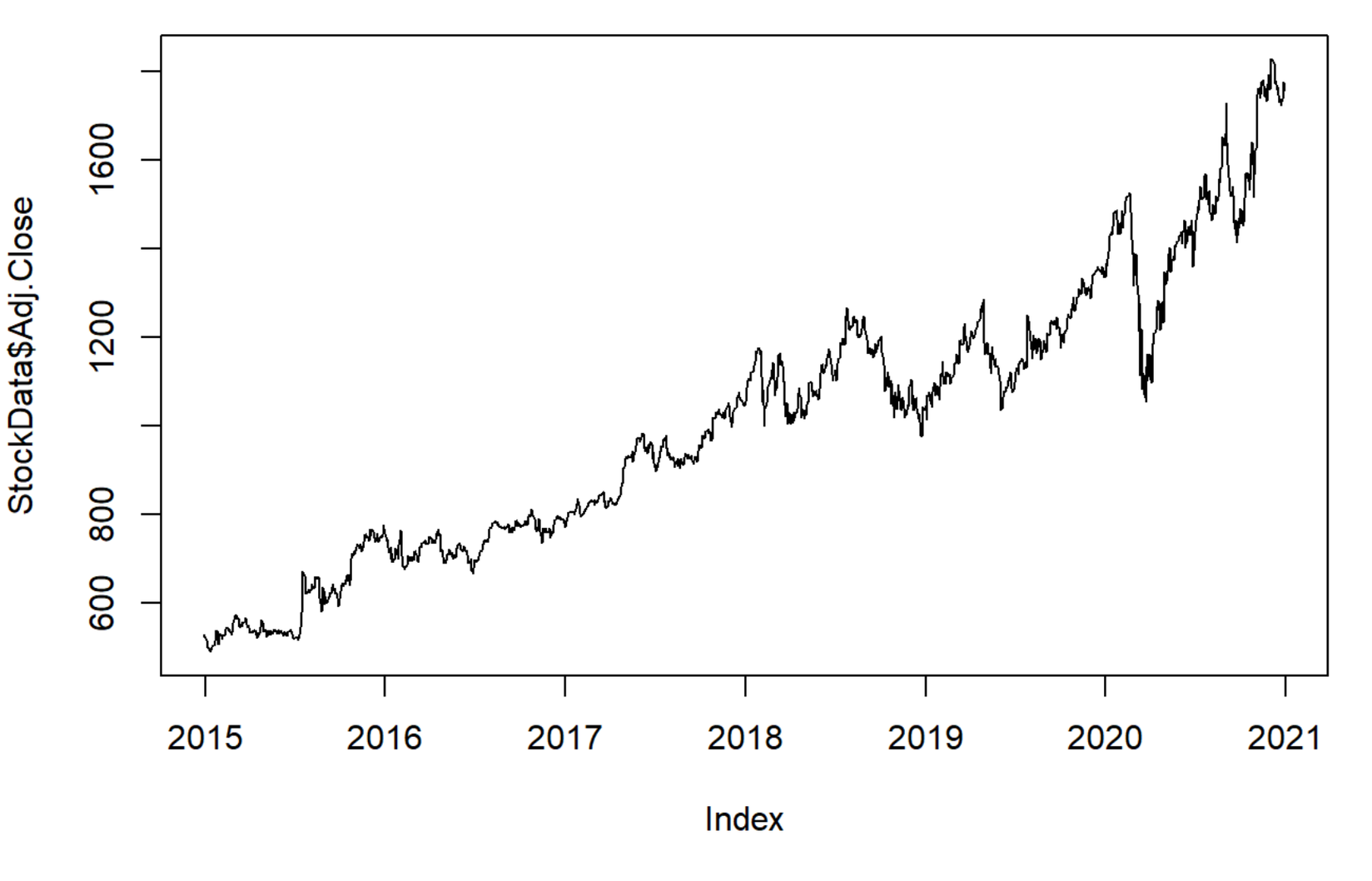
Volatility Forecasting

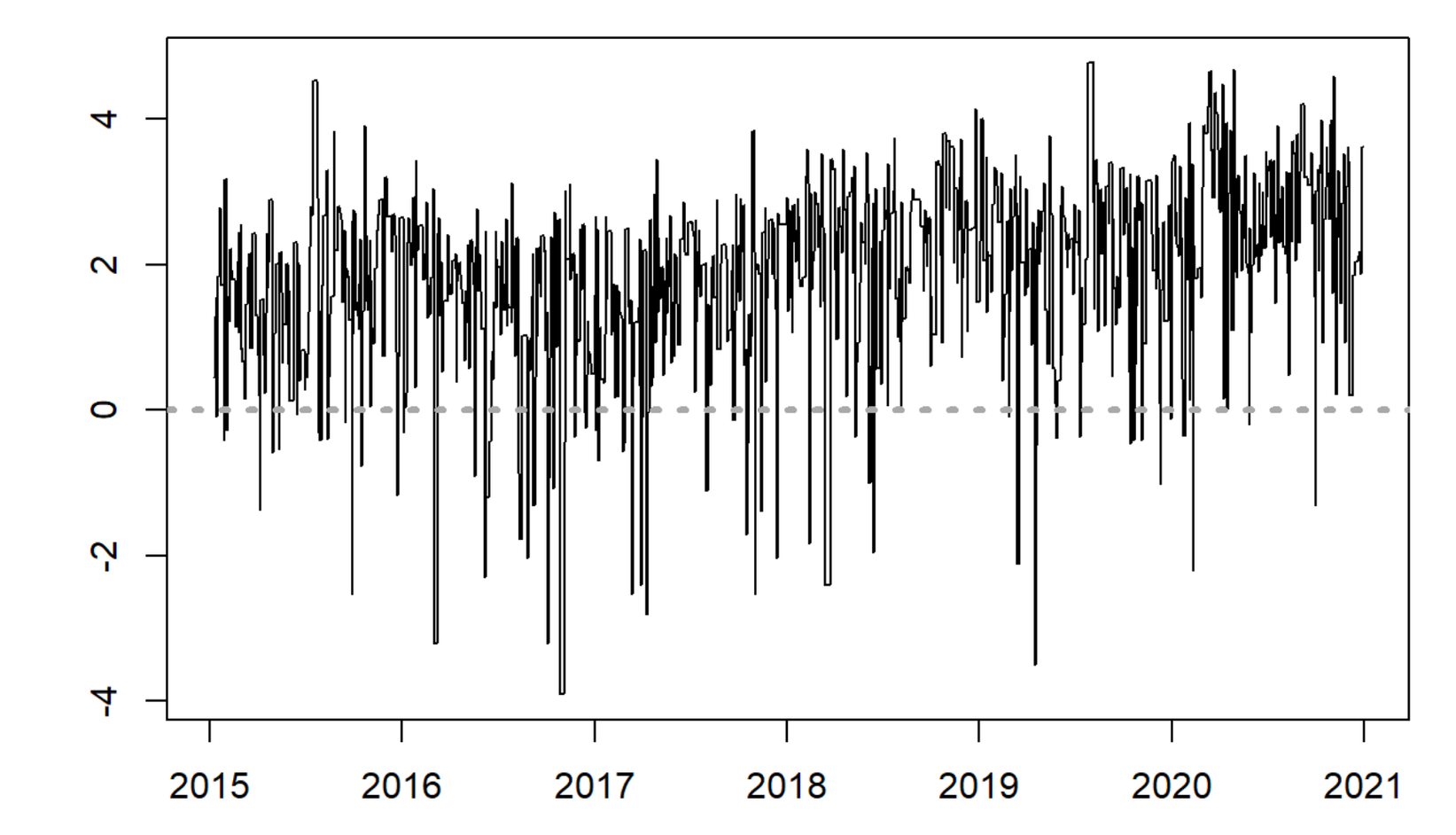
Convert data into *time series*

```r
################################  Convert data into time series  ######################################################
############
setwd("C:/Users/ddaya/OneDrive/Data Science Portfolio/Quantitative Finance")
# import dataset
library(zoo)
StockData <- read.zoo("GOOG.csv",header = TRUE, sep = ",",format="%Y-%m-%d")
PriceData<-ts(StockData$Adj.Close, frequency = 5) # Frequency=5 because it is set on only business days

# Note:
#Frequency = 12 means that data is at monthly level
# Frequency = 4 means that data is at quarterly level
# Frequency = 6 means data points for every 10 minutes of an hour
# Frequency = 5 means that data is at daily level business day

plot(StockData$Adj.Close,type="l") # plots stock movement through time
```



