Assignment 3

ANS-1

a) Not Stationary as mean is not constant (it is increasing).

b) Stationary as all conditions of stationarity are satisfied.

c) Not Stationary as variance is different in the middle portion.

d) Not Stationary as the data is periodic.

e) Not Stationary as mean is not constant (is is decreasing).

f) Not Stationary as there is a peak at the beginning.

g) Stationary as the data is periodic with irregular intervals.

h) Not Stationary as the data is periodic.

i) Not Stationary as mean is increasing.

ANS-2

```
library('forecast')
```

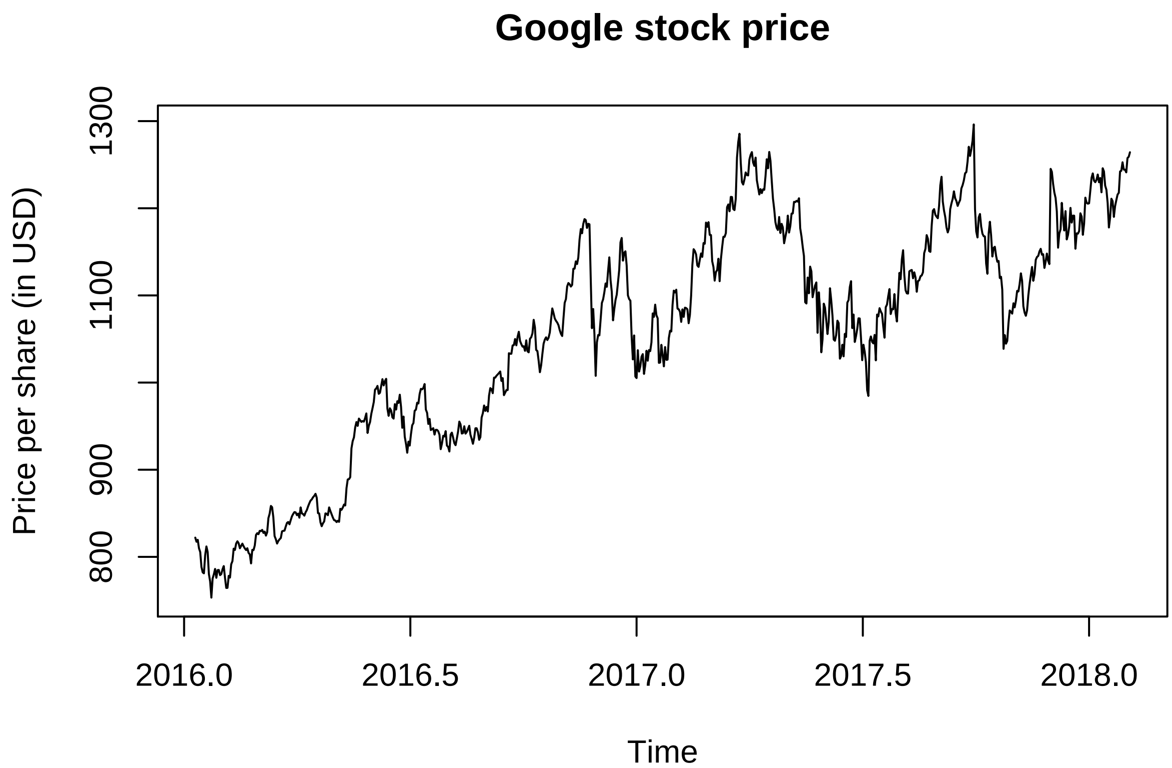
```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```
library('tseries')
```

