JEM017 assignment 1: 1)
a) Generate an AR(1) and ARMA(1,1) process. Do not use predefined functions. Estimate both the processes

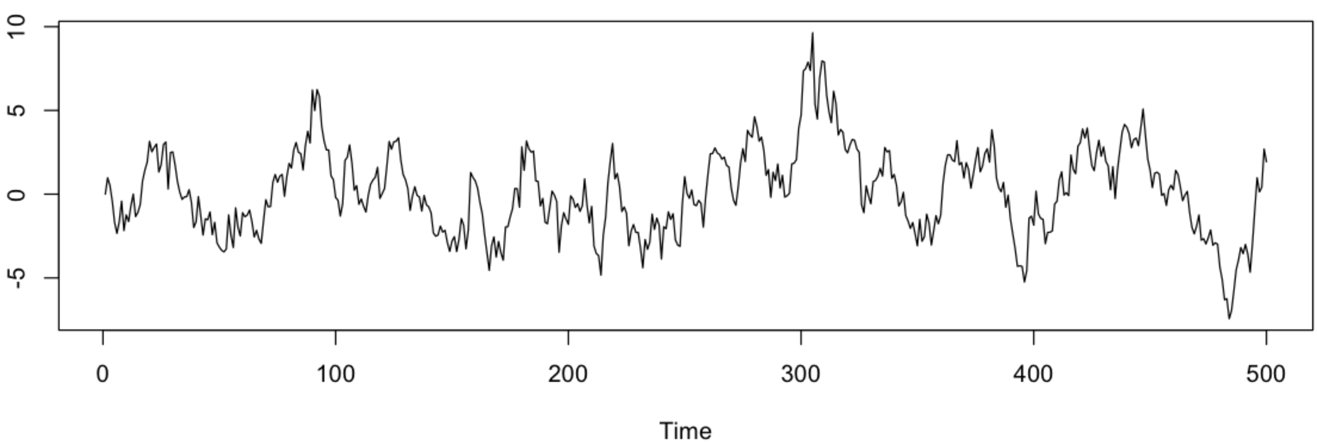
In [1]:

```
if (!require(forecast)) install.packages("forecast")
if (!require(forecast)) install.packages("uroot")

library(stats)
library(repr) -
library(tseries)
library(readr)
library(forecast)
library(uroot)

options(repr.plot.width = 10, repr.plot.height = 4)
```

```
  File "<ipython-input-1-e605420c576c>", line 1
    if (!require(forecast)) install.packages("forecast")
        ^
SyntaxError: invalid syntax
```

In [79]:

```
#AR(1) process
set.seed(7257)                                                 #setting seed for replic
ability
l <- 500                                                       #number of observations
e <- rnorm(l)                                                  #white noise process
yt <- vector()                                                 #create blanc vector to
store values of yt for t=1,...,l
phi1 <- 0.9                                                    #set value of coefficie
nt phi 1
yt[1] <- 0                                                     #set value of yt for t=
1
yt[2] <- phi1 * e[1]                                           #set value of yt for t=
2
for (i in 3 : l){                                              #loop to fill in yt dat
a for 2<t<l
    yt[i] <- phi1 * yt[i - 1] + e[i]
}
plot.ts(yt, ylab = NA)                                         #plot the yt-s
```