log the difference of the variable to get the percentage change

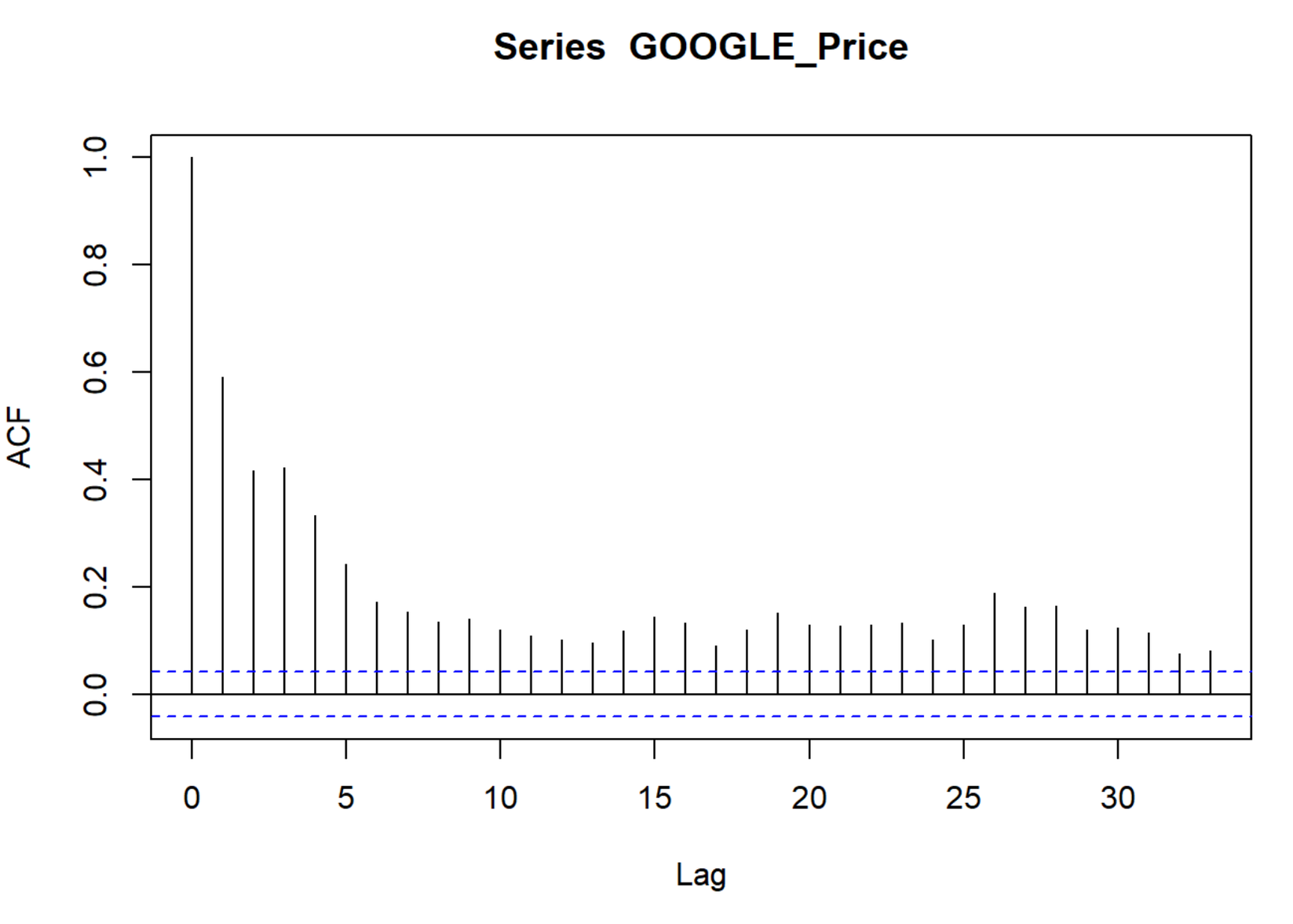
```r
# log and difference the variable
GOOGLE_Price<-log(diff(StockData$Adj.Close))
 library(zoo)
   GOOGLE_Price <- na.locf(GOOGLE_Price) # to handle NA values (from weekends and holidays)
plot(GOOGLE_Price, ylab="",xlab="")
abline(h=0, col="dark grey", lty=3, lwd=3)
```



