1

R INSTALLATION AND BASICS

In this chapter, we offer a brief introduction to R. First, we argue why any finance-major student should master at least one computer language. To support this argument, we will discuss two concepts: Open-Source and Open-Source-Finance. What is the future of finance equation? We use three words to summarize it: Finance, Programming and Data. Then, we will explain why two top choices are R and Python in terms of programming languages for finance students. After that, we will explain how to install R software, how to launch and quit R, whether R is case sensitive, and how to assign a value or values to a variable. In a sense, we assume that any reader, especially a finance-major student, knows nothing about this wonderful software. In particular, we will cover the following:

- Why finance-major students should master at least one programming language
- Comparisons between R, Python, Matla and SAS
- Concepts of Open-Source and Open Source Finance
- Five step to download and install R
- How to launch and quit R
- R basics and some embedded functions
- Three ways to assign values to a variable
- Choosing meaningful variable names
- Numerical vs. string variables
- Inputting data via `scan()`
- Copy-and-paste from notepad or Excel
- Finding embedded n-letter functions
- Author’s programming path
- List of all chapters
- Finding help

1.1 ONE PROGRAMMING LANGUAGE PLEASE

Our society has entered a so-called big data era. This means that we could use big data to solve many issues, such as choose a better way to develop good drugs, optimize our operations by analyzing a huge amount of data. Here using finance as an example. For a topic called Financial Statement Analysis, traditionally, investors or financial analysts just analyze a few companies. Is it possible to analyze ALL companies available? Another example is to apply the Benford Law, the Law Of the First Digit. It is easy to apply it to a few dozen companies. Since the SEC (Securities and Exchange Commissions) has many ALL financial statement available from 2009 to today, could we apply the Benford Law to all public companies filed Balance Sheets, Income Statement and Cash Flow Statement in the first quarterly 2020? By the way, in the first quarter in 2020, there are xx companies filed various types of reposts. Interested readers could try one quarterly index and the location is here: https://www.sec.gov/Archives/edgar/full-index/.

Using the SEC quarterly index as an example, could our students process those files in the first place? There is research area called Market Microstructure. The database used by this area is called TAQ (Trade and Quote) database. The size of this type of financial data bae is huge, about several Giga bites. Again, interested readers could download one or two such data sets at ftp://ftp.nyxdata.com/Historical%20Data%20Samples/. The third example is related to Census data. Could we link the demographic data from Census data in 2010 and 2020 to predict the success or failure of private schools? To accomplish those tasks, researchers/students need a programming language. It is our prediction that within 3 years, all finance-major students would be required to learn one programming language. Within 5 to 8 years, all business school students will have to meet the standard.

Later in this book we will show how to process relatively big data sets such as SEC filing, Census and NYSE high-frequency data.

1.2 CONCEPT OF OPEN SOURCE AND OPEN-SOURCE-FINANCE

Open Source is defined as free and everyone could use it. First, we use YouTube as an example. Many of use have watched many interesting, educational or funny vides by using its platform, http://youbute.com. To editing those videos, producers could use free software to do so. The top three open-source video editing software are Davinci Resolve, https://www.blackmagicdesign.com/products/davinciresolve/, Shotcut, http://shotcut.com, and Lightworks, https://www.lwks.com. Those software is more than enough for new learners to master skills of video editing.

For finance, the related concept is called Open-Source-Finance (OSF), first mentioned by Kane and Masters (2009). OSF has three components: open software, open data and open code. For the open software component, we have R, Python, Perl, Octave, and Julia. All of them are free. For the open data, we devoted a whole chapter to it: *Chapter 16*, *Open Source for finane/economics/accounting*. Open code means that researchers show their code with others. A typical example is Github, http://github.com. The objective of the platform is:

> *GitHub is a development platform inspired by the way you work.*
> *From open source to business, you can host and review code,*
> *manage projects, and build software alongside 50 million developers.*

1.3 FUTURE OF FINANCE EDUCATION: 3-WORD SUMMARY

In terms of future of finance education in the future (for the next 5 to 10 years) at college/university levels, we could use three words to summary it: finance, programming and data. Those three areas are vital important for our next level's development. The first word represents our current situation.

Word #1: Finance

For the first word, students would take related courses, such as Corporate Finance, Investment, Portfolio Theory, Financial Modeling using Excel, Options Theory, Econometrics, Fixed Income plus other related courses.

Word #2: Programming

For programming, as we just mentioned in the previous, students should master at least one programming language. If we could design a 4-year finance major program, we would demand two programming languages: R or Python plus SAS. It means student should learn R and SAS or Python and SAS. One related question is why SAS since SAS is not free. The major reason is SAS's super ability to handle data. Another question is how to measure their level of programming skills. Our approach is through a good term project. For term project, we have devoted a whole chapter: *Chapter 29*, *Term Projects.* Here are few typical topics: Test if whether so-called Momentum trading strategy profitable or not; Apply Benform Law to all companies that filed the Financial Statement to SEC in 2020; Conduct Financial Statement Analysis for 50 companies by using SEC filing data from 2009 up to today;

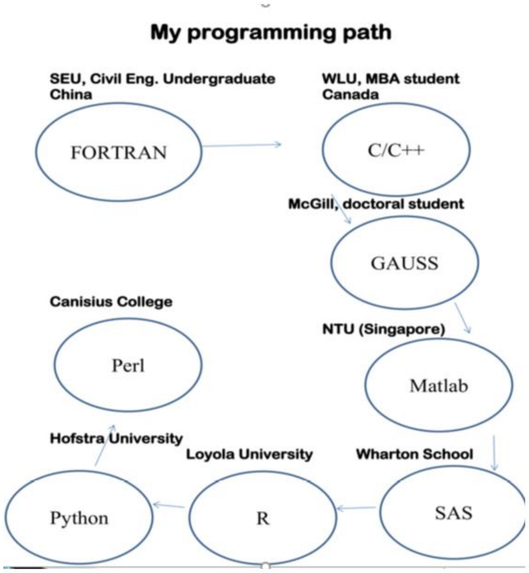
Word #3: Data

For the last word of "data", students will be trained to handle big data sets by writing various programs using their programming skills. Here are several typical examples. Download all SEC filing quarterly index files from 1994 up to today and generate related R data sets, see the data sets at https://www.sec.gov/Archives/edgar/full-index/. Download one or two days high frequency data at ftp://ftp.nyxdata.com/Historical%20Data%20Samples/, then estimate stocks spreads by using bid and ask prices. Download and process 2020 Census Summary File #1 and their data sets at https://www.census.gov/data/datasets/2010/dec/summary-file-1.html. For students at research schools, they should understand CRSP, Computat and TAQ quite well. For the CRSP databases, see our paper titled 'CRSP for Teaching", Yan(2018).

1.4 AUTHOR'S PROGRAMMING PATH

The author's background in terms of programming might inspire finance major students to learn at least one programming language. His major is finance since he earned a Doctoral Degree from McGill. On the other hand, he is good at several computer languages, such as SAS, R, Python, Matlab, Octave and C. Below is his programming path.

His first programming language is Fortran learnt when he was an undergraduate student at the Department of Civil Engineering, Southeast University (in China). During his MBA study, he learnt C by himself when a finance professor asked him to speed up data retrieval process. When teaching, the professor like to show stock price data to his students. However, it took a few minutes to retrieve one stock's data. After the author learnt that an index could be a good solution. After he produced such an index file by using C (languge), it took just a few second to retrieve a stock's data. Here is the logic: before, the underlying software had a sequential search, i.e., search each stock until a match. Now, it is called "random search", the program could pin point the starting and ending points by using an index file. The following graph shows his path.



When studying finance at McGill University, he learnt GAUSS and switched to Matlab at Nanyang Technical University (in Singapore). After joined Wharton School, he spent 8-year to polish his SAS skills since it is the main language to organize various financial databases. When came back to teaching in 2010, he went to several teaching schools. It means those schools have no SAS, Matlab. In addition, no expensive financial databases, such as CRSP, Compustat nor TAQ. Since then, he focused on open-source software, such as R and Python. So far, he has published 3 books: Financial Modeling using R (2018), Python for Finance (2nd edition, 2017), and Hands-on Data Science with Anaconda (2018, with James). The Chinese and Korean translation of "Python for Finance" were published as in 2017. For the third book, the publisher ask the authors to apply 4 open-source languages: R, Python, Octave and Julia. It is kind of crazy, isn't it?

Based on his programming path, it is quite possible for a 4-year finance-major student to master two computer languages. Usually, it takes much longer time to learn the first one. Assume that it takes two years to study and polish the first programming language. For the second one, it takes about one-year. For the third one, it might be just half year! The author still remember vividly the conversation between him and a finance professor when he just started his PhD program at McGill.