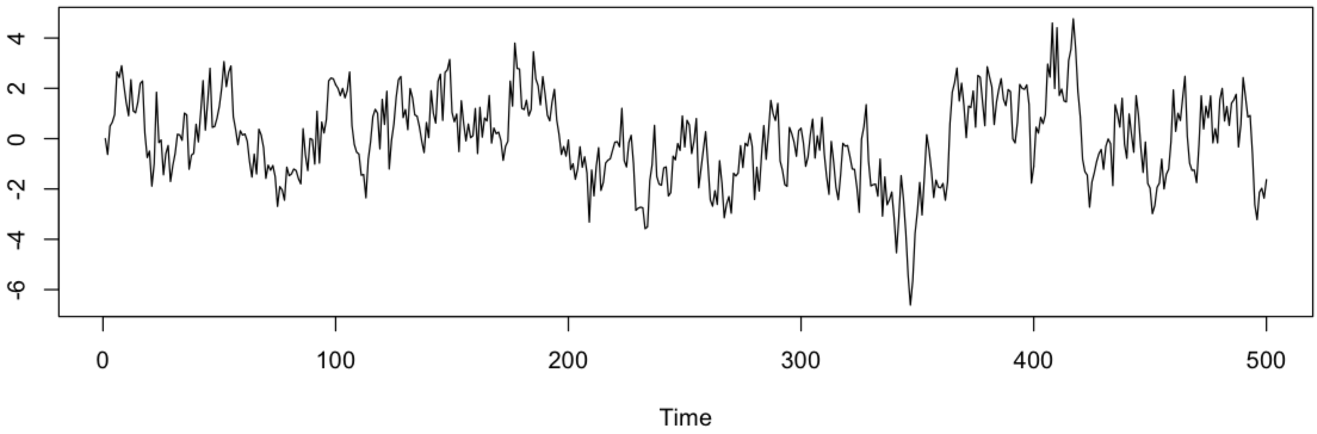


ARMA(1,1) process

In [80]:

```
set.seed(8087)                                                 #setting seed for replic
```

```
ability
l <- 500                                                         #number of observations
e <- rnorm(l)                                                    #white noise process
yt <- vector()                                                   #create blanc vector to
store values of yt for t=1,...,l
phi <- 0.9                                                       #set value of coefficie
nt phi
theta <- 0.2                                                     #set value of coefficie
nt theta
yt[1] <- 0                                                       #set value of yt for t=
1
yt[2] <- phi * e[1]                                              #set value of yt for t=
2
for (i in 3 : l){                                                #loop to fill in yt dat
a for 2<t<l
    yt[i] <- phi * yt[i - 1] + e[i] - theta * e[i-1]
}
plot.ts(yt, ylab = NA)                                           #plot the yt-s
```



b) Choose one real time series and estimate the best linear model (AR, MA, ARMA, ARMIA) that fits the time series.

In [81]:

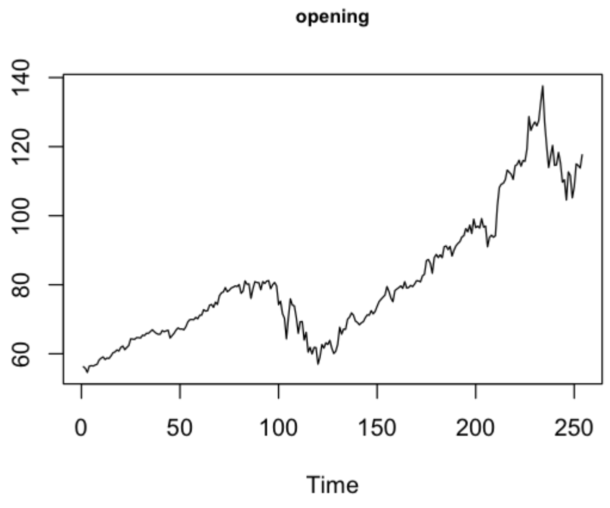
```
data <- read.csv("AAPL.csv")                                     #load Apple stock data
opening <- data[,2]                                              #assign the opening pri
ces
dopening <- diff(opening, lag = 1)                               #take first differences
par(mfrow = c(1, 2))                                             #make graphs show in pa
irs

plot.ts(opening, ylab = NA, main = 'opening', cex.main = 0.8)    #plot the original openi
ng prices as time series
plot.ts(dopening, ylab = NA, main = 'dopening', cex.main = 0.8) #plot the first differen
ces of opening prices as time series

adf.test(opening)                                                #confirm stacionarity wi
th a test
adf.test(dopening)                                               #confirm stacionarity wi
th a test
```