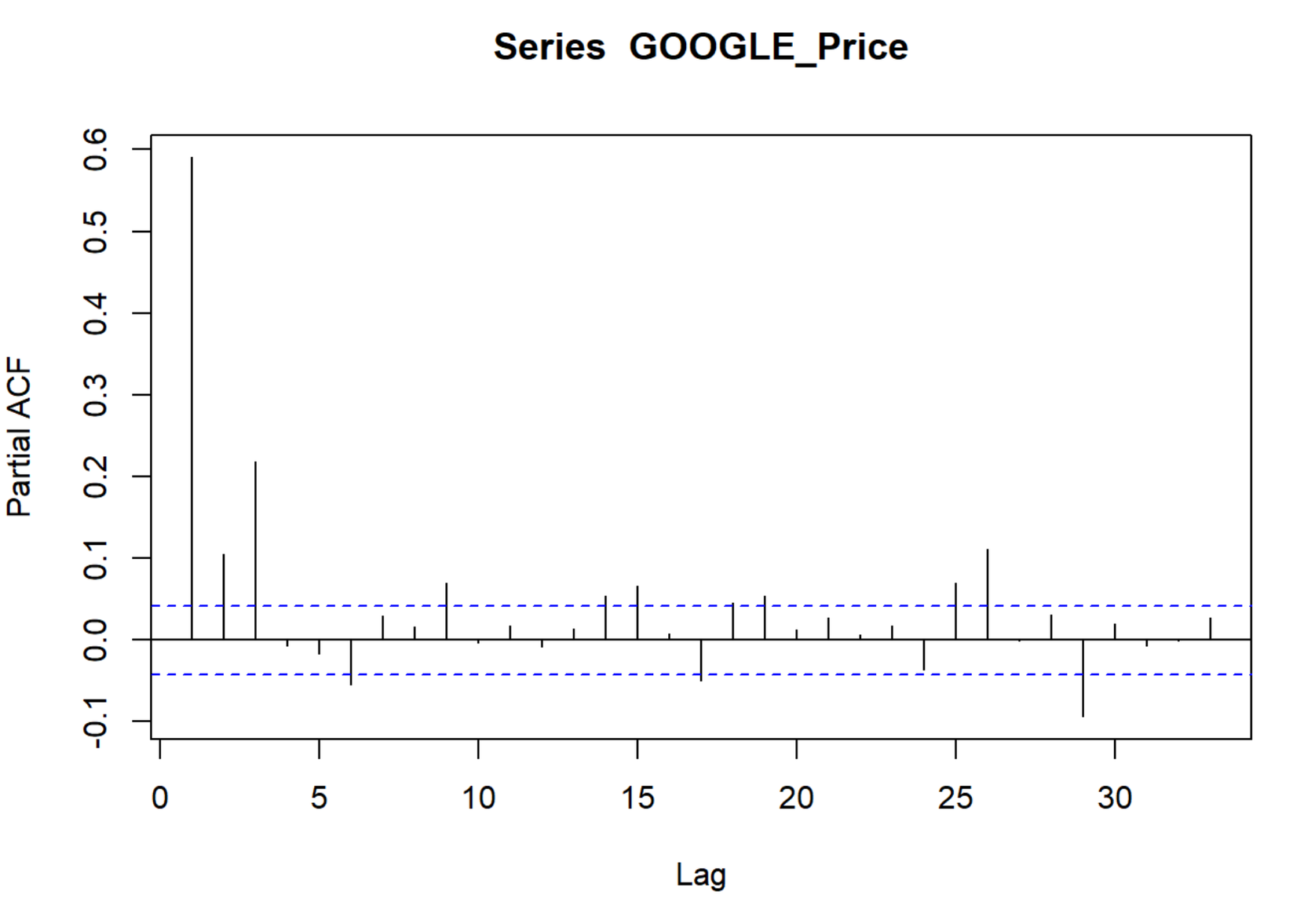
**Key Takeaway*: For the most part, GOOGLE stock performs at +2% – +4% above average whenever the market has a positive shock. Similarly, it almost always performs above average, at about +1.5% – +2%. Additionally, it has some very bad negative performance, at around -3% – -4% when the market suffers a negative shock, but it immediately recuperates (by performing above average) the next day.

Diagnostics

```r
acf(GOOGLE_Price, na.action=na.pass)
```




```r
pacf(GOOGLE_Price, na.action=na.pass)
```



Note: Our *ACF* portrays a non-stationary process. For an AR(p) model we need and ACF plot that decays slowly, and a PACF plot that cuts off after x lags; which would then lead us to conclude that the order of AR(p=x). We do not have any of those conditions, so we can't operate an AR(p) model.

Our ACF decays slowly and ends at 3. Similarly, our PACF decays suddenly after the first spike, so it could be ARIMA(3,1,0)

ARIMA Model

```r
ar3<-arima(GOOGLE_Price, c(3,1,0))
ar3
```

```
##
## Call:
## arima(x = GOOGLE_Price, order = c(3, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3
##       -0.4635  -0.3397  -0.1493
## s.e.   0.0289   0.0325   0.0370
##
## sigma^2 estimated as 1.201:  log likelihood = -2358.87,  aic = 4725.75
```

GARCH Model

```
##
## *---------------------------------*
## *       GARCH Model Spec          *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ------------------------------------
## GARCH Model      : sGARCH(1,1)
## Variance Targeting   : FALSE
##
## Conditional Mean Dynamics
## ------------------------------------
## Mean Model       : ARFIMA(3,0,0)
## Include Mean     : TRUE
## GARCH-in-Mean        : FALSE
##
## Conditional Distribution
## ------------------------------------
## Distribution :  norm
## Includes Skew    :  FALSE
## Includes Shape   :  FALSE
## Includes Lambda  :  FALSE
```

```r
GARCH_Fit<-ugarchfit(GARCH_Model, data=GOOGLE_Price)
GARCH_Fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      1.865536    0.061208 30.47884 0.000000
## ar1     0.457560    0.032453 14.09912 0.000000
## ar2     0.064042    0.032981  1.94178 0.052164
## ar3     0.049186    0.025587  1.92232 0.054565
## omega   0.784056    0.120344  6.51512 0.000000
## alpha1  0.240069    0.050723  4.73295 0.000002
## beta1   0.114800    0.119318  0.96213 0.335984
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      1.865536    0.076108 24.51177 0.000000
## ar1     0.457560    0.036269 12.61569 0.000000
## ar2     0.064042    0.034002  1.88349 0.059634
## ar3     0.049186    0.025930  1.89687 0.057845
## omega   0.784056    0.173369  4.52248 0.000006
## alpha1  0.240069    0.078205  3.06974 0.002142
## beta1   0.114800    0.166008  0.69153 0.489231
##
## LogLikelihood : -2221.593
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.9616
## Bayes        2.9863
## Shibata      2.9615
## Hannan-Quinn 2.9708
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.462  0.1166
## Lag[2*(p+q)+(p+q)-1][8]     5.257  0.1117
## Lag[4*(p+q)+(p+q)-1][14]    9.699  0.1455
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1727  0.6777
## Lag[2*(p+q)+(p+q)-1][5]    0.5742  0.9457
## Lag[4*(p+q)+(p+q)-1][9]    1.4640  0.9585
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.05281 0.500 2.000  0.8182
## ARCH Lag[5]   0.80596 1.440 1.667  0.7913
## ARCH Lag[7]   1.28745 2.315 1.543  0.8628
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  6.2269
## Individual Statistics:
## mu     3.73786
## ar1    0.17723
## ar2    0.09460
## ar3    0.31019
## omega  0.07028
## alpha1 0.07264
## beta1  0.06565
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:         1.69 1.9 2.35
## Individual Statistic:    0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.7164 0.4738
## Negative Sign Bias  0.6830 0.4947
## Positive Sign Bias  1.0428 0.2972
## Joint Effect        1.6024 0.6588
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     220.4    2.854e-36
## 2    30     242.1    1.740e-35
## 3    40     265.5    1.760e-35
## 4    50     283.6    9.083e-35
##
##
## Elapsed time : 0.7886171
```

