Assignment 4: Return, Autocorrelation and GBM

- **By Sandeep Joshi**

Question 1: Returns and Autocorrelations

1. Download data for any symbol for at least 2 years

```r
#-----------------------------------#
#     A S S I G N M E N T   4       #
#-----------------------------------#

cat("\014")

# Question 1
# Returns and autocorrelation

# Download stock for 2 years
rm(list = ls())
library(quantmod)

getSymbols(Symbols = "MSFT", from = "2013-10-01", to = "2015-10-01")
# of all the various prices given in the data fram I am choosing the adjusted price
# to calculate all the values in the tasklist
# msft.price <- data.frame(MSFT)$MSFT.Adjusted
msft <- data.frame(MSFT)
# msft <- msft[0:5,]  # For testing purposes
```

Output

```
> head(msft)
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2013-10-01     33.35     33.61    33.30      33.58    36718700      31.79213
2013-10-02     33.36     34.03    33.29      33.92    46946800      32.11403
2013-10-03     33.88     34.00    33.42      33.86    38703800      32.05723
2013-10-04     33.69     33.99    33.62      33.88    33008100      32.07616
2013-10-07     33.60     33.71    33.20      33.30    35069300      31.52704
2013-10-08     33.31     33.33    32.80      33.01    41017600      31.25248
> 
```

2. Calculate One period simple and log returns

```r
# This is in ascending order.. flipping it
msft <- msft[nrow(msft):1,]

# making another column for n - 1 prices vector as for n enteries we will have n - 1 deltas
msft.shift_price <- msft$MSFT.Adjusted[1:length(msft$MSFT.Adjusted) - 1]
# make another column for t -1 prices so that we get two vectors to get delta
msft.shift_price1 <- msft$MSFT.Adjusted[2:length(msft$MSFT.Adjusted)]
length = length(msft.shift_price1)
# Calculating simple return
msft.simple_return <- (msft.shift_price - msft.shift_price1)/ msft.shift_price1  # Rt

# Calculating log return
msft.log_return <- log(msft.shift_price) - log(msft.shift_price1)       # rt
```

Output

```
> msft.simple_return <- (msft.shift_price - msft.shift_price1)/ msft.shift_price1  # Rt
> head(msft.simple_return)
[1]  0.0079078856  0.0188765888  0.0034649572 -0.0147928542  0.0006831929  0.0009118076
> # Calculating log return
> msft.log_return <- log(msft.shift_price) - log(msft.shift_price1)       # rt
> head(msft.log_return)
[1]  0.0078767821  0.0187006368  0.0034589681 -0.0149033596  0.0006829596  0.0009113922
```

3. Calculate multi period returns for log and simple and verify their relationship with one periods calculated earlier

```r
# Calculating multiperiod simple return
msft.simple_return_m <- msft.shift_price/ msft$MSFT.Adjusted[length(msft$MSFT.Adjusted)] - 1

# Calculating multi period log return
msft.log_return_m <- log(msft.shift_price/ msft$MSFT.Adjusted[length(msft$MSFT.Adjusted)])
#print(msft.log_return_m)

# Re-Calculating multiperiod returns based on simple returns calculated above
size <- length(msft.simple_return)   # should be length - 1
msft.simple_return_m2 <- double(size)
msft.log_return_m2 <- double(size)

for (i in 1:size)
{
    msft.simple_return_m2[i] <- prod((msft.simple_return[i:size] + 1)) - 1
    print (msft.simple_return_m2[i])
}

for (i in 1:size)
{
    msft.log_return_m2[i] <- sum(msft.log_return[i:size])
    print (msft.log_return_m2[i])
}

cbind.data.frame(msft$MSFT.Adjusted[1:size], msft.shift_price, msft.shift_price1,
                 msft.simple_return, msft.simple_return_m, msft.simple_return_m2,
                 msft.log_return_m, msft.log_return_m2)

# Comparing columns for simple_return_m and simple_return_m2 and log_return_m and log_return_m2 we c
# the relationship between single and multiperiod.
```