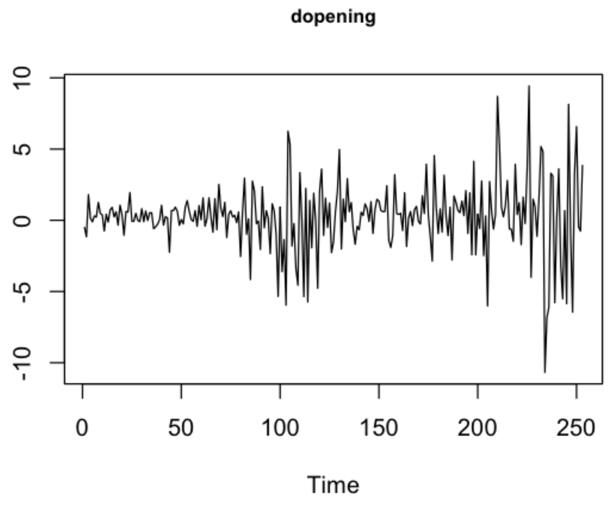
```
 Augmented Dickey-Fuller Test

data:  opening
Dickey-Fuller = -1.6281, Lag order = 6, p-value = 0.7324
alternative hypothesis: stationary
```

```
Warning message in adf.test(dopening):
“p-value smaller than printed p-value”
```

```
 Augmented Dickey-Fuller Test
```

```
data:  dopening
Dickey-Fuller = -5.6958, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```





In [82]:

```
kpss.test(opening)                                             #confirm stacionarity w
ith a test (double-checking because of suspicious rising trend of variance)
kpss.test(dopening)                                            #confirm stacionarity w
ith a test (double-checking because of suspicious rising trend of variance)
```

```
Warning message in kpss.test(opening):
"p-value smaller than printed p-value"
```

```
 KPSS Test for Level Stationarity

data:  opening
KPSS Level = 3.2184, Truncation lag parameter = 5, p-value = 0.01
```

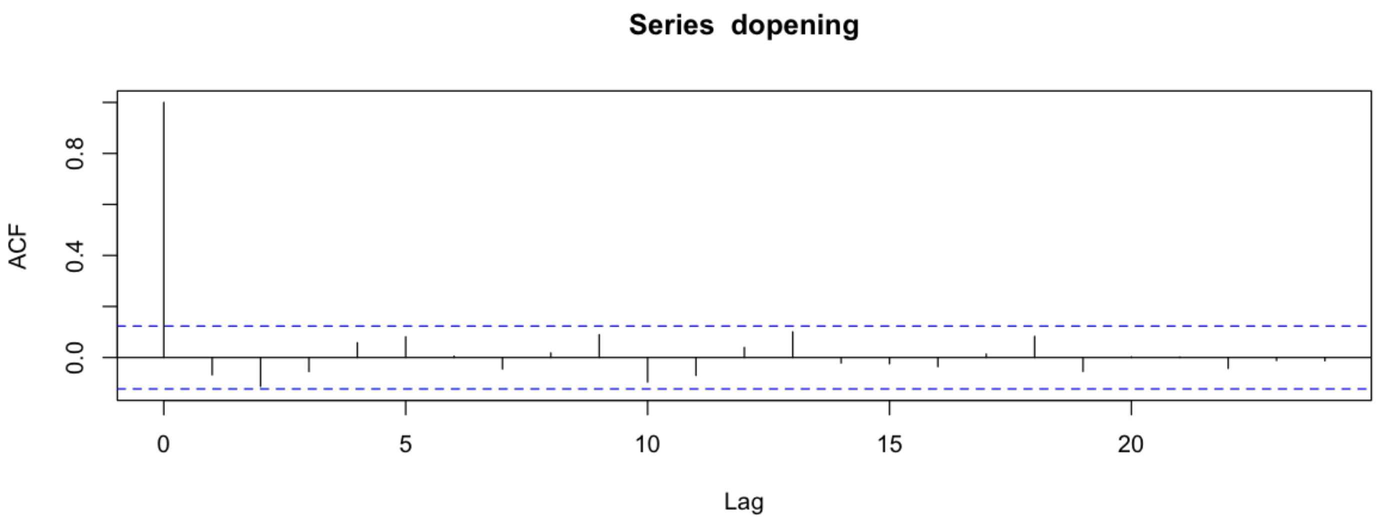
```
Warning message in kpss.test(dopening):
"p-value greater than printed p-value"
```

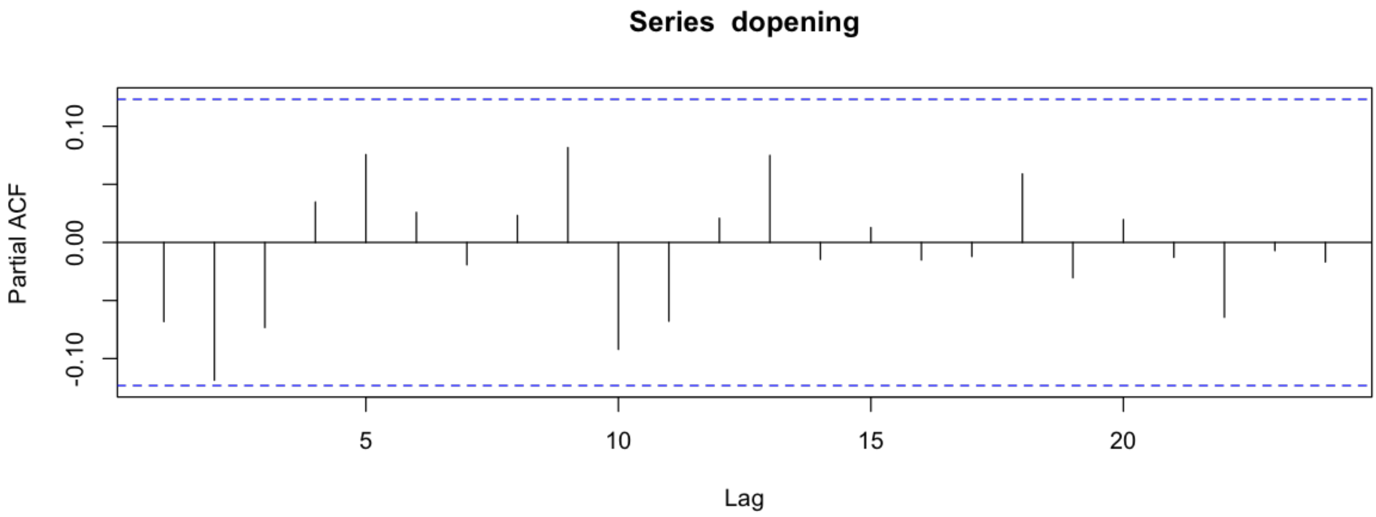
```
 KPSS Test for Level Stationarity

