Section 3 Documentation

Release 1.0

Yuting Ye

Sep 14, 2016

Contents:

CHAPTER

ONE

INTRODUCTION TO R

1.1 Rstudio and R

Hopefully, you have downloaded R and Rstudio already. If you do not have chance for it yet. You can now download R from R and Rstudio from Rstudio. Please check your Operation System before downloading. R is one of the best softwares for statistical analysis. You can find all the implementations of the methods you've learned in STAT 135. Rstudio is said to be the best platform for R.

1.1.1 R basics

- Assignment

  The most straight forward way to store a list of numbers is through an assignment using the c command. (c stands for "combine.") The idea is that a list of numbers is stored under a given name, and the name is used to refer to the data. A list is specified with the c command, and assignment is specified with the "<-" symbols. Another term used to describe the list of numbers is to call it a "vector."

  The numbers within the c command are separated by commas. As an example, we can create a new variable, called "bubba" which will contain the numbers 3, 5, 7, and 9:

```
> bubba <- c(3, 5, 7, 9)
```

  When you enter this command you should not see any output except a new command line. The command creates a list of numbers called "bubba." To see what numbers is included in bubba type "bubba" and press the enter key:

```
> bubba
[1] 3 5 7 9
```

  If you wish to work with one of the numbers you can get access to it using the variable and then square brackets indicating which number:

```
> bubba[2]
[1] 5
> bubba[1]
[1] 3
> bubba[0]
numeric(0)
> bubba[3]
[1] 7
> bubba[4]
[1] 9
```

Notice that the first entry is referred to as the number 1 entry, and the zero entry can be used to indicate how the computer will treat the data. You can store strings using both single and double quotes, and you can store real numbers.

- Operation

  R's basic operation like "+", "-", "*", "/", "^", are the same as those of other languages such as Python, C, C++ and so on. There are also basic math function like "log", "sin":

```
> 2^5
[1] 32
> log(2)
[1] 0.6931472
> sin(pi)
[1] 1.224647e-16
> cos(pi)
[1] -1
```

- Read Data from a CSV file

  Please download the data from bcourse. Now I store it in "~/Desktop/temp" (You can store it in anywhere you like). Then we read this file by:

```
xom <- read.csv("~/Desktop/temp/sp500Historical.csv")
```

  The data file is in the format called "comma separated values" (csv). That is, each line contains a row of values which can be numbers or letters, and each value is separated by a comma. We also assume that the very first row contains a list of labels. The idea is that the labels in the top row are used to refer to the different columns of values. We can use a command to take a look at the data or, a good thing for Rstudio, is that you can just click:

```
View(xom)
```

  The data is a 252 by 7 matrix. The first 5 rows look like:

Date	Open	High	Low	Close	Adj.Close	Volume
12-Sep-16	2,120.86	2,150.76	2,119.12	2,148.10	2,148.10	321,471,413
9-Sep-16	2,169.08	2,169.08	2,127.81	2,127.81	2,127.81	4,233,960,000
8-Sep-16	2,182.76	2,184.94	2,177.49	2,181.30	2,181.30	3,727,840,000
7-Sep-16	2,185.17	2,187.87	2,179.07	2,186.16	2,186.16	3,319,420,000
6-Sep-16	2,181.61	2,186.57	2,175.10	2,186.48	2,186.48	3,447,650,000

  For more details and more advanced knowledge, please refer to R Tutorial. This is a course tutorial by Kelly Black in University of Georiga.

