ORIE 4630: Spring Term 2019
Homework #1
Due: Thursday, January 31, 2019

Students are required to work independently on homework. You should not give or receive help from other students. You should also not receive help from students or former students who took this course in previous years and who may have solutions to similar problems. The solutions you submit should be your own work and not copied from elsewhere.

Homework is due at the end of lecture (12:55pm) on the due date. You will usually have one week to do the assignments. Please don't wait until the homework is nearly due to start. Late homework is not accepted. Also, homework is not accepted by email. You can submit your assignment in lecture or in the drop box in Rhodes Hall.

Please print your name on the front of your homework so that it is legible.

Include your `R` code, output, graphs, and other work with your homework. This will allow the grader to find any errors you make and to give partial credit.

The goal of this assignment is to introduce you to some `R` functions that are useful for the analysis of financial markets data. Most practitioners use `Rstudio`, because they find it easier to implement `R` this way. Although is not necessary for you to use `Rstudio`, if you try it, you will probably find that you prefer using `Rstudio` to running `R` directly.

The documents *A (Very) Short Introduction to R* and *Rstudio101*, which are posted on the course Blackboard site, will introduce you to `Rstudio` as well as to `R`. Also posted on the course Blackboard site is the book *An Introduction to R*, which provides a more comprehensive introduction to `R`; this book can be downloaded from CRAN and should be available from `R`'s help menu.

The first ten questions of this assignment use log gross returns for several "assets" contained in a comma separated values (csv) file named `returns`. You should download this file from the course Blackboard site and put it into your `R` or `Rstudio` working directory. The file has 35 columns. The first column shows the date (Date), and the next 30 columns are for the stocks that are the components of the Dow Jones Industrial average (DOW). The final four columns are for the Dow Jones Industrial average index (DOW), the NASDAQ composite index (NASD), the NASDAQ 100

index (NASD100), and the S&P 500 index (SP500). The daily log gross returns are from January 4, 2006 to August 18, 2017. There are 2927 days of returns. The log gross returns are calculated from adjusted closing prices downloaded from Yahoo.

Start `R` or `Rstudio` and run the following code:

```
1 Returns = read.csv("returns.csv")
2 names(Returns)
3 class(Returns$Date)
4 Returns$Date = as.Date(Returns$Date, format="%m/%d/%Y")
5 class(Returns$Date)
6 head(Returns,n=5)
7 tail(Returns)
8 mode(Returns)
9 class(Returns)
10 summary(Returns)
11 str(Returns)
```

Line 1 reads the data into a data frame named `Returns`. If `returns.csv` is not in your working directory, then you need to give a complete path to that file in line 1. Line 2 prints the names of the columns (variables) of `Returns`.

Line 6 prints the names of the columns and the first `n` lines of `Returns`; here `n` is given to be 5. If `n` is unspecified, then the default value is 6. Line 7 prints the last six lines of `Returns`, as the value of `n` in the `tail()`[1] command is unspecified.

Lines 8 and 9 print the mode and class of `Returns`. The most common modes are `list`, `numeric`, `logical`, and `character`. Examples of classes are `numeric`, `ts`, `data.frame`, and `matrix`. You will learn more about modes and classes in this and future homework assignments. There are `R` functions for converting an object from one class to another. One example is the function `as.Date()`, at line 4 and another example is the function `as.ts()` further below at line 15. The outputs from lines 3 and 5 show that the class of `Returns$Date` changes from `factor` to `Date`; you will soon see why this is useful.

At line 10, `summary()` outputs summary statistics (minimum, first quartile, median, mean, third quartile, and maximum) of the variables in `Returns`. At line 11, `str()` prints the structure of this data frame. The function `str()` can be especially useful when working with complex data structures, for example, a list of lists.

Run the following lines:

```
12 stock.names=c("AAPL", "CAT", "GS", "DOW", "NASD100")
13 Returns2 = Returns[ ,stock.names]
```

[1]It is common to add "()" to the name of a function to indicate that it is an `R` function and takes arguments.

```
14 plot(Returns2)
15 plot(as.ts(Returns2))
```

Line 13 creates a new data frame named `Returns2` containing the columns of `Returns` that correspond to the symbols in `stock.names`. We are using the returns for only a subset of the "assets" in `Returns` so that the `R` output is more compact.