data:  dopening
KPSS Level = 0.11993, Truncation lag parameter = 5, p-value = 0.1
```

Plotting ACF and PACF to help with estimation of model.

In [83]:

```
acf(dopening)
pacf(dopening)
```





In [101]:

```
y_model <- Arima(dopening, order = c(1, 0, 0))
summary(y_model)
```

```
Series: dopening
ARIMA(1,0,0) with non-zero mean

Coefficients:
          ar1    mean
      -0.0685  0.2419
s.e.   0.0629  0.1443

sigma^2 estimated as 6.059:  log likelihood=-585.89
AIC=1177.78   AICc=1177.87   BIC=1188.38

Training set error measures:
                        ME     RMSE     MAE      MPE     MAPE      MASE
Training set -0.0001913391 2.451791 1.58959 100.2983 124.2024 0.6403158
                     ACF1
Training set -0.008359758
```

In [100]:

```
y_model1 <- Arima(dopening, order = c(0, 0, 2))
summary(y_model1)
```

```
Series: dopening
ARIMA(0,0,2) with non-zero mean

Coefficients:
          ma1      ma2    mean
      -0.0868  -0.1071  0.2410
s.e.   0.0621   0.0581  0.1235

sigma^2 estimated as 5.994:  log likelihood=-584.03
AIC=1176.06   AICc=1176.22   BIC=1190.19

Training set error measures:
                       ME   RMSE      MAE      MPE     MAPE      MASE
Training set -0.001148756 2.4337 1.575369 96.97446 126.3212 0.6345872
                    ACF1
Training set 0.002714232
```

In [99]:

```
y_model2 <- auto.arima(dopening)
summary(y_model2)
```

```
Series: dopening
ARIMA(0,0,0) with non-zero mean
```

```
Coefficients:
        mean
      0.2426
s.e.  0.1545

sigma^2 estimated as 6.064:  log likelihood=-586.48
AIC=1176.96   AICc=1177.01   BIC=1184.03