Professor: *Do you know GAUSS?*

Student: *You mean GAUSS distribution? Yes, I heard it.*

Professor: *No, GAUSS is a programming language.*

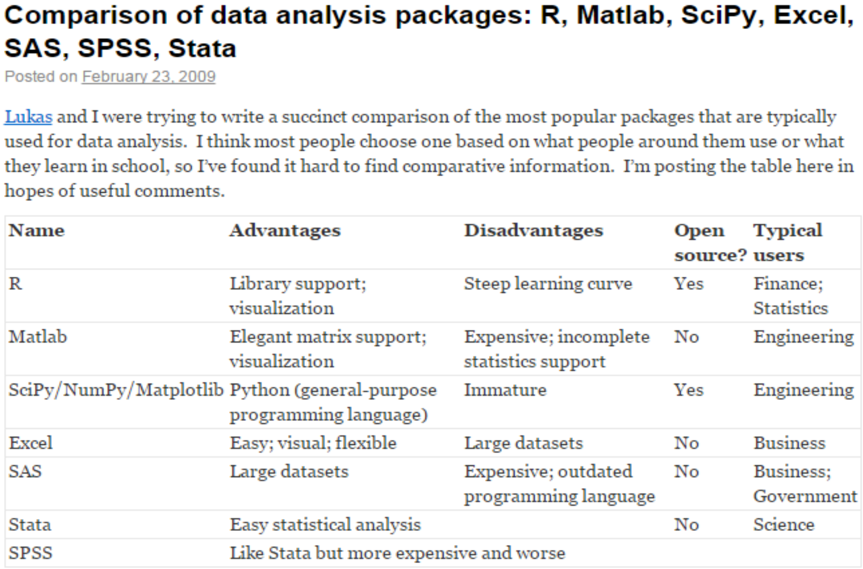
Student: *Sorry, never heard of it.*

Professor: *No problem. Since you know two programing languages already, you will learn it in no time. Here are three volumes of the manuals and my copy-machine card. Come back to my office in two weeks.*

Two weeks later, he started to write various programs in Gauss as an RA (research assistant).

1.5 WHY R?

In the previous section, we mentioned a few software: R, Python, Octave, Perl and Julia. R and Python are top choices for many master degree program from quantitative finance, business analytics and data science. Lucas and Jetee (2009) have the following comparison.



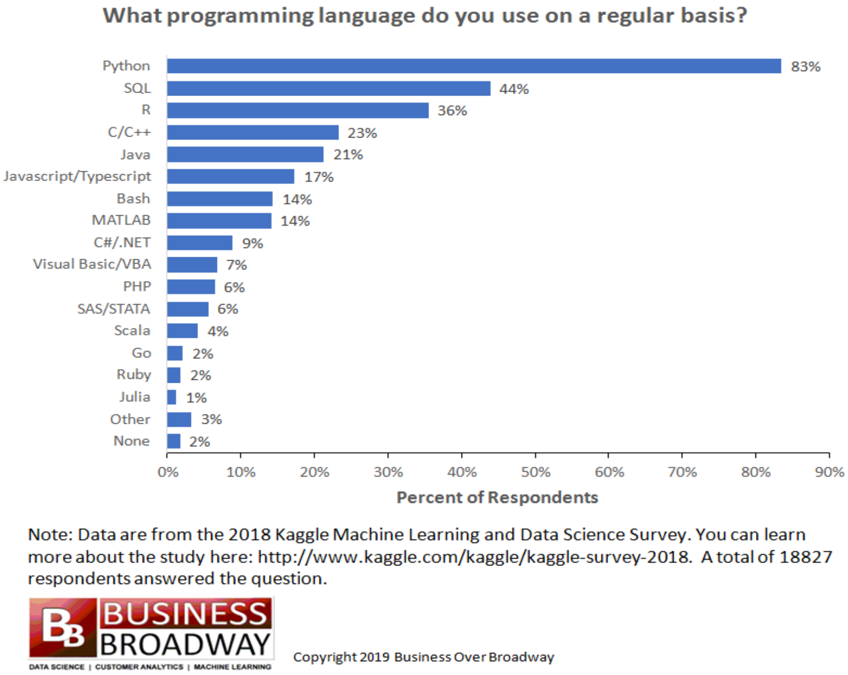
Comparison of data analysis packages: R, Matlab, SciPy, Excel, SAS, SPSS, Stata

Posted on February 23, 2009

Lukas and I were trying to write a succinct comparison of the most popular packages that are typically used for data analysis. I think most people choose one based on what people around them use or what they learn in school, so I've found it hard to find comparative information. I'm posting the table here in hopes of useful comments.

Name	Advantages	Disadvantages	Open source?	Typical users
R	Library support; visualization	Steep learning curve	Yes	Finance; Statistics
Matlab	Elegant matrix support; visualization	Expensive; incomplete statistics support	No	Engineering
SciPy/NumPy/Matplotlib	Python (general-purpose programming language)	Immature	Yes	Engineering
Excel	Easy; visual; flexible	Large datasets	No	Business
SAS	Large datasets	Expensive; outdated programming language	No	Business; Government
Stata	Easy statistical analysis		No	Science
SPSS	Like Stata but more expensive and worse			

Hayes (2019) shows the following image. From it, we can see that Python is number 1 while R is number 3.



Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

Copyright 2019 Business Over Broadway

Based on our many years' teaching experience, we have the following table. SAS and Matlab are two expensive software. However, they are used intensively by financial industry. For SAS, it is used intensively by Banking industry. Since they are paid software, the support will be best. In the following table 5 is the best grade.

Table 1.1 Comparison between R, Python, Matlab and SAS.

	R	Python	SAS	Matlab
Cost	5	5	2	2
Easy to learn	5	4	2	2
Data handling	4	4	5	2
Big data handling	2	2	5	2
Current Job	4	4	5	5
Future trend	5	5	3	4
Support	2	2	5	5

In the above table, there are two entries related to data: Data handling and Big Data Handling. The big data is defined here at least several Giga bit. For example, the daily TAQ data could be treated as big data in terms of this book. Based on our multi-year teaching experience of teaching programming language to finance major students, R is the best one to start with. There are several reasons behind this. First, R is relatively easy to learn, compared with Python. Second, R is used quite intensively in many areas. Third, exists many R packages that could help new learners greatly.

Actually, we end up with two choices: R or Python. Based on our multiple years teaching experience, R is better choice than Python. This does mean that R is promising than Python. Here our objective is "which programing language, R or Python, is a better choice for the FIRST programming language for a finance-major student? We have three keywords: FIRST, and FINACE STUDENT

1.6 RELATED COURSE MATERIALS

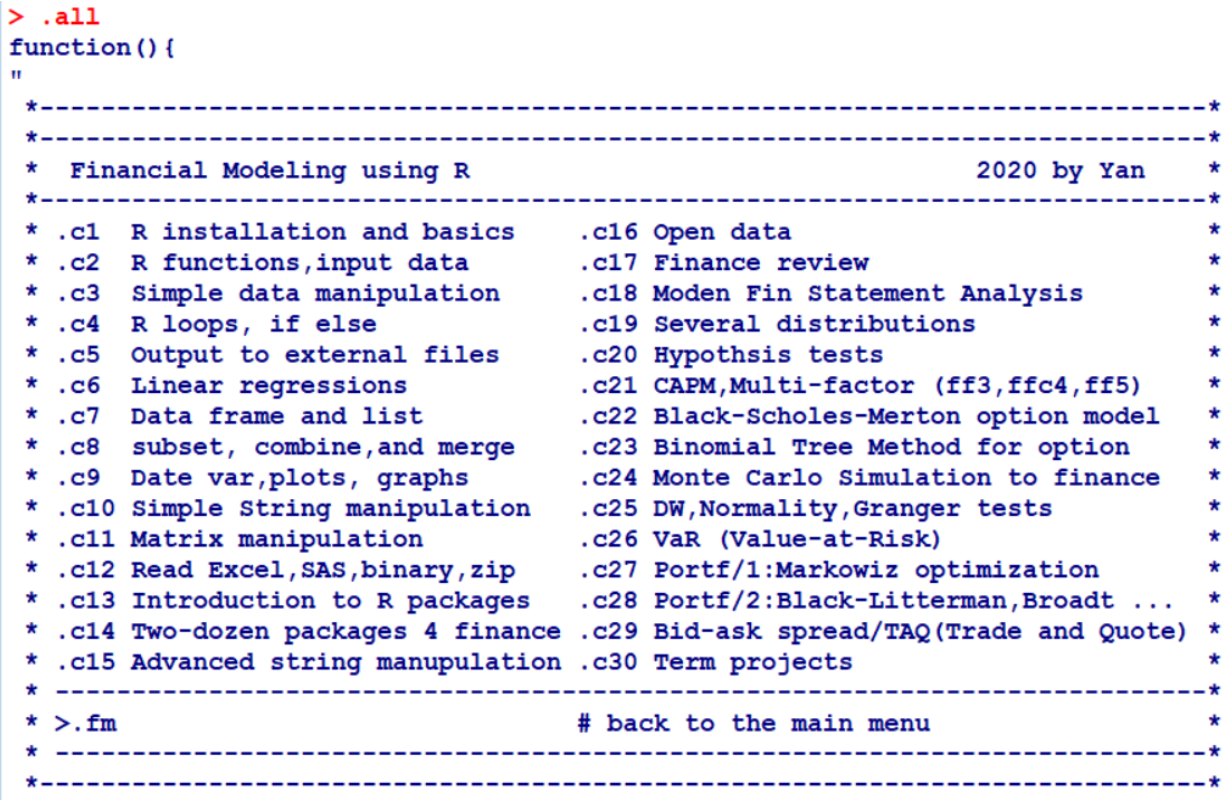
To help a potential student to learn this course online, we have generated many data sets in various forms, such as csv, RData, sas7bdat and binary. For more detail, please see the chapter, : *Chapter 16, Open Source for finane/economics/accounting*. We have generated many in-class exercises. For each lecture, we expect to have at least two in-class exercises. Hands-on is the most important aspect of learning a computer language. In our paper, *Teaching programming skills to finance students: how to design and teach a great course*, we have summarized 7 critical factors to make programming course a success. Those 7 factors are: strong motivation, a good textbook, hands-on learning environment, being data intensive, a challenging term project, multiple supporting R datasets, and an easy way to upload such R datasets. In this book, we would target those objectives. The related website is shown below.

http://datayyy.com/fmr/

We plan to generate two videos for each week's lecture. Thus, for 15 weeks, readers should expect 30 videos. Do a good term project is an integral part of taking this course. To help students choose a good term project, we would generate around 5 videos.