CHAPTER

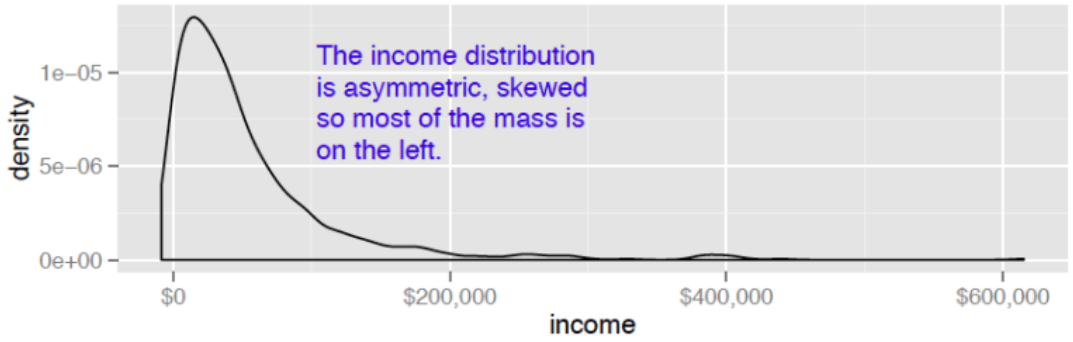
TWO

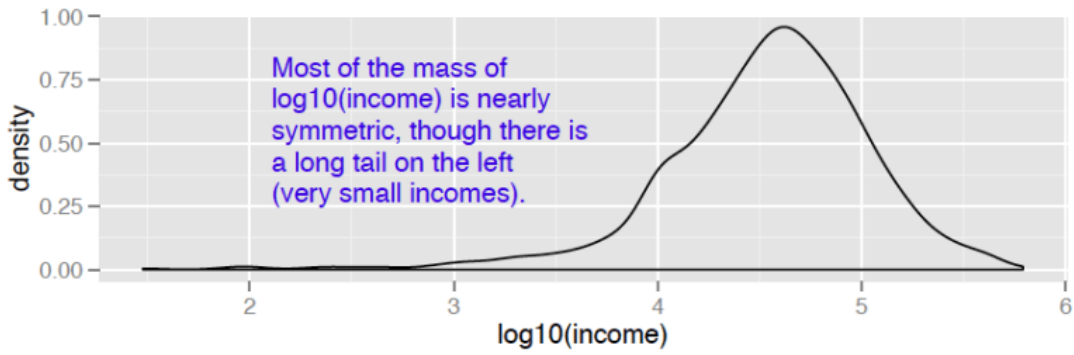
BLACK SCHOLES

A convenctional way to compare the price of each day is to compare the "log Returns":

```
> close.prices <- as.numeric(gsub(",", "", xom$Close))
> length(close.prices)
[1] 252
>logReturns<-log(close.prices[2:250]/close.prices[1:249])
```

- We need to transform our data from a comma separated number, e.g., 2,000 into 2000 before any data analysis.
- Why log returns? We can compare the prices of two consecutive days in many ways, e.g., ratio, difference and what we did, log ratio. There are two reasons why we choose log ration.
  - Compare to difference, ratio is normalized. For example, we have no idea if the difference of 1 big enough. But ratio of 1.1 means the return of today has increased by 10% compared to that of yesterday.
  - log transformation can also makes skewed data more 'normal'.

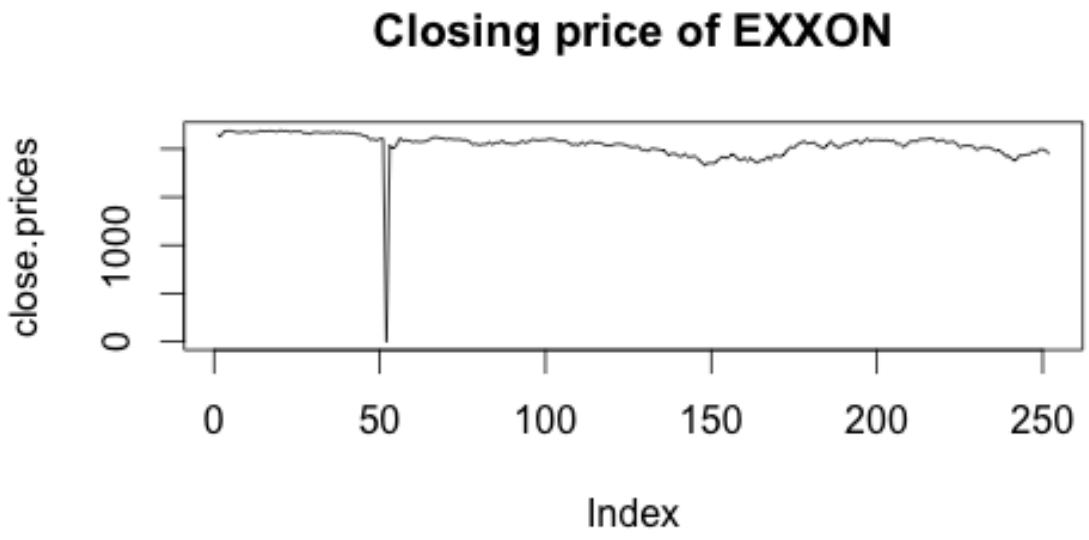


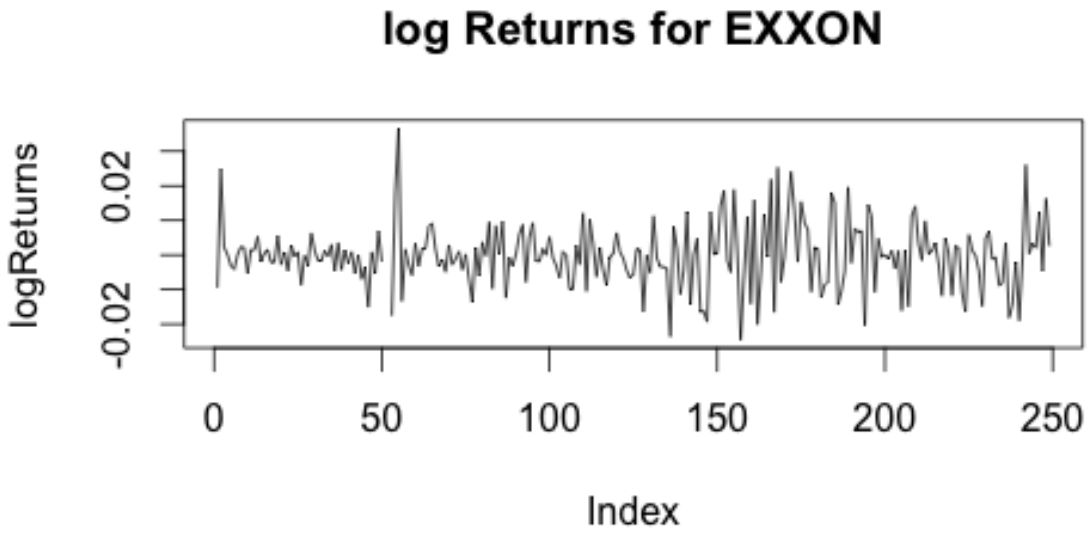


- R manipulates data in a vectorized way, including addition, multiplication and so on.

  We take a look at the logReturns:

```
plot(logReturns,type="l",main="log Returns for EXXON")
plot(close.prices,type="l",main="Closing price of EXXON")
```





  - We can see that log Returns looks much better than the Close prices because of the normalization.

Now suppose we're traders and and want to choose whether to buy an option or not. Recall Black Scholes formula: Let S_t be the price of a stock at time t. Popular statistical model:

$Z_t = log\frac{S_{t+dt}}{S_t} \overset{iid}{\sim} N((\mu - \sigma^2/2)dt, \sigma^2 dt)$

Great result by Black-Scholes and Merton: consider an option that gives the right to buy the stock S at time, for a price K. Then

$C(S_0, T, K, r, \sigma) = S_0\Phi(d_+) - K\Phi(d\_)exp(-rT)$

where

$d_\pm = \frac{log(S_0/K)+(r\pm\frac{\sigma^2}{2})}{\sigma\sqrt{T}}$

Here, we have K = \$95, T = 3 months, r = 0.25% pear year, $S_0 = 88.65$, What would you do?

- First we need to estimate the variance $\hat{\sigma}^2$:

```
sigmahat <- sqrt(var(logReturns[!is.infinite(logReturns)]) * 365) # make the unit␣
↪in years
```

  Notice there are some outliers, which we do not want, and thus removing. Here, for simplicity, we treat the close prices are daily data, that is, the time interval for any two consecutive is one day. In reality, some intervals are one day, some are three days because of weekends and holidays.

- Use Black-Scholes:

```
> K <- 95
> T <- 3/12 # make the unit in years
> r <- 0.25/100
> S0 <- 88.65
>
> d1 <- (log(S0/K) + (r + sigmahat^2/2) * T)/(sigmahat * sqrt(T))
> d2 <- (log(S0/K) + (r - sigmahat^2/2) * T)/(sigmahat * sqrt(T))
>
> C <- S0 * pnorm(d1) - K * pnorm(d2) * exp(-r * T)
> C
[1] 1.011341
```

- Use the delta method to assess the variablility of this estimator. Suppose that $\hat{\sigma} \sim N(\sigma, D^2)$, with D known. We fix other quantities and let:

  $$g(\sigma) = C(S_0, T, K, r, \sigma)$$

  Then by delta method, we have:

  $$g(\hat{\sigma}) \sim N(g(\sigma), D^2[\frac{\partial}{\partial\sigma}g(\sigma)]^2)$$

CHAPTER

THREE

SECTION 8.10, PROBLEM 44

The file bodytemp contains normal body temperature readings (degrees Fahrenheit) and heart rates (beats per minute) of 65 males (coded by 1) and 65 females (coded by 2) from Shoemaker (1996). Assuming that the population distributions are normal (an assumption that will be investigated in a later chapter), estimate the body temperature means and standard deviations of the males and females. Form 95% confidence intervals for the means. Standard folklore is that the average body temperature is 98.6 degrees Fahenheit. Does this appear to be the case.

3.1 Solution

We first load the data and extract a basic information:

```
data <- read.csv("~/Desktop/temp/bodytemp.csv")

n_m <- length(data[data[, 2] == 1, 1])
n_fm <- length(data[data[, 2] == 2, 1])
```

Then we calculate the means and standard deviations respectively:

```
## means
mean_temp_m <- mean(data[data[,2] == 1,1])
mean_temp_fm <- mean(data[data[,2] == 2,1])

## standard deviations

sd_temp_m <- sd(data[data[,2] ==1, 1])
sd_temp_fm <- sd(data[data[,2] ==2, 1])
```

Recall that if

$$X_i \overset{iid}{\sim} N(\mu, \sigma^2), i = 1, \ldots, n$$

then we have

$$\sqrt{n}\frac{\bar{X}-\mu}{S_n} \sim t_{n-1}$$

where

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i,\ S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

This conclusion is important. You may use them as well in HW3.

Let's form the confidence interval now:

```
> interval_temp_m <- c(mean_temp_m - sd_temp_m * qt(0.975, n-1)/sqrt(n),
+                      mean_temp_m + sd_temp_m * qt(0.975, n-1)/sqrt(n))
```

```
> interval_temp_m
[1] 97.93147 98.27776
>
> interval_temp_fm <- c(mean_temp_fm - sd_temp_fm * qt(0.975, n-1)/sqrt(n),
+                       mean_temp_fm + sd_temp_fm * qt(0.975, n-1)/sqrt(n))
> interval_temp_fm
[1] 98.20962 98.57807
```

The result seems not the case for standard folklore.