Training set error measures:
                       ME     RMSE      MAE      MPE     MAPE      MASE
Training set 5.757756e-17 2.457552 1.611239 104.9005 127.7979 0.6490362
                    ACF1
Training set -0.06813247
```

In [98]:

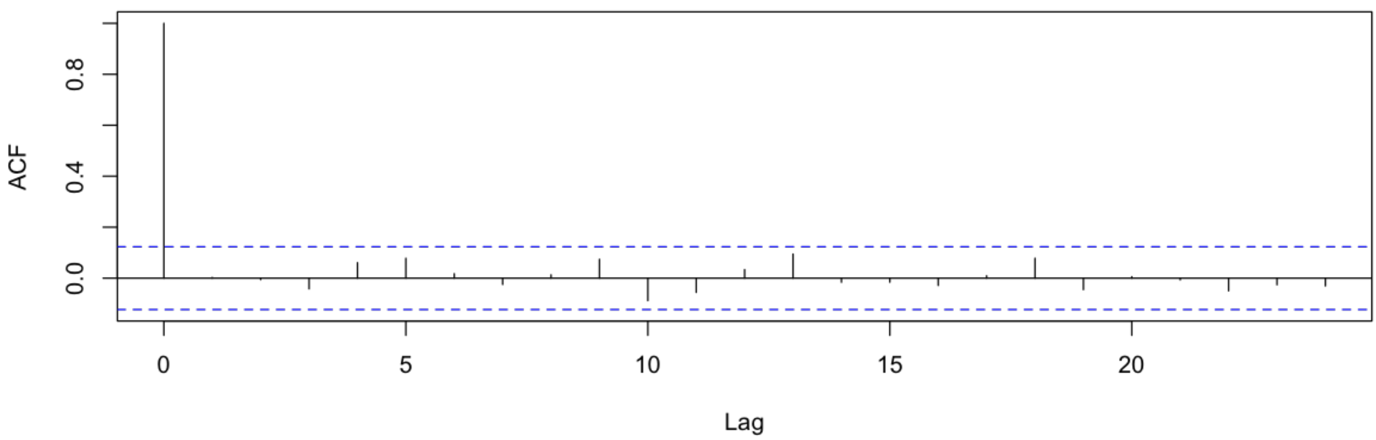
```
y_model3 <- auto.arima(dopening, approximation=FALSE, stepwise =FALSE )
summary(y_model3)
```

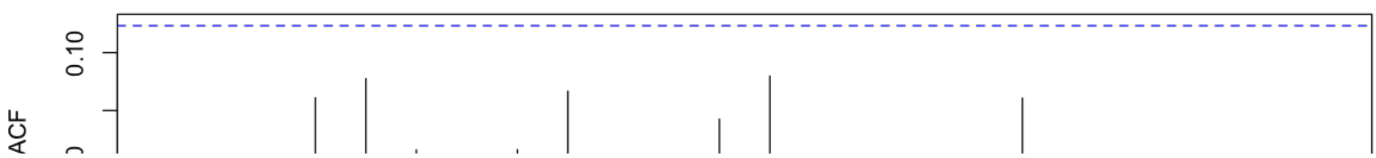
```
Series: dopening
ARIMA(0,0,2) with non-zero mean

Coefficients:
          ma1      ma2    mean
      -0.0868  -0.1071  0.2410
s.e.   0.0621   0.0581  0.1235

sigma^2 estimated as 5.994:  log likelihood=-584.03
AIC=1176.06   AICc=1176.22   BIC=1190.19

