Lab2.

```r
# Set Working Directory
setwd("~/Desktop/CSC425/week2/intelc7303")
```

1. Load Library

```r
# Library
library(tseries)
library(fBasics)
```

```
## Loading required package: timeDate

## Loading required package: timeSeries

##

## Rmetrics Package fBasics

## Analysing Markets and calculating Basic Statistics

## Copyright (C) 2005-2014 Rmetrics Association Zurich

## Educational Software for Financial Engineering and Computational Science

## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.

## https://www.rmetrics.org --- Mail to: info@rmetrics.org
```

```r
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':
##
##     time<-

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

2. Import Data

```r
# Load data with no variable names into the data frame "da"
da = read.table("d-intc7303.txt",header=F)
head(da)
```

```
##         V1       V2
## 1 19730102  0.03015
## 2 19730103 -0.00976
## 3 19730104 -0.00985
## 4 19730105  0.00000
## 5 19730108  0.00995
## 6 19730109  0.02956
```

```r
# prints the first row of the data
head(da[1,],10)
```

```
##         V1      V2
## 1 19730102 0.03015
```

```r
# prints the fist col that contains dates
head(da[,1],10)
```

```
##  [1] 19730102 19730103 19730104 19730105 19730108 19730109 19730110
##  [8] 19730111 19730112 19730115
```

```r
# prints the second col that contains intel returns
head(da[,2],10)
```

```
##  [1]  0.03015 -0.00976 -0.00985  0.00000  0.00995  0.02956 -0.01914
##  [8]  0.00000  0.00000  0.00976
```

```r
# computes log returns
rt = log(da[,2]+1)
head(rt,10)
```

```
##  [1]  0.029704423 -0.009807941 -0.009898832  0.000000000  0.009900825
##  [6]  0.029131527 -0.019325541  0.000000000  0.000000000  0.009712679
```

3. Creat Time Series Variable "Intel"

```r
# Zoo package is included in tseries. See R document in week 2 for more information.
intel = zoo(da [, 2], as.Date(as.character(da[,1]), format = "%Y %m %d"))
head(intel,10)
```

```
## 1973-01-02 1973-01-03 1973-01-04 1973-01-05 1973-01-08 1973-01-09
##    0.03015   -0.00976   -0.00985    0.00000    0.00995    0.02956
## 1973-01-10 1973-01-11 1973-01-12 1973-01-15
##   -0.01914    0.00000    0.00000    0.00976
```

```r
#start date
start(intel)
```

```
## [1] "1973-01-02"
```

```r
#end date
end(intel)
```

```
## [1] "2003-12-31"
```

```r
# computes log returns
rts = log(intel + 1)
#to retrieve dates only
head(time(rts),20)
```

```
##  [1] "1973-01-02" "1973-01-03" "1973-01-04" "1973-01-05" "1973-01-08"
##  [6] "1973-01-09" "1973-01-10" "1973-01-11" "1973-01-12" "1973-01-15"
## [11] "1973-01-16" "1973-01-17" "1973-01-18" "1973-01-19" "1973-01-22"
## [16] "1973-01-23" "1973-01-24" "1973-01-26" "1973-01-29" "1973-01-30"
```

```r
head(tail(rts),20)
```

```
##    2003-12-23    2003-12-24    2003-12-26    2003-12-29    2003-12-30
##  0.0224266326 -0.0009604611  0.0089696521  0.0248779613 -0.0034258616
##    2003-12-31
##  0.0003099520
```

The intel log returns have been stored in to two variables: "rt" is a simpole data variable (no time information), and "rts" is a time object that contains log returns and date information. For most functions both variables can be sued interchangeably. However, time series objects have specific functions that utilize the time information, such as "plot()"

4. Compute Summary Statistics