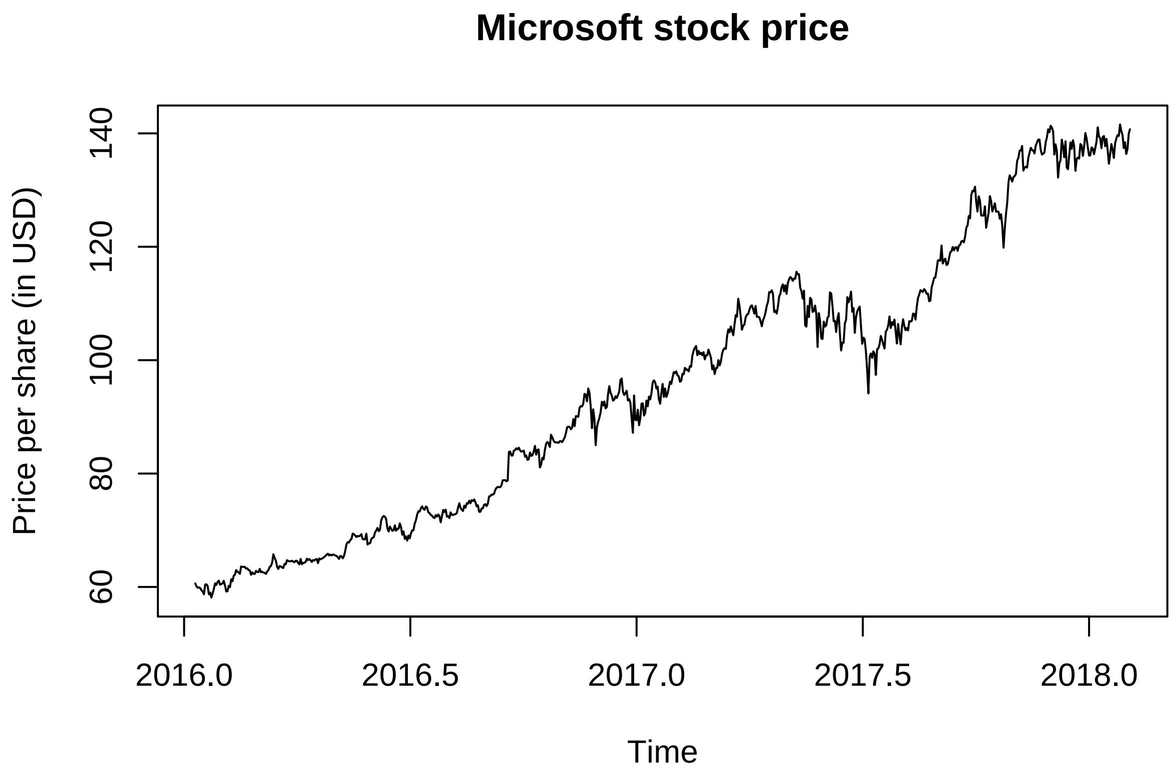
```
amazon=read.csv('amazon.csv')
amazon=amazon[which(amazon$date=='26/10/2016'):nrow(amazon),'close']
amazon=ts(amazon,start=c(2016,10,26),frequency=365)

google=read.csv('google.csv')
google=google[which(google$date=='26/10/2016'):nrow(google),'close']
google=ts(google,start=c(2016,10,26),frequency=365)

micro=read.csv('microsoft.csv')
```

```r
micro=micro[which(micro$date=='26/10/2016'):nrow(micro),'close']
micro=ts(micro,start=c(2016,10,26),frequency=365)

plot(amazon,type = 'l',ylab='Price per share (in USD)',main = 'Amazon stock price')
```


```r
plot(google,type = 'l',ylab='Price per share (in USD)',main = 'Google stock price')
```




```r
plot(micro,type = 'l',ylab='Price per share (in USD)',main = 'Microsoft stock price')
```