1.7 LIST OF ALL CHAPTERS

Later in the chapter, we will show how to list all chapters. For now, we just give a list shown below. The whole course/book is divided into two parts: related R and related its various applications to finance. The first 14 chapters belong to Part I and the rest chapters belong to Part II.



```
> .all
function(){
"
 *-----------------------------------------------------------------------------*
 *-----------------------------------------------------------------------------*
 *  Financial Modeling using R                                 2020 by Yan     *
 *-----------------------------------------------------------------------------*
 * .c1  R installation and basics    .c16 Open data                            *
 * .c2  R functions,input data       .c17 Finance review                       *
 * .c3  Simple data manipulation     .c18 Moden Fin Statement Analysis         *
 * .c4  R loops, if else             .c19 Several distributions                *
 * .c5  Output to external files     .c20 Hypothsis tests                      *
 * .c6  Linear regressions           .c21 CAPM,Multi-factor (ff3,ffc4,ff5)     *
 * .c7  Data frame and list          .c22 Black-Scholes-Merton option model    *
 * .c8  subset, combine,and merge    .c23 Binomial Tree Method for option      *
 * .c9  Date var,plots, graphs       .c24 Monte Carlo Simulation to finance    *
 * .c10 Simple String manipulation   .c25 DW,Normality,Granger tests           *
 * .c11 Matrix manipulation          .c26 VaR (Value-at-Risk)                  *
 * .c12 Read Excel,SAS,binary,zip    .c27 Portf/1:Markowiz optimization        *
 * .c13 Introduction to R packages   .c28 Portf/2:Black-Litterman,Broadt ...   *
 * .c14 Two-dozen packages 4 finance .c29 Bid-ask spread/TAQ(Trade and Quote)  *
 * .c15 Advanced string manupulation .c30 Term projects                        *
 * ----------------------------------------------------------------------------*
 * >.fm                              # back to the main menu                   *
 * ----------------------------------------------------------------------------*
 *-----------------------------------------------------------------------------*
```

The whole books in divided into two parts. Part I is from chapters 1 to 16, they are about R. Part II is about its application to finance. This book is designed for a one-semester course. For each week, instructor could use two chapters: one from Part I and one from Part II. For more detail, see the Appendix related to our syllabus.

1.8 HOW TO DOWNLOAD AND INSTALL R?

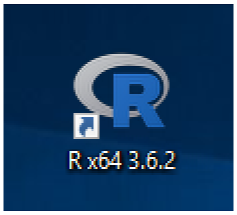
To install R, we have the following 5 steps.

Step 1: Go to http://www.r-project.org
Step 2: Click "CRAN" under "Download" (left-hand side)
Step 3: Choose a mirror address
Step 4: Choose appropriate software (PC, Mac)
Step 5: Click "base"

After you have finished installation, an R icon will appear on your desktop.

1.9 HOW TO LAUNCH R AND QUIT R?

To start R, double click the R icon on your desktop.



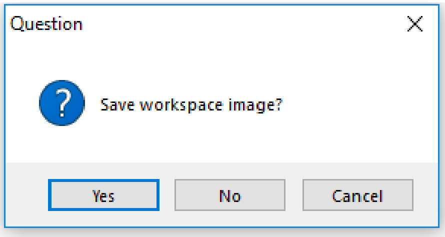
To quit, just type q() from the R prompt (>).

```
> q() # first way to quit
```

Anything after # is a comment.

```
# this is a comment line
# >  is the R prompt
```

When quitting, the program will ask you whether to “Save the workspace image” which means to keep all your variables or functions for future usage. At this stage, just answer no.



Save workspace image?

See below for another way to quit.

```
# [click] "file" on the menu bar - - > "exit"
```

To quit R without saving, we use q(“no”) command.

```
> q("no")  # quit R without saving variables and functions
> q("yes") # quit R and keep variables and functions
```

1.10 R BASICS AND SOME EMBEDDED FUNCTIONS

To assign a value to a variable, we use “<-“, “=” or “->”.

```
> x<-10           # assign 10 to x
> y=20            # assign 20 to y
> 30 ->z          # assign 30 to z
> title<-"Hello"  # title is a character (string) variable
```

To show the value of a variable, simply type its name.

```
> x<-10           # assign 10 to x
> x
[1] 10
```

In R, we don’t need to define a variable before use it.

```
# a variable is not formally defined before its assignment
> fv<-100
```

R is case sensitive which means that capital X and low case x are different variables.

```
> x<-10     # lower case x
> X         # capital letter of x
Error: object 'X' not found
```

To put several R commands on one line, semi-colons are used

```
# semi-colon (;) can be used to separate different lines
> fv<-10; pv<-100; n<-10; rate<-0.05
```

To assign values to a vector, we use the c() function, where c means column.

```
> # assign values to a vector
> x<-c(1,2.5,3.4,6.2)
```

If the increment value is one, we could use n1:n2.

```
> y<-1:50  # assign 1, 2, 3, . . ., 50 to y (y is a vector)
```

We could reverse the order.

```
> z<-10:1  # assign 10,9, 8, . . ., 1 to z  (z is a vector)
```

Try the following codes and print x to see the result.

```
> x<-1.5 :10
```

1.11 LIST FUNCTION, ls() AND REMOVE FUNCTION, rm()

Sometimes, we need to check all existing variables (objects). For this reason, we use the ls() function.

```
> ls()
```

When a variable is no longer needed, we could remove it from the memory.

```
> rm(x) # remove variable called x
```

To remove several variables (objects) simultaneously, we use comma to separate them.

```
> rm(x,y,pv)     # remove x, y and pv
```

To remove all variables (objects), we have the codes below.

```
> rm(list=ls())  # remove all variables (objects)
```

The 2nd way to remove all objects (variables) is given below.

```
# [click] "Misc" - - > "Remove all objects … "
```

To print a character variable (a string) on the screen, we could use the cat() or the print() functions. Remember to put our sentences in double or single quotation marks.

```
> cat("hello, world!\n\n\n") #\n is for a new line
hello, world!
```

```
>
```

The print() function could be used instead.

```
> print('hello R!')
[1] "hello R!"
```

Note that “\n” is not working for the print() function.

```
> print('hello world\n')
[1] "hello world\n"
```

We could also print a defined variable.

```
> x<-'this is great'
> print(x)
[1] "this is great"
```

1.12 NEXT LINE SYMBOL (+), GO BACK TO THE R PROMPT

When one command occupying multiple lines, the symbol of + will appear. Assume that we intend to assign 1 to 10 to x.

```
> x<-1:10 # assign 1,2,… 10 to x
```

For some reasons, we hit the enter-key before we finish the whole command show below. In other words, we use several lines to finish the command.

```
> x<-1:10 # assign 1,2,… 10 to x
```

It is often, especially for a beginner, that we type a few wrong keys, such as a double or single quotation mark without a matching one. Sometimes, we simply don’t want figure out where the issue is since it might be too time-consuming. Instead, we just want to go back to R prompt and retype the command. In those cases, we hit the ‘Esc’ key, on top-left of our keyboard, to return to the R prompt (>).

```
> x<-‘9”(999asdfklj

+ >   # use ‘Esc’ to come back to the R prompt
```

1.13 SEVERAL WAYS TO ASSIGN A VALUE TO A VARIABLE

The simplest way to assign a value to a variable is to use “<-“.

```
> x<-10
```

To show the value of a variable, simply type its name.

```
> x
[1] 10
```

To assign a value to an variable, we could use “=” or “->” as well.

```
> y=2
> 10->x
```

The “->” assignment could make our debug efforts easier. Assume that we want to test a program to estimate the present value of $100 received in two years with an 8% annual discount rate. We could

```
> 100/(1+0.08)^2
[1] 85.73388
```

After that we change our mind trying to sign the result to a variable, such as pv. To save time, we simply use upper arrow key to recall the previous command. Then add “->pv” at the end of the above command.

```
> 100/(1+0.08)^2->pv
> pv
[1] 85.73388
```

To assign a set of values, we use “c(1,2.6,4.3,5.25)” where “c” stands for column.

```
# assign a vector (column values)
> X<-c(1,2,4,6)
```

To assign a set of consecutive integers, we could use n1:n2, such as 1:10.

```
> y<-1:50
> x<-c(1:5,8:12)
> x
[1]  1  2  3  4  5  8  9 10 11 12
```

We can input data from high to low, i.e., reverse the order.

```
> y<-5:1
```

The `rev()` function could be used to reverse an input data set.

```
> x<-5:1
> x<-rev(1:5)  # same as above
```

1.14 TO VIEW OBJECTS WE USE THE ls() FUNCTION

We can use the *ls()* function to list all objectives including variables.

```
> ls()      # list all variables
```

1.15 seq() FUNCTION

The *seq()* function is used to generate a set of values, see an example below.

```
> x<-seq(1, 19, by = 2)
> x
[1]  1  3  5  7  9 11 13 15 17 19
```

The following command use pi as an incremental value.

```
> x<-5:1
  > x<-seq(1, 11, by = pi)
  > x
[1]  1.000000  4.141593  7.283185 10.424778
```

The complete command has the following format.

```
>seq(from=1,to=3, by =0.5)
[1] 1.0 1.5 2.0 2.5 3.0
```

1.16 POSITION AND KEYWORD APPROACHES

There are two ways to input data: position and key-word. The following one-line code, we use the position-variable approach. In other words, the meaning of the input variable depends on its position in the set of input variables.

```
> x<-seq(1,3,0.5)    # position variable approach
```

For the keyword approach, we add a key-word in front of each input value, such as *from=1*. One advantage of the key-word approach is that the order of input variables does not play a role. The following three statements are equivalent.

```
> seq(from=1,to=3,by=0.5) # they are equivalent
> seq(to=3,from=1,by=0.5)
> seq(by=0.5,to=3,from=1)
```

1.17 INPUTTING DATA VIA scan()

Another easy way to input data from your keyboard is to use the *scan()* function.

```
> x<-scan()
1: 1
2: 3
3: 4
4: 2.5
5: 5
6:
Read 5 items
> x
[1] 1.0 3.0 4.0 2.5 5.0
```

If you plan to input multiple columns, you can input as a vector first. Then use the *matrix()* function to convert it to what you need. The desired input format (two columns) is given in the right panel below.

```
> x<-scan()
1: 1 3 3 6 5 6 7 8
9:
Read 8 items
> y<-matrix(x,4,2,byrow=T)
> y
     [,1] [,2]
[1,]    1    3
[2,]    3    6
[3,]    5    6
[4,]    7    8
```

The output is shown below.

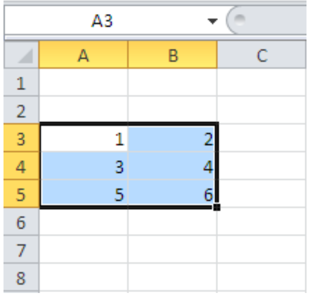
```
1 3
3 6
5 6
7 8
```

In the above example, we input data according to row. On the other hand if we input data according to column, we have to change our codes a little bit.

```
> x<-scan()
1: 1 3 5 7 3 6 6 8
9:
Read 8 items
> y<-matrix(x,4,2,byrow=F) # or use default y<-matrix(x,4,2)
```

1.18 GETTING DATA FROM clipboard

Here we discuss a very simple way to get data from Excel. This works for a small data set. For more complex data set from an Excel file, we will explain related functions in *Chapter 2*, *Simple function, Input data from external files*. Assume that we have the following Excel spread sheet (in the right panel below). To input the data into R, we can highlight and copy the data, shown below.

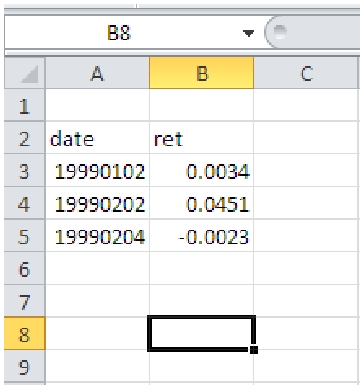


	A	B	C
1			
2			
3	1	2	
4	3	4	
5	5	6	
6			
7			
8			

Then issue the following command.

```
> x<-read.table("clipboard")
> x
  V1 V2
1  1  2
2  3  4
3  5  6
```

Our data might have a header (column names), shown below.

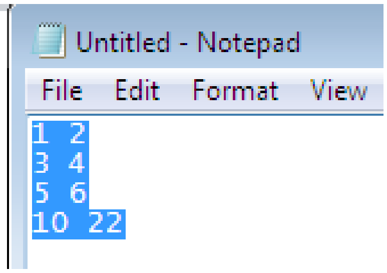


	A	B	C
1			
2	date	ret	
3	19990102	0.0034	
4	19990202	0.0451	
5	19990204	-0.0023	
6			
7			
8			
9			

For this case, we simpy add “header=T”, shown below.

```
> x<-read.table("clipboard", header=T)
> x
      date     ret
1 19990102  0.0034
2 19990202  0.0451
3 19990204 -0.0023
```

Above example is true when we open a notepad or MS-word file, shown below.

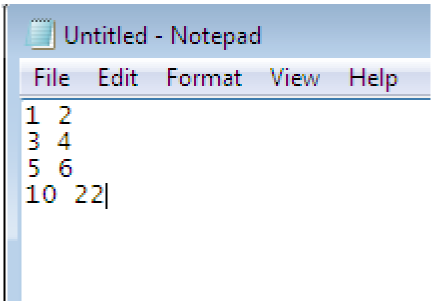


```
1 2
3 4
5 6
10 22
```

For the following code, we use an R function called `read.table()` and the input source is ‘clipboard’.

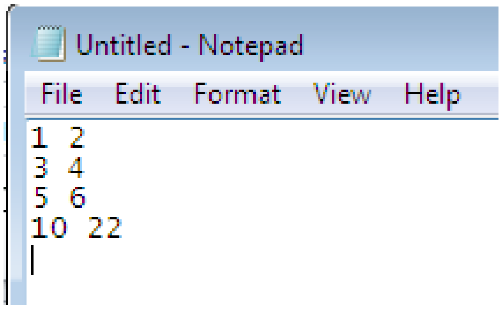
```
> y<-read.table("clipboard")
```

We should pay attention to the last row which should be the entry only, see the comparison for the two formats below. The wrong format is shown first. Pay attention to the cursor position of the last line.



```
1 2
3 4
5 6
10 22|
```

The correct input file is shown below.



```
1 2
3 4
5 6
10 22
|
```

For the format shown in the above left panel, we will get the following warning message when issuing the command of *x<-read.table("clipboard")*. Fortunately, the variable will take values it is supposed to get.

```
> x<-read.table("clipboard")
Warning message:
In read.table("clipboard") :
  incomplete final line found by readTableHeader on 'clipboard'
```

1.19 USING R AS A CALCULATOR

R can be used as a calculator since it is straight forward to call various embedded R functions. For example, the mean() function is for average.

```
> x<-1:50
> mean(x)
  [1] 25.5
```

You can try other functions as well, such as `max()`, `min()`, `median()`, `sd()` and `var()`.

```
> x<-1:50
> max(x)
  [1] 50
> min(x)
[1] 1
> median(x)
[1] 25.5
> sd(x)
[1] 14.57738
```

The following table summarizes a set of most widely used functions.

1.1: A list of some basic functions

function	Meaning	Examples
mean(x)	Mean	x<-1:10 mean(x) # [1] 5.5
median(x)	Median	median(x) # [1] 5.5
min(x)	Minimum	min(x) # [1] 1
max(x)	Maximum	max(x) # [1] 10
var(x)	Variance	>var(x) # [1] 9.166667
sd(x)	Standard deviation	sd(x) # [1] 3.027650
exp(x)	Exponential function	exp(2.3) # [1] 9.974182
log(x)	Natural log function	log(4.5) # [1] 1.504077
log10(x)	Log function based on 10	log10(4.3) # [1] 0.6334685
sum(x)	Take the summation	sum(x) # [1] 55
sort(x)	Sort in ascending order	
range(x)	Range of a variable	x<-1.5:10 range(x) # [1] 1.5 9.5
diff(x)	Calculate the difference for a vector	x<-c(1,2.3,4.5) diff(x) # [1] 1.3 2.2
ceiling(x)	Get the smallest integer larger than x	x<-9.5 ceiling(x) # [1] 10
floor(x)	Get the largest integer smaller than x	x<-9.5 floor(x) # [1] 9
as.integer(x)	Take the integer value	x<-9.5 as.integer(x) # [1] 9
prod()	Get product of a vector	> x<-1:3 > prod(x) # [1] 6
quantile(x)	> quantile(x) 0% 25% 50% 75% 100% -2.3210170 -0.6862249 0.1460511 0.7151348 2.0682096	
	> quantile(x,probs=c(0.01,0.05,0.95,0.99)) 1% 5% 95% 99% -2.000058 -1.609246 1.256627 1.476727	

Sometimes, we need to change the directory for convenience. The related procedure is given below.

```
# change the directory
# [click] File -- > "Change dir…" [choose your working directory]
```

We could change our starting directory by modifying the properties of our R icon.

Step 1: right click R icon on your desktop
Step 2: click "properties"
Step 3: choose your directory in "Start in", e.g., C:\test\

1.20 USING THE UP- OR DOWN-ARROW KEYS

We can use the up-arrow and down-arrow key to recall the previous command and modify it.

```
> x<-1:500
> y<-10:510
```

After issued a set of commands lines, we can use both Up and Down Arrow keys to move back and forth to recall and correct the ‘old’ commands. This is extremely convenient to check and modify our codes since we could recall the previous command with a new input or a minor modification.

```
# use upper (down) arrow keys to recall previous commands
```

1.21 FINDING HELP

There exist several ways to find information for a specific R function. If we are interested in the mean function, we could issue ?mean, help(mean) or example(mean).

```
>?mean
```

The command of help(mean) achieves the same goal as ?mean.

```
>help(mean)
```

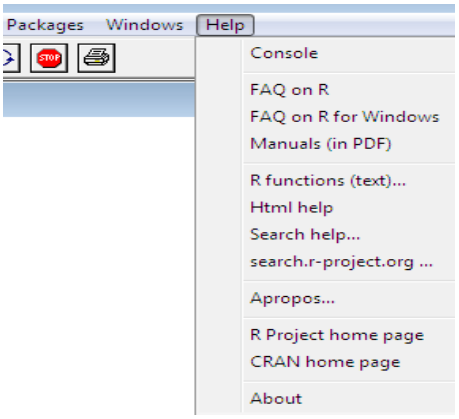
To get examples for a specific function, we use the example() function.

```
>example(mean)
```

We could also use the help on the menu bar.

```
> # [click] "Help" - -> "FAQ on R"
```

The following picture shows all the entries after clicking the “help” icon on the menu bar.



When not sure about the spelling of a function in question, we use the *apropos()* function.

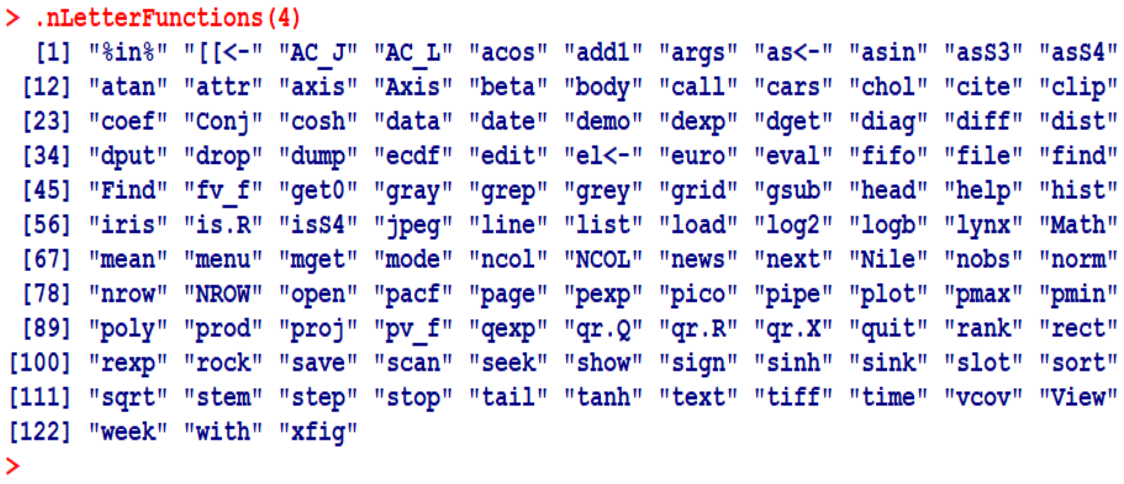
```
 > apropos("mea")
  [1] "colMeans"          "influence.measures" "kmeans"
  [4] "mean"              "mean.data.frame"    "mean.Date"
  [7] "mean.default"      "mean.difftime"      "mean.POSIXct"
 [10] "mean.POSIXlt"      "rowMeans"           "weighted.mean"
```

One alternative is to use the `find()` function which would achieve the same goal.

1.22 A FUNCTION CALLED .nLetterFunction()

To help readers to collect all n-letter embedded functions, we have generated a function called `.nLetterFunction`. For example, if we want to know all 4-letter functions, we could issue the following command. Note that there is a dot in front of `nLetterFunction`. Later in the book, we will explain why we add a dot in from of a function.

The result is shown below.



```
> .nLetterFunctions(4)
  [1] "%in%" "[[<-" "AC_J" "AC_L" "acos" "add1" "args" "as<-" "asin" "asS3" "asS4"
 [12] "atan" "attr" "axis" "Axis" "beta" "body" "call" "cars" "chol" "cite" "clip"
 [23] "coef" "Conj" "cosh" "data" "date" "demo" "dexp" "dget" "diag" "diff" "dist"
 [34] "dput" "drop" "dump" "ecdf" "edit" "el<-" "euro" "eval" "fifo" "file" "find"
 [45] "Find" "fv_f" "get0" "gray" "grep" "grey" "grid" "gsub" "head" "help" "hist"
 [56] "iris" "is.R" "isS4" "jpeg" "line" "list" "load" "log2" "logb" "lynx" "Math"
 [67] "mean" "menu" "mget" "mode" "ncol" "NCOL" "news" "next" "Nile" "nobs" "norm"
 [78] "nrow" "NROW" "open" "pacf" "page" "pexp" "pico" "pipe" "plot" "pmax" "pmin"
 [89] "poly" "prod" "proj" "pv_f" "qexp" "qr.Q" "qr.R" "qr.X" "quit" "rank" "rect"
[100] "rexp" "rock" "save" "scan" "seek" "show" "sign" "sinh" "sink" "slot" "sort"
[111] "sqrt" "stem" "step" "stop" "tail" "tanh" "text" "tiff" "time" "vcov" "View"
[122] "week" "with" "xfig"
>
```

1.23 VIDEOS FOR THIS CHAPTER

Since this course could be learnt as a online course, we have generated a few related videos. In addition, we include others' videos as well.

Video #1: (add later)

Video #2: (add later)

Video #3: How to download and install R (6m39s) [for windows)

https://www.youtube.com/watch?v=ZoPJGmpYJzw

Video #4: Programming in R - Getting Started - Installing R and RStudio on a Mac (5m59s)

https://www.youtube.com/watch?v=Ywj6yNfc5nM

1.24 PRECISION OF R

Most of times, precision of our R software is not an issue for most researchers or calculation. However, knowing how to find it would be helpful is you have such an issue in the future.

```
> .Machine$double.eps
[1] 2.220446e-16
```

SUMMARY

In this chapter, we discuss how to install R, basics and value assignments. Those are most basic concepts for learning a computer language. Later in the book, we will use those concepts repeatly. Thus, a new user should have certain confidence if he/she feels overwhelmed by those new concepts.

In the next chapter, *Chapter 2*, *Simple function and inputting data from an external file*, first we will discuss how to write one-line R function. Then, we explain how to extend it to a multi-line function. In addition, we explain how to add comments to make our functions more readable. It means that we offer the objective of the functions, definitions of input variables, any default values, plus a few examples. Because of those extra comments, our functions become self-explanatory. After that we will discuss several ways to retrieve data from external files, such as csv (comma separated values), txt (text), RData, rds, sas7bdat (SAS data format), and .xlsm, xlsx, and .xlsm (Excel) files.

REFERENCES

Kane, David and Joseph D. Masters, 2009, Open Source Finance, Investing, https://joi.pm-research.com/content/18/1/92

Lucas, and Jetee, 2009, Comparison of data analysis packages: R, Matlab, SciPy, Excel, SAS, SPSS, Stata, https://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

Kane, David, 2019, Open Source Finance, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=966354

Hayes, Bob, 2019,Programming Languages Most Used and Recommended by Data Scientists,https://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/

Yan, Yuxing, 2018, CRSP for Teaching, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3303504

Yan, Yuxing,2018, Financial Modeling using R, Lagar Book.

Yan, Yuxing, and James Yan, 2018, Hands-on Data Science with Anaconda, *Packt Publishing*.

Yan, Yuxing, 2017, Python for Finance (2nd edition) Packt Publishing.

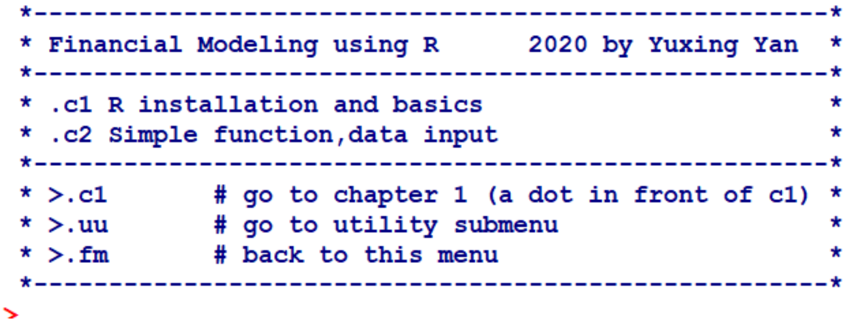
Yan, Yuxing, 2016, Teaching programming skills to finance students: how to design and teach a great course, https://link.springer.com/article/10.1186/s40854-017-0081-x

APPENDIX A

For a 15-week course, copy and paste the following line on to the R window.

```
source('http://datayyy.com/fmr/week1.R')
```

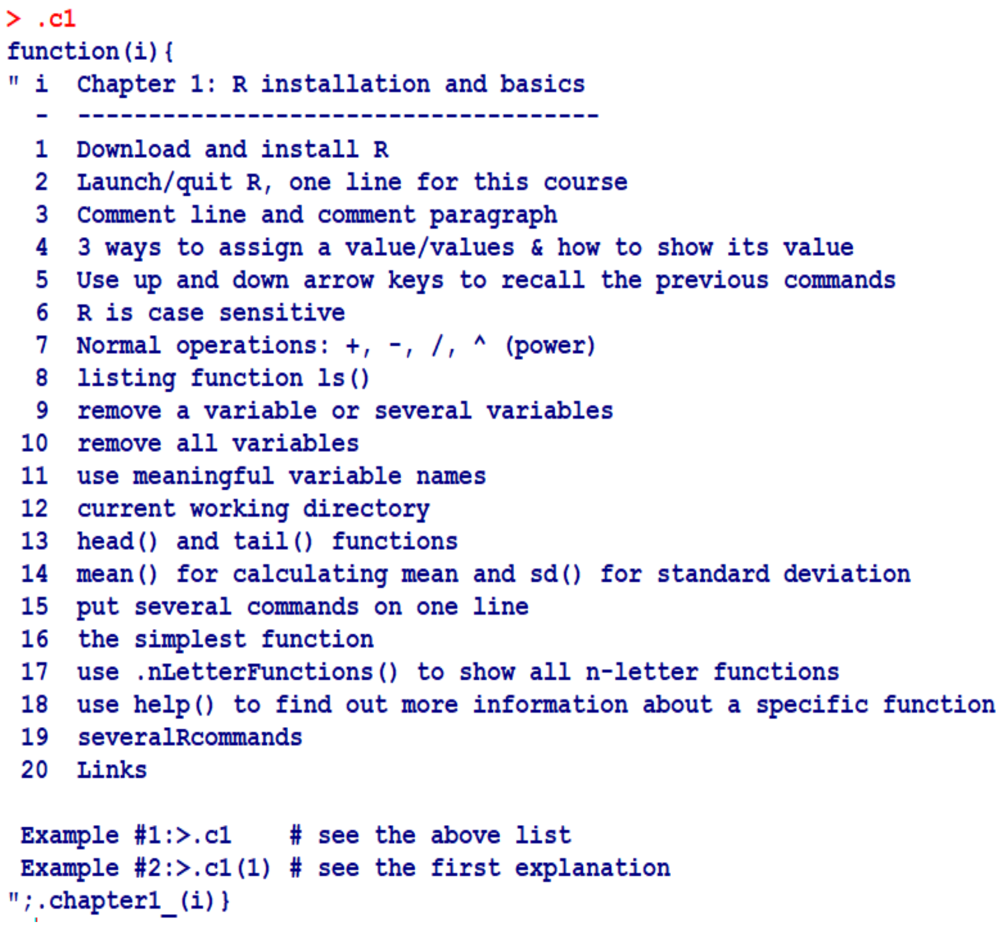
After hitting the enter-key, the following menu would pop up. Obviously, for the first week, we will cover two chapters.



```
*----------------------------------------------------*
* Financial Modeling using R      2020 by Yuxing Yan  *
*----------------------------------------------------*
* .c1 R installation and basics                       *
* .c2 Simple function,data input                      *
*----------------------------------------------------*
* >.c1        # go to chapter 1 (a dot in front of c1) *
* >.uu        # go to utility submenu                  *
* >.fm        # back to this menu                      *
*----------------------------------------------------*
>
```

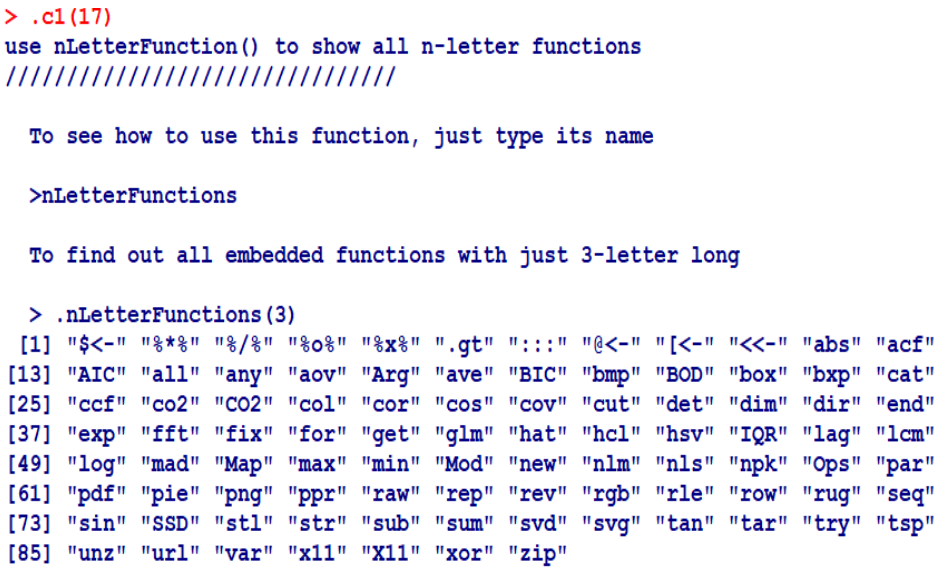
Appendix B: finding the contents of chapter 1.

To view the contents of chapter 1, we type `.c1`. Note that there is a dot in front of `c1`



```
> .c1
function(i){
" i  Chapter 1: R installation and basics
  -  -------------------------------------
  1  Download and install R
  2  Launch/quit R, one line for this course
  3  Comment line and comment paragraph
  4  3 ways to assign a value/values & how to show its value
  5  Use up and down arrow keys to recall the previous commands
  6  R is case sensitive
  7  Normal operations: +, -, /, ^ (power)
  8  listing function ls()
  9  remove a variable or several variables
 10  remove all variables
 11  use meaningful variable names
 12  current working directory
 13  head() and tail() functions
 14  mean() for calculating mean and sd() for standard deviation
 15  put several commands on one line
 16  the simplest function
 17  use .nLetterFunctions() to show all n-letter functions
 18  use help() to find out more information about a specific function
 19  severalRcommands
 20  Links

 Example #1:>.c1    # see the above list
 Example #2:>.c1(1) # see the first explanation
";.chapter1_(i)}
```

To see each entry, we simply use the upper-arrow key to recall our previous command and modify the command according to you needs. For example, we are interested in a function called `.nLetterFunction`, we could issue .c1(17), shown below.

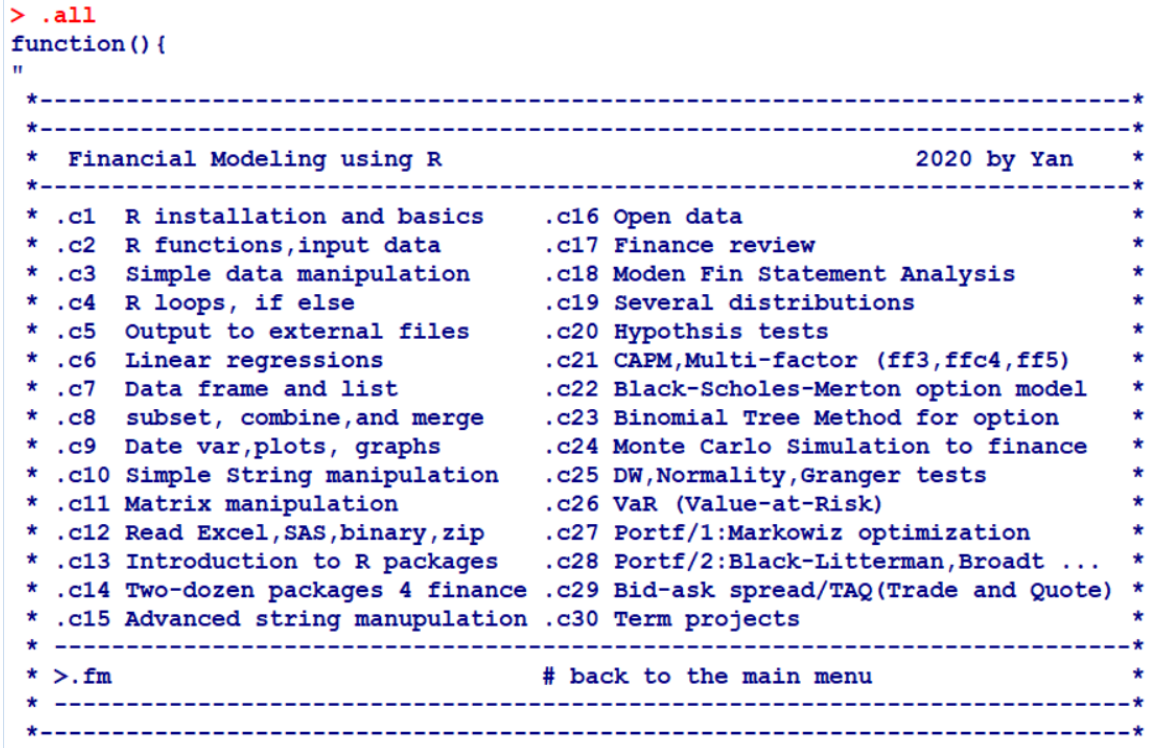


```
> .c1(17)
use nLetterFunction() to show all n-letter functions
////////////////////////////////

  To see how to use this function, just type its name

  >nLetterFunctions

  To find out all embedded functions with just 3-letter long

  > .nLetterFunctions(3)
 [1] "$<-" "%*%" "%/%" "%o%" "%x%" ".gt" ":::" "@<-" "[<-" "<<-" "abs" "acf"
[13] "AIC" "all" "any" "aov" "Arg" "ave" "BIC" "bmp" "BOD" "box" "bxp" "cat"
[25] "ccf" "co2" "CO2" "col" "cor" "cos" "cov" "cut" "det" "dim" "dir" "end"
[37] "exp" "fft" "fix" "for" "get" "glm" "hat" "hcl" "hsv" "IQR" "lag" "lcm"
[49] "log" "mad" "Map" "max" "min" "Mod" "new" "nlm" "nls" "npk" "Ops" "par"
[61] "pdf" "pie" "png" "ppr" "raw" "rep" "rev" "rgb" "rle" "row" "rug" "seq"
[73] "sin" "SSD" "stl" "str" "sub" "sum" "svd" "svg" "tan" "tar" "try" "tsp"
[85] "unz" "url" "var" "x11" "X11" "xor" "zip"
```

Later in the book, we will discuss how to write such functions.

Appendix C: Title of all chapters

To see the titles of all chapters, just type `.all`, shown below.



```
> .all
function(){
"
 *------------------------------------------------------------------------*
 *------------------------------------------------------------------------*
 *  Financial Modeling using R                               2020 by Yan    *
 *------------------------------------------------------------------------*
 * .c1  R installation and basics    .c16 Open data                         *
 * .c2  R functions,input data       .c17 Finance review                    *
 * .c3  Simple data manipulation     .c18 Moden Fin Statement Analysis      *
 * .c4  R loops, if else             .c19 Several distributions             *
 * .c5  Output to external files     .c20 Hypothsis tests                   *
 * .c6  Linear regressions           .c21 CAPM,Multi-factor (ff3,ffc4,ff5)  *
 * .c7  Data frame and list          .c22 Black-Scholes-Merton option model *
 * .c8  subset, combine,and merge    .c23 Binomial Tree Method for option   *
 * .c9  Date var,plots, graphs       .c24 Monte Carlo Simulation to finance *
 * .c10 Simple String manipulation   .c25 DW,Normality,Granger tests        *
 * .c11 Matrix manipulation          .c26 VaR (Value-at-Risk)               *
 * .c12 Read Excel,SAS,binary,zip    .c27 Portf/1:Markowiz optimization     *
 * .c13 Introduction to R packages   .c28 Portf/2:Black-Litterman,Broadt ...  *
 * .c14 Two-dozen packages 4 finance .c29 Bid-ask spread/TAQ(Trade and Quote) *
 * .c15 Advanced string manupulation .c30 Term projects                     *
 * -----------------------------------------------------------------------*
 * >.fm                              # back to the main menu                *
 * -----------------------------------------------------------------------*
 *------------------------------------------------------------------------*
```

Appendix D: a sample syllabus

University at Buffalo
Office of Academic Services
The Graduate School

Course Subject Code: MGF,Course Number: 694
Type of Instruction: Lecture
Course Title: Financial Modeling using R
Class Number: 1,Semester: Fall 2020

Course Information

- Instructor: Paul Yan
- Contact information :
  - Office: 359 Jacobs Management Center (Monday afternoon only)
  - Emails: yyan6@buffalo.edu and pyan@geneseo.edu
  - Phones: 716 645 3277 (Monday afternoon), 716 888 2604 (other weekdays)
- Date(s)/Time(s): Mondays: 5:00pm – 7:50pm @ Jacobs 214
- Office hours: Mondays: 2:00pm – 4:00pm. @ Jacobs 359 or by appointment
- Delivery mode: traditional lecture. Each class will be consist of two parts: lecture (including discussion of homework & potential topics for term projects) and hands-on (in-class exercises).
- Number of credits: 3

Course Description

- Unlike many Financial Modeling courses that using Excel, this course uses R as the computational tool. R is free and powerful software. In terms of data, students learn how to download and process public data associated with economics, finance and accounting. Over the past several years, many business schools have established many business analytics programs.

The trend is obvious: students at business school should master at least one computer language. From this course, students learn how to apply R to various finance theories.

- Course prerequisites: Minimum two finance courses, such as Corporate Finance and Portfolio Analysis

Course Materials

- For Fall 2020, I will supply all chapters in a PDF format [new edition of the book]
- Reference: Financial Modeling using R by Yuxing Yan, ISBN: 9-78-1-94696454, Lagaia Books, 2018, Amazon link: http://datayyy.com/webs/amazonR2018.html
- Websites:
    - http://datayyy.com/fmr/, http://datayyy.com/webs/R.shtml
    - https://ublearns.buffalo.edu/, https://www.r-project.org/,
- References:
    - An Introduction to R http://cran.wustl.edu/doc/manuals/r-release/R-intro.pdf
    - R Language Definition http://cran.wustl.edu/doc/manuals/r-release/R-lang.pdf
- One-line R code source("http://datayyy.com/abc.R") # I will explain it during the 1st lecture.

Software

- R, open source statistical and computational software . Students are expected to spend at least 1 hour per day on R outside the classroom.

QR Code (Later, I will show how to generate a QR code for a given webpage)

UB learns My book on Amazon Course webpage

Student Learning Outcome

- Learn/review basic financial concepts such as Ratio Analysis, Portfolio Theory, CAPM, Fama-French-Carhart Factor Model, Monte Carlo simulation, Options Theory, VaR (Value at Risk) and Market microstructure
- Learn and apply R to finance
- Focus on publicly available financial data such as Yahoo Finance, Google Finance, Prof. French's Data Library and Federal Reserve Economic Data Library (FRED).
- Learn how to use CRSP, Compustat and/or TAQ (Trade and Quote) databases.
- Learn how to make a good presentation

Course Requirements

- Regular attendance since hands-on is critical
- Class participation (10% of the final grade)
- About 6-7 Home work
- Mid-term and final exams
- Term project (a group project) and term project presentation

Grading Policy

Homework	30%
Midterm	20%
Final exam	25%
Group project	10%
Group presentation	5%
Class participation	10%
--	
Total	100%

From a percentage grade to a letter grade

Percentage grade	Letter grade
$grade \geq 90\ \%$	A
$85\% \leq grade < 90\%$	A-
$80\% \leq grade < 85\%$	B+
$75\% \leq grade < 80\%$	B
$70\% \leq grade < 75\%$	B-
$60\% \leq grade < 70\%$	C
$grade < 60\%$	F

Mid-term and Final

All exams (midterm and final) will be conducted in the computer lab. Those are open book exams with three types of questions: related to 1) finance; 2) R or 3) financial data.

Group project

Each group can have up to three members. A topic should be closely associated with this course. The maximum number of pages of your report is 15 with 12-point font. Please discuss with me your topic before you start to work on it. Some basic criterions are listed below.

Real world topics are especially encouraged. Three parts are essential:

1) theory and background of the topic,
2) R programs with a short explanation of the codes,
3) final data set (plus the codes to process the data, the source of raw data)
   Note: please do not send me your raw data.

The second type of projects is to study one of R packages. Three parts are critical:

1) why this specific package is useful in finance
2) a summary of all or most important functions offered by the package
3) examples to use them

Note: see a list of potential topics, at the end of the syllabus, for the group projects.

Weekly Course Schedule (tentative and subject to change)

#	Date	Topics	Description (F for Finance)	Data case
1	8/31	Syllabus discussion, introduction to R Open data	A short survey, self-intro, syllabus, course structure, mid-term/final Chapter 1: R Installation, basics and value assignments assignment, basic math functions: mean(), min(),max(), median(), sd(), as a scientific calculator Chapter 16: Open data for finance/economics/accounting	
2	**9/7**		**Labor Day (no Class)**	
3	9/14	Writing simple R functions	Chapter 2: Simple function, input data from external files Writing one-line functions, multi-lines, add help double_f(), pv_f(), fv_f(), IRR(), pv_annuity(), fv_annuity(), pv(perpetuity), pv(perpetuity_due), how to call your functions? use R as a financial calculator, input data from a text file, simple programs to get data, French's data library Chapter 17: Review of basic finance concepts, several decision rules Time-value, NPV, IRR, Payback rules	Data case #1
4	9/21	Data manipulation, Fin Statement Analysis	Chapter 3: Simple data manipulation Chapter 18: Modern financial statement analysis ratio analysis, probability ratio: operating margin, net profit margin, ROA, ROE, current ratio, book debt-	Data Case #2

			equity ratio, SEC filings (from 2009 to today)	
5	9/28	R loops, if else if, several distributions	Chapter 4: R loops, if else, if else if Chapter 19: Several distributions: Normal, student-t, Chisq and F-distributions	Data Case #3
6	10/5	Output, Hypothesis tests	Chapter 5: Output data to an external file Chapter 20: Hypothesis Tests	Data Case #4
7	10/12	Linear regressions, CAPM,ff3,ffc4	Chapter 6: Linear Regression Chapter 21: CAPM (Capital Asset Pricing Model) β estimation, rolling/portfolio β, hedging portfolio market risk, ff3, ffc4, ff5	Data Case #5
8	10/19	Multi-factor models Ratios	Chapter 7: Data frame and list Chapter 22: Option theory, Black-Scholes-Merton model trading strategies with options, implied volatility, put-call parity, hedging strategy, pnorm() , bs_f.R, implied_vol.R, greeks.R	Data Case #6
9	**10/26**		**Midterm**	

Continued

Week	Date	Topic	Chapters
10	11/2	T-test, F-test, Autocorrelation, Causality	Chapter 8: subset, combine data sets, and merge Chapter 22: Binomial Tree Method for option Chapter 30: Term project (please choose one topic)
11	11/9	Monte Carlo Simulation	Chapter 9: date variable, simple plots and graphs Chapter 10: Simple string manipulation Chapter 24: Applications of Monte Carlo Simulation to finance assumptions, normality test, estimate variance-covariance matrix, conversion variances between different frequencies, path dependent options, sensitivity analysis, scenario analysis, random number from normal, uniform distribution , price European and Asian options
12	11/16	CRSP for teaching using R	Chapter 11: Matrix manipulation Chapter 12: Reading Excel data, SAS data, binary data and zip files Chapter 25: Durbin-Watson, Normality,Granger causality tests T-test for significance, equality of means, F-test for difference of volatility, Granger causality test, Durbin-Watson autocorrelation test, t.test(), var.test(), dwtest(), Wilcoxon.test(), granger_test() Chapter 25: VaR (Value-at-Risk) standard normal distribution, thick tail distribution VaR_01.R, VaR_02.R, several R packages
13	11/23	Portfolio theory(2)	Chapter 13: Introduction to R packages Chapter 14: Two-dozen R packages related to finance Chapter 27: Portfolio Theory (Part I)

			var, std, correlation, return matrix, portfolio return, portfolio vol of 2-stock (n-stock),var-cov matrix, portfolio optimization, several R package, fPortfolio Extra: Introduction to CRSP What is CRSP? CRSP monthly, daily time series data, event data (more topics for term projects), stockMonthly, indexMonthly, indexDaily, stockD1925 to stockD2014, various R program to retrieve/process data efficiently
14	11/30	Spread estimation Task view for finance	Chapter 28: Portfolio Theory (Part II): Black-Litterman, Broadt et al. Chapter 29: Bid-ask spread and TAQ (Trade and Quote) Extra : Introduction to Compustat, annual data, several data sets **group presentation?**
15	**12/7**	**Presentation**	**Rest of groups**
	Extra	Text Analysis	Number of lines, number of words, word frequency, search keywords, distance between words etc.
	Extra	TAQ for teaching using R	TAQ (Trade and Quote) are high frequency database by NYSE MTAQ (up to second) DTAQ (up to millisecond) R data sets: TAQct, TAQcq, DTAQ22ct, DTAQ22cq TORQct, TORQcq, TORQcd, TORQsod R: loadTAQ, filterCT, filterCQ, spread, relativeSpread,leeRead
	Extra	Spread estimation from low-frequency	F: spread estimation from low frequency data, Roll (1984), Corwin and Schultz (2012) high-low spread R: Roll.R, Corwin_Schultz.R
	Extra	Liquidity measure	F: Amihud (2002) illiquidity measure, Pastor and Stambaugh (2002) liquidity measure. R: Amihud.R, PS.R
	12/9	**Final**	Final-exam (7:15PM - 10:15PM Jacobs 214)

Add UB learn here

Appendix A: A list of potential topics for term projects

Warm up		1	Financial statement analysis
		2	An interned connected financial calculator (Yan, 2012)
		3	Correlations among stocks in US, UK, Canada, France, China, Japan and Australia
		4	A Business cycle indicator (Yan and Zhang, 2015)
		5	Use journal ranking data efficiently (SCImago Journal and Country Ranking)
		6	Find an optimal portfolio
		7	How much you need when you retire? Social Security Benefit calculator
		8	Which party, Republican or Democratic, could manage the economy better?
		9	Monte Carlo Simulation (standard normal distribution, one variables) VaR
		10	PCA (Principal Component Analysis)
Data	Public data	11	Generate R data sets for 200 stocks, CPI, GDP, Unemployment rate etc.
		12	Generate R data sets for Fama-French 3-factors, 5 factors etc.
		13	Generate R data sets for all SEC 10Q and 10K index files from SEC (1993-2015)
		14	Generate R data sets for TORQ (Trade, Order, Report and Quote) database
		15	Generate R data sets for TDAQ (millisecond by millisecond transaction data)
		16	Parse 10K data from SEC filings, generate related R data sets
	CR	17	Generate R data sets for one month's TAQ data (MTAQ)
		18	Generate R data sets for crspInfo, stockMonthly, indexMonthly for CRSP

		19	Generate R data sets for stockDaily, indexDaily for CRSP
		20	Generate R data sets for TDAQ for several months
Using public data		21	Are annual beta mean reversion?
		22	Test the January and weekday effects
		23	Does size effect exist?
		24	Tracking errors
		25	Z-score (bankruptcy prediction, Altman, 1968)
		26	52-week High trading strategy using more than 200 stocks
		27	estimate Roll (1984) spread from daily data (Roll, 1984)
		28	Assessment of multiple choice questions using R
		29	Monte Carlo Simulation (capital budgeting, replicate a Slot Machine)
		30	Monte Carlo Simulation (one variables) VaR, n correlated stocks
Using CRSP or TAQ		31	Replicate S&P500 EW (equal-weighted) and VW (value-weighted) returns
		32	Is liquidity factor priced? (Amihud, 2002)
		33	What is the color of your firm, blue or red? (Yan, 2014)
		34	Which model is the best, CAPM, FF3, FFC4, or FF5?
		35	Estimate spread, relative spread, expected spread etc. by using TAQ
		36	Process TAQ efficiently, how to process 30 year MTAQ data efficiently ?
		37	Replicate momentum trading strategy (Jegadeesh and Titman, 1993)
		38	Replicate industry momentum trading strategy (Moskowitz and Grinblatt, 1999)
		39	Replicate 52-week high trading strategy (George and Huang, 2004)
		40	Replicate max- trading strategy (Bali, Cakici and Whitelaw, 2011)
		41	Impact of business cycle on the above four trading strategies, (Yan and Zhang , 2015)

References (most of them are for term projects)

Altman, Edward, Altman, E. (1968), Financial ratios, discriminant analysis and the prediction of corporate bankruptcy, *Journal of Finance*, 23(4), pp 598-608.

Amihud, Yakov, 2002, Illiquidity and Stock returns, *Journal of Financial Markets* 5, 31-56.

Bali, Turan G., Nusret Cakici, and Robert F. Whitelaw, 2011, Maxing Out: Stocks as Lotteries and the Cross-Section of Expected Returns, *Journal of Financial Economics* 99 427-446.

George, Thomas J, and Chuan-Yang Huang, 2004, The 52-week High and Momentum Investing, *Journal of Finance* 54, 5, 2145-2176.

Jegadeesh, N., and S. Titman, 1993, Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency, *Journal of Finance* 48, 65-91.

Moskowitz, Tobias, and Mark Grinblatt, 1999, Do industries explain momentum? *Journal of Finance* 54, 2017-2069.

Pastor, L. & Stambaugh, R., 2003, Liquidity risk and expected stock returns, *Journal of Political Economy* 111, 642-685.

Roll, Richard, 1984, A simple implicit measure of the effective bid-ask spread in an efficient market, *Journal of Finance* 39, 1127-1139.

Yan, Yuxing, 2019, CRSP for Teaching, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3303504

Yan, Yuxing, 2018, Financial Modeling using R, ISBN: 9-78-1-94696454, *Lagaia Books*, Amazon link at http://datayyy.com/webs/amazonR2018.html

Yan, Yuxing, 2017, Teaching programming skills to finance students: how to design and teach a great course, *Financial Innovation*, https://link.springer.com/article/10.1186/s40854-017-0081-x

Yan, Yuxing, 2015, Red vs. Blue Stocks: Politics and Profitability of Firms, *Journal of Business and Policy Research* 10 (1), 117-138.

Yan, Yuxing, An internet connected financial calculator, 2012, *Journal of Accounting and Finance* 12(5), 59-70.

EXERCISES

1.1 What are the advantages of using R?
1.2 What does it mean 'Open-Source'?
1.3 List minimum three open-source software.
1.4 Compare R with Python and SAS. What are their advantages an disadvantages?
1.5 What is the home page of R software?
1.6 What is the difference between functions of `ls()` and `rm()`?
1.7 Generating a vector from 2 to 15 then from 20 to 40. Estimate its mean, standard deviation and median.
1.8 What might be the disadvantages of using R?
1.9 How to assign a value to a new variable?
1.10 Is R case-sensitive?
1.11 Is R free?
1.12 How to get help for R?
1.13 How to add a comment?
1.14 Is it difficult to install R?
1.15 Will the R compiler compile a comment line?
1.16 Does a space play a role in R's command?
1.17 How to download manuals related to R?
1.18 Use the scan() function to input 20 pairs of x and y.
1.19 Write an R program to input an Excel data set.
1.20 Input values for x range from 1 to 100 and 202 to 300.
1.21 Reverse the input values in the above exercise.
1.22 How many 5-letter embedded functions in R?
1.23 How to find the titles of all chapters?
1.24 How to get support when using R?