```
basicStats(rt)
```

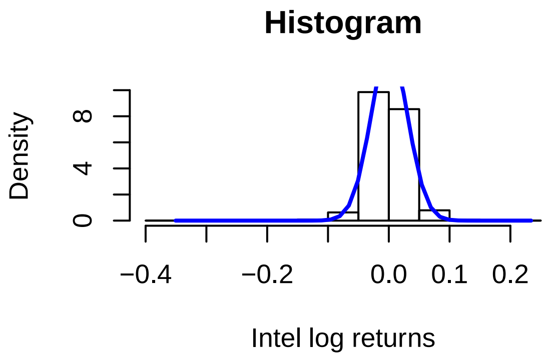
```
##                      rt
## nobs        7828.000000
## NAs            0.000000
## Minimum       -0.350579
## Maximum        0.234107
## 1. Quartile   -0.015266
## 3. Quartile    0.017781
## Mean           0.000855
## Median         0.000000
## Sum            6.692218
## SE Mean        0.000341
## LCL Mean       0.000187
## UCL Mean       0.001522
## Variance       0.000908
## Stdev          0.030129
## Skewness      -0.544240
## Kurtosis       7.536558
```

```
# same results if we use time series object "rts"
basicStats(rts)
```

```
##                       x
## nobs        7828.000000
## NAs            0.000000
## Minimum       -0.350579
## Maximum        0.234107
## 1. Quartile   -0.015266
## 3. Quartile    0.017781
## Mean           0.000855
## Median         0.000000
## Sum            6.692218
## SE Mean        0.000341
## LCL Mean       0.000187
## UCL Mean       0.001522
## Variance       0.000908
## Stdev          0.030129
## Skewness      -0.544240
## Kurtosis       7.536558
```

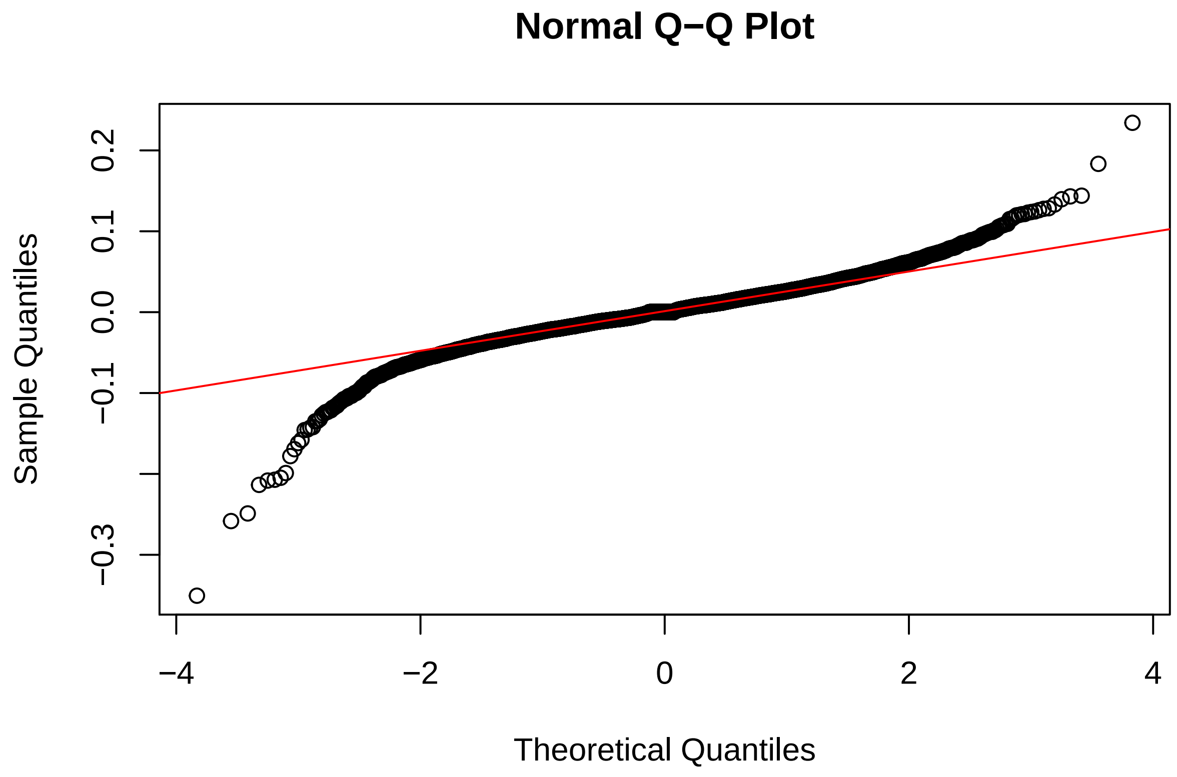
5. Create Histogram