Training set error measures:
                       ME   RMSE      MAE      MPE     MAPE      MASE
Training set -0.001148756 2.4337 1.575369 96.97446 126.3212 0.6345872
                    ACF1
Training set 0.002714232
```

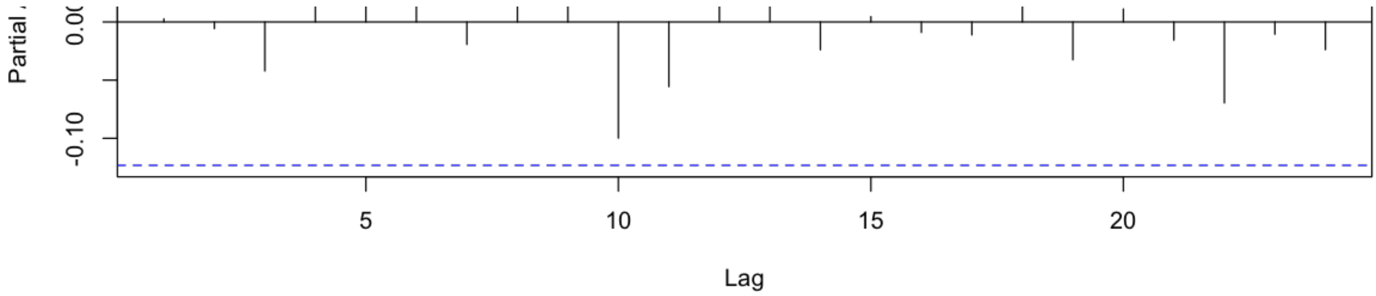
Lowest AICc in y_model3, ARIMA(0,0,2) with non-zero mean

In [116]:

```
acf(y_model3$residuals, main = NA)                #plotting the residuals in ACF and PACF to check whether the residuals show WN behavior
pacf(y_model3$residuals, main = NA)
```





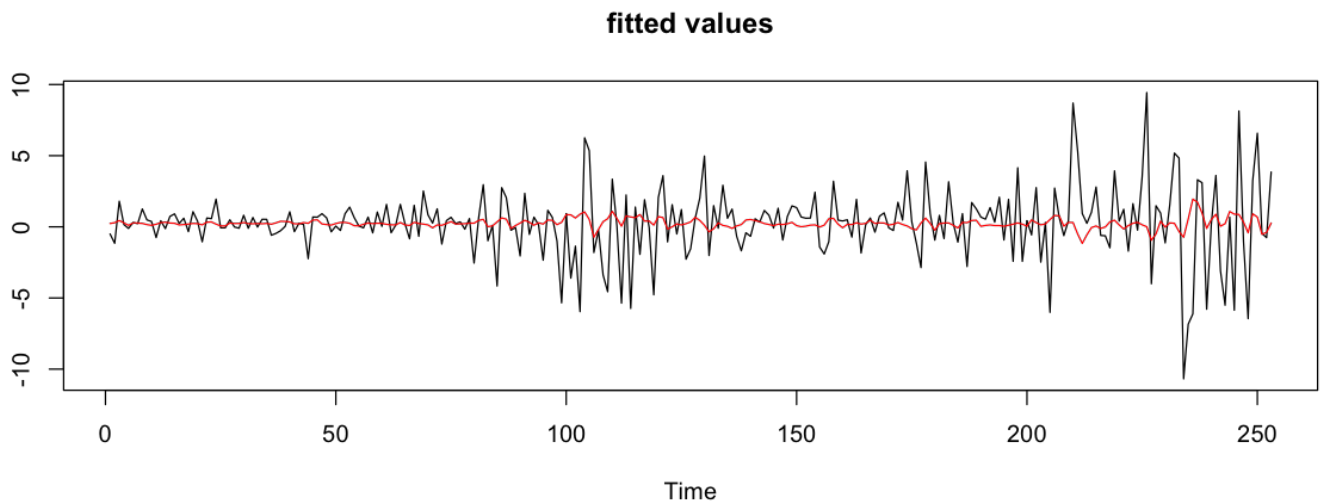


They do show WN behavior, using Ljung-Box test to check for lack of fit"

In [119]:

```
Box.test(y_model3$residuals, type = "Ljung-Box", lag = 4)
Box.test(y_model3$residuals, type = "Ljung-Box", lag = 8)
Box.test(y_model3$residuals, type = "Ljung-Box", lag = 12)
plot.ts(dopening, ylab = NA, main = 'fitted values')
lines(y_model3$fitted, col = 'red')
```

```
 Box-Ljung test

data:  y_model3$residuals
X-squared = 1.4265, df = 4, p-value = 0.8396

 Box-Ljung test

data:  y_model3$residuals
X-squared = 3.2962, df = 8, p-value = 0.9144

 Box-Ljung test

data:  y_model3$residuals
X-squared = 7.9595, df = 12, p-value = 0.7883
```



High p-values don't indicate lack of fit.

In [ ]: