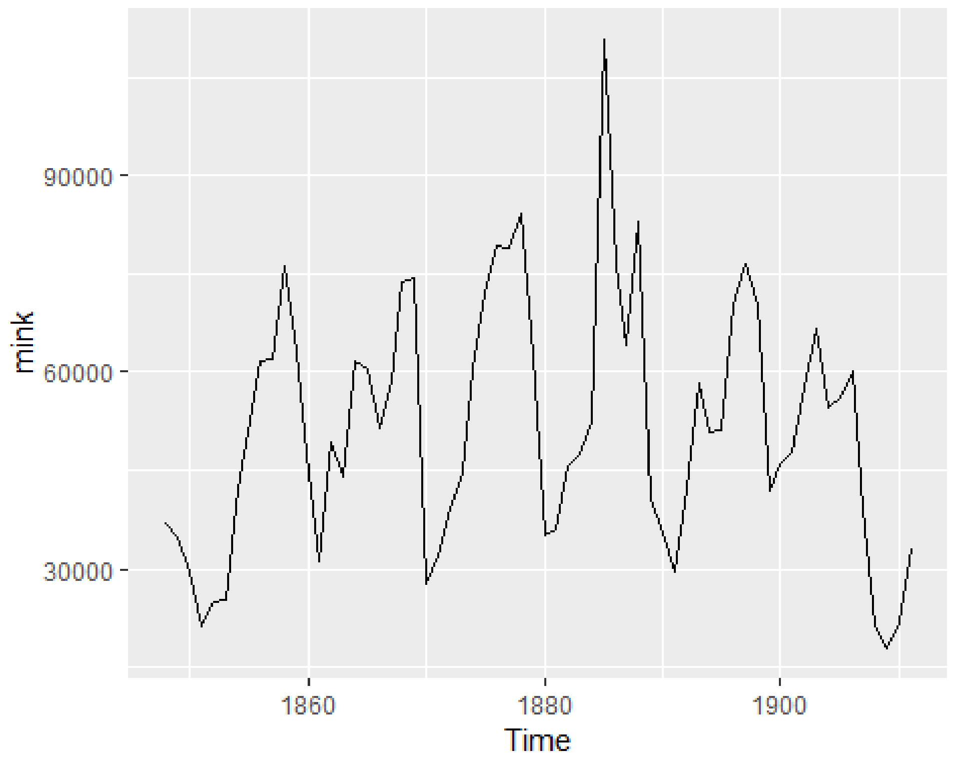
You can also plot correlation assocaited with lag plots. This is Auto Correlation. The Autocorrelation between yt and yt-k for different values of k can be written as:

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$

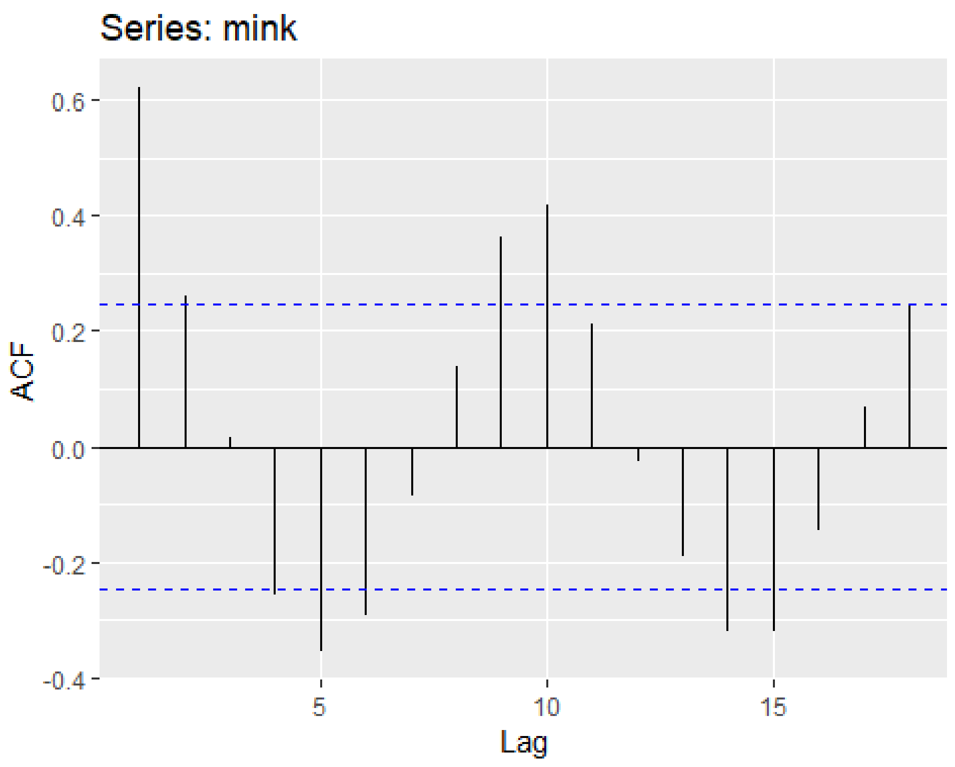
where T is the length of the time series. And similar to correlation, autocorrelation will always between +1 and -1.