```r
#creates 2 by 2 display for 4 plots
par(mfcol = c(2,2))
#to make RStudio open up a new graphics device with default settings
hist(rts, xlab="Intel log returns", prob=TRUE, main="Histogram")
#add approximating normal density curve
xfit <- seq(min(rt), max(rt), length = 40)
yfit <- dnorm(xfit, mean = mean(rt), sd = sd(xfit))
lines(xfit, yfit, col = "blue", lwd = 2)
```



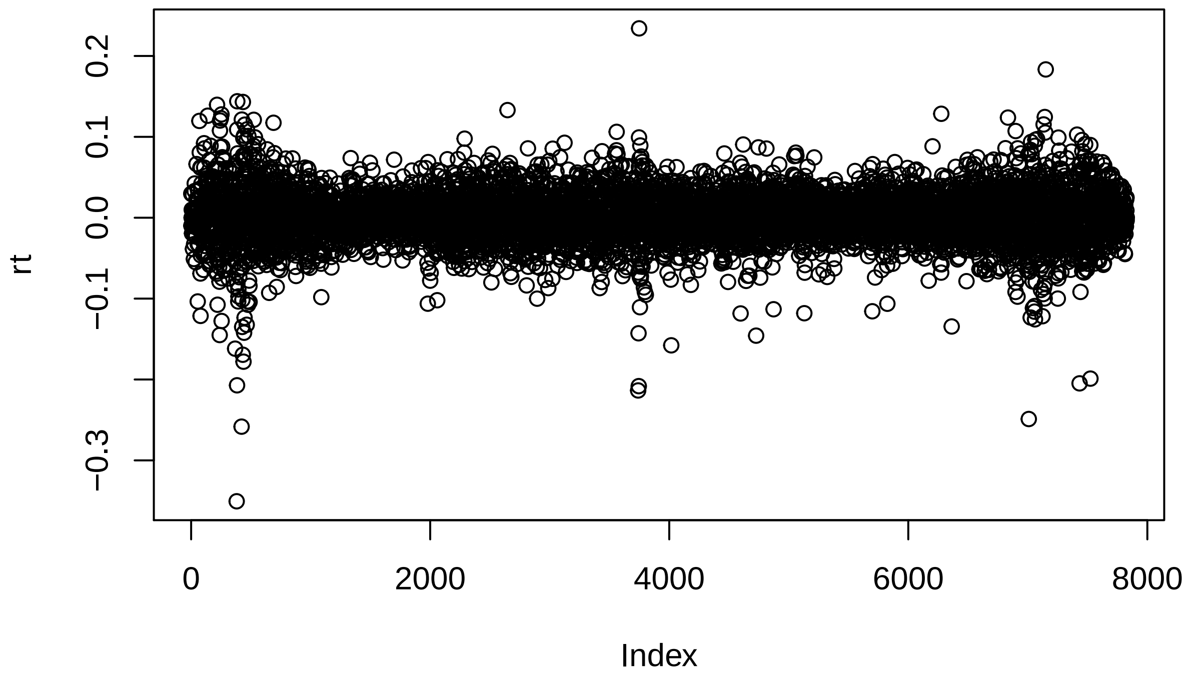
6. Create Normal Probability Plot