Notice:

Based on the output, the estimated mean of the series is μ=3.73786, and the estimated variance is $\omega + \alpha_1 + \beta_1$ =0.07028+0.07264+0.06565=0.20857

Variance volatility series

```r
# convert the fitted values into time series
library(readr)
    StockData <- read_csv("GOOG.csv")
      library(zoo)
          dt = as.Date(StockData$Date, format="%Y-%m-%d")
            Stockdataz = zoo(x=GARCH_Fit@fit$sigma^2, order.by=dt)
```

```r
# view the first values of our new time series
              head(Stockdataz)
```

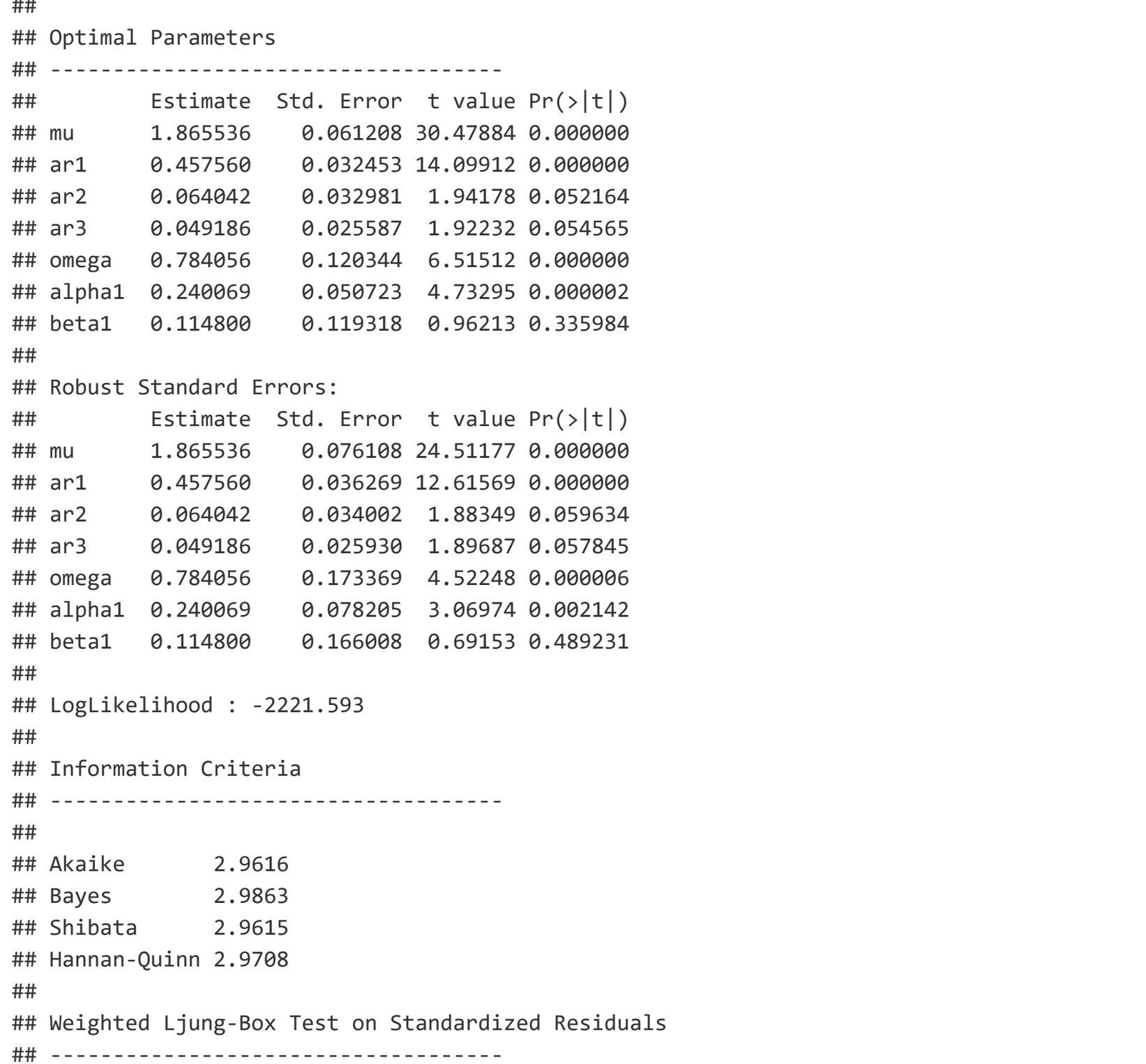
```
## 2014-12-30 2014-12-31 2015-01-02 2015-01-05 2015-01-06 2015-01-07
##  1.2034182  1.2034182  1.2034182  1.4000630  0.9570711  0.8964175
```

```r
# view the last values of our new time series
tail(Stockdataz)
```

```
## 2020-12-21 2020-12-22 2020-12-23 2020-12-24 2020-12-28 2020-12-29
##  1.2034182  1.2034182  1.2034182  1.4000630  0.9570711  0.8964175
```

Visualize our model

```r
plot(Stockdataz, xlab="", ylab="", main=" GOOGLE Volatility - GARCH(1,1)")
```



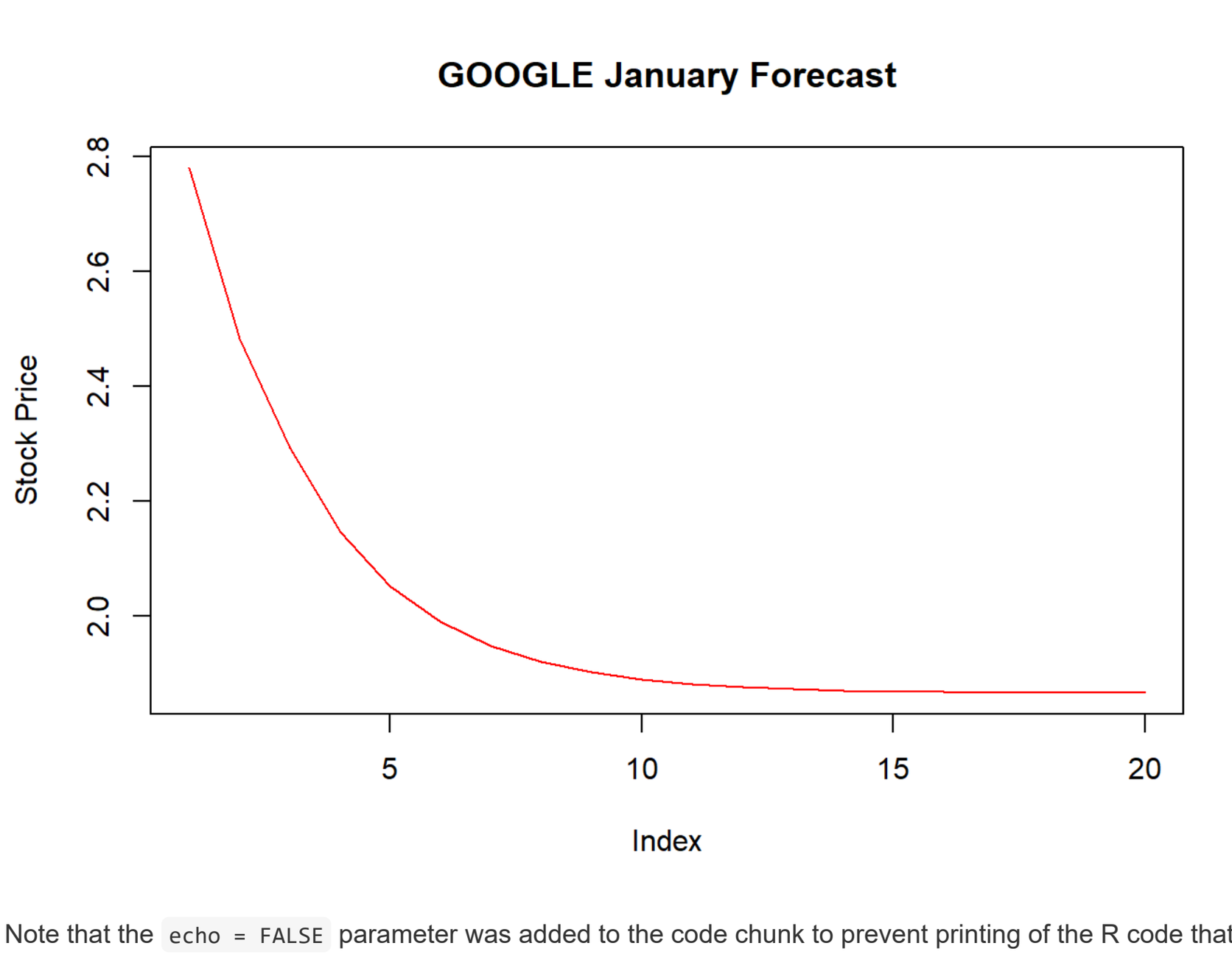
Forecast

We will forecast the percentage change in GOOGLE stock for January

```r
FutureForecast=ugarchforecast(GARCH_Fit, n.ahead = 20)
FutureForecast
```

```
##
## *------------------------------------*
## *       GARCH Model Forecast         *
## *------------------------------------*
## Model: sGARCH
## Horizon: 20
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=2020-12-29]:
##      Series Sigma
## T+1   2.779 1.085
## T+2   2.482 1.096
## T+3   2.292 1.100
## T+4   2.145 1.102
## T+5   2.051 1.102
## T+6   1.989 1.102
## T+7   1.948 1.102
## T+8   1.920 1.102
## T+9   1.902 1.102
## T+10  1.890 1.102
## T+11  1.882 1.102
## T+12  1.876 1.102
## T+13  1.873 1.102
## T+14  1.870 1.102
## T+15  1.869 1.102
## T+16  1.868 1.102
## T+17  1.867 1.102
## T+18  1.866 1.102
## T+19  1.866 1.102
## T+20  1.866 1.102
```

```r
    'Stock Price'<-as.data.frame(FutureForecast@forecast)
          `Stock Price`<-`Stock Price`$X2020.12.29.1
            plot(`Stock Price`, col='red', type='l')
              title("GOOGLE January Forecast")
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.