I'll illustrate that below using the seasonal impact on the mink population every 10 years using the mink dataset that's built in.

```
autoplot(mink)
```

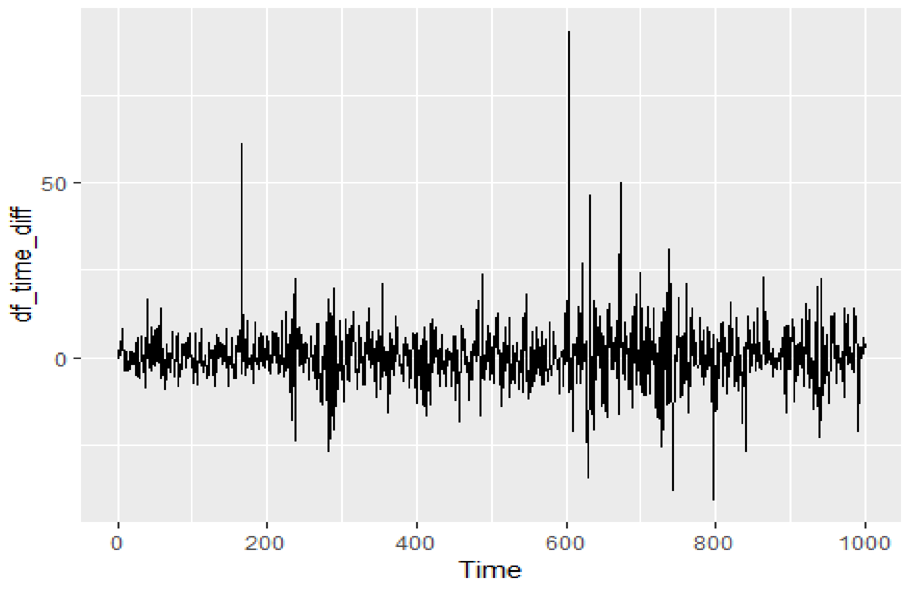


```
ggAcf(mink)
```

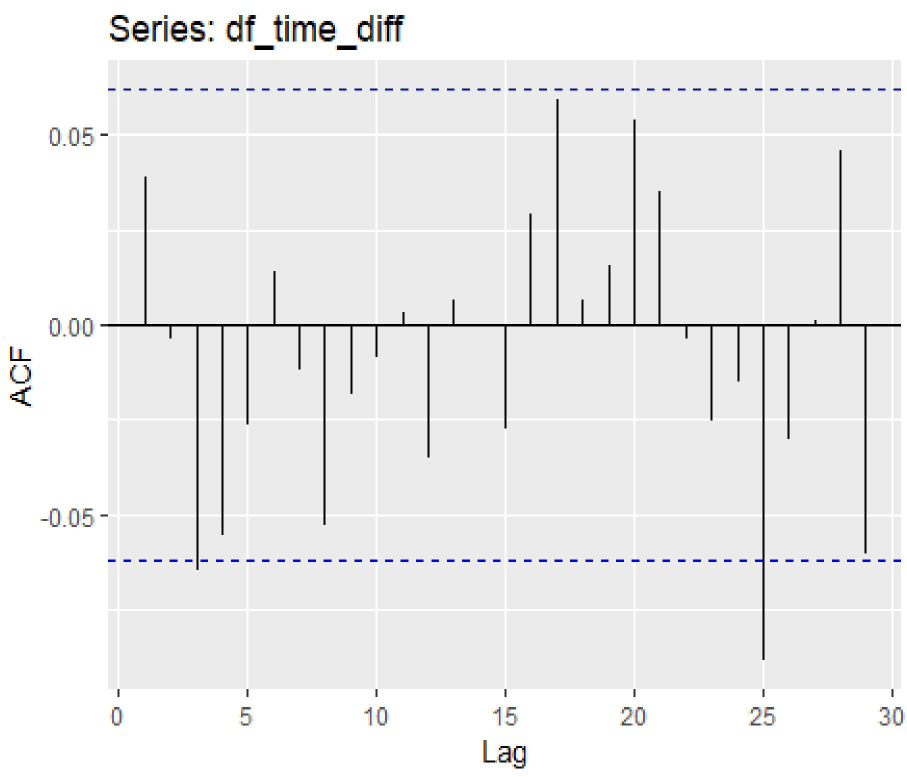


Applying the diff function to our data will give us daily changes in google stock prices. Let's see the daily changes and if there's any white noise We can also apply the ggAcf() function mentioned above

```
df_time_diff <- diff(goog)
autoplot(df_time_diff)
```



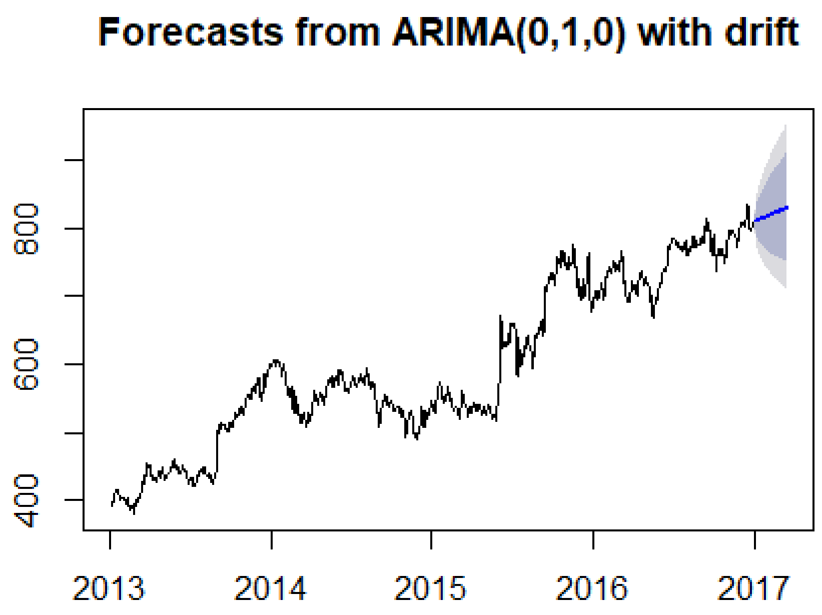
```
ggAcf(df_time_diff)
```



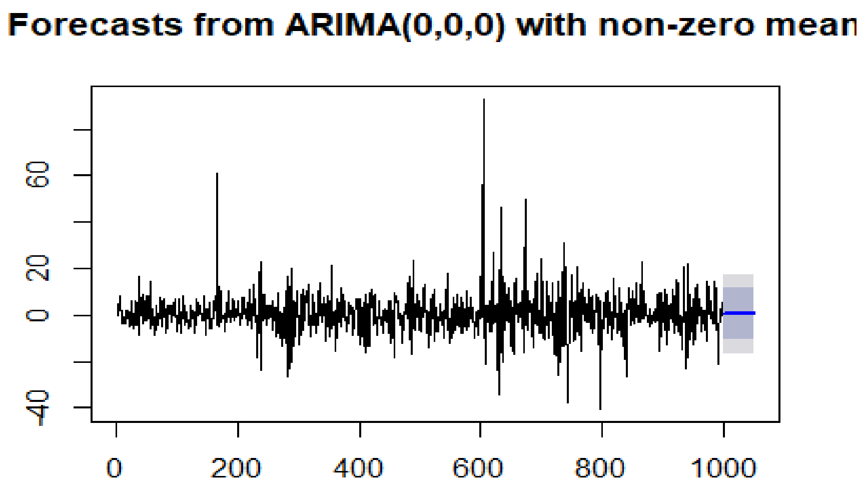
Let’s forecast our time series using an Auto ARIMA fit.

We will also forecast the diff variable we created above.

```
autoArimaFit <- auto.arima(df_time)
autoArimaFitDiff <- auto.arima(df_time_diff)
plot(forecast(autoArimaFit, h=50))
```



```
plot(forecast(autoArimaFitDiff, h=50))
```


Forecasts from ARIMA(0,0,0) with non-zero mean