```r
qqnorm(rt)
qqline(rt, col = 2)
```

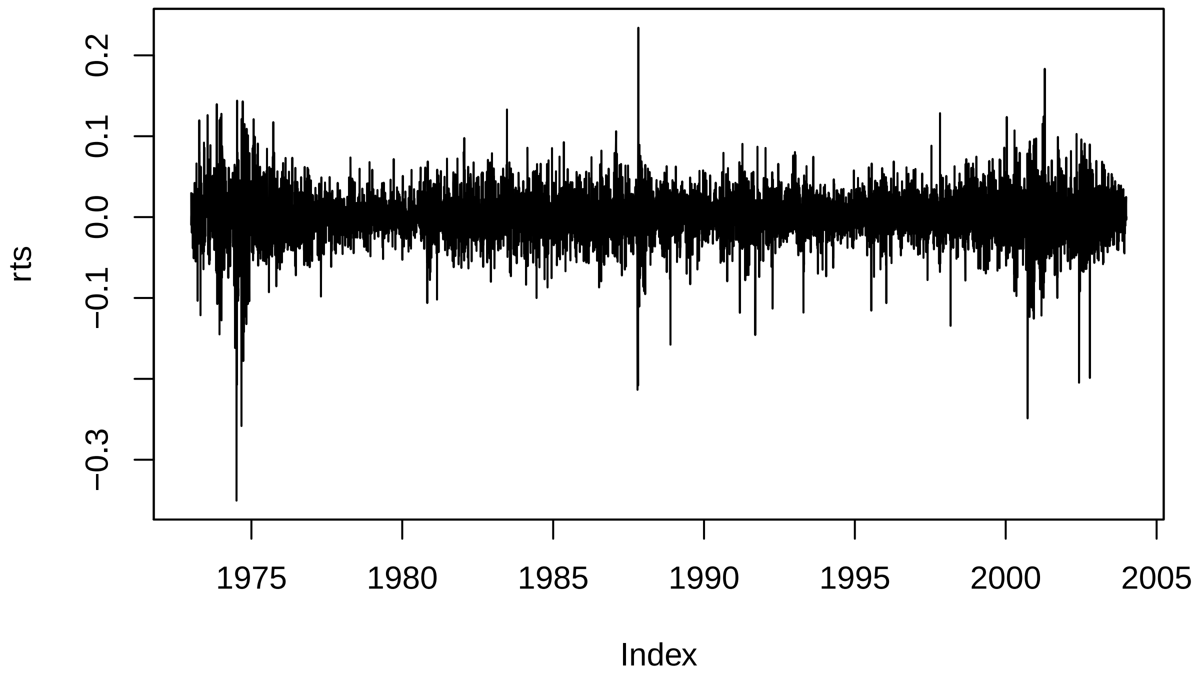


7. Creat Time Plot