Output

A4.r* result

Filter

	msft$MSFT.Adjusted[1:size]	msft.shift_price	msft.shift_price1	msft.simple_return	msft.simple_return_m	msft.simple_return_m2	msft.log_return_m	msft.log_return_m2
1	44.61000	44.61000	44.26000	0.0079078856	0.4031773	0.4031773	0.3387392	0.3387392
2	44.26000	44.26000	43.44000	0.0188765888	0.3921682	0.3921682	0.3308624	0.3308624
3	43.44000	43.44000	43.29000	0.0034649572	0.3663757	0.3663757	0.3121618	0.3121618
4	43.29000	43.29000	43.94000	-0.0147928542	0.3616576	0.3616576	0.3087028	0.3087028
5	43.94000	43.94000	43.91000	0.0006831929	0.3821028	0.3821028	0.3236061	0.3236061
6	43.91000	43.91000	43.87000	0.0009118076	0.3811593	0.3811593	0.3229232	0.3229232
7	43.87000	43.87000	43.90000	-0.0006834396	0.3799010	0.3799010	0.3220118	0.3220118
8	43.90000	43.90000	44.11000	-0.0047608024	0.3808448	0.3808448	0.3226955	0.3226955
9	44.11000	44.11000	43.48000	0.0144894434	0.3874501	0.3874501	0.3274676	0.3274676
10	43.48000	43.48000	44.25000	-0.0174011299	0.3676339	0.3676339	0.3130822	0.3130822
11	44.25000	44.25000	44.30000	-0.0011286456	0.3918537	0.3918537	0.3306365	0.3306365
12	44.30000	44.30000	43.98000	0.0072760118	0.3934264	0.3934264	0.3317658	0.3317658
13	43.98000	43.98000	43.04000	0.0218401250	0.3833611	0.3833611	0.3245161	0.3245161
14	43.04000	43.04000	43.48000	-0.0101195722	0.3537940	0.3537940	0.3029110	0.3029110
15	43.48000	43.48000	43.29000	0.0043889812	0.3676339	0.3676339	0.3130822	0.3130822
16	43.29000	43.29000	43.07000	0.0051079870	0.3616576	0.3616576	0.3087028	0.3087028
17	43.07000	43.07000	43.89000	-0.0186830490	0.3547376	0.3547376	0.3036078	0.3036078
18	43.89000	43.89000	42.61000	0.0300398491	0.3805301	0.3805301	0.3224676	0.3224676
19	42.61000	42.61000	43.50000	-0.0204597471	0.3402687	0.3402687	0.2928701	0.2928701
20	43.50000	43.50000	43.36000	0.0032287592	0.3682630	0.3682630	0.3135420	0.3135420
21	43.36000	43.36000	41.82000	0.0368245098	0.3638594	0.3638594	0.3103185	0.3103185
22	41.82000	41.82000	43.52000	-0.0390625000	0.3154197	0.3154197	0.2741558	0.2741558
23	43.52000	43.52000	43.93000	-0.0093330298	0.3688921	0.3688921	0.3140017	0.3140017
24	43.93000	43.93000	43.90000	0.0006833257	0.3817883	0.3817883	0.3233786	0.3233786
25	43.90000	43.90000	42.71000	0.0278623982	0.3808448	0.3808448	0.3226955	0.3226955

It could be easily seen from the result shot above that values in column 5 & 6 as well as 7 & 8 are clearly matching validating our two formulae given in the study material.