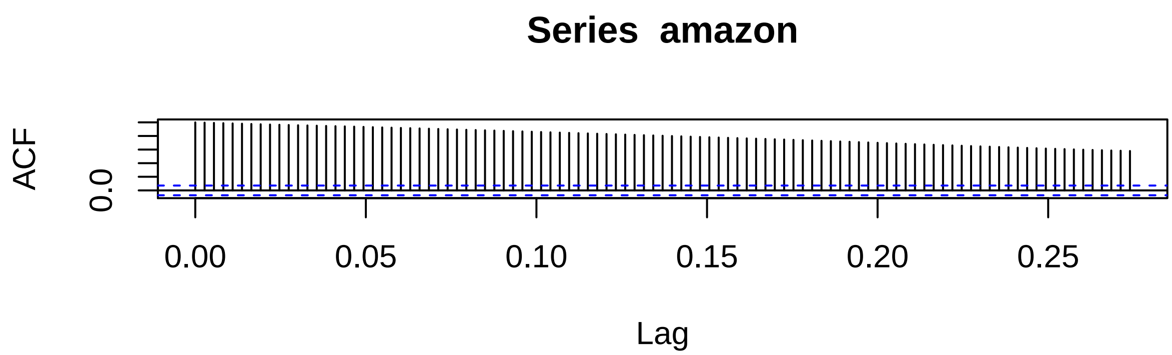


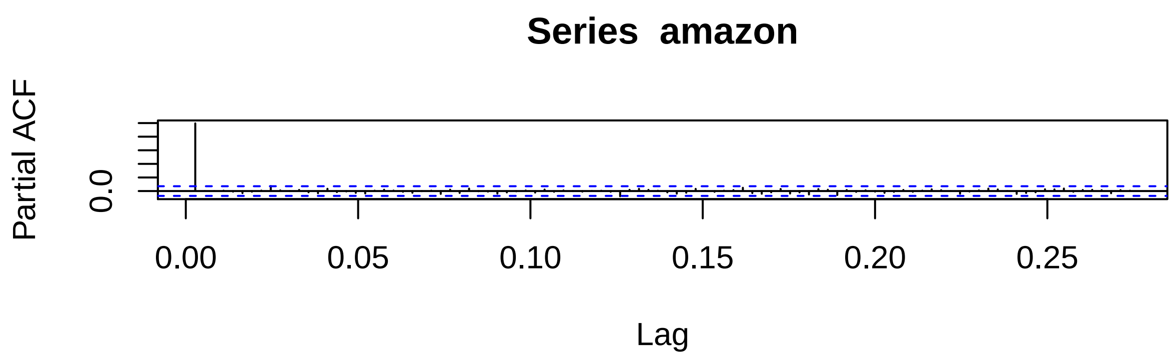
a) All the stocks are performing very well as they are showing an upward trend. Microsoft's stocks are very stable, whereas Google's stock shows a small periodic decline and generally has higher variance.

Amazon's stock value is fluctuating around a constant value for the past 1 year and hasn't seen growth recently.

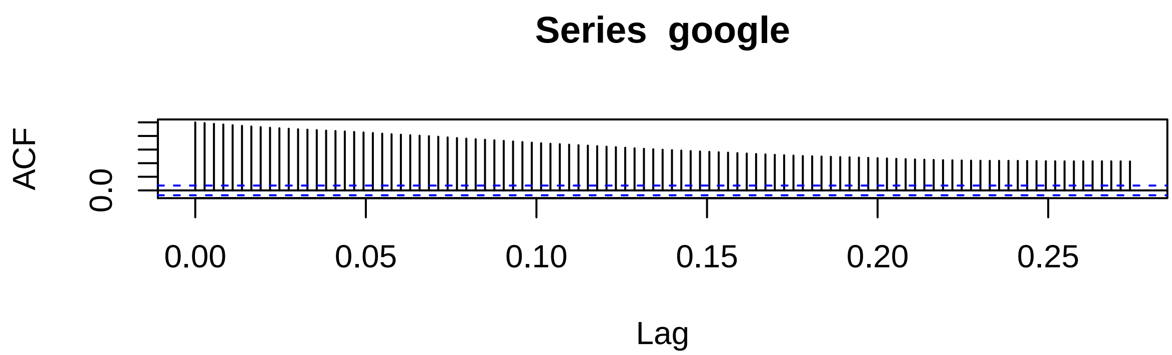
b)

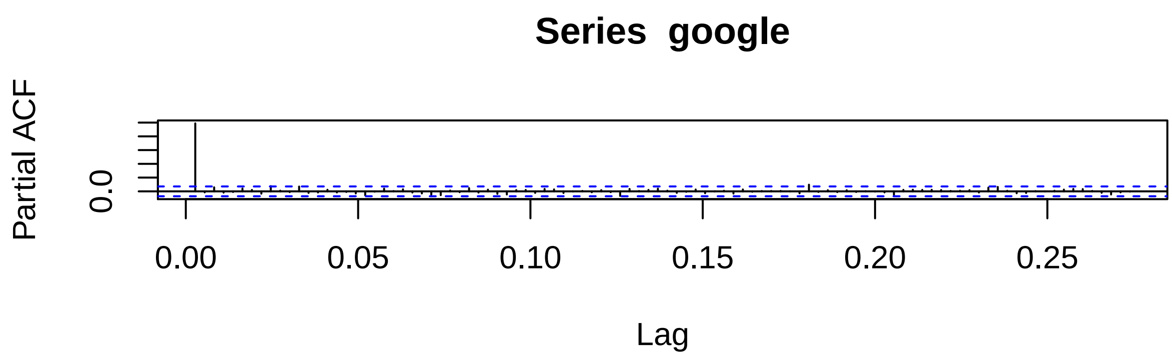
```
acf_amazon=acf(amazon,lag.max=100, plot=FALSE)
pacf_amazon=pacf(amazon,lag.max=100, plot=FALSE)
par(mfrow=c(2,1))
plot(acf_amazon)
plot(pacf_amazon)
```



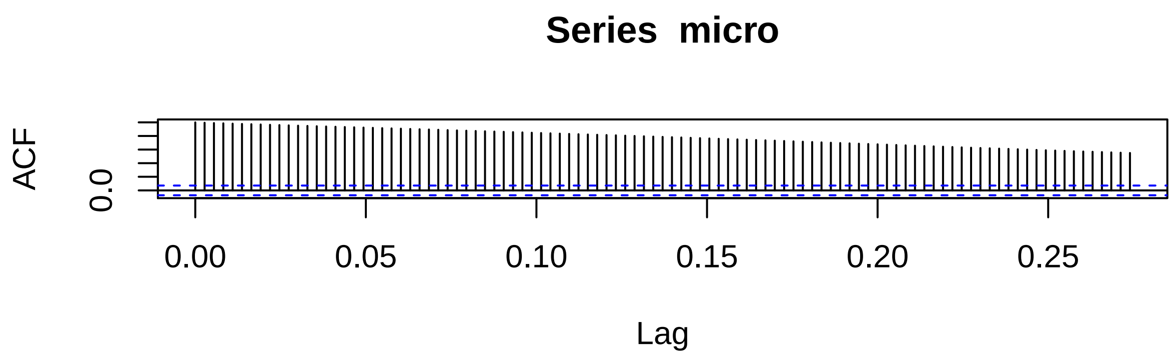


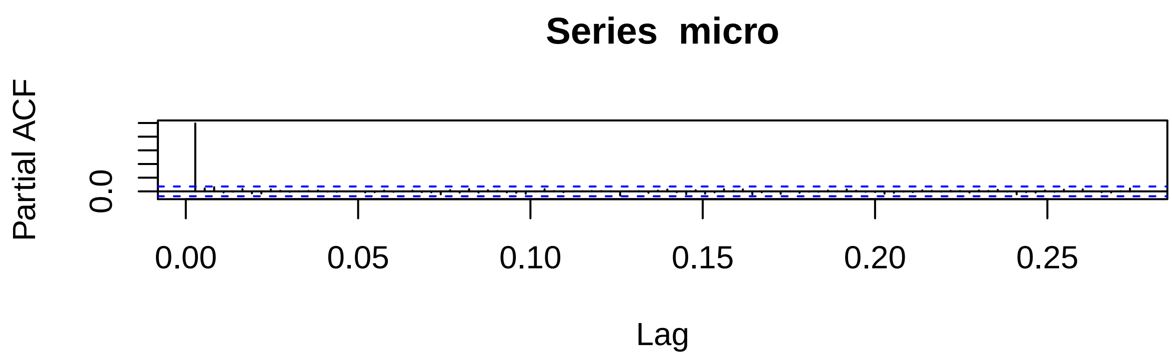

```r
acf_google=acf(google,lag.max=100, plot=FALSE)
pacf_google=pacf(google,lag.max=100, plot=FALSE)
par(mfrow=c(2,1))
plot(acf_google)
plot(pacf_google)
```






```r
acf_micro=acf(micro,lag.max=100, plot=FALSE)
pacf_micro=pacf(micro,lag.max=100, plot=FALSE)
par(mfrow=c(2,1))
plot(acf_micro)
plot(pacf_micro)
```

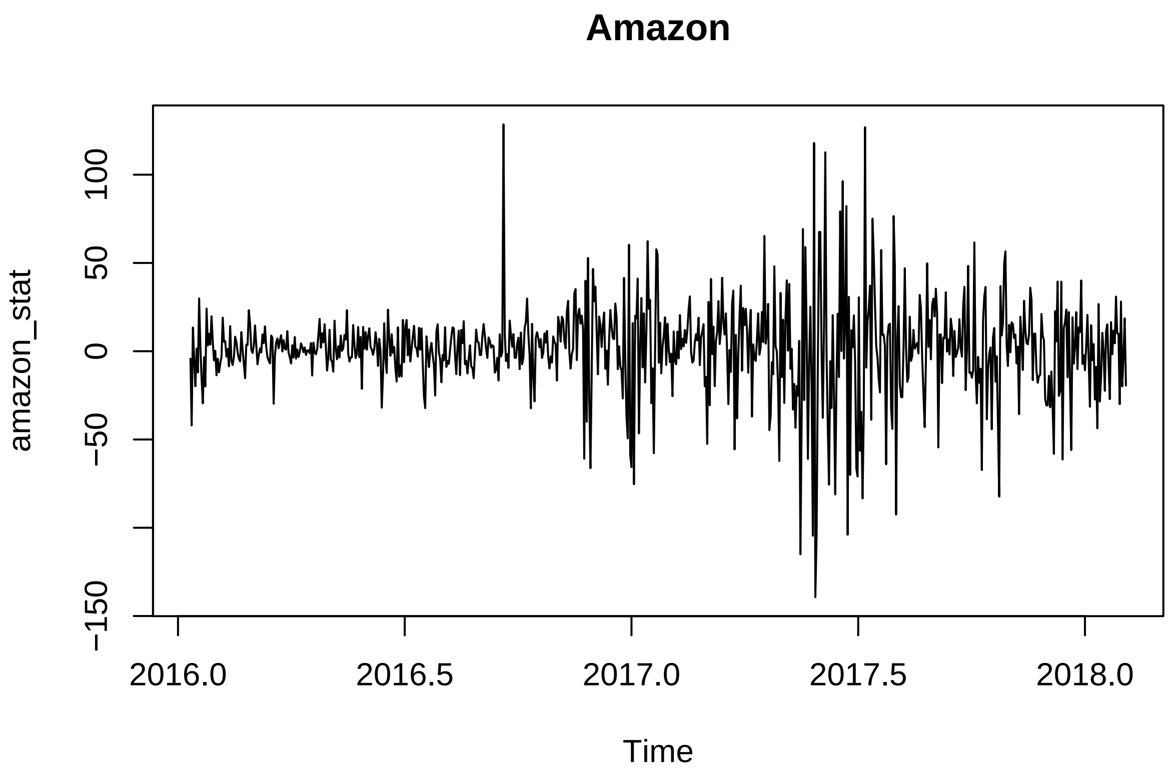




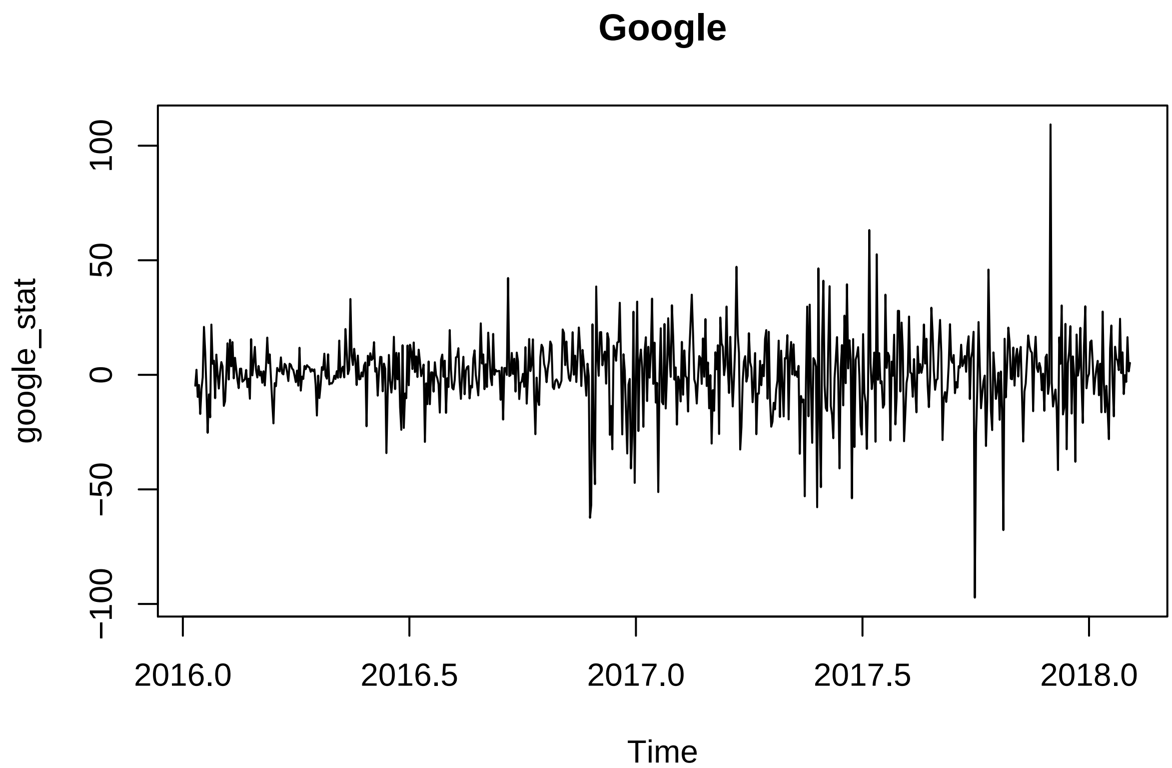
In all of the above cases we see that auto correlation is decreasing with increasing lags. In partial autocorrelation we see that the autocorrelation is high at lag=1. This implies that value of y at timt=t depends only on the value at time=(t-1). q=1 for our AR model.