```r
# simple plot where x-axis is not labeled with time
plot(rt)
```

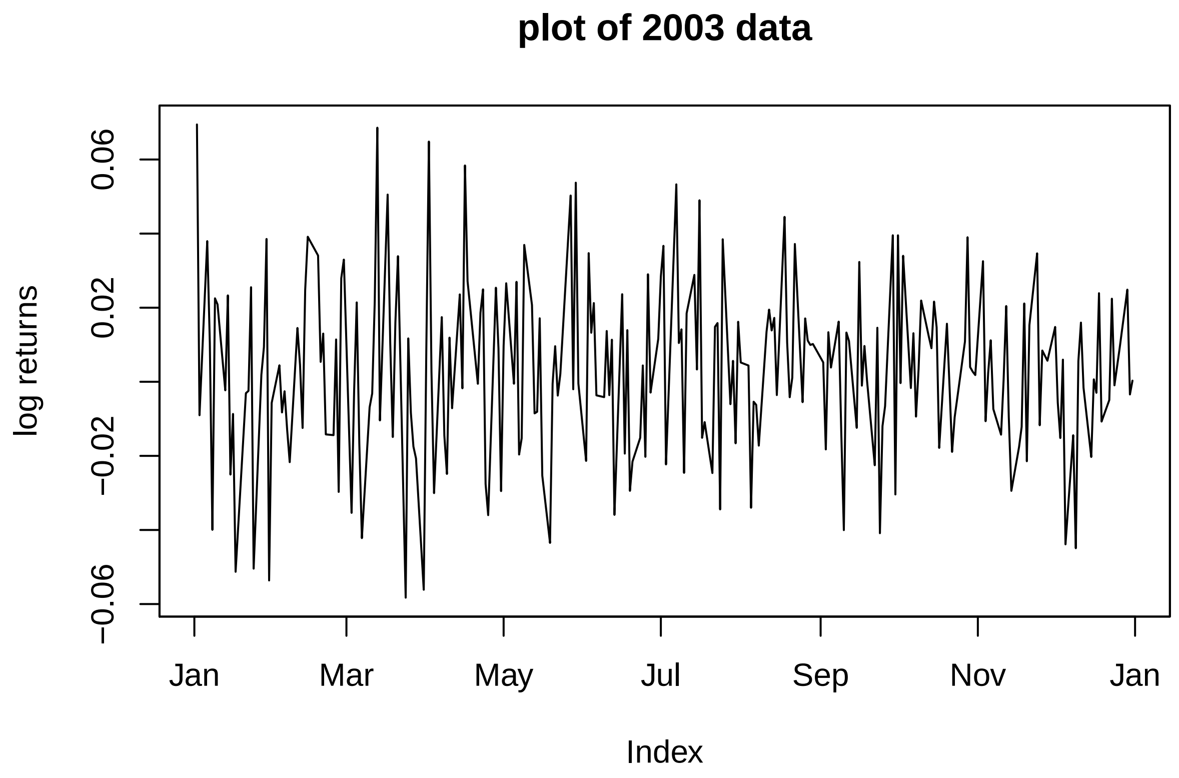


```r
# use time series object "rts" to draw time plot indexed with time
plot(rts)
```



```r
# creates subsets of data in smaller period of time
rts_03 = window(rts, start = as.Date("2003-01-01"), end = as.Date("2003-12-31"))
#plot the subset of data
```

```r
plot(rts_03, type = 'l', ylab = "log returns", main = "plot of 2003 data")
```



8. Normality Tests

```r
# Perform Jarque-Bera normality test
normalTest(rt, method = c("jb"))
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 18925.9999
##   P VALUE:
##     Asymptotic p Value: < 2.2e-16
##
## Description:
##  Thu Sep 21 14:40:38 2017 by user:
```

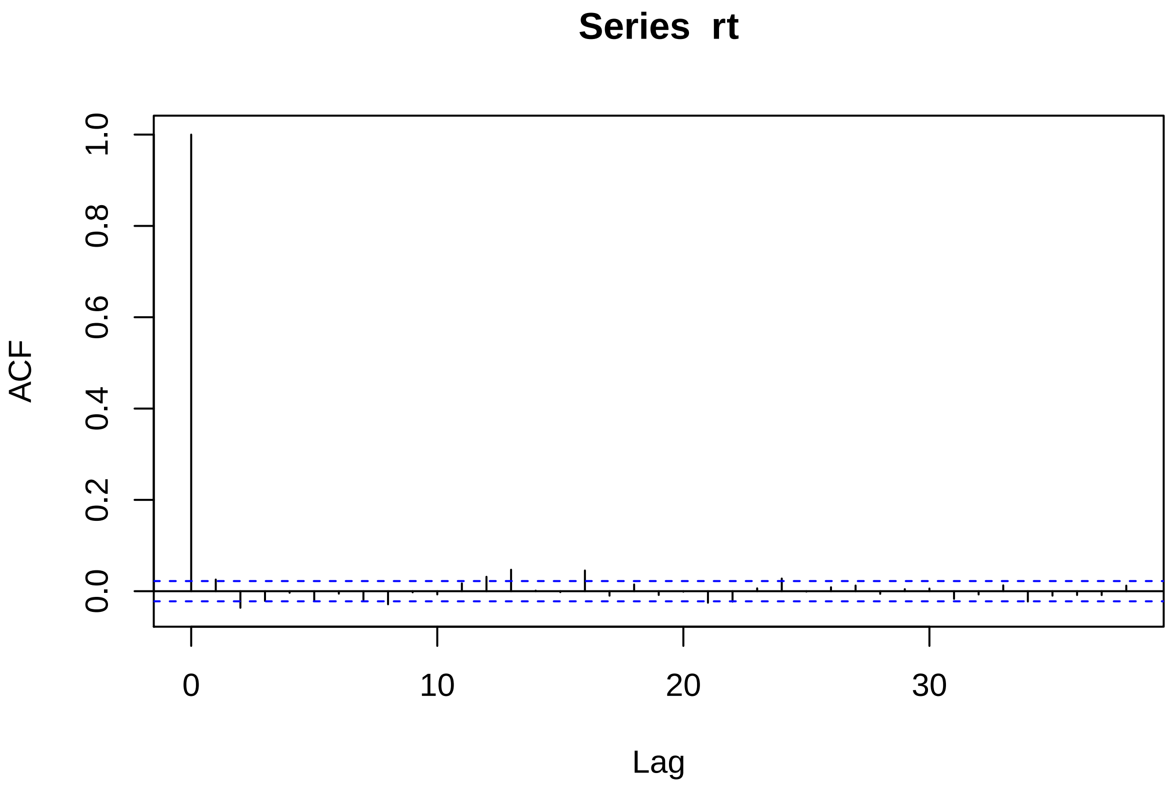
9. Compute ACF and Plot Correlogram

```r
# prints acf to console
acf(rt, plot = F)
```

```
##
## Autocorrelations of series 'rt', by lag
```

```
##
##      0      1      2      3      4      5      6      7      8      9
##  1.000  0.026 -0.036 -0.021 -0.003 -0.020 -0.005 -0.019 -0.029 -0.002
##     10     11     12     13     14     15     16     17     18     19
## -0.007  0.017  0.032  0.047  0.001 -0.002  0.045 -0.010  0.015 -0.008
##     20     21     22     23     24     25     26     27     28     29
## -0.001 -0.025 -0.023  0.006  0.028 -0.001  0.009  0.012 -0.006  0.005
##     30     31     32     33     34     35     36     37     38
##  0.006 -0.017 -0.007  0.013 -0.023 -0.010 -0.008 -0.008  0.012
```

```r
# plots acf (correlogram)
acf(rt, plot = T)
```



10. Compute Ljung-Box Test for White Noise (No Autocorrelation)

```r
# to lag 6
Box.test(rt, lag = 6, type = 'Ljung')
```

```
##
##  Box-Ljung test
##
## data:  rt
## X-squared = 22.497, df = 6, p-value = 0.0009835
```

```r
# to lag 12
Box.test(rt, lag = 12, type = 'Ljung')
```

```
##
##  Box-Ljung test
```

```
##
## data:  rt
## X-squared = 42.266, df = 12, p-value = 3.004e-05
```

Extra Commands to Aggregate Time Series

11. To Aggregate Daily log Returns To Monthly Returns

```r
rtm=aggregate(rts, as.yearmon, mean)
```

12. To Aggregate Daily Log Returns to Quarterly Returns

```r
rtq = aggregate(rts, as.yearqtr, mean)
```

13. To aggregate Daily log Returns to Weekly Returns (Next Friday)

```r
# function nextfri() compute for each day the next Friday
nextfri <- function(x) 7 * ceiling(as.numeric(x - 5 + 4) / 7) + as.Date(5 - 4)
rtw = aggregate(rts, nextfri, mean)
```

14. Aggregate Daily Log Returns to Last Trading Friday of Each Week

```r
rtw = aggregate(rts, nextfri, tail, 1)
```