4. Calculate auto correlation functions for log and log squared returns.

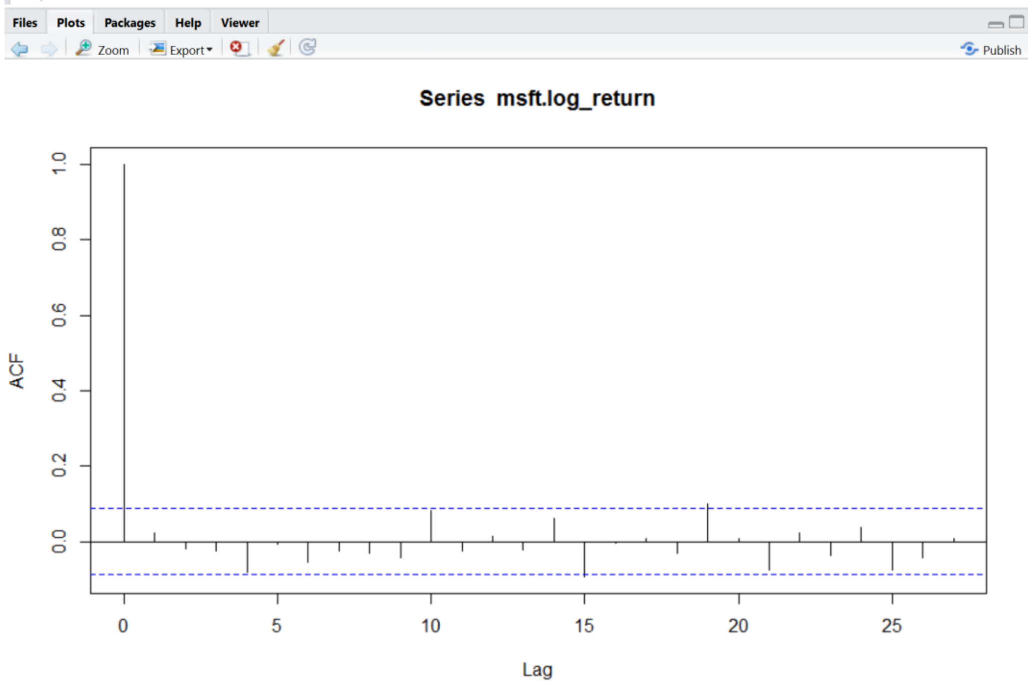
P.S. Library acf functions used to perform this.

```r
# Auto correlation function
# log return series
# Reusing the log vector here from earlier endeavors and employing acf function
# msft.log_return
acf_log <- acf(msft.log_return)
print(acf_log)
# Squared log return series
acf_log_sq <- acf(msft.log_return^2)
print(acf_log_sq)
```

Output:

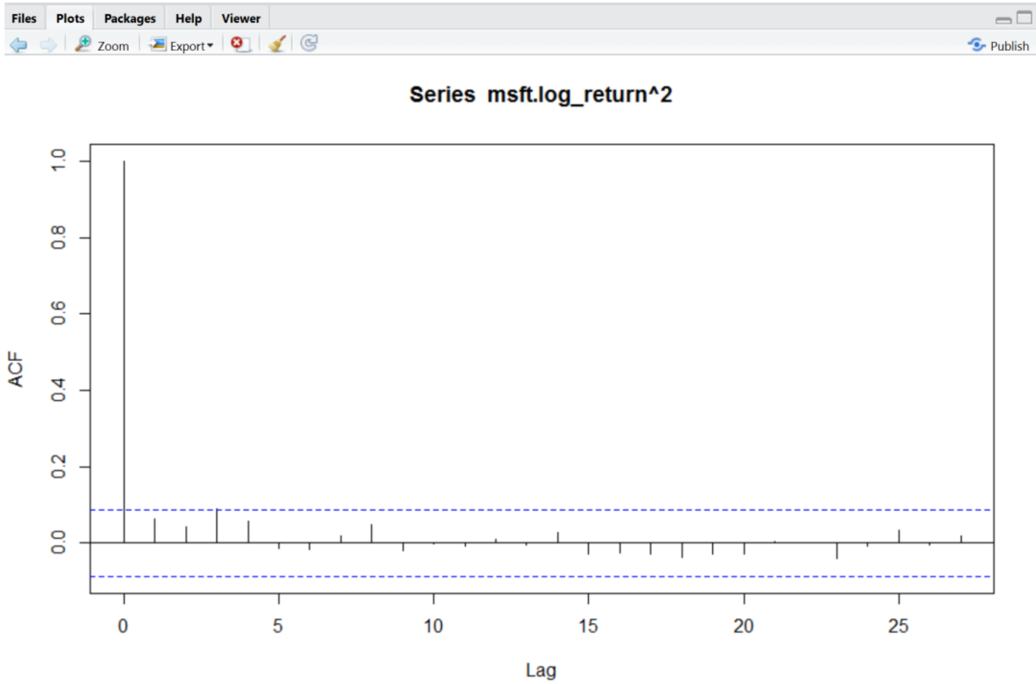
1.

```
Autocorrelations of series ‘msft.log_return’, by lag

     0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15 
 1.000  0.024 -0.019 -0.023 -0.082 -0.006 -0.054 -0.023 -0.029 -0.042  0.082 -0.023  0.014 -0.020  0.061 -0.092 
    16     17     18     19     20     21     22     23     24     25     26     27 
-0.004  0.007 -0.031  0.100  0.009 -0.073  0.023 -0.036  0.039 -0.074 -0.042  0.010 
>
```



2.

```
Autocorrelations of series ‘msft.log_return^2’, by lag

     0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15 
 1.000  0.064  0.043  0.089  0.057 -0.013 -0.016  0.020  0.047 -0.020 -0.003 -0.008  0.011 -0.005  0.029 -0.028 
    16     17     18     19     20     21     22     23     24     25     26     27 
-0.025 -0.029 -0.039 -0.029 -0.029  0.004  0.002 -0.040 -0.009  0.032 -0.004  0.019 
>
```


Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Series msft.log_return^2
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
Lag

Question 2: Geometrical Brownian Motion

Disclaimer: For this solution I have understood and borrowed code from the lecture 6 material. I have just seeded the two methods equally and looped them couple of times with different but consistent seeds to compare the final termination values.

```r
# Question 2
# Geomatrical Brownian Motion
# Disclaimer: Used code provided in lecture 6'

rm(list = ls())
cat("\014")
## parameters
r <- 0.05
sigma <- 0.2
Maturity <- 1
steps <- 252
S0 <- 100
iterations <- 20
result = matrix(nrow = 20, ncol = 2)

# Start the loop
for (i in 1:iterations)
{
    set.seed(i)  # Set seed so we get same rnorm

    ## method 1: Euler Method
    dt <- Maturity / steps
    epsilon_t_vec <- rnorm(steps)
    epsilon_t_vec <- append(0, epsilon_t_vec)
    dWt_vec <- epsilon_t_vec * sqrt(dt)
    St_vec <- c()
    St_vec[1] <- S0
    for(j in 1:steps)
    {
        dWt <- dWt_vec[j+1]
        St_vec[j+1] <- St_vec[j] + r * St_vec[j] * dt + sigma * St_vec[j] * dWt
    }
    result[i, 1] = St_vec[steps+1]

    set.seed(i)
    ## method 2: Solution to GBM
    dt <- Maturity / steps
    epsilon_t_vec <- rnorm(steps)
    epsilon_t_vec <- append(0, epsilon_t_vec)
    cum_Wt_vec <- cumsum(epsilon_t_vec) * sqrt(dt)
    ST <- S0 * exp(r*Maturity + cum_Wt_vec[steps + 1] * sigma)

    result[i, 2] = ST
}

# Check terminal values by both methods side by side
print (result)
```

Output

```
> # Check terminal values by both methods side by side
> print (result)
          [,1]      [,2]
 [1,] 111.44277 113.50458
 [2,] 115.44639 118.12764
 [3,] 117.10242 119.45490
 [4,] 105.82910 107.75000
 [5,] 111.32775 113.52884
 [6,]  81.76372  83.33268
 [7,] 131.66559 134.21623
 [8,]  90.60343  92.51594
 [9,]  75.92925  77.37672
[10,]  79.66654  81.11691
[11,] 100.50049 102.33094
[12,]  98.72282 100.47425
[13,]  83.32588  85.05723
[14,]  99.11526 101.08123
[15,] 101.59396 103.85768
[16,] 116.48931 118.78816
[17,]  94.77287  96.96509
[18,]  81.27659  82.98324
[19,] 106.27734 108.58006
[20,] 111.74490 114.15058
>
```

The first column contains terminal value using method 1 and the second column shows terminal value using method 2. As we can see they are pretty same and consistently shows same pattern for same seeds.