c)

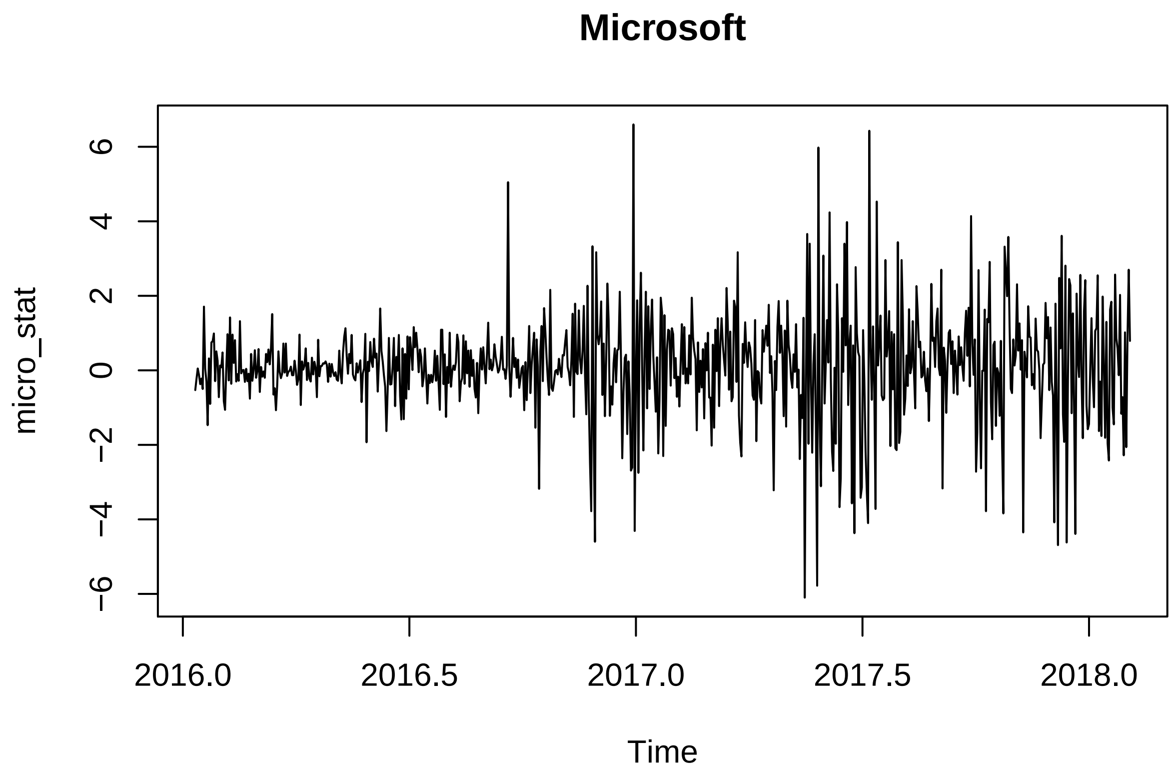
```
amazon_stat=diff(amazon,differences = 1)
google_stat=diff(google,differences = 1)
micro_stat=diff(micro,differences = 1)
plot(amazon_stat, main='Amazon')
```




```r
plot(google_stat, main='Google')
```




```r
plot(micro_stat, main='Microsoft')
```



All of the above can be made approximately stationary by differencing the data by lag=1.