Line 14 creates a scatterplot matrix of the returns for the "assets" in `Returns2`. The `plot()` function plots data frames as scatterplot matrices and time series as time series plots. This is an example of how the output of a function depends on the class of the object it is acting upon.[2]

For each of the "assets" in `Returns2`, line 15 plots the returns in time order, that is, as a time series plot; the function `as.ts()` in 15 causes `Returns2` to be plotted as if it had class `ts` (times series).

You can create a more detailed scatterplot matrix of the returns in `Returns2` by using the function `pairs.panel()` in the `psych` package. You can install the `psych` package by running the line

```
16 install.packages("psych")
```

Once the `psych` package is installed, you can obtain the detailed scatterplot matrix by running the lines

```
17 library(psych)
18 pairs.panels(Returns2, breaks = 50)
```

In addition to plotting the scatterplots, `pairs.panels()` shows histograms, kernel density estimates, loess curves (the red curves on the scatterplots), and correlation ellipses (although these are obscured by the large number of points). The argument `breaks` in `pairs.panels()` is the number of bins used for the histograms. You can experiment with a smaller or larger number of bins than 50, say 15 or 150, to see why 50 is a sensible choice.

The correlation ellipses are evident if there are fewer points, as is the case if the most recent 150 returns are used, i.e., the last 150 rows of `Returns2` are used, which is accomplished by using the following lines

```
19 n=dim(Returns2)
20 pairs.panels(Returns2[(n[1]-149):n[1], ], breaks=50)
```

[2]More precisely, `plot()` is a generic function that calls specific functions depending upon the class of an object. Thus, `plot()` calls `plot.data.frame()` line 14, and `plot()` calls `plot.ts()` at line 15.

Note that `Returns2` has 2927 rows corresponding to the dates and 5 columns for the five "assets" under consideration. Consequently, `n=dim(Returns2)` produces a variable `n` having two entries, 2927 and 5, and `n[1]` is the first of these entries. The variable `(n[1]-149):n[1]` has 150 entries, running from 2778 to 2927 in steps of 1, so `Returns2[(n[1]-149):n[1], ]` selects the last 150 rows of `Returns2`, i.e., rows 2778 to 2927, and all of the columns of `Returns2`.

The time series plot produced by line 15 is deficient because the horizontal axis is indexed by the period number t and not by the actual date to which the period corresponds; note that $t = 1$ is for the date 01/04/2006 and $t = 2927$ is for 08/18/2017. To plot the returns against the actual date, instead of against the index t, use the following lines:

```
21 par(mfcol=c(3,2))
22 for(i in 1:5)
23 {
24 plot(Returns$Date, Returns2[ ,i], type="l", xlab="Date", ylab=stock.names[i])
25 }
```

For these lines to work properly, it is essential that line 4 has been run first so that `Returns$Date` has been converted from class `factor` to class `Date`. Remember that volatility clustering corresponds to non-constancy of variance of the log gross returns. The dates of the periods for which the volatility is large can be easily identified by examining the plots produced by using these lines. In line 21, `c(3,2)` stipulates that the plots are presented in a 3×2 array, and the `mfcol` command (make figure by columns) causes the plots to be placed in the array by filling each column before moving to the next. On the other hand, the command `mfrow` (make figure by rows), which you might try, would cause the plots to be placed in the array by filling each row before moving to the next. The command `type="l"` causes the successive points in the plots to be joined by lines.

Note that the scatterplot matrix generated by using `pairs.panels()` in line 18 gives contemporaneous correlations. For example, if $r_{\mathrm{GS},t}$ is return at time t from GS and $r_{\mathrm{NASD100},t}$ is the return at time t from NASD100, then the sample estimate of the correlation between $r_{\mathrm{GS},t}$ and $r_{\mathrm{NASD100},t}$ is 0.66. The covariance matrix, correlation matrix, and means of the returns in `Returns2` can be obtained directly by using the lines:

```
26 n=getOption("digits")
27 options(digits = 3)
28 cov(Returns2)
29 cor(Returns2)
```

```
30 colMeans(Returns2)
31 options(digits=n)
```

In line 26, the present setting for the number of significant digits to be displayed is stored in `n`; line 27 changes that setting to 3 to make the output more compact. The covariance matrix is obtained from line 28, the correlation matrix is obtained from line 29, and the means are obtained from line 30. In line 31, the setting for the number of significant digits is returned to its original value. The default value for `digits` is 7.

The returns for any particular "asset" can be examined for serial correlation by creating a scatterplot of r_t against r_{t-1} for the "asset." The following lines produce such a scatterplot for GS:

```
32 ret = Returns$GS;  n = length(ret); min(ret); max(ret)
33 par(mfcol=c(1,1))
34 plot(ret[-n], ret[-1], xlab = expression(r[t-1]),
35      ylab = expression(r[t]), xlim = c(-0.22, 0.24),
36      ylim = c(-0.22, 0.24))
```

Line 32 defines a variable `ret` that is a copy of `Returns$GS`; this line is merely for convenience so that the simpler name `ret` can be used instead of `Returns$GS` on subsequent lines. Of course, any other of the "assets" could be used in place of GS; for instance, using `ret = Returns$CAT` at line 32 would produce the scatter plot for CAT.

At line 34, `ret[-n]` contains the first $n-1$ returns[3] and `ret[-1]` contains the last $n-1$ returns. Thus, the plot is of r_t against r_{t-1} as desired. Mathematical notation can be specified by using the function `expression()`; for example, in line 34, `expression(r[t-1])` will print r_{t-1} as the label on the x-axis.

A more detailed scatter plot for GS can be obtained by using the `pairs.panels()` function as shown in the following lines:

```
37 ret = Returns$GS
38 n = length(ret)
39 pairs.panels(cbind(ret[-n],ret[-1]),breaks=50)
```

In line 39, the variables `ret[-n]` and `ret[-1]`, each of length 2926, are pasted together as two columns by the command `cbind(ret[-1],ret[-n])` to form a variable of dimension 2926×2.

[3]The "$-n$" means delete the `n`-th element.

Suppose that the initial value of an asset is P_0 and that the log gross returns for the asset are $r_1, r_2, \ldots$ i.i.d. with $r_t \sim N(\mu, \sigma^2)$. Consider P_t, the value of the asset at period t. Recall that, after t periods, the log gross return for the asset is

$$\log\left(\frac{P_t}{P_0}\right) = r_t(t) = r_1 + \cdots + r_t \sim N(t\mu, t\sigma^2).$$

Then, the probability that the value of the asset at period t is at most x is

$$P(P_t \leq x) = P\left(\log\left(\frac{P_t}{P_0}\right) \leq \log\left(\frac{x}{P_0}\right)\right),$$

which can be calculated as the c.d.f. of a normal variable with mean $t\mu$ and variance $t\sigma^2$ evaluated at $\log\left(\frac{x}{P_0}\right)$. To illustrate the function that calculates the normal c.d.f. in `R`, suppose that $P_0 = 1000$, $\mu = 0.0005$, $\sigma = 0.02$, and $t = 100$. Then $P(P_t \leq 990)$ is given by the command `pnorm(log(990/1000), mean=0.05, sd=0.2)`. The function `pnorm` calculates the normal c.d.f. for specified mean and standard deviation; it produces right-tail areas under the normal curve. The function `qnorm` produces quantiles of the normal distribution. The function `rnorm` simulates random values from the normal distribution.

Consider again the case $P_0 = 1000$, $\mu = 0.0005$, and $\sigma = 0.02$. Suppose the goal is to calculate the probability that the value of the asset falls below 990 during the first 100 periods, i.e., that at some period $t \in \{1, 2, \ldots, 100\}$, the value of the asset P_t is less than 990. There is no analytical formula for this probability, but it can be simulated by using the following lines:

```
t1=proc.time()
set.seed(4630)
niter=100000
ind=matrix(nrow=niter, ncol=1)
for (i in 1:niter)
{
logR=rnorm(100, mean=0.0005, sd=0.02)
cumR=cumsum(logR)
logP=log(1000)+cumR
ind[i]=(min(logP) < log(990))
}
estProb=mean(ind)
se=sqrt(estProb*(1-estProb)/niter)
UpperCL=estProb+qnorm(0.975)*se
LowerCL=estProb+qnorm(0.025)*se

estProb
se
```

```
58 UpperCL
59 LowerCL
60 t2=proc.time()
61 t2-t1
```

Lines 40, 60, and 61 are unnecessary for simulating the desired probability, but they are useful for calculating how long the simulation takes. Line 40 records the starting time in the variable `t1`, line 60 records the ending time in the variable `t2`, and line 61 writes the difference `t2-t1` between the two times. Line 41 sets the starting seed for the random number generator; by setting the starting seed, the results are identical over repeated runs of the simulation. The number of iterations, i.e., the simulation size, is stored in the variable `niter` in line 42. Of course, using a larger simulation size causes the simulation to take longer to run, but the resulting probability estimate is more precise. Lines 53 and 54 computes the upper and lower endpoints of the 95% confidence limit for the desired probability. For each iteration of the simulation, line 46 generates the random returns $r_1, r_2, \ldots r_{100}$, line 47 calculates the sums multi-period returns $r_1(1), r_2(2), \ldots, r_{100}(100)$, line 48 calculates $\log(P_1), \log(P_2), \ldots \log(P_{100})$, and `ind` is an indicator of whether the minimum of $\log(P_1), \log(P_2), \ldots \log(P_{100})$ is less than $\log(990)$. The estimate of the probability is calculated in line 51 as the mean of the indicators across all iterations.

Questions:

1. [**10 points**] Run lines 1 to 18, and submit the output from lines 14, 15, and 18.

2. [**5 points**] Based on the output from line 18, comment on the contemporaneous correlations between the Apple (AAPL) returns and the returns of the other four "assets."

3. [**10 points**] Run lines 21 to 25, and submit the output.

4. [**10 points**] Based on the output from lines 21 to 25, and is there evidence of volatility clustering? When did the most pronounced volatility occur? Do you have an explanation for this timing?

5. [**10 points**] Run lines 26 to 31, and submit the output.

6. [**5 points**] Based on the output from line 30, what is the mean of the returns for Caterpillar index (CAT)?

7. [**5 points**] Based on the output from line 28, what is the variance of the returns for Goldman Sachs (GS)?

8. [**5 points**] Based on the output from line 29, what is the correlation between the returns for the Dow Jones Industrial index and the returns of the NASDAQ 100 index?

9. [**10 points**] Adapt and run lines 37 to 39 for Caterpillar (CAT), and submit the output.

10. [**5 points**] Based on the output from lines 37 to 39, is there any evidence of serial correlation in the returns for Caterpillar (CAT)?

11. [**10 points**] Suppose that daily log returns for an asset are independent and normally distributed with mean $\mu = 0.004$ and standard deviation $\sigma = 0.012$. If the initial value of the asset is $P_0 = \$100,000$, what is the probability that the value of the asset is at least \$108,000 after 16 days, i.e., what is $P(P_{16} \geq \$108,000)$?

12. [**15 points**] Suppose that daily log returns for an asset are independent and normally distributed with mean $\mu = 0.004$ and standard deviation $\sigma = 0.012$. If the initial value of the asset is \$100,000, what is the probability that the value of the asset is at least \$108,000 during the next 30 days, i.e., what is the probability that at least one of $P_1, P_2, \ldots, P_{30}$ is at least \$108,000? Adapt lines 40 to 61 to simulate the probability. Provide your code. Use a simulation size of 250,000, and set the seed for random number generation to 8765. Give your estimate of the probability and a 99% confidence interval for the probability.