d) Learn only from first 600 data points and then use it to predict future samples. Finally, calculate mae error.

```
# Amazon prices
amazon_fit=arima(amazon[1:600],order=c(1,1,1))
forecast_amazon=forecast(amazon_fit,h=155)
sprintf('MAE error on future samples of Amazon: %s',sum(abs(forecast_amazon[["mean"]]-amazon[601:755])))
```

```
## [1] "MAE error on future samples of Amazon: 17409.0814325496"
```

```
google_fit=arima(google[1:600],order=c(1,1,1))
forecast_google=forecast(google_fit,h=155)
sprintf('MAE error on future samples of Google: %s',sum(abs(forecast_google[["mean"]]-google[601:755])))
```

```
## [1] "MAE error on future samples of Google: 6787.32233402785"
```

```
micro_fit=arima(micro[1:600],order=c(1,1,1))
forecast_micro=forecast(micro_fit,h=155)
sprintf('MAE error on future samples of Microsoft: %s',sum(abs(forecast_micro[["mean"]]-micro[601:755])))
```

```
## [1] "MAE error on future samples of Microsoft: 2368.64804756622"
```

ANS-3

```r
# load the data
library(mlbench)
data("BreastCancer")
cancer=data.frame(data.matrix(BreastCancer[1:nrow(BreastCancer),-1]))
cancer[is.na(cancer)]<-1      # Replace NA values with mode only bare.nuclei columns has na values
```

```r
model=glm(as.factor(Class)~.,data=cancer, family='binomial')

# predicted=predict.glm(model,cancer,type='response')
# predicted=ifelse(predicted>0.5,2,1)
```

```r
# [501:699,1:9]
# (predicted>0.5)==cancer[501:699,10]
# table(cancer[501:699,10],predicted>0.5)

# sum(predicted==cancer[1:699,10])

summ=summary.glm(model)
summ
```

```
##
## Call:
## glm(formula = as.factor(Class) ~ ., family = "binomial", data = cancer)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3324  -0.1269  -0.0658   0.0272   2.3958
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -9.72056    1.07329  -9.057  < 2e-16 ***
## Cl.thickness     0.53489    0.13482   3.967 7.27e-05 ***
## Cell.size        0.01162    0.19327   0.060  0.95205
## Cell.shape       0.32323    0.21391   1.511  0.13077
## Marg.adhesion    0.23765    0.11672   2.036  0.04175 *
## Epith.c.size     0.05850    0.15252   0.384  0.70131
## Bare.nuclei      0.42816    0.09054   4.729 2.25e-06 ***
## Bl.cromatin      0.41228    0.15694   2.627  0.00862 **
## Normal.nucleoli  0.15826    0.10416   1.519  0.12866
## Mitoses          0.53953    0.30397   1.775  0.07591 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 900.53  on 698  degrees of freedom
## Residual deviance: 113.10  on 689  degrees of freedom
## AIC: 133.1
##
## Number of Fisher Scoring iterations: 8
```

We fit a logit link function to the data as the output variable is a binary variable and obtain the summary as above.

Estimate column tells us the coefficients of each independent variable including a bias term.

Std. Error is the standard deviation of that variable.

z-value is obtained by dividing the estimate and the std. error.

Pr(>|z|) tells the probability of obtaining that z value.

Variables with *'s at the end denote that they are statistically significant with a significance level of 0.05.

```r
prob=summ[["coefficients"]][31:40]
oddsratio=(prob/(1-prob))**2
oddsratio
```

```
## [1] 1.803638e-38 5.279680e-09 3.942752e+02 2.263327e-02 1.898273e-03
## [6] 5.512622e+00 5.079552e-12 7.551718e-05 2.180144e-02 6.746989e-03
```

Variables with high odds ratio also have low value of coefficients meaning these variables are less important in predicting the output class.

More the important a variable is in predicting the class less the odds ratio it has.

```r
# mean=summ[["coefficients"]][1:10]
std=summ[["coefficients"]][11:20]

low=oddsratio-(std*1.96)/(sqrt(nrow(cancer)))
high=oddsratio+(std*1.96)/(sqrt(nrow(cancer)))

low
```

```
## [1]  -0.079566975  -0.009994731 394.260892682   0.006775216  -0.006754885
## [6]   5.501315250  -0.006711759  -0.011559318   0.014079600  -0.015787411
```

```r
high
```

```
## [1] 7.956698e-02 9.994741e-03 3.942895e+02 3.849133e-02 1.055143e-02
## [6] 5.523930e+00 6.711759e-03 1.171035e-02 2.952328e-02 2.